Vectorized Painting with Temporal Diffusion Curves

Yingjia Li, Xiao Zhai, Fei Hou, Yawen Liu, Aimin Hao, and Hong Qin, *Senior Member, IEEE*

Abstract—This paper presents a vector painting system for digital artworks. We first propose Temporal Diffusion Curve (TDC), a new form of vector graphics, and a novel random-access solver for modeling the evolution of strokes. With the help of a procedural stroke processing function, the TDC strokes can achieve various shapes and effects for multiple art styles. Based on these, we build a painting system of great potential. Thanks to the random-access solver, our method has real-time performance regardless of the rendering resolution, provides straightforward editing possibilities on strokes both at runtime and afterward, and is effective and straightforward for art production. Compared with the previous Diffusion Curve, our method uses strokes as the basic graphics primitives, which are able to intersect each other and much more consistent with the intuition and painting habits of human. We finally demonstrate that professional artists can create multiple genres of artworks with our painting system.

Index Terms—Vector graphics, heat equation, procedural model, real-time application

1 Introduction

As the craving for digital art grows with the rapid development of computer graphics, digital painting systems have been widely developed and used for decades. To date, these systems are capable of successfully mimicking various painting styles, and many existing methods focus on improving immersive and real-life like experience using physics-based simulations of fluid behaviors. To name a few, Curtis et al. [1] adopted a multi-layer paper model and the shallow water equations for creating watercolor effects. Chu et al. [2] did some impressive ink simulations using Lattice-Boltzmann method, and they also implemented a real-time watercolor painting system Expresii [3] that incorporates impacts from gravity. Huang et al. [4] reproduced Chinese calligraphy by replicating the diffusion of ink on "xuan" papers. Chen et al. [5] achieved real-time 3D oil painting in the Wetbrush system using a hybrid of Eulerian and Lagrangian approaches in simulating oil pigment.

Although these digital painting systems can produce good results, they all have common problems. On one hand, these systems use discretized basis in computation and final results, making it nontrivial to change resolution once the painting session begins. On the other hand, the discrete representation makes editing each individual stroke on the fly impractical without re-simulation. Vector graphics and vector painting systems are developed to overcome the above drawbacks. As the underlying representation, vector graphics have many advantages over the discretized basis, including the nature of being resolution independent, the ability for stroke editing and the computational efficiency. Traditional vector graphics are generated from rasterized images using meshes [6], but they lack the fundamental ability for editing. Later, Orzan et al. [7] proposed Diffusion Curve (DC) which solves the Laplace's equation using curves as the boundary conditions for color spreading. Although this technique excels in producing high-quality results, the DC images are composed of boundary curves rather than strokes, which inevitably contradicts the human intuition and painting habits, see Fig. 10. Moreover, the intersection of DCs results in artifacts, which restrains the flexibility of DC. Recently, DiVerdi et al. [8] put forward a vector watercolor painting engine using procedural stroke configurations, bringing great realism in the creation of the digital artworks.

This paper presents a more generic digital painting measure through a novel vector model for strokes. We first propose a new model of vector graphics, Temporal Diffusion Curve (TDC), which represents not only the graphics but also their evolution over time using continuous functions. This new model is piece-wisely parameterized and inherently supports random-access solving in real-time. The TDCs represent strokes similar to human habits of painting and they are able to intersect and overlap each other as usual strokes. Therefore, it is very suitable for modeling strokes. Meanwhile, we devise a procedural model for processing TDC strokes to realize richer visual effects, including smooth paint

• *Y. Li and X. Zhai are with the State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing 100191, China. E-mail: {liyingjiasss, zhaixiao43}@gmail.com.*
• *F. Hou is with the State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China, and also with the University of Chinese Academy of Sciences, Beijing 100049, China. E-mail: houfei@ios.ac.cn.*
• *Y. Liu is with the Beijing Piesat Information Technology Co., Ltd., Beijing 100195, China. E-mail: 945664721@qq.com.*
• *A. Hao is with the State Key Laboratory of Virtual Reality Technology and Systems, and Research Institute of Frontier Science, Beihang University, Beijing 100191, China. E-mail: ham@buaa.edu.cn.*
• *H. Qin is with the Department of Computer Science, Stony Brook University, Stony Brook, NY 11794 USA. E-mail: qin@cs.sunysb.edu.*

Manuscript received 22 Feb. 2019; revised 2 July 2019; accepted 11 July 2019. Date of publication 18 July 2019; date of current version 24 Nov. 2020. (Corresponding authors: Fei Hou and Hong Qin.) Recommended for acceptance by K. Zhou. Digital Object Identifier no. 10.1109/TVCG.2019.2929808

diffusion, irregular paint scattering, inter-stroke color mixing, etc. Based on these, we build a painting system of great potential. Specifically, our method has real-time performance regardless of the rendering resolution, provides straightforward editing possibilities on strokes both at runtime and afterward, and delivers various stroke effects for art production of multiple genres. In contrast to the former DC which solves the static Laplace's equation for color spreading, our method integrates the temporal heat equation instead with TDCs being the diffusion source. More concretely, we find the closed-form solution of the heat equation by using Fourier transform and only compute the numerical density right before the procedural stroke processing. In this way, the painting can use evolving strokes as the basic primitives as opposed to the counterintuitive motionless boundary curves in DC images, see Fig. 10 for details. We finally demonstrate that professional artists can create satisfying artworks of various kinds with our painting system.

In short, the main contributions of this work include:

- A new form of vector graphics, Temporal Diffusion Curve, which models the evolution of strokes;
- A random-access solver of the heat equation, which is efficient and suitable for vector graphics;
- A procedural TDC stroke processing function to provide richer visual effects, and;
- A real-time vector painting system which is efficient, easy to use, capable of editing on the fly, and can be used to create artworks of multiple styles.

2 Related Works

The vector graphics and digital painting systems are the most related topics of this paper. We briefly review the existing works of these categories in this section.

2.1 Vector Graphics

Vector graphics have many advantages over rasterized images, such as resolution-independence, sparse representation, compact storage and geometric editability. Traditional vector graphics are generated based on meshes [6], [9] and can be represented by vector primitives with colors, such as points, curves, and polygons. Lately, Favreau et al. [10] proposed a line drawing vectorization method that explicitly balances the fidelity to the input and result simplicity which is measured by the number of curves and their degrees.

Another important research direction of vector graphics is based on the idea of Diffusion Curve (DC) images which create vector images of smooth color gradients, as proposed by Orzan et al. [7]. The aim of DC is to solve a 2D Laplace's equation with known boundaries to obtain the desired vector image. For instance, Bowers et al. [11] presented a stochastic ray tracing strategy in which the curves define source radiance whose visible contribution will be integrated at a shading pixel to produce color. Sun et al. [12] used boundary element method where the Green's function is taken to transform the Laplace's equation into a boundary integral along DCs. Recent researches focus on controlling the color changes away from the determined boundaries by solving the bi-Laplace equation using thin-plate splines [13] or BEM [14]. Jeschke et al. [15] proposed a method of mixing multiple DCs, achieving higher degrees of freedom and similar results as solving the bi-Laplace equation, but with higher efficiency and better numerical stability. Most recently, Hou et al. [16] proposed a new DC extension, Poisson Vector Graphics (PVG), which provides more control over the resulting images through multiple sub-regions. PVG can easily produce photorealistic effects such as specular highlights, core shadows, translucency and halos. However, the DCs depict region boundaries contradicting human habits of painting and they cannot intersect each other restraining the production of artists.

2.2 Digital Painting Systems

Digital painting systems have been thoroughly studied in recent years. Most of the existing works focus on reproducing the painting experience in reality. To model pigments, these painting systems use various physics-based fluid simulation methods, including the diffusion equation [4], [17], the shallow water equation [1], [18] or the Navier-Stokes equation [5].

Curtis et al. [1] presented a very effective system for painting western watercolors. They used the shallow-water equation and a multi-layer paint model to simulate the pigment behaviors. Although it was not a strict simulation of physics, their system achieved very good visual results with the Kubelka-Munk diffuse reflection model [19] for rendering. Van Laerhoven et al. [20] used a similar multi-layer pigment model and the faster stable fluid method [21] to solve the Navier-Stokes equation, realizing a real-time watercolor painting system. In order to reproduce more realistic drawing experience, Chu et al. [2] proposed a real-time ink dispersion simulation on GPU by using the lattice Boltzmann method to process the percolation of ink on paper. Later, researchers are able to produce more complicated phenomenon during painting. For example, Blakovi [22] achieved wetting, drying and re-wetting already dried colors, while urikovi et al. [23] considered the vortex of watercolor pigments and used a combination of 2D grids and particles [24] to simulate the diffusion of ink. Different from these grid-based methods, DiVerdi et al. [8] present a procedural algorithm for generating watercolor-like dynamic paint behaviors using a particle-based model. Their stroke representation is also vectorized, which allows rendering at arbitrary resolutions.

Besides watercolor, the oil painting is also a very popular direction. However, digital oil painting systems are usually more difficult to implement due to the high viscosity. Baxter et al. [25] implemented the DAB system, which used a physics-based spline brush model and a triangular grid canvas model for paint transfer. Later, Baxter et al. [26] proposed the improved IMPaSTo system using a 2.5D fluid model to simulate pigment propagation. In addition, Chen et al. [5] implemented a real-time 3D oil painting system Wetbrush which used a new Eulerian-Lagrangian approach for simulating detailed liquid effects. In order to achieve real-time performance on portable hardware, Stuyck et al. [18] adopted Shallow Water Equation and a multi-layered structure to model the oil pigments.

There are also other art genres which draw the researchers' attention. For instance, to simulate Chinese calligraphy, Huang et al. [4] presented a GPU-based real-time system that

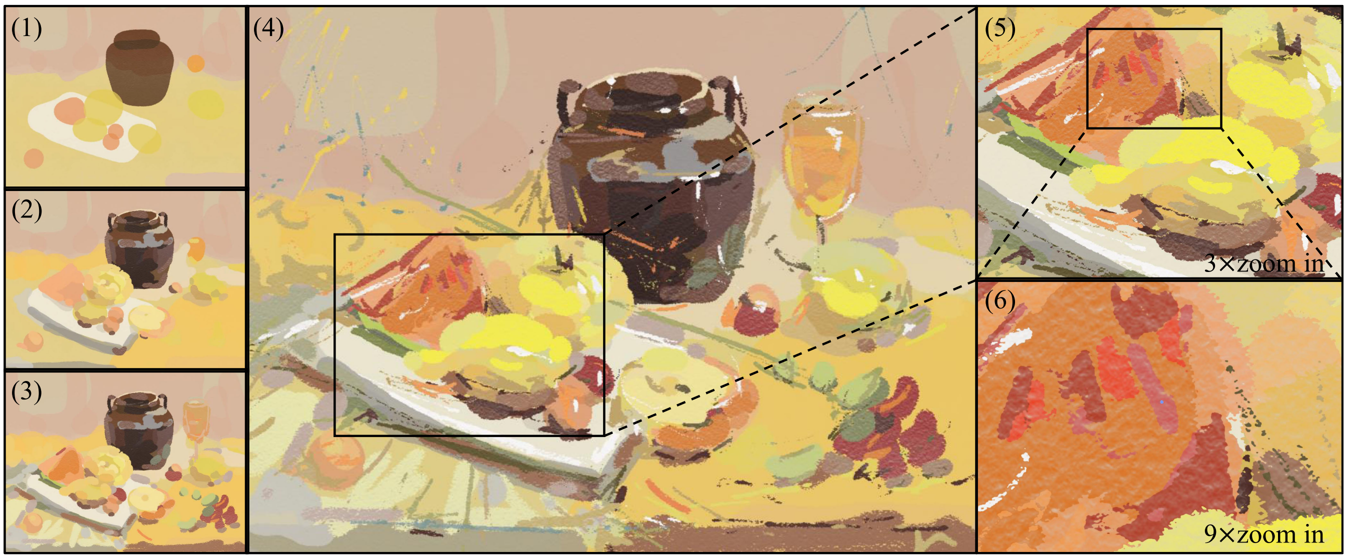


Fig. 1. (1)-(4) The painting session of a still-life artwork and (5)-(6) the zoom-in views of the final result. Our system recreates a very similar workflow to the real-life painting with brushes and papers and remains sharp when zoomed in significantly due to the nature of vector graphics.

includes physically-based brush deformation and seamless integration with ink diffusion rendering on "Xuan" paper structure. DiVerdi [27] conducted a very detailed analysis of the existing digital painting system. They decomposed the digital painting system into five components, including input control, tip shape, pigment transfer, canvas propagation, and pigment rendering. Furthermore, the results of digital paintings such as calligraphy, oil painting, watercolor, airbrush, pencil, and crayon have been categorized. Please refer to their survey for a more detailed summary on this topic.

3 Method Overview

As illustrated in Fig. 2, the drawing of strokes with our method comprises three major steps in cascade: the curve setup, the TDC solving and the procedural stroke processing. At the very beginning, a sequence of points sampled with a fixed time interval are taken as the input of our system. The curve setup step builds a curve based on these points and treats all the quantities along the stroke, including position, color, transparency and density, as continuous functions by using the cubic-spline fitting.

With the source curve determined, the second step is to solve the diffusive evolution process using the TDC formulation. Conventionally, this phenomenon is usually handled with finite-difference discretization and approximated with iterative solvers. However, this obviously contradicts the goal to keep our method vectorized. Instead, we develop a new method based on Fourier transform to reproduce the diffusion process, which does not require discretization during calculation. Therefore, the result of our solver can be exported at any resolution without losing details. Having the diffusion solved, an extra procedural model is applied to the result to render different effects on strokes, such as feathering, edge darkening, non-uniform scattering, etc. The TDC solving and procedural stroke modeling will be explained in depth in the following sections. We list the symbols used in our method in Table 1.

4 Temporal Diffusion Curve

Temporal Diffusion Curve, which can be considered as an extension of the DC [7], is proposed in this section for strokes modeling. There are mainly two differences between TDC and its predecessor. For one thing, DC handles the Laplace's equation for the static result, while TDC integrates the heat equation which represents the temporal evolution of color spreading. For another, DC uses color defined on boundary curves, which is fairly counterintuitive. In contrast, our TDC models the strokes directly, offering a user-friendly tool for painting purpose. In this section, to be suitable for vector graphics, we propose a random-access solver for the heat equation in 2D infinite domains with Fourier transform method, followed by the details of our TDC diffusion solver.

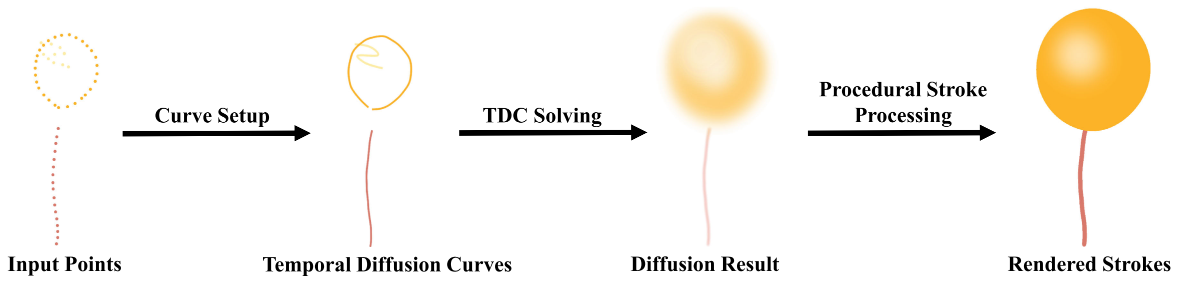


Fig. 2. The drawing of strokes comprises three major steps in cascade: The curve setup, the TDC solving and the procedural stroke processing.

4.1 Continuous Diffusion in 2D

The heat equation models the spatial distribution and temporal changes of the density variable $\phi(x, y, t)$ in 2D infinite domains, where the density is used to handle color opacity in this paper. Given the initial condition $\phi^0(x, y)$, the density can be seen as the following initial-value problem

$$\begin{cases} \frac{\partial\phi}{\partial t} - D(\frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2}) = 0 \\ \phi(x, y, 0) = \phi^0(x, y) \end{cases}, \tag{1}$$

where ϕ is short for $\phi(x, y, t)$ and D is the diffusion coefficient.

It is well known that the Fourier transform method can be used to solve the heat equation. We review the derivation here for ease of understanding. Applying 2D Fourier transform on both sides of the partial differential equation leads to an ordinary differential equation

$$\frac{d\Phi}{dt} + D(k_1^2 + k_2^2)\Phi = 0, \tag{2}$$

where k_1, k_2 are spatial frequencies and $\Phi(k_1, k_2, t)$ is the Fourier transform of $\phi(j_1, j_2, t)$

$$\Phi(k_1, k_2, t) = \iint_{-\infty}^{+\infty} \phi(j_1, j_2, t)\frac{e^{-i(k_1j_1+k_2j_2)}}{(2\pi)^2} dj_1dj_2, \tag{3}$$

where j_1, j_2 are integral variables standing for spatial coordinates. The initial condition is determined by

$$\Phi(k_1, k_2, 0) = \iint_{-\infty}^{+\infty} \phi^0(j_1, j_2)\frac{e^{-i(k_1j_1+k_2j_2)}}{(2\pi)^2} dj_1dj_2. \tag{4}$$

Therefore, we have

$$\Phi(k_1, k_2, t) = \Phi(k_1, k_2, 0)e^{-D(k_1^2+k_2^2)t}. \tag{5}$$

Applying the inverse Fourier transform on Φ gives

$$\begin{aligned} \phi(x, y, t) &= \iint_{-\infty}^{+\infty} \Phi(k_1, k_2, t)e^{i(k_1x+k_2y)}dk_1dk_2 \\ &= \iint_{-\infty}^{+\infty} \left[\iint_{-\infty}^{+\infty} \phi^0(j_1, j_2)\frac{e^{-i(k_1j_1+k_2j_2)}}{(2\pi)^2} dj_1dj_2\right] \\ &\quad e^{-D(k_1^2+k_2^2)t}e^{i(k_1x+k_2y)}dk_1dk_2 \\ &= \frac{1}{(2\pi)^2}\iint_{-\infty}^{+\infty} \phi^0(j_1, j_2)\left[\iint_{-\infty}^{+\infty} e^{-i(k_1j_1+k_2j_2)}\right. \\ &\quad \left. e^{-D(k_1^2+k_2^2)t}e^{i(k_1x+k_2y)}dk_1dk_2\right]dj_1dj_2 \\ &= \frac{1}{(2\pi)^2}\iint_{-\infty}^{+\infty} \phi^0(j_1, j_2)\left[\iint_{-\infty}^{+\infty} e^{i(x-j_1)k_1}\right. \\ &\quad \left. e^{i(y-j_2)k_2}e^{-Dk_1^2t}e^{-Dk_2^2t}dk_1dk_2\right]dj_1dj_2. \end{aligned} \tag{6}$$

According to the integral formula

$$\int_{-\infty}^{+\infty} e^{-ax^2}e^{bx}dx = \sqrt{\frac{\pi}{a}}e^{\frac{b^2}{4a}}, \tag{7}$$

TABLE 1
Definition of Notations

Notation	Description	Defined in	Type
ϕ^s	Segment density	TDC segment	Scalar
ϕ	Density	Canvas	Scalar
Φ	Fourier transform of ϕ	Canvas	Scalar
C^s	Segment color	TDC segment	4D vector
C	Stroke color	Canvas	4D vector
ω	Granulation opacity	TDC segment	Scalar
$\bar{\omega}$	Granulation opacity	Canvas	Scalar
D	Diffusion coefficient	TDC segment	Scalar
t^{max}	Maximum diffusion time	TDC segment	Scalar
η	Canvas texture	Canvas	Scalar

we have

$$\begin{aligned} \phi(x, y, t) =& \frac{1}{(2\pi)^2}\iint_{-\infty}^{+\infty} \phi^0(j_1, j_2)\left[\sqrt{\frac{\pi}{tD}}e^{\frac{(x-j_1)^2}{4tD}}\right. \\ & \left.\sqrt{\frac{\pi}{tD}}e^{\frac{(y-j_2)^2}{4tD}}\right]dj_1dj_2 \\ =& \frac{1}{4\pi tD}\iint_{-\infty}^{+\infty} \phi^0(j_1, j_2)e^{-\frac{(x-j_1)^2}{4tD}}e^{-\frac{(y-j_2)^2}{4tD}}dj_1dj_2, \end{aligned} \tag{8}$$

which is the continuous solution of the diffusion problem in Equation (1).

4.2 Diffusion with Temporal Diffusion Curves

We use TDCs as the diffusion source in our solver. A TDC is made up of several cubic splines, and each spline L can be represented by a pair of parametric equations of the parameter p

$$\begin{cases} x &= f(p) \\ y &= g(p) \end{cases}, \qquad p \in [p^{min}, p^{max}], \tag{9}$$

where $[p^{min}, p^{max}]$ denotes the domain of p. Initially, the density is 0 everywhere else but on the curve. $\phi^0(x, y)$ should hence be written using the 2D Dirac delta function as

$$\phi^0(x, y) = \phi^0(p)\delta(x - f(p))\delta(y - g(p)). \tag{10}$$

According to the properties of the δ function [28], Equation (8) can be rewritten as the line integral over L

$$\begin{aligned} \phi(x, y, t) =& \frac{1}{4\pi tD}\int_L \phi^0(p)e^{-\frac{(x-f(p))^2}{4tD}}e^{-\frac{(y-g(p))^2}{4at}}dl \\ =& \frac{1}{4\pi tD}\int_{p^{min}}^{p^{max}} \left[\phi^0(p)e^{-\frac{(x-f(p))^2}{4tD}}e^{-\frac{(y-g(p))^2}{4tD}}\right. \\ & \left.\sqrt{f'(p)^2 + g'(p)^2}\right]dp. \end{aligned} \tag{11}$$

With Equation (11), the diffusion result of any given value (x, y, t) reduces to a single-variable integral and are independent of other values. Thus, with the random-access solver, the output resolution can be infinitely magnified as long as the memory allows since each point can be calculated separately. Additionally, considering the display resolution is fixed, zooming-in only requires the pixel coordinates to be updated with no extra boundary processing, which ensures real-time performance for our system.

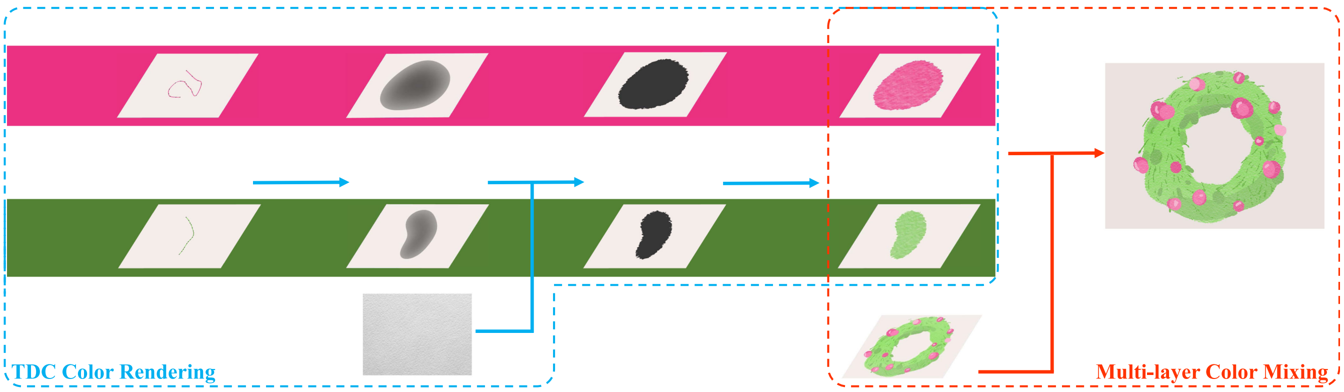


Fig. 3. A pipeline of the procedural stroke processing. The color rendering of a single stroke is divided into three steps. First, we calculate the TDC diffusion based on the parameters of TDC segments and rasterize the results according to the display resolution. The diffusion results are then modified towards stylization. Later, the stroke color is determined. Having the individual strokes updated, we use a multi-layer model to handle the color mixing among strokes and the existing canvas.

5 Procedural Stroke Processing

To extend the usability of our painting tool, we implement a procedural processing on TDC strokes to render various effects. In this section, we introduce the color rendering of a single TDC and the subsequent color mixing between the active curves and the existing canvas. As listed in Algorithm 1, the TDC color rendering comprises 3 steps, namely TDC integration, density modification, and stroke colorization. First, we calculate the TDC diffusion following the aforementioned model and rasterize the resulting function using the display resolution. The density modification is then applied to achieve a target stroke style. Later, the stroke color is determined based on the TDC density and the color parameters. Having the individual strokes updated, we use a multi-layer color mixing to finally complete the color update of one iteration, see Algorithm 2. A pipeline of this section can be found in Fig. 3.

Algorithm 1. The Framework of Single TDC Color Rendering

```
   Input: The parametric equation and parameters of the $j$th TDC.
   Output: The stroke color $C_j$ of the display resolution.
   /* TDC Integration */
1: for the $k$th TDC segment in $j$th TDC do
2:     Compute TDC segment density $\phi_{jk}$ using Equation (13).
3: Compute TDC density $\phi_j$ using Equation (14).
   /* Density Modification */
4: Incorporate the canvas texture using Equation (16).
5: Perform density modification using Equations (17), (18) or (19).
   /* TDC Colorization */
6: Compute averaged color $\bar{C}_j^s$ using Equation (20)
7: Compute stroke color $C_j$ using Equation (21)
```

5.1 TDC Integration

A TDC is made up of several spline segments, and the parameters are hence defined on each TDC segment, including the diffusion coefficient D_{jk}, the initial density ϕ^s_{jk}, the maximum diffusion time t^{max}_{jk}, the 4-channel color C^s_{jk} and the granulation opacity $\omega_{jk} \in [0, 1]$, where subscript jk indicates the kth segment on the jth TDC. The diffusion coefficient D_{jk} affects the width and blurring of the diffusion. The larger the diffusion coefficient, the larger the diffusion width and the more blurred the edge and vice-versa, as shown in Fig. 5. The initial density ϕ^s_{jk} is the value of $\phi^0(p)$ in Equation (11) on the TDC segment k. t^{max}_{jk} controls the life span of a TDC segment, beyond which the TDC segment freezes and is no longer taken into consideration in the diffusion. In all demonstrated results of this paper, t^{max}_{jk} is set to 1 second.

To calculate the diffusion, we write the parametric equation for each TDC segment k as

$$\begin{cases} x &= f_{jk}(p) \\ y &= g_{jk}(p) \end{cases}, \qquad p \in [p^{min}_{jk}, p^{max}_{jk}]. \tag{12}$$

By substituting the parametric equation and the parameters including D_{jk}, ϕ^s_{jk} and the segment age t_{jk} into Equation (11), we get the diffusion result $\phi_{jk}(x, y, t)$ of the TDC segment k through

$$\begin{aligned}\phi_{jk}(x,y,t_{jk}) = &\frac{1}{4\pi t_{jk} D_{jk}} \int_{p^{min}_{jk}}^{p^{max}_{jk}} \Bigg[\phi^s_{jk} e^{-\frac{(x-f_{jk}(p))^2}{4t_{jk}D_{jk}}} \\ &e^{-\frac{(y-g_{jk}(p))^2}{4t_{jk}D_{jk}}} \cdot \sqrt{f'_{jk}(p)^2 + g'_{jk}(p)^2} \Bigg] dp. \end{aligned} \tag{13}$$

The TDC density ϕ_j is defined as the sum of the TDC segment density of all the segments contained in the TDC

$$\phi_j = \sum_k \phi_{jk}. \tag{14}$$

5.2 Density Modification

To incorporate the canvas texture in strokes, we implement the granulation effect as follows. ϕ_{jk} and ϕ_j are first calculated and rasterized according to the display resolution. We use the segment density ϕ_{jk} as the weights to calculate the granulation opacity $\bar{\omega}_j$ of the jth TDC through

$$\bar{\omega}_j = \frac{\sum_k \phi_{jk} \omega_{jk}}{\phi_j}. \tag{15}$$

And then we use the $\bar{\omega}_j$ and an underlying canvas texture $\eta(x, y)$ to modulate the density through

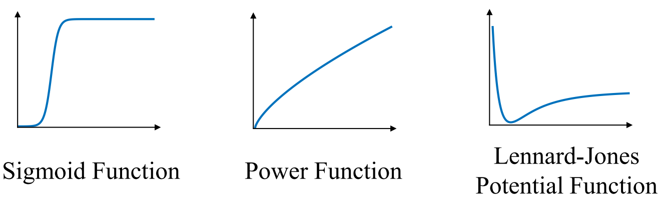


Fig. 4. Density modification functions.

$$\phi_j^{'} = [(1-\bar{\omega}_j)\eta(x,y) + \bar{\omega}_j]\phi_j, \tag{16}$$

where $\phi_j^{'}$ is the resulting density. As shown in Fig. 5, the amount of granulation can be adjusted on individual segments. If $\bar{\omega}_j = 1$, $\phi_j^{'}$ equals ϕ_j and the texture plays no part in the density; if $\bar{\omega}_j = 0$, $\phi_j^{'}$ equals $\eta(x,y)\phi_j$ and the influence from texture reaches maximum. In this paper, we use a gray-scale photo of a piece of paper as the canvas texture $\eta(x,y)$ which can be seen in Fig. 3.

In order to achieve different styles of strokes, we design three density modification functions shown in Fig. 4 for nonlinear brightness mapping. Hereinafter, we use ϕ_j^{mod} to indicate the density of TDC j after modification. The first density modification is the sigmoid function for uniform coloring, where the density remains almost constant near the stroke and drops sharply on edges

$$\phi_j^{mod} = \frac{1}{1+e^{-\frac{\phi_j^{'}-\xi_{u1}}{\xi_{u2}}}}, \tag{17}$$

where ξ_{u1} affects the stroke width and ξ_{u2} affects the rate of density drop on edges. The next one is the power function for the feathering effect, where the color of the curve fades gradually from near to far

$$\phi_j^{mod} = (\frac{\phi_j^{'}}{\xi_{f1}})^{\xi_{f2}}, \tag{18}$$

where ξ_{f1} affects the stroke width and ξ_{f2} affects the changing speed of color. The last one is a Lennard-Jones potential function for the edge darkening effect, where the density rises abruptly at the edges of the curve

$$\phi_j^{mod} = \xi_{d3}((\frac{\xi_{d1}}{\phi_j^{'}})^6 - (\frac{\xi_{d2}}{\phi_j^{'}})^3 + 1), \tag{19}$$

where ξ_{d1} and ξ_{d2} affects the width of dark edge and ξ_{d3} affects the shade of the stroke color.

5.3 Stroke Colorization

Now we introduce the colorization of strokes based on TDC density and colors. Given that each TDC segment has its own color C_{jk}^s, we use the segment density ϕ_{jk} as the weights to calculate the averaged color $\bar{C}_j^s$ through

$$\bar{C}_j^s = \frac{\sum_k \phi_{jk} C_{jk}^s}{\phi_j}. \tag{20}$$

After density modification, ϕ_j^{mod} is clipped to [0, 1] and is subsequently used to modify the opacity of $\bar{C}_j^s$. In this paper,

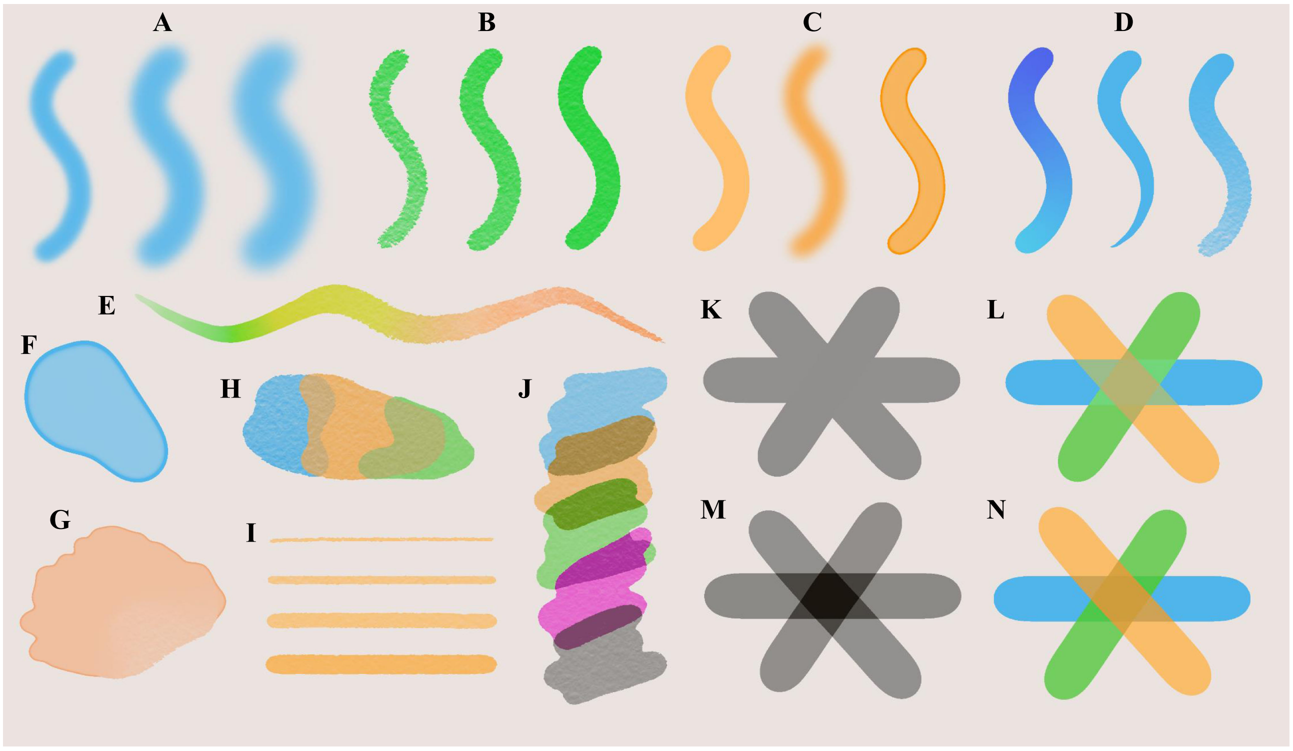


Fig. 5. The demonstration of strokes with different effects. (A) compares strokes using different diffusion coefficient (0.001, 0.003 and 0.005, respectively). (B) displays strokes with varied granulation opacity (0, 0.25 and 0.5, respectively). (C) exhibits different stroke styles (including uniform coloring, feathering and edge darkening, respectively). (D)(E) show the gradual change on different parameters along the stroke. (F)(G) present the areal edge darkening effect. (I) exhibits strokes of different widths. (H)(K)(L) display the result of transparency color mixing and (J)(M)(N) display the approximate color mixing of real pigments.

we use the RYB color model [29] of 4 channels, namely red, yellow, blue, and opacity. The stroke color C_j of TDC j is

$$\begin{aligned} C_j^{ryb} &= (\bar{C}_j^s)^{ryb}, \\ C_j^w &= \phi_j^{mod}(\bar{C}_j^s)^w, \end{aligned} \tag{21}$$

where the superscript ryb represents the red, yellow and blue channels and w the opacity channel.

5.4 Multi-Layer Color Mixing

We use a multi-layer model to handle the color mixing among strokes and the existing canvas, see Fig. 3. We first define a TDC as an expired TDC if all its segments' ages t_{jk} exceed t_{jk}^{max}; otherwise, the TDC is active. Each active TDC corresponds to an active layer in the multi-layer model, and all expired TDCs are mixed into the expired layer before they are excluded from the update. During the update, the layers are sorted according to the drawing order with the expired layer in front. Thus, we only consider the mixing between two color layers, with the foreground color being an active layer and the background color being the intermediate mixing result of all the previous layers.

In this paper, we use two different color mixing formulas from which artists can choose according to their needs, as proposed by Sugita et al. [29]. The first one is the transparency mix commonly used in computer graphics

$$C_\alpha = \alpha_{fore} C_{fore} + (1 - \alpha_{fore}) C_{back}, \tag{22}$$

where C_α is the result color, C_{fore} and C_{back} are the foreground and background color respectively, and $\alpha_{fore} \in [0, 1]$ is the transparency. The second one is an approximate mixing of real pigments

$$C_\beta = C_{fore} + \beta_{fore} C_{back}, \tag{23}$$

where C_β is the mixing result and $\beta_{fore} \in [0, 1]$ is the mixing parameter. Mixing using this method makes the resulting color darker than both the foreground color and background color.

When an active TDC expires, we delete the corresponding active layer and mix its color into the expired layer. The algorithm for one color-mixing iteration is listed in Algorithm 2.

Algorithm 2. The Framework of One Color-Mixing Iteration

```
Input: The active layers, the expired layer and the drawing order.
Output: The background color.
1 Sort the color layers according to the drawing order with the expired layer in front.
2 Set the background color as the first color layer.
3 for the $i$th pixel $(x, y)$ in parallel do
4   for the $j$th active layer do
5     if the $j$th TDC is expired then
6       Mix the $j$th active layer into the expired layer using Equations (22) or (23).
7       Delete the corresponding active layer.
8   for the $j$th active layer do
9     Perform color mixing between this layer and background using Equations (22) or (23).
```

6 Implementation and Results

In this section, we present the implementation details and some artworks created with our vector painting system. We further offer comparisons against the finite-difference solver and the DC method which proves that our method is not only effective and efficient but also intuitive and straightforward to use.

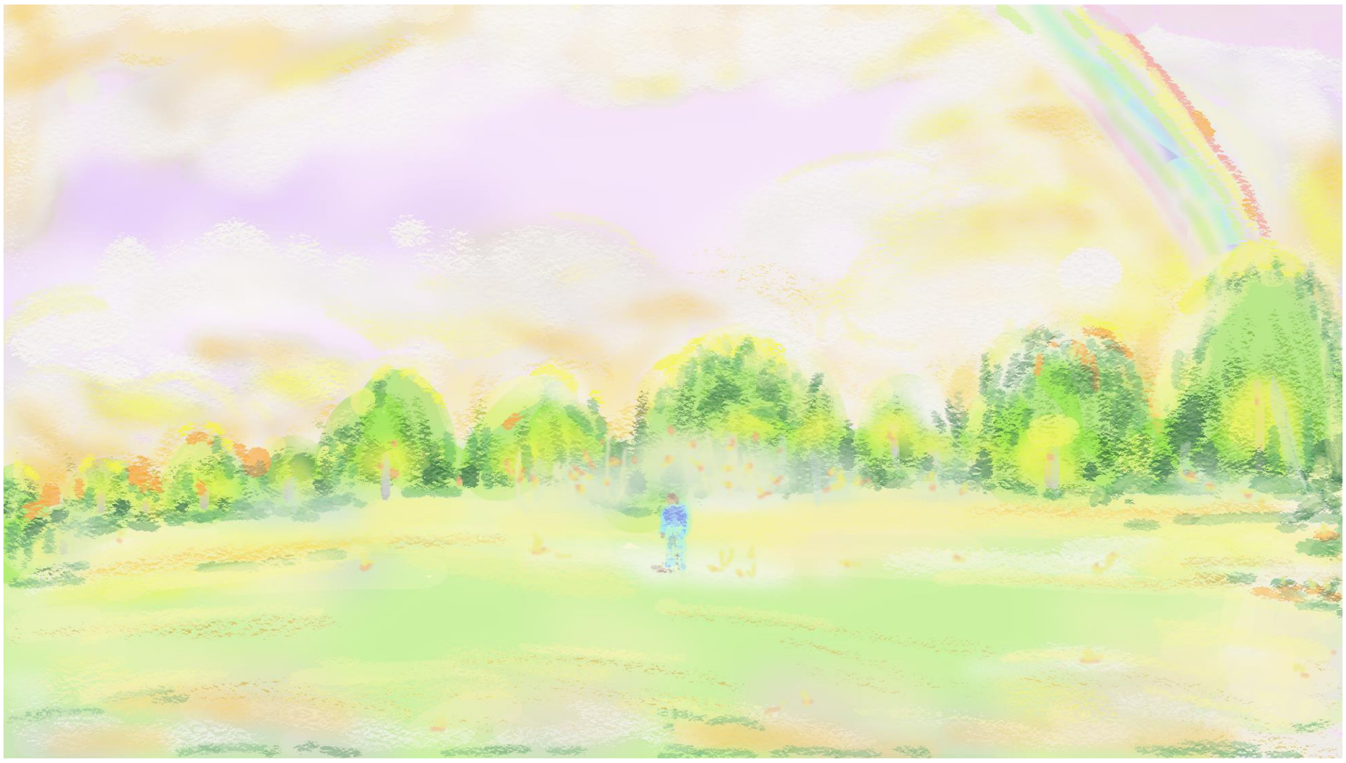
6.1 Implementation

We implemented our system using C++ and CUDA, with the real-time rendering using OpenGL and the user interface using Dear ImGui. We invoke a CUDA thread for each screen pixel so all pixels can be updated in parallel. To simplify the computation, a TDC segment only affects its nearby pixels within distance $d_{jk} = 0.5 l_{jk} + 0.1 W$, where l_{jk} is the length of segment jk, W is the canvas width, and d_{jk} is the distance between a pixel and the midpoint of segment jk. Our system runs at a stable 60 FPS under 1024×768 resolution on a PC with an Nvidia Geforce GTX 1070 GPU and Intel Core i7-7700K CPU. Meanwhile, we test the performance of our system under 3940×2160 resolution and show the runtime framerate changes in Fig. 6. The system runs above 30 FPS as long as the number of active TDC segments does not exceed 180, which is more than enough for painting purpose. Our system also provides powerful editing functions by which we can insert or delete TDC segments arbitrarily and modify all the parameters on each TDC segment individually. When zooming in or out, the display resolution remains unchanged and the performance of our system is therefore unaffected.

The states of our system, including the positions of all TDC segments and the relevant parameters, take very small space for storage and can be easily saved to a file. When reloading, we restore the painting session by redo all the calculations, which is pretty fast thanks to the random-access feature of our solver. For example, the painting Fig. 11 F, which contains 778 TDCs and 17917 TDC segments, only requires 426.3 KB to store in binary format and 0.138 seconds to reload to a 1024×768 canvas. The computing time for recreating the entire paintings in this paper from their TDC inputs is listed in Table 2.

6.2 Results and Comparisons

As demonstrated in Fig. 5, our painting tool can easily achieve multiple stroke styles, such as blurred edges, granulation, overlaid colors, etc. Using these effects, TDC can be exploited for a variety of painting styles, including but not limited to watercolor (Fig. 1), flat style (Figs. 7 and 10), sketch (Fig. 11 B) and ink wash painting (Fig. 11 E). Fig. 1 shows the painting session of a still-life artwork. In this example, the workflow is fairly similar to the real-life painting with brushes and papers where the artwork is drawn layer by layer onto the canvas with abundant color mixing and stroke combination. Fig. 7 illustrates a flat-style result created with our system, where the diffusion process can be clearly noticed during the drawing of each stroke. Continuous temporal feedback is usually a desireable feature for painting systems since it can be very helpful for artists to adjust their drawing shape on the fly, and the temporal expressiveness of TDC enables satisfying visual feedback once a stroke



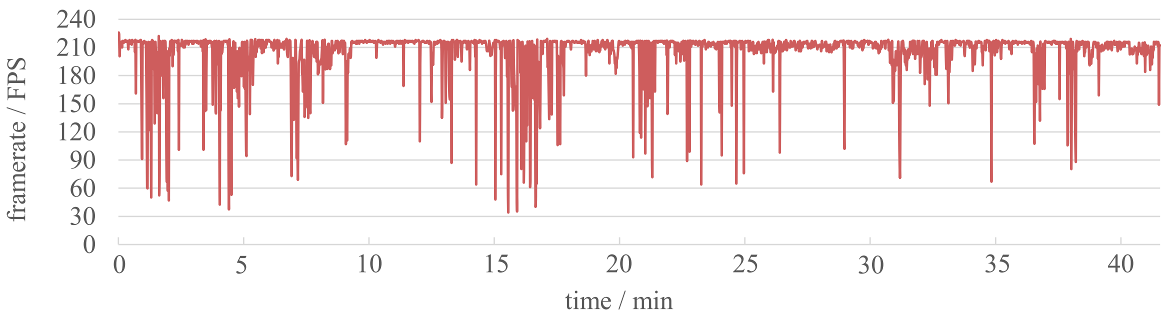


Fig. 6. A TDC painting under 4 K rendering resolution along with the framerate plot of the system during painting.

starts. Please refer to the supplementary video, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG.2019.2929808, to see more details on the temporal evolution of the strokes. Fig. 8 displays a TDC painting using a combination of various stroke effects. In Fig. 11 we show more painting results of our system.

TABLE 2
Performance of TDC

Result	# TDCs	# TDC segments	Total Time
Fig. 1	752	17065	0.142s
Fig. 7	197	5523	0.051s
Fig. 8	590	5228	0.044s
Fig. 9	206	1696	0.015s
Fig. 10	83	3780	0.036s
Fig. 11 A	479	13560	0.115s
Fig. 11 B	645	6938	0.066
Fig. 11 C	177	6033	0.061
Fig. 11 D	183	13617	0.091
Fig. 11 E	597	7769	0.067
Fig. 11 F	778	17917	0.138s
Fig. 11 G	431	14987	0.129s
Fig. 11 H	128	3723	0.035s

This table lists the number of TDCs, the number of TDC segments and the total computing time of the paintings rendered to a 1024×768 *canvas.*

Compared with the finite-difference method for solving the heat equation, our method is free of numerical dissipation, offers continuous solutions in both spatial and temporal dimensions, can have random-access evaluations at any given time and position, and keeps the computational burden constant when zoom-in applied. Fig. 9 demonstrates a visual comparison between our method and the finite-difference method, where both methods achieve similar results at 1024×768 resolution but our method remains sharp and displays more details when amplified 5 times. In this example, the finite-difference diffusion requires 10 time steps and totally 2.248 seconds to compute, while our method completes in 0.015 seconds.

As an extension of the DC method, our TDC solves the heat equation, while DC solves the Laplace's equation which is essentially equivalent to the steady-state heat equation where time approaches infinity. In other words, TDC is capable of modeling the temporal diffusion of strokes, which is one of the biggest advantages of our method. In

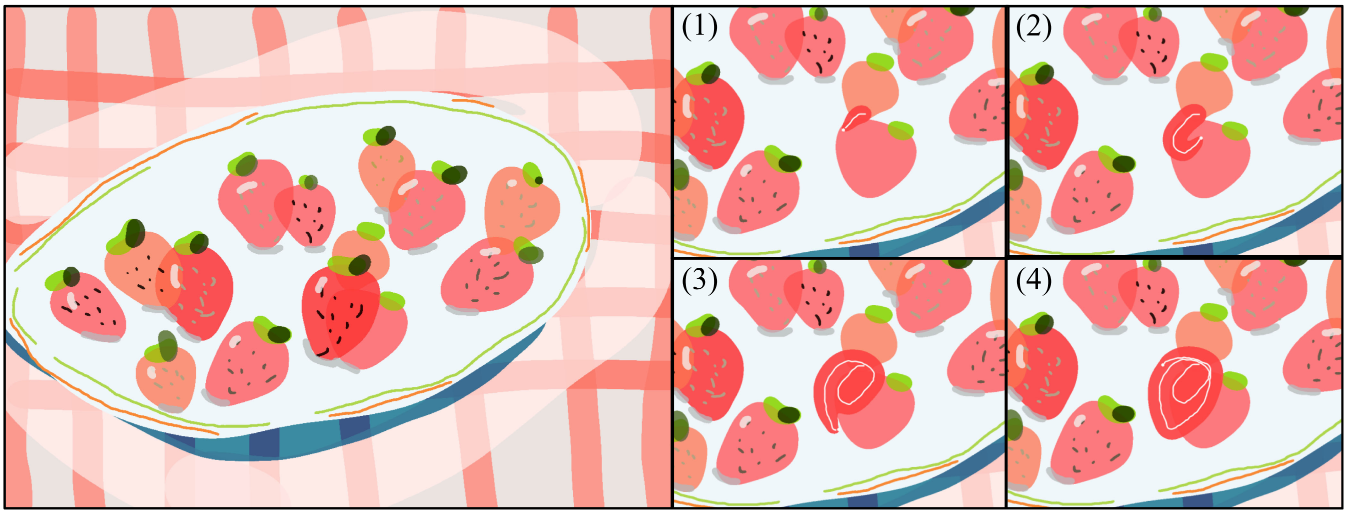


Fig. 7. Flat-style painting created by temporal-evolving TDCs. The four images on the right show the zoomed-in evolution of a single stroke, with the white curve being the TDC being drawn. In our system, the temporal expressiveness of TDC provides satisfying feedback once a stroke start which allows artists to adjust their drawing shape on the fly. Please refer to the supplementary video, available online, to see the complete drawing progress.

contrast, when using DCs, the colors diffuse to infinity until they are prevented by other DCs, which makes them unsuitable for controlling stroke width. On the one hand, our TDC method overcomes the inconvenience in drawing by using strokes as the primitive to generate vector graphics. As shown in Fig. 10, when using DCs, artists need to mark all the boundaries where colors are discontinuous. This behavior contradicts the intuition and drawing habits of human, resulting in a steep learning curve. In contrast, artists can directly draw strokes with TDCs, which is consistent with the real-life painting experience. On the other hand, intersections of DCs usually result in undesired results. To deal with this issue, artists need to break the curves and change colors, which complicates the drawing procedure significantly. TDC strokes, however, are able to intersect and overlap arbitrarily accordant with artist's intention. In addition, while zooming in, the DC using multi-grid solver is necessary to confine the current viewport boundary values. With our random-access solver, the diffusion result is only determined by the pixel coordinates and the parameters of the

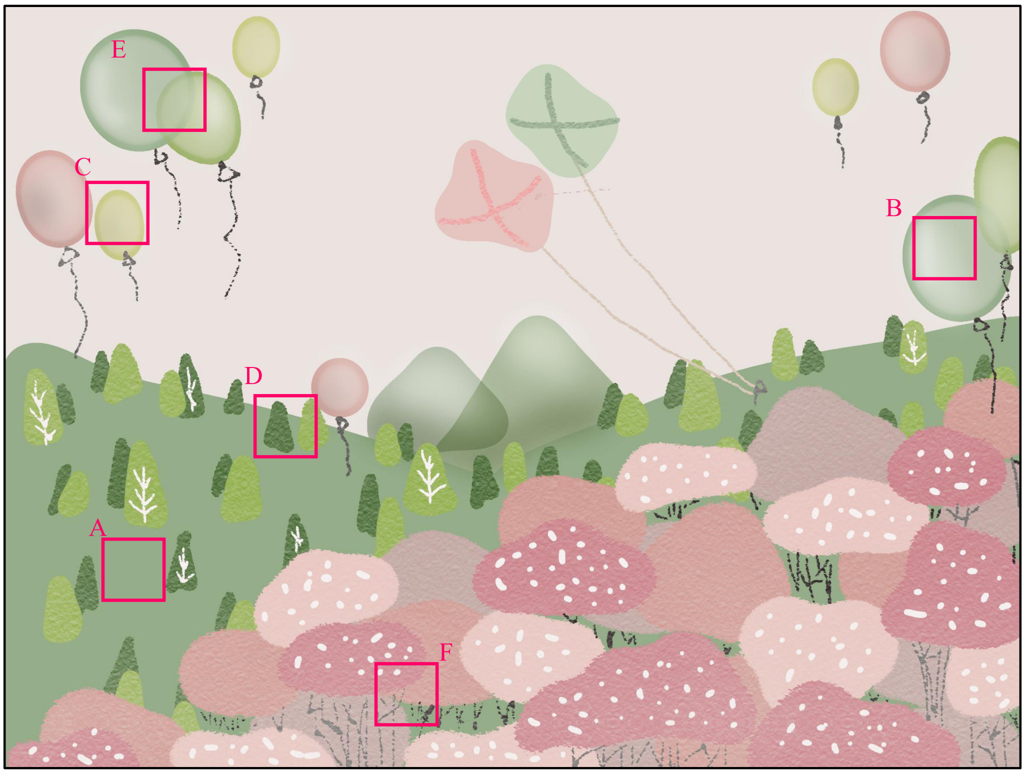


Fig. 8. A TDC painting using a combination of various stroke effects, including uniform coloring (A), feathering (B), edge darkening (C), granulation (D), transparency color mixing (E) and approximate color mixing of real pigments (F).

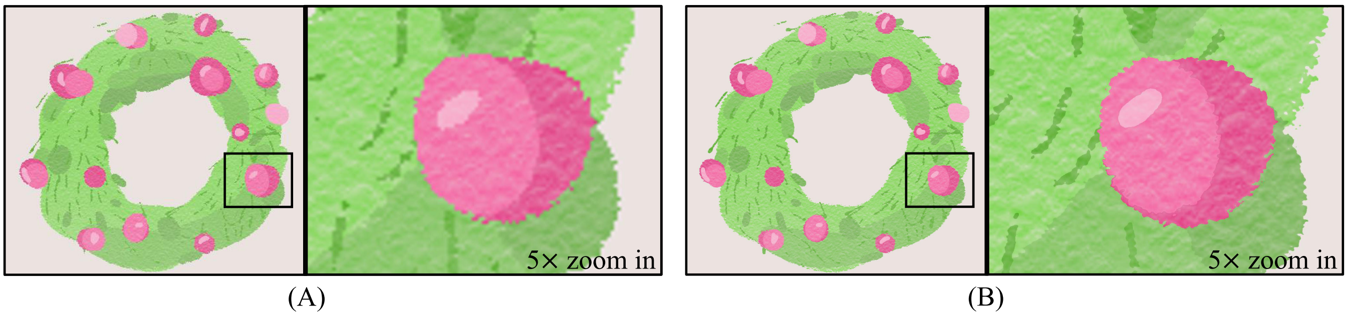


Fig. 9. Comparison between the finite-difference method (A) and TDC (B). Using the same diffusion setup, TDC and the conventional finite-difference method yield similar results. However, the TDC result is resolution independent and remains sharp after zooming in significantly.

TDCs. There is no need to provide values on the viewport boundary.

6.3 User Feedback

To evaluate the usability of our TDC painting system, we invited professional artists to try our system and compare it with the PVG software [16], the latest extension of the DC method. After a 5-minute introduction to the basic functions, artists were able to use our TDC painting tool to draw Figs. 7, 11 A, 11 B, 11 F, and 11 G in half an hour and Fig. 1 in 50 minutes. On the contrary, we had to spend more than 20 minutes introducing the basic theory and the painting process of PVG in case the artists had not been exposed to DC methods before, and the artists still needed 30 minutes to familiarize themselves with the complicated software operations before painting.

Artists gave a high rating to our TDC painting system. According to the feedbacks, the system is very easy to use and stroke effects are more than enough for various styles of artworks. Compared with the commonly used drawing software Adobe Photoshop, our painting tool is much more intuitive and convenient as we recreate the real-life painting experience by modeling strokes directly while in Photoshop user has to manage the layers manually. Meanwhile, our system supports rendering at arbitrary resolution, thus the artwork can be infinitely magnified during the painting session without compromising sharpness or details.

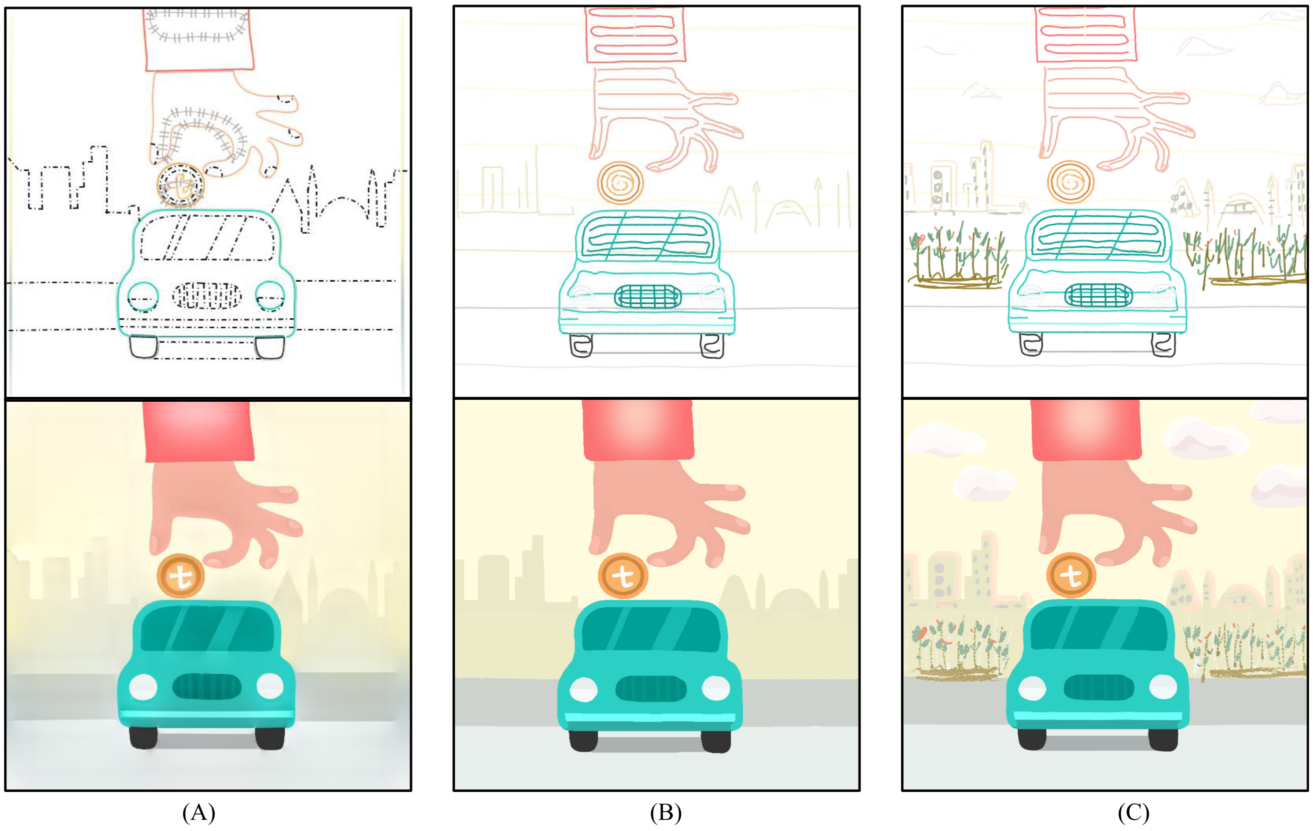


Fig. 10. Comparison of inputs and results among Diffusion Curve (PVG) [16] (A), our TDC method (B) and our TDC result with extra details (C). The DC method uses region boundaries and auxiliary curves to determine the color and its variation. In contrast, TDC models the strokes directly, which is much more consistent with the real-life painting experience using brushes and papers. Moreover, it is perfectly feasible to recreate the DC results with TDC. Nevertheless, some results of our method, including the canvas texture, crossing strokes, color layers, and the temporal evolution, are not trivial to implement with DCs.

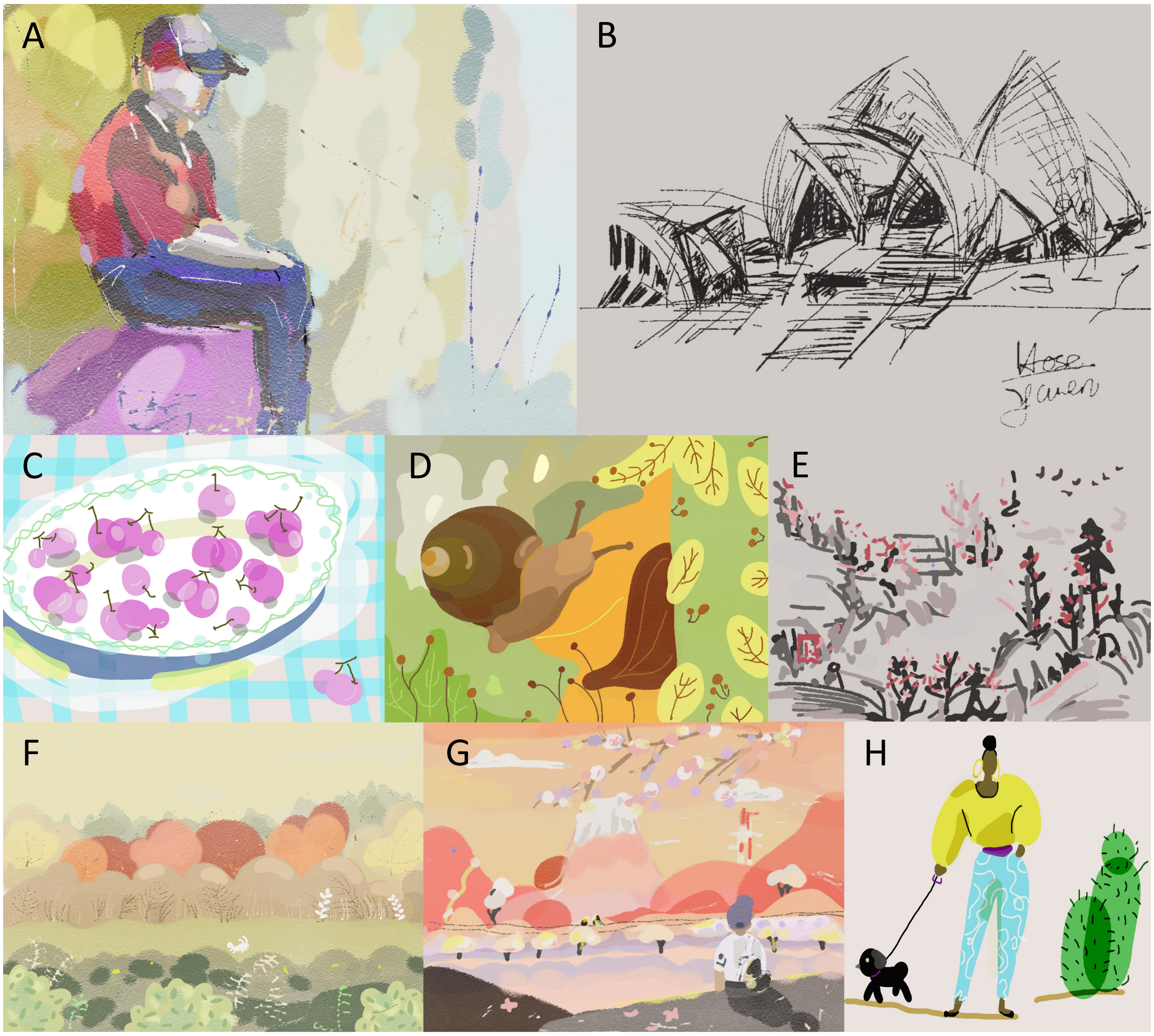


Fig. 11. A gallery of artworks using our painting system.

7 Conclusion and Future Works

To meet the growing demand for digital artwork creation, this paper presents a powerful, efficient and editable vectorized painting system which is capable of producing convincing paintings of various genres. At the core lies the Temporal Diffusion Curve, our novel stroke-based model for modeling evolving vector graphics. By further integrating a procedural stroke processing function into the system, we achieve a digital art creation tool that has quite a few merits over the existing solutions. Being one of the vector solvers, our method is able to render at arbitrary resolutions, holds real-time performance regardless of the scale of the art piece, and provides straightforward editing possibilities on strokes both at runtime and afterward. Meanwhile, our method also has advantages over the existing vector painting tools. Compared with the Diffusion Curve technique, our method not only supports reproducing the evolution of strokes but also is more intuitive to use. Additionally, our system is suitable for multiple styles of artworks as opposed to DiVerdi et al. [8], which only offers watercolor in their vector painting engine.

Our work still has much room for improvement. Although this paper mainly focuses on reproducing the painting session on flat surfaces, many art forms, including the oil painting, are famous for their intriguing 3D texture. We would like to design a more advanced 3D procedural model for this purpose in the future. Additionally, the TDC diffusion has to be isotropic in this paper since the Fourier-Transform-based diffusion solver cannot handle anisotropic diffusion coefficients. Therefore, another future topic could be seeking the fundamental solution of a less constrained TDC formulation. Currently our work is mainly about modeling of strokes, we would like to cover the automated vectorization of images, a hot application of current DC researches, in the future.

Acknowledgments

This project was partially supported by NSFC Grants (61872347, 61532002, 61672077), Special Plan for the Development of

Distinguished Young Scientists of ISCAS (Y8RC535018), USA National Science Foundation IIS-1715985, IIS-1812606, National Key R&D Program of China (2017YFF0106407), CAS Key Research Program of Frontier Sciences (QYZDY-SSW-JSC041), Applied Basic Research Program of Qingdao (161013xx), Capital Health Research and Development of Special (2016-1-4011), Fundamental Research Funds for the Central Universities and Beijing Natural Science Foundation-Haidian Primitive Innovation Joint Fund (L182016). We would like to thank Jiawen Zeng for helping record the voice over of the supplementary video, available online. Y. Li and X. Zhai contributed equally and should be regarded as co-first authors.

REFERENCES

[1] C. J. Curtis, S. E. Anderson, J. E. Seims, K. W. Fleischer, and D. H. Salesin, "Computer-generated watercolor," in *Proc. 24th Annu. Conf. Comput. Graph. Interactive Techn.*, 1997, pp. 421–430.

[2] N. S.-H. Chu and C.-L. Tai, "MoXi: Real-time ink dispersion in absorbent paper," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 504–511, 2005.

[3] N. S. H. Chu, "Expresii watercolor," in *Proc. ACM SIGGRAPH Appy Hour*, 2017, Art. no. 1.

[4] Z. Huang, et al., "A GPU-based method for real-time simulation of eastern painting," in *Proc. 5th Int. Conf. Comput. Graph. Interactive Techn. Australia Southeast Asia*, 2007, pp. 111–118.

[5] Z. Chen, B. Kim, D. Ito, and H. Wang, "Wetbrush: GPU-based 3D painting simulation at the bristle level," *ACM Trans. Graph.*, vol. 34, no. 6, 2015, Art. no. 200.

[6] T. Xia, B. Liao, and Y. Yu, "Patch-based image vectorization with automatic curvilinear feature alignment," *ACM Trans. Graph.*, vol. 28, no. 5, 2009, Art. no. 115.

[7] 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.*, vol. 27, no. 3, 2008, Art. no. 92.

[8] S. Diverdi, A. Krishnaswamy, R. Mech, and D. Ito, "A lightweight, procedural, vector watercolor painting engine," in *Proc. ACM SIGGRAPH Symp. Interactive 3D Graph. Games*, 2012, pp. 63–70.

[9] Z. Liao, H. Hoppe, D. Forsyth, and Y. Yu, "A subdivision-based representation for vector image editing," *IEEE Trans. Vis. Comput. Graph.*, vol. 18, no. 11, pp. 1858–1867, Nov. 2012.

[10] J.-D. Favreau, F. Lafarge, and A. Bousseau, "Fidelity versus simplicity: A global approach to line drawing vectorization," *ACM Trans. Graph.*, vol. 35, no. 4, 2016, Art. no. 120.

[11] J. C. Bowers, J. Leahey, and R. Wang, "A ray tracing approach to diffusion curves," *Comput. Graph. Forum*, vol. 30, no. 4, pp. 1345–1352, 2011.

[12] X. Sun, G. Xie, Y. Dong, S. Lin, W. Xu, W. Wang, X. Tong, and B. Guo, "Diffusion curve textures for resolution independent texture mapping," *ACM Trans. Graph.*, vol. 31, no. 4, 2012, Art. no. 74.

[13] M. Finch, J. Snyder, and H. Hoppe, "Freeform vector graphics with controlled thin-plate splines," *ACM Trans. Graph.*, vol. 30, no. 6, 2011, Art. no. 166.

[14] P. Ilbery, L. Kendall, C. Concolato, and M. McCosker, "Biharmonic diffusion curve images from boundary elements," *ACM Trans. Graph.*, vol. 32, no. 6, 2013, Art. no. 219.

[15] S. Jeschke, "Generalized diffusion curves: An improved vector representation for smooth-shaded images," *Comput. Graph. Forum*, vol. 35, no. 2, pp. 71–79, 2016.

[16] F. Hou, Q. Sun, Z. Fang, Y.-J. Liu, S.-M. Hu, A. Hao, H. Qin, and Y. He, "Poisson vector graphics (PVG)," *IEEE Trans. Vis. Comput. Graph.*, 2018, doi: 10.1109/TVCG.2018.2867478.

[17] T. L. Kunii, G. V. Nosovskij, and T. Hayashi, "A diffusion model for computer animation of diffuse ink painting," in *Proc. Comput. Animation*, 1995, pp. 98–102.

[18] T. Stuyck, F. Da, S. Hadap, and P. Dutré, "Real-time oil painting on mobile hardware," *Comput. Graph. Forum*, vol. 36, no. 8, pp. 69–79, 2017.

[19] P. Kubelka, "New contributions to the optics of intensely light-scattering materials. Part II: Nonhomogeneous layers," *J. Opt. Soc. Amer.*, vol. 44, no. 4, pp. 330–335, 1954.

[20] T. Van Laerhoven and F. Van Reeth, "Real-time simulation of watery paint," *Comput. Animation Virtual Worlds*, vol. 16, no. 3/4, pp. 429–439, 2005.

[21] J. Stam, "Stable fluids," in *Proc. 26th Annu. Conf. Comput. Graph. Interactive Techn.*, 1999, pp. 121–128.

[22] P. Blaškovič, "Rebelle: Real watercolor and acrylic painting software," in *Proc. ACM SIGGRAPH Appy Hour*, 2016, Art. no. 3.

[23] R. Ďurikovič and Z. Páleníková, "Real-time watercolor simulation with fluid vorticity within brush stroke," in *Proc. 21st Int. Conf. Inf. Vis.*, 2017, pp. 158–163.

[24] S. Xu, X. Mei, W. Dong, Z. Zhang, and X. Zhang, "Interactive visual simulation of dynamic ink diffusion effects," in *Proc. 10th Int. Conf. Virtual Reality Continuum Appl. Ind.*, 2011, pp. 109–116.

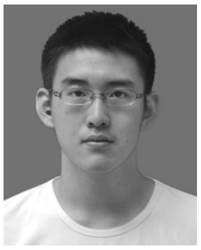
[25] B. Baxter, V. Scheib, M. C. Lin, and D. Manocha, "DAB: Interactive haptic painting with 3D virtual brushes," in *Proc. 28th Annu. Conf. Comput. Graph. Interactive Techn.*, 2001, pp. 461–468.

[26] W. Baxter, J. Wendt, and M. C. Lin, "IMPaSTo: A realistic, interactive model for paint," in *Proc. 3rd Int. Symp. Non-Photorealistic Animation Rendering*, 2004, pp. 45–148.

[27] S. DiVerdi, "A modular framework for digital painting," *IEEE Trans. Vis. Comput. Graph.*, vol. 21, no. 7, pp. 783–793, Jul. 2015.

[28] L. Onural, "Impulse functions over curves and surfaces and their applications to diffraction," *J. Math. Anal. Appl.*, vol. 322, no. 1, pp. 18–27, 2006.

[29] J. Sugita and T. Takahashi, "Paint-like compositing based on RYB color model," in *Proc. ACM SIGGRAPH Posters*, 2015, Art. no. 83.



Yingjia Li received the BS degree in applied physics from Beihang University, in 2017. He is currently working toward the MS degree in the State Key Laboratory of Virtual Reality Technology and Systems, Beihang University. His research interests include data vectorization and physics-based fluid simulation.

Xiao Zhai received the BS degree in computer science and engineering from Beihang University, in 2013. He is currently working toward the PhD degree in the State Key Laboratory of Virtual Reality Technology and Systems, Beihang University. His research interests include physics-based fluid simulation, data-driven fluid animation, and all the relevant topics in computer graphics.

Fei Hou received the PhD degree in computer science from Beihang University, in 2012. He is currently a research associate professor of Institute of Software, Chinese Academy of Sciences. He was a postdoctoral researcher with Beihang University from 2012 to 2014 and a research fellow with the School of Computer Science and Engineering, Nanyang Technological University from 2014 to 2017. His research interests include geometry processing, image-based modeling, data vectorization, medical image processing, etc.

Yawen Liu received the BA degree in industrial design from Jingdezhen Ceramic Institute, in 2015. She is currently a UI designer with Beijing Piesat Information Technology Co., Ltd.

Aimin Hao received the BS, MS, and PhD degrees in computer science from Beihang University. He is a professor with the School of Computer Science and Engineering, Beihang University, and the associate director with the State Key Laboratory of Virtual Reality Technology and Systems. His research interests include virtual reality, computer simulation, computer graphics, geometric modeling, image processing, and computer vision.

Hong Qin received the BS and MS degrees in computer science from Peking University, and the PhD degree in computer science from the University of Toronto. He is a professor of computer science with the Department of Computer Science, Stony Brook University. His research interests include geometric and solid modeling, graphics, physics-based modeling and simulation, computer-aided geometric design, human-computer interaction, visualization, and scientific computing. Currently, he serves as an associate editor of the *Visual Computer*, the *Graphical Models*, and the *Journal of Computer Science and Technology*. He is a senior member of the IEEE and IEEE Computer Society.