MoXi: Real-Time Ink Dispersion in Absorbent Paper

Nelson S.-H. Chu Chiew-Lan Tai

The Hong Kong University of Science and Technology*

Abstract

This paper presents a physically-based method for simulating ink dispersion in absorbent paper for art creation purposes. We devise a novel fluid flow model based on the lattice Boltzmann equation suitable for simulating percolation in disordered media, like paper, in real time. Our model combines the simulations of spontaneous shape evolution and porous media flow under a unified framework. We also couple our physics simulation with simple implicit modeling and image-based methods to render high quality output. We demonstrate the effectiveness of our techniques in a digital paint system and achieve various realistic effects of ink dispersion, including complex flow patterns observed in real artwork, and other special effects.

Keywords: Eastern Ink Painting, Fluid Simulation, Lattice Boltzmann Equation, Physically-Based Modeling

CR Categories: I.3.3 [Computer Graphics] Picture/Image Generation; I.3.4 [Computer Utilities]: Paint Systems

1 Introduction

Eastern brushwork is unique among the world's art traditions. The economy of brush strokes, the use of expressive lines, and the fascinating ink dispersion all contribute to its distinctive appeal. The tools and materials, and how they are manipulated are crucial in producing these features. Progressive painters are always on the quest for new effects or techniques to heighten their expression.

In this paper, we describe various effects of ink dispersion and show how we simulate them computationally. Our goal is to simulate ink spreading in painting paper so that artists can paint in the spontaneous style of Eastern ink painting on a computer interactively (Figure 1). Accomplishing this goal is challenging because of the complexity of the art medium and the real-time requirement. Apart from replicating existing artistic effects, we also hope to further develop the art by creating new digital effects, in particular those that carry the spirit of ink – its fluidity.

Contributions: We develop an ink flow model that can simulate more complex effects than possible with previous work. Our ink flow simulation is based on the method of lattice Boltzmann equation (LBE), a relatively new computational fluid dynamics method [Succi 2001]. This method is easy to implement, highly efficient on parallel processors, and amenable to additional fluid physics modeling. We modify the basic LBE for the physics of ink flow in absorbent paper. The modifications include the incorporation of variable permeability, modulated advection, and boundary roughening. Various ink effects including complex branching patterns, spontaneous shape evolution are achievable under this unified model.

*e-mail: {cpegnel, taicl}@ust.hk

© ACM, 2005. This is the author's version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution. The definitive version was published in ACM Transactions on Graphics, Vol. 24, No. 3, August 2005. http://doi.acm.org/

Figure 1: A sample painting created with our system.

We have implemented our ink flow model and a brush dynamics model [Chu and Tai 2004] in a real-time paint system. Both the GPU and the CPU are utilized for these computation-intensive simulations. We also develop simple implicit modeling and image-based methods to enhance the output quality. These methods are applied on the fly to give immediate user feedback. The resulting system allows a very realistic and intuitive painting experience not attained before in the digital realm.

Organization: After giving some background on Eastern ink painting, we discuss previous work in Section 3. We then describe the LBE fluid simulation approach in Section 4 and present the details of our methods in Sections 5 and 6. Results are presented in Section 7. Finally, Section 8 gives directions for future work.

2 Physical Properties of Ink Painting

In the spontaneous style of Eastern ink painting (or *ink painting* for short), artists utilize flexible brushes to create expressive lines and shapes, and exploit the interplay of ink and water to produce shades and patterns. To provide readers with the necessary background, we next discuss the art materials, the artistic effects related to ink dispersion, and the physical processes involved.

2.1 Eastern Ink and Paper

Eastern ink in its solid form is a mixture of soot and glue [Swider et al. 2003]. The solid is grinded together with water to get black liquid ink. The soot is composed of 10-150 nm carbon particles and dissolves readily in water. Because of their small sizes, the carbon particles seep into paper fibers easily, giving the most prominent dispersion effects in comparison with other (color)

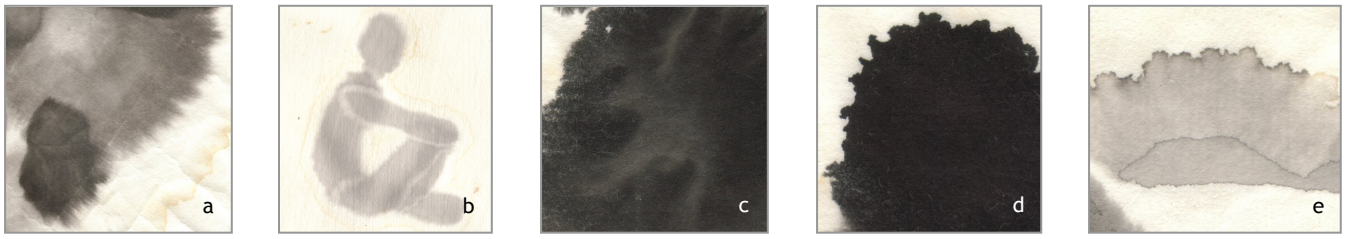


Figure 2: Real ink effects. (a) Feathery pattern. (b) Light fringes. (c) Branching pattern. (d) Boundary roughening. (e) Boundary darkening.

pigments. The glue functions as a dispersing agent to provide a stable liquid suspension of pigment particles and as a binder to fix the pigment during its application on paper. More glue in the ink makes the ink more viscous.

Ink dispersion is closely determined by paper absorbency; it is mainly the imbibition of water that causes the ink to flow through the paper fibers. Very thin and highly absorbent *Xuan* paper is commonly used to vividly pick up the special charm of ink dispersion. To reduce absorbency, the paper can be treated with *alum*.

2.2 Painting Effects

For a faithful simulation, we need to model the essential effects of the art medium, which include:

- Feathery pattern (Figure 2a): When diluted ink is absorbed into the paper, delicate ink streaks that follow the direction of the water flow appear. This feathery quality is caused by the pigments being hindered or used up while the ink spreads.
- Light fringes (Figure 2b): When a stroke is made with diluted ink, water percolates on paper forming a light fringe around the stroke. If another stroke is painted over the first, the wet region of the paper will be less receptive to the new stroke, making the light fringe stands out.
- Branching pattern (Figure 2c): Branching patterns appear because the water flow is obstructed by irregularities in the paper or trapped ink ingredients. Artists may apply glue or other materials unevenly onto the paper to enhance this effect.
- Boundary roughening (Figure 2d): As ink percolates through paper, the wet-dry boundary is pinned at different points by the irregularities in the paper. This is more prominent with concentrated ink, forming toes around the boundary.
- Boundary darkening (Figure 2e): At the pinned boundary of a wet ink mark, water is lost to the surrounding dry fibers via capillary attraction. This water evaporates immediately without expanding the mark, and the attraction induces a migration of pigments that darkens the boundary.

There are painting techniques that artists can use to produce a few of these effects at once, giving a dynamic feel of the art medium. One general technique is *ink-breaking*, in which the artist first paints a stroke with a certain ink concentration and, while it is still wet, paints another stroke over it with a different ink concentration. The two strokes then interact to break the monotony in each of them.

2.3 Physics of Ink Dispersion

In the graphics community, the term 'ink diffusion' has been traditionally used to describe ink dispersion in absorbent paper (e.g. [Guo and Kunii 1991; Lee 2001]). However, in physics, 'diffusion' refers to the random walk of particles towards the less concentrated regions. It is mathematically described by Fick's law. In effect, diffusion simply averages out any spatial difference in the concentration, making the concentration approach a constant with time.

In reality, the actual process of ink dispersion is a complex interplay between paper, water and ink constituents – a process that Fickian diffusion alone cannot describe adequately. Ink dispersion can be viewed as a two-part process: the percolation of water and the movement of pigments within the water. Water flows through the paper fibers in low speed due to water pressure and capillary attractions. Voids in the paper, alum, and the accumulation of ink constituents all may impede the flow, inducing momentum exchange. When faced with obstacles, water branches into streams (Figure 3). The flowing water carries pigments with it. The dispersion of pigments in the water is mainly caused by the spatial difference in water velocity and the hindrance by paper fibers; pigment diffusion only plays a minimal role.

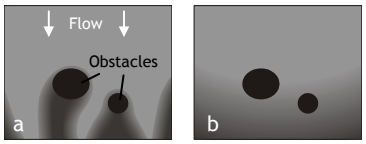


Figure 3: Impeded flow. (a) Modeled with momentum. (b) Just as diffusion or without momentum.

To better appreciate the above phenomena, we video record the ink dispersion process and play it back in fast-forward mode. We refer the readers to our accompanying video for sample footage. Realizing the complex nature of ink flow has motivated us to use a full-fledged fluid simulation to produce realistic ink effects.

3 Previous Work

Eastern Ink Simulation: Guo and Kunii [1991] proposed the first model specifically for Eastern ink dispersion in paper. Assuming that ink consists of pigment particles of different sizes, they modeled ink spreading as a one-dimensional filtering process. Later, Kunii et al. [2001] used partial differential equations (PDE) in an attempt to better describe the phenomenon. Their PDE's for water spreading in paper and pigment movement within water are essentially Fick's law of diffusion. However, these PDE's could only produce blurry images without the interesting flow patterns observed in reality.

Other researchers [Zhang et al. 1999; Yu et al. 2002] used cellular automation to simulate ink dispersion. They essentially treat ink flow as a relaxation process without momentum exchange. Without considering momentum, these methods are only suitable for situations where ink mixes to form blurry images. Yet other research efforts [Lee 2001; Guo and Kunii 2003] processed only the fronts of the spreading strokes to speedup rendering, but this also limits the range of producible effects.

Western Watercolor Simulation: Curtis et al. [1997] did some very successful simulations of Western watercolor paint. Their model simulates the flow of water in two parts: on the paper

surface, and through the paper fibers. They solve the Navier-Stokes (N-S) equations for the on-surface flow and use a cellular automation for the capillary flow through fibers. The latter is used only to produce an effect called *back-run*.

Curtis et al.'s model does not simulate Eastern ink effects well. First, they use a fixed wet-dry boundary for their N-S equation solver, making shape evolution of ink marks impossible. Using their capillary flow for boundary expansion would produce unrealistic spreading because the two flow processes are only loosely coupled. In particular, the water supply from the surface to the capillary layer is not tracked, so a larger puddle of ink on the surface would not result in a larger blob. Second, their model omits medium permeability, which is essential for generating subtle ink patterns.

Laerhoven et al. [2004] recently performed watercolor simulations similar to Curtis et al. [1997] on a grid of processing units. They employed a semi-Lagrangian method [Stam 1999] for faster simulation. A frame rate of 25 frames per second for a canvas size of 256^2 with 6 processors is reported.

Fluid Simulation: In the past decade, graphics researchers have adopted numerical fluid simulation to add realism to computer generated animations (e.g. [Harris 2003]). They typically followed the traditional approach of beginning with a macroscopic description of the fluid, namely, the Navier-Stokes (N-S) equations. Because the N-S equations are hard to solve, various approximate numerical techniques have been proposed [Ferziger and Peric 1999]. One popular method among the graphics researchers is the scheme presented by Stam [1999]. Real-time applications favor this scheme for its ability to take large simulation time step stably. Yet, too large steps result in visible artifacts, hence real-time simulations often have to be done at 256^2 or lower resolutions.

Stam's scheme has also been implemented on the GPU for acceleration [Harris 2003]. However, this scheme has to solve two Poisson equations: one for the pressure (often the performance bottleneck) and one for the viscous diffusion. Solving the Poisson equations involves global operations, making the method not intrinsically suitable for parallel GPU processing.

An increasingly popular alternative approach for fluid simulation is the method of lattice Boltzmann equation (LBE) [Succi 2001; Wei et al. 2004]. Instead of starting with a macroscopic description of the fluid, the LBE models the physics of fluid particles at a mesoscopic level. We base our ink flow simulation on the LBE for its advantages detailed in the next section.

4 Lattice Boltzmann Equation Approach

We surveyed various fluid simulation techniques and chose to simulate water percolation using the LBE method. Advantages of the LBE approach over the traditional N-S equation solvers include [Yu et al. 2003]: (1) it does not involve Poisson equations, (2) all operations are simple and local, and (3) easy to incorporate physics that is hard to describe macroscopically. Items (1) and (2) facilitate very efficient implementation on parallel graphics hardware, while item (3) eases the incorporation of extra physics for new paint effects. Fluid compressibility, the main feature of the LBE that gives it a performance advantage, however, also creates an issue for simulating incompressible flow: the flow speed must stay low to keep the compressibility effect negligible. Fortunately, this is not a problem for us, since ink flows slowly through fibers.

We adapt the basic LBE method to incorporate various features needed for the special case of percolation (described in Section 5.3). The main idea of the LBE approach is to model fluid dynamics using a simplified particle kinetic model. This approach divides the simulation domain into a regular lattice. At each lattice site $\mathbf{x}$ and time t, the fluid particles moving at arbitrary velocities are modeled by a small set of particle distribution functions $f_i(\mathbf{x}, t)$, each of which is the expected number of particles moving along a lattice vector $\mathbf{e}_i$. We use a standard square lattice model for 2D flow, called D2Q9 (Figure 4). In this model, there are nine lattice vectors: $\mathbf{e_1}$, $\mathbf{e_2}$, $\mathbf{e_3}$, $\mathbf{e_4}$ pointing to the four nearest neighbor sites, $\mathbf{e_5}$, $\mathbf{e_6}$, $\mathbf{e_7}$, $\mathbf{e_8}$ pointing along the diagonals to the next nearest sites, and the zero vector $\mathbf{e}_0$ corresponding to the stationary particles. During each time step Δt, two operations are performed at each lattice site: (1) streaming f_i's to the next lattice site along their directions of motion, and (2) colliding f_i's that arrive at the same site. The collision redistributes f_i's toward their equilibrium distribution functions $f_i^{(eq)}$. The two operations of streaming and collision are mathematically described by the LBE:

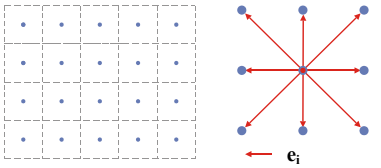


Figure 4: The D2Q9 lattice model. Left: The lattice. Right: The nine lattice vectors of a lattice site.

$$f_i(\mathbf{x}+\mathbf{e}_i\Delta t, t+\Delta t) = (1-\omega) f_i(\mathbf{x},t) + \omega f_i^{(eq)}(\mathbf{x},t) \quad , \qquad (1)$$

where ω is the relaxation parameter. There are numerous variants of the LBE flow model. We employ the 'incompressible' variant of He and Luo [1997], which can minimize the compressible effect inherent in the LBE. In this model, the equilibrium distributions $f_i^{(eq)}$ are

$$f_i^{(eq)} = w_i \left\{ \rho + \rho_0 \left[\frac{3}{c^2}\mathbf{e}_i \cdot \mathbf{u} + \frac{9}{2c^4}(\mathbf{e}_i \cdot \mathbf{u})^2 - \frac{3}{2c^2}\mathbf{u}\cdot\mathbf{u} \right] \right\} , \qquad (2)$$

where c equals $\Delta x/\Delta t$, Δx is the lattice spacing, w_i are constants determined by the lattice geometry, ρ and $\mathbf{u}$ are fluid density and velocity, respectively, and ρ_0 is a predefined average fluid density. For simplicity, we set $\Delta x = \Delta t = c = \rho_0 = 1$ in our simulation. The constants w_i are set as 4/9 for i = 0, 1/9 for i = 1,2,3,4, and 1/36 for i = 5,6,7,8. Fluid density and velocity at each site are given by

$$\rho = \sum_{i=0}^{8} f_i \quad , \qquad \mathbf{u} = \frac{1}{\rho_0}\sum_{i=1}^{8} \mathbf{e}_i f_i \ . \qquad (3), (4)$$

5 Ink Dispersion Simulation

In this section, we describe our ink simulation techniques. Like previous graphics research (e.g. [Curtis et al. 1997]), we do not aim to do strict scientific simulations. We bring in physics only to improve the digital painting experience.

Physicists have proposed various models for the sub-problems of imbibition [Davis and Hocking 2000], porous media flow [Adler 1992], hydrodynamic dispersion [Sun 1996], and boundary roughening [Alava et al. 2004]. Only some of these models are practical for our application owing to modeling efficiency, leaving various holes to be filled in a good solution. Our challenge is therefore to design a unified model that can capture the essence of the physical processes involved so as to produce the desired results, and yet is computationally efficient to work in real time. An additional requirement is that our ink simulation has to work well together with an accurate brush dynamics simulation for intuitive deposition of ink from brush to paper.

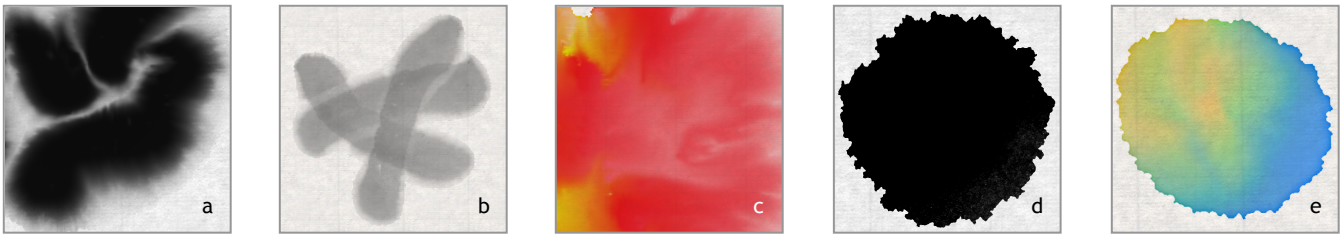


Figure 5: Digital ink effects created using our system that are similar to the real effects illustrated in Figure 2.

5.1 Simulation Overview

Our ink simulation is based on a three-layer paper model. The layers are *surface*, *flow* and *fixture*. Ink is first deposited onto the surface layer. For simplicity, we assume that the ink flows only in the flow layer and not in the surface layer. The ink on the surface gradually seeps into the flow layer, where water percolates through paper and ink constituents are advected by the water. Finally, as the ink dries, ink constituents in the flow layer are moved slowly into the fixture layer.

We develop an LBE-based fluid flow model to simulate the water percolation. We also couple this flow model with a simple method for simulating the movement of ink constituents. For realistic ink deposition from brush to paper, we employ an existing physically-based brush dynamics model [Chu and Tai 2004]. Among the three simulation parts – ink deposition, water percolation, and movement of ink constituents – the simulation of water percolation is the most important for capturing the dynamic nature of the art medium.

The quantity fields used in our ink flow simulation (e.g. water velocity) are discretized spatially on the lattice in the LBE method. These data fields are updated iteratively using the GPU. The CPU is dedicated to the brush simulation. In designing our ink simulation algorithms, we assume a stream-processing model in modern GPU architecture [Mark et al. 2003]. All our discretized data fields are stored as texture maps. In the description of our methods below, we omit the spatial indices to data fields (like ρ) when we refer to the quantity at the current lattice site as in a data streaming model.

5.2 Ink Deposition

The amount of ink deposited onto the surface layer is determined by the brush footprint and the saturation of water in the brush and in the paper. By simulating the variation in paper receptivity caused by saturation, we can produce the effect of a light stroke fringe (Figure 5b). Ink is supplied from the surface to the flow layer according to the capacity of the paper fibers. The surface layer stores excess ink not yet absorbed, which acts as a reservoir.

To generate the brush footprints, we have chosen the brush model of Chu and Tai [2004] for its effectiveness. To model variable paper receptivity, each pixel in the footprint is masked by the value $max(1-\rho/\lambda, m)$, where ρ is the water density (amount of water) at the corresponding lattice site in the flow layer, λ is a receptivity parameter, and m is a base mask value. The amount of water supplied from the surface to the flow layer φ is taken as $clamp(s, 0, \pi-\rho)$, where s denotes the amount of water on the surface, π is the capacity of the paper fibers, and $clamp(x, min, max)$ is a function that returns the clamped value of x against the limits of min and max. In our simulation, we use $\pi = 1$, $0.3 \le \lambda \le 1$, and $m = 0.1$. After φ is determined, s and ρ are updated accordingly. Ink constituents carried by the water that seeps to the flow layer are also moved as described in Section 5.4.1.

5.3 Water Percolation

We apply the LBE to model the water movement in the flow layer. The original LBE describes fluid flows without considering medium permeability and free boundary evolution. To deal with the more complex situation of percolation, we made several modifications to the basic LBE: variable permeability, advection modulation, boundary evolution, and uneven evaporation.

5.3.1 Permeability and Viscosity

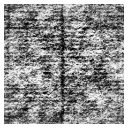
Variable permeability is one key element in our model that makes the creation of interesting flow patterns possible. Recall that the water flow is impeded by various irregularities in the paper. This impediment can be modeled as the permeability of the simulation domain. Like Dardis and McCloskey [1998], we use fractional (rather than Boolean) values to represent the permeability of each lattice site. The permeability is realized by blocking the streaming process; thus we associate each site with a blocking factor κ. Varying κ allows us to model a wide range of media, from those that allow no water movement to those where perturbation in ρ causes ripples.

Specifically, we use the half-way-bounce-back scheme [Succi 2001] during the streaming process to simulate the variable permeability. The blocking is performed as if the link to each neighboring site is partially blocked with a blocking factor $\bar{\kappa}_i$. Rather than having $\bar{\kappa}_i$ equal to the blocking factor of the destination site as in [Dardis and McCloskey 1998], we let $\bar{\kappa}_i$ be the average of the blocking factors of the two linked sites. This guarantees the same amount of blocking in both streaming directions, hence conserving the momentum and density. The streaming step with bounce-back is mathematically described by

$$f_i(\mathbf{x}, t+1) = \bar{\kappa}_i(\mathbf{x})\, f_k(\mathbf{x}, t) + (1-\bar{\kappa}_i(\mathbf{x}))\, f_i(\mathbf{x}-\mathbf{e}_i, t) \quad , \qquad (5)$$

where f_k is the distribution function pointing in the opposite direction of f_i.

Two basic factors that determine the paper permeability are the voids in the fibers and the alum deposited. We store such information in texture maps. Because most types of Eastern painting paper are very thin and semi-transparent, we can obtain the paper thickness patterns simply by scanning the paper against a dark background. We use the normalized thickness images as *grain textures* (a sample is shown on the right), which serve as patterns for the paper grain or voids in the fibers. *Alum textures* that record alum deposition can be generated procedurally (e.g. with some random dots). Precisely, the blocking factor κ at each site is defined as



$$\kappa = k_1 + k_2 \cdot G + k_3 \cdot A + k_4 \cdot g + k_5 \cdot h, \qquad (6)$$

where G and A are values of the grain texture and the alum texture, respectively, k_i are weights that define the blocking, g is the glue concentration in the flow layer, and h is the ink constituent accumulation in the fixture layer.

Another factor that affects the ink flow is the viscosity. Artists sometimes add extra glue to the ink to limit its spread. For a good control, we want the flow to vary from being completely stationary when glue = 1 to normal when glue = 0. In the LBE model, viscosity is given by $(1/\omega - 1/2) / 3$, assuming $\Delta t = c = 1$. Lowering the relaxation parameter ω increases the viscosity. However, we found that modulating ω with glue does not give the desired behavior: the flow cannot remain still when glue = 1 since f_i's from the areas with less glue continue to stream in.

Our solution is to modulate κ with glue, which forms part of our formulation for permeability. The modulation of viscosity with the relaxation parameter ω is still used for adjusting the viscosity globally. In our simulation, ω = 0.5 normally, and reaches 1.5 for simulations of more fluent flow.

5.3.2 Free Boundary and Advection

As water percolates, free boundaries exist between the air (in the fibers) and the water. In fluid dynamics literature, air and water are referred to as two *phases*. Since the effect of air is negligible, we use only a single-phase model, for water. Devising a single-phase free-boundary LBE model is, nevertheless, not straightforward. The LBE model was originally designed for situations where the simulated fluid fills the whole domain, with small local deviation in velocity and density (within 10 to 20 percent) from the mean. If we use the fluid density to represent how much a site is filled (zero density for empty sites), applying the original LBE would cause negative density in certain sites. This is because the advection built into the LBE carries density away even from sites with near-zero density. Existing single-phase models deal with negative densities by simply averaging among neighboring sites [Thuerey 2003] or by extrapolation [Ginzburg and Steiner 2003].

To avoid the unphysical situation of negative density from happening altogether, we modify the basic LBE to reduce the strength of the advection when the density is low. Our rationale is that the advection of water drops in less saturated areas. Specifically, we add a weight ψ to the terms responsible for advection:

$$f_i^{(eq)} = w_i \left\{ \rho + \rho_0 \psi \left[\frac{3}{c^2} \mathbf{e_i} \cdot \mathbf{u} + \frac{9}{2c^4} (\mathbf{e_i} \cdot \mathbf{u})^2 - \frac{3}{2c^2} \mathbf{u} \cdot \mathbf{u} \right] \right\} \quad (7)$$

$$\psi = smoothstep(0, \alpha, \rho) , \quad (8)$$

where α is a user parameter for adjusting this effect. It can be readily checked that the conservation of total water density still holds. The range $0.2 \le \alpha \le 0.5$ works well in our simulation.

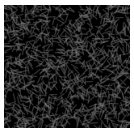
We define a *boundary site* to be a wet lattice site (i.e. $\rho > 0$) with at least one dry site among its eight neighbors. All single-phase free-boundary LBE models require interfacial boundary conditions to determine the particle distribution functions f_i of the boundary sites, including those that are streaming inward from outside the boundary. In our model, this is taken care of by our full-bounce-back pinning mechanism, which is detailed next.

5.3.3 Boundary Pinning and Roughening

Boundary roughening is caused by the spreading front being pinned at different points [Alava et al. 2004]. A front is depinned when there is enough water pressure to overcome the pinning. In reality, the process involves complex interaction between fiber web and ink constituents, and other factors (e.g. evaporation).

We use simple local rules to model pining and depinning, which can be efficiently integrated into the LBE. We say a lattice site is a *pinning site* if it is dry and the water density ρ at each of its eight neighbors is below a threshold. The four nearest neighbors share the same threshold, denoted by σ. We set the threshold for the four next nearest neighbors as $\sqrt{2}\,\sigma$ to compensate for geometric difference of the links. To effect the pinning, we overwrite the blocking factor κ at all the pinning sites by a large value to fully block all their neighbor links.

For the toe patterns found in real ink marks made with concentrated ink (Figure 2d), we introduce one more texture map called *pinning texture* to model the effect of paper disorder. The map is generated by sprinkling light line segments on a dark background (a sample is shown on the right). Modulating the threshold σ with this map gives the effect of easier ink flow at certain locations and directions. We have found this to work well in producing the desired patterns (Figure 5d) when we also model boundary evaporation, which is described in the next subsection. This evaporation avoids the toes from simply taking the shapes of the overlapping line segments in the pinning texture.



We also have σ dynamically depend on the glue concentration in the flow layer, denoted by g, and the ink constituent accumulation in fixture, denoted by h. Precisely, σ is defined as

$$\sigma = q_1 + q_2 \cdot h + q_3 \cdot lerp(G, P, smoothstep(0, \vartheta, g)) , \quad (9)$$

where G and P are values of the grain texture and the pinning texture, respectively, q_i are weights that define the roughening behavior, and ϑ is a parameter controlling the effect of the glue concentration on the appearance of toes. The function $lerp(a, b, f)$ returns the linear interpolation $(1-f)\,a + f\,b$. For smooth spreading fronts (Figure 11b), we can simply set $q_2 = q_3 = 0$.

5.3.4 Evaporation

As a stroke dries, a small extra loss of water occurs at the pinned boundary via capillary attraction. This induces a migration of pigments towards the boundary resulting in a darkened edge.

We model this by having different evaporation rates for the pinned boundary sites and the rest of the wet sites. We do this by reducing the water density ρ at a rate of ε_s as we update ρ using Eq. (3), and reducing those f_i's that bounce back due to pinning at a rate of ε_b during the streaming step. This process is similar to the way Curtis et al. [1997] produced their edge-darkening effect, except that they use a smooth spatial falloff to vary the evaporation rate. Our simulation uses $0 \le \varepsilon_s \le 0.005$ and $\varepsilon_b = 5 \times 10^{-5}$.

5.4 Pigments and Glue Movement

Since pigments and glue are moved likewise, we will only describe how pigments are moved. Concentrations of different pigments are stored in different channels of RGBA textures. We will use a bold symbol (e.g. $\mathbf{p_f}$) to denote the array of concentrations at a site. In our simulation, the movement of pigments can be divided into three parts: *supply*, *advection*, and *fixture*.

5.4.1 Pigment Supply

The ink deposited on the surface layer serves as a reservoir that provides ink supply to the flow layer. After determining the amount of water supplied from the surface, φ (Section 5.2), the pigments in the flow layer is updated according to the ratio of ρ to φ as: $\mathbf{p_f} \leftarrow (\mathbf{p_f}\,\rho + \mathbf{p_s}\varphi) / (\rho + \varphi)$, where $\mathbf{p_f}$ and $\mathbf{p_s}$ denote the pigment concentrations in the flow layer and the surface layer, respectively.

5.4.2 Pigment Advection

The movement of pigments in the flow layer is governed by

hydrodynamic dispersion, in which advection dominates diffusion. In our advection scheme, we first differentiate sites that are already wet from those that are becoming wet in the current time step. For the latter, we derive $\mathbf{p_f}^*(\mathbf{x})$, the pigment concentrations newly advected to site $\mathbf{x}$, as:

$$\mathbf{p_f}^*(\mathbf{x}) = \frac{1}{\rho}\sum_{i=1}^{8} f_i\, \mathbf{p_f}(\mathbf{x}-\mathbf{e}_i) \qquad (10)$$

For the former, $\mathbf{p_f}^*(\mathbf{x})$ is obtained by tracing the velocity backward as in the *method of characteristics* [Sun 1996]: $\mathbf{p_f}^*(\mathbf{x}) = \mathbf{p_f}(\mathbf{y})$, where $\mathbf{y} = \mathbf{x} - \mathbf{u}(\mathbf{x})$. Since our data fields are stored as textures, it is very efficient to use hardware interpolation to sample $\mathbf{p_f}(\mathbf{y})$. But this interpolation would give a wrong $\mathbf{p_f}^*(\mathbf{x})$ when $\mathbf{y}$ is within half a lattice spacing from the boundary. The reason is that $\mathbf{p_f}(\mathbf{y})$ would be the result of a cross-boundary interpolation, but any $\mathbf{p_f}$ outside the boundary should never contribute to $\mathbf{p_f}^*(\mathbf{x})$. Our solution is simply to use $\mathbf{p_f}^*(\mathbf{x}) = \mathbf{p_f}(\mathbf{x})$ when this happens.

Finally, we update $\mathbf{p_f}(\mathbf{x})$ with an interpolation between $\mathbf{p_f}^*(\mathbf{x})$ and the old $\mathbf{p_f}(\mathbf{x})$ to model the hindrance of pigments by the irregularities in paper according to the pseudo-code:

```
Proc SIMULATEHINDRANCE(p_f, p_f*, γ, ς, u) {
  γ* ← lerp(1, γ, smoothstep(0, ς, |u|))
  p_f ← lerp(p_f*, p_f, γ*)
}
```

Here γ is the hindrance rate, ς is a parameter for blocking the advection when the flow speed is low.

5.4.3 Pigment Fixture

Pigments in the flow layer are gradually transferred to the fixture layer as the ink dries. Given the fact that real dried ink mark cannot be washed away by water, we assume that the transfer is a one-way process. For realistic pigment behaviors, we want to satisfy the following conditions: (1) the transfer rate is higher when the strokes become drier, (2) the transfer rate is higher when the glue is more concentrated, and (3) all pigments are settled when a stroke dries.

We devise a simple pigment fixture algorithm that satisfies the above conditions. It updates the pigment concentrations in the flow layer and the fixture layer, denoted by $\mathbf{p_f}$ and $\mathbf{p_x}$, respectively, according to several parameters: ρ and ρ', the water density in the flow layer in the current frame and the last frame, respectively; g, the glue concentration in the flow layer; η, a base fixture rate; μ and ξ, parameters for modulating the fixture rate by dryness and glue, respectively. The algorithm expressed in pseudo-code reads:

```
Proc SIMULATEFIXTURE(p_f, p_x, ρ, ρ′, g, η, μ, ξ) {
  wLoss ← max(ρ′ − ρ, 0)
  if wLoss > 0 then
    FixFactor ← wLoss / ρ′
  else
    FixFactor ← 0
  μ* ← clamp(μ + ξ × g, 0, 1)
  FixFactor ← max(FixFactor × (1 − smoothstep(0, μ*, ρ)), η)
  p_x ← p_x + FixFactor × p_f
  p_f ← p_f − FixFactor × p_f
}
```

6 High Quality Rendering

It is important that a paint system can render high resolution output since artworks are often produced at print resolution. However, doing a physics simulation down to the scale of tiny paper fibers for fine details would be prohibitive. Besides, excessive simulation is undesirable if similar results can be obtained at a much lower cost. Therefore, we enhance the resolution of our ink simulation output with implicit modeling and image-based methods. We choose to keep our methods simple and fast so as to give immediate user feedback; this contributes towards a better painting experience.

6.1 Boundary Trimming

To generate a higher resolution output, we first take the original simulation output I and produce a larger version I' by hardware interpolation. The interpolation blurs away sharp features and the boundaries look blocky when enlarged (Figure 7a). To keep the boundaries sharp and avoid blockiness, we trim away pixels that are interpolated between stained and unstained sites in I'. Conceptually, the trimming scheme is similar to iso-surface methods [Whitaker 2002].

To trim the ink marks in I' on the fly, we must first decide when and where to trim. Basically, the life of an ink mark has three stages: (1) expanding as the paper imbibes it, (2) having its boundary pinned, and (3) having its wet area shrinking as the water evaporates. Clearly, the trimming should only be done during the second stage. When this condition is detected (by keeping track of some site variables), trimming is performed according to an implicit curve defined by a scalar function ϕ derived from the water density field. Each lattice site is associated with a variable ρ_τ, which is updated at each time step to $max(\rho_\tau, \rho)$ if the site is wet, and to a small negative value if it is dry. The function ϕ interpolates the values of ρ_τ in different sites. We use $max(\rho_\tau, \rho)$ instead of ρ so that ρ_τ would not change as the mark dries (during its second life stage). We first shrink the implicit shape a bit by overwriting ρ_τ at the sites just inside the boundary with zero, and then trim the pixels of I' where $\phi < 0$ to give a sharp and natural-looking boundary.

For fine boundary roughening that is not captured by the coarser resolution of I, we also add a roughening term when deriving ϕ:

$$\phi = \rho_\tau + r_{scale}(r + r_{bias}) \ , \qquad (11)$$

where r_{scale} and r_{bias} are parameters for adjusting the effect, and r is the value of a *rougher texture,* which is simply a blurred image of some random spots. To generate smooth boundaries, we use $r_{scale} = 0$ and replace the term ρ_τ in Eq. (11) by the value sampled at the same point from a low-pass-filtered $[\rho_\tau]$, with $[\rho_\tau]$ denoting the texture storing the data field ρ_τ. Figure 7b shows a trimming result.

In practice, the texture storing the pigment concentrations in fixture, denoted $[\mathbf{p_x}]$, is used as the input for generating I'. Ideally, I' should be updated at every time step to reflect the changes in $[\mathbf{p_x}]$. However, for better performance and to avoid a hardware precision issue (described in the supplementary material), we only

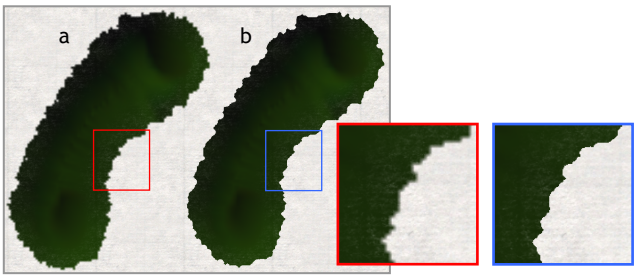


Figure 7: Boundary trimming. (a) Before trimming. (b) After trimming.

update I' every 200-400 frames. For rendering the painting scene, we also apply boundary trimming on $[p_f]$ and $[p_x]$ (every frame) so that the user sees nice boundaries while painting.

6.2 Textural Details

When ink percolates paper, tiny hairs of the paper fibers become apparent in regions not fully soaked by the ink (as in the left part of Figure 2c, and the lower left corner of Figure 2d). We render this effect by modulating the pigment concentrations in regions that are partially soaked with a *hair texture*, which is prepared from real images of stained paper. The modulation also depends on paper thickness to give venetian blind patterns (Figure 2c). Sample results are shown in Figures 5d and 7.

7 Implementation and Results

We have implemented a paint system, named *MoXi* (pronounced *Maw See*), based on our ink dispersion model using OpenGL and the Cg shading language [Mark et al. 2003]. The simulation operations are implemented in fragment programs running on the GPU. The simulated quantities are stored as textures, which are updated with the result of the fragment programs. We also incorporated a brush dynamics simulation [Chu and Tai 2004], which runs on the CPU, into our system to allow realistic brush stroke generation. For manipulating the virtual brush, our system supports graphics tablet (Figure 8), which is popular among digital artists.

GPU Processing: Currently, we limit our dispersion simulation to three pigments (colors) at a time so that we can fit pigments and glue concentrations into one RGBA texture. A large gamut of colors, however, is still available by mixing the three pigments (e.g. cyan, magenta, yellow). During each time step, we perform six texture updates for the LBE flow simulation, and another six texture updates for moving the pigments and glue. The boundary trimming requires one more texture update for I'. Thanks to the simplicity of the method, the six fragment programs for the LBE operations have an average assembly instruction count of only 29.8. Further details are presented in the supplementary material.

Performance: We conducted performance test on a machine with an Althlon XP 2600+ CPU and a GeForce 6800Ultra GPU with 256MB video memory. We denote the ratio of the dimensions of I to those of I' by *d-scale*. For real-time generation of output at 1536^2 with a simulation resolution of 512^2 and a *d-scale* of 3, we are able to have an overall system frame rate (including the high-quality rendering techniques described in Section 6) of 44 frames per second. More performance data are shown in Table 1.

Results: Figure 1 was painted by first drawing a circle with glue in the center to create a 'wall'. Then, we painted a few strokes with a brush loaded with an ink gradient inside the wall. Notice the unevenness of the wall allowed some of the ink to seep through near the top part of the wall. We then painted the rest of the painting with different ink concentrations. Figure 9 shows real branching patterns made with *ink catalyst* (a relatively new art material), and some digital ink marks mimicking that effect made with our system. We model catalyst as a material that blocks the ink flow. The digital marks were made by first spraying some catalyst onto the paper, and then adding red and yellow inks and letting them flow. In Figure 10a, glue was used to hold some parts of the painting stationary. All the above figures were painted with a simulation resolution of 512^2 and a *d-scale* of 3.

Special Effects: We have experimented with several special effects or controls that are hard or impossible to get in real life:

- Rewetting and moisture control: We provide the option to transfer pigments in the fixture layer back to the flow layer, so that artists can rework dried ink marks. We also add the controls of moisture redoubling (Figure 11a) and quick drying. Repeated addition and reduction of moisture can also create certain ring patterns.
- Physics tinkering: By switching the paper parameters suddenly during painting, we are able to produce some interesting patterns like Figure 11b. By making certain (unphysical) modifications to our pigment advection scheme, we are also able to obtain nice flow patterns like Figure 11c.
- Splash and spray: We also incorporated simple physics for ink drops (Figure 10c) so that artists can splash or spray art materials as in real life or have the ink drops emitting in some exotic patterns.

Additional controls we implemented that are not possible in real-world painting include sucking and pushing of ink with a virtual brush. We refer the reader to our accompanying video for demonstrations.

8 Future Work

Resolution: Since the printing of large format digital artwork requires much higher resolution outputs, we are interested in further increasing the output resolution. Possible directions include texture synthesis techniques (e.g. [Hertzmann et al. 2001]) for fine details, and the use of Lagrangian particles [Sun 1996] in the simulation to resolve very fine streaks.

Rendering: Although *color glazing* is not generally used in ink paintings, it would be interesting to incorporate the Kubelka-

Sim. Resol.	*d-scale*	I' Update Interval (fr.)	fr./sec.
256^2	1	-	70
512^2	1	-	48
512^2	3	300	44
512^2	3	1	32
512^2	4	300	42

Table 1: Performance of our system. Frame rates are measured when there is no concurrent brush deformation.

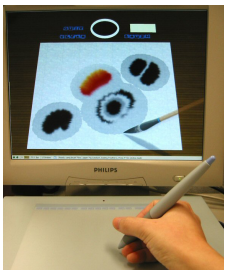
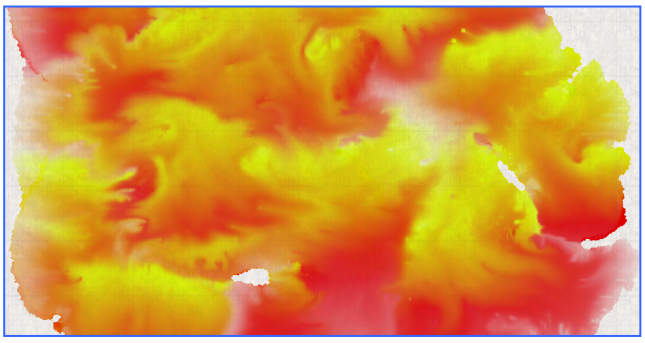


Figure 8: System setup.



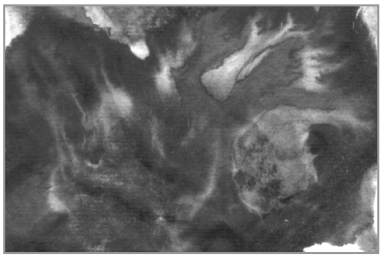


Figure 9: Severe branching patterns. Left: Real ink marks made with ink catalyst. Top: Marks of similar effect made with our system. No manual pushing or sucking of ink with brush was used.

Munk model for simulating optical blending [Curtis et al. 1997]. The rendering can also benefit from the use of Wang tiles [Wei 2004] to give non-repeating paper textures.

New Effects: Physicists have explored various fluid behaviors, including viscous fingering and surfactant physics, using the LBE. We are interested in exploiting LBE's advantage of easy addition of new fluid physics for new painting effects. In particular, simulated ink on water would allow digital development of floating-ink prints (see e.g. www.suminagashi.com).

Acknowledgements

We thank Xavier Granier for his help in improving the clarity of this paper and Michael S. Brown for providing the video voice-over. This work was supported by a grant from the Research Grant Council of Hong Kong, China.

References

ADLER, P., 1992. *Porous Media: geometry and transports*. Butterworth-Heinemann.

ALAVA, M., DUBÉ, M., AND ROST, M. 2004. Imbibition in disordered media, *Advances in Physics 53*, 83-175.

CHU, N. S., AND TAI, C.-L., 2004. Real-Time Painting with an Expressive Virtual Chinese Brush, *IEEE Computer Graphics and Applications 24*, 5, 76-85.

CURTIS, C., ANDERSON, S., SEIMS, J., FLEISCHER, K., AND SALESIN, D., 1997. Computer-Generated Watercolor, In *Proceedings of ACM SIGGRAPH 97*, ACM Press, 421-430.

DARDIS, O., AND MCCLOSKEY, J., 1998. Lattice Boltzmann scheme with real numbered solid density for the simulation of flow in porous media. *Phys. Rev. E: Lett. 57* (14), 4834–4837.

DAVIS, S. H, AND HOCKING, L. M., 2000. Spreading and imbibition of viscous liquid on a porous base. II. *Physics of Fluids 12*, 7, 1646-1655.

FERZIGER, J.H., AND PERIC , M., 1999. *Computational methods for fluid dynamics*. Springer-Verlag.

GINZBURG, I., AND STEINER K. 2003. Lattice Boltzmann model for free surface flow and its application to filling process in casting, *J. Comput. Phys. 185*, 61–99.

GUO, Q., AND KUNII, T. L., 1991. Modeling the diffuse painting of sumie, *IFIP Modeling in Computer Graphics,* 329-338.

GUO, Q., AND KUNII, T. L., 2003. Nijimi rendering algorithm for creating quality black ink paintings, in *Proceedings of Computer Graphics International 2003,* 152-159.

HARRIS, M. J., 2003. *Real-Time Cloud Simulation and Rendering*. Technical Report #TR03-040, University of North Carolina.

HE, X, AND LUO, L.-S., 1997. Lattice Boltzmann model for the incompressible Navier-Stokes equation, *J. Stat. Phys. 88*, 927-944.

HERTZMANN, A. , JACOBS, C. , OLIVER, N. , CURLESS, B., AND SALESIN, D., 2001. Image Analogies. In *Proceedings of SIGGRAPH 2001 Conference,* ACM Press, 327-340.

KUNII, T. L., NOSOVSKIJ, G. V., AND VECHERININ, V. L., 2001. Two-dimensional diffusion model for diffuse ink painting. *Int. J. of Shape Modeling, 7*, 1, 45-58.

LAERHOVEN, T., LIESENBORGS, J., AND REETH, F., 2004. Real-Time Watercolor Painting on a Distributed Paper Model. In *Proceedings of Computer Graphics International 2004*, 640-643.

LEE, J., 2001. Diffusion Rendering of Black Ink Paintings Using New Paper and Ink Models, *Computers and Graphics 25,* 2, 295-308.

MARK, W. R., GLANVILLE, R. S., AKELEY K., AND KILGARD, M. J., 2003. Cg: a system for programming graphics hardware in a C-like language. *ACM Transactions on Graphics,* 22, 3, 896-907.

STAM, J. 1999. Stable Fluids, In *Proceedings of ACM SIGGRAPH 99*, ACM Press, 121-128.

SUCCI, S., 2001. *The lattice Boltzmann equation for fluid dynamics and beyond.* Oxford University Press.

SUN, N.-Z., 1996. *Mathematical modeling of groundwater pollution.* Springer-Verlag.

SWIDER, J. R, HACKLEY, V. A., AND WINTER, J., 2003. Characterization of Chinese Ink in size and surface, *J. of Cultural Heritage 4*, 175-186.

THUEREY, N., 2003. *A single-phase free-surface lattice-Boltzmann method.* Master-thesis, Erlangen Germany.

WEI, L.-Y., 2004. Tile-Based Texture Mapping on Graphics Hardware, *SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware 2004*.

WEI, X., ZHAO, Y., FAN, Z., LI, W., QIU, F., YOAKUM-STOVER, S., AND KAUFMAN, A., 2004. Lattice-Based Flow Field Modeling. *IEEE Transactions on Visualization and Computer Graphics*, 10, 6, 719-729.

WHITAKER, R. T., 2002. *Isosurfaces and Level-Set Surface Models.* Technical report, School of Computing, University of Utah.

YU, D., MEI, R., LUO, L.-S., AND SHYY, W., 2003. Viscous flow computations with the method of lattice Boltzmann equation, *Progress in Aerospace Science 39*, 329-367.

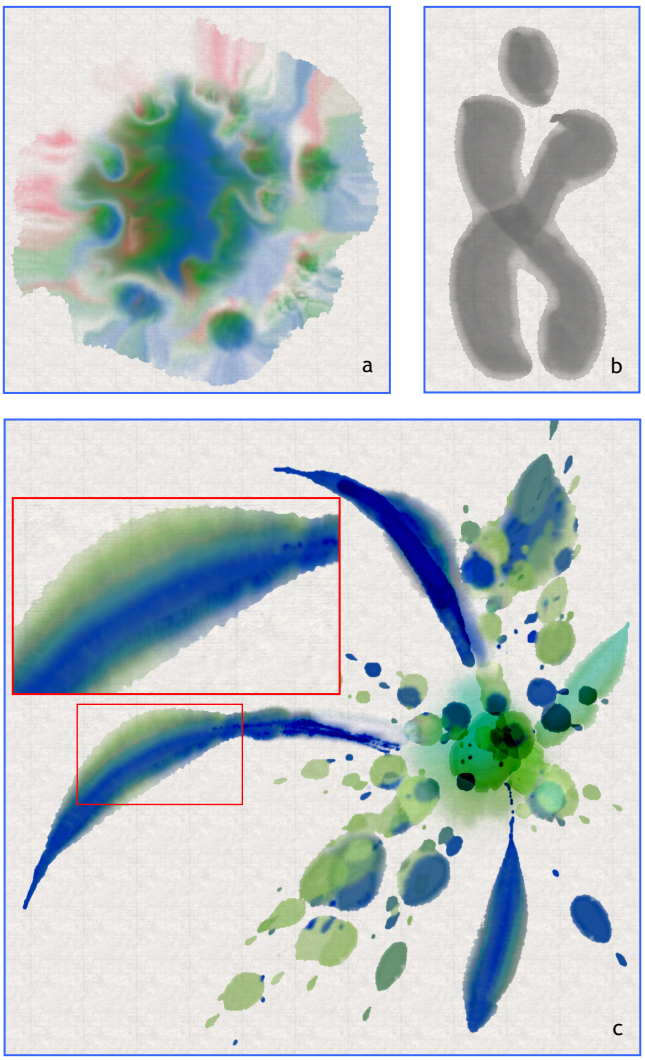


Figure 10: Sample paintings created with our system.

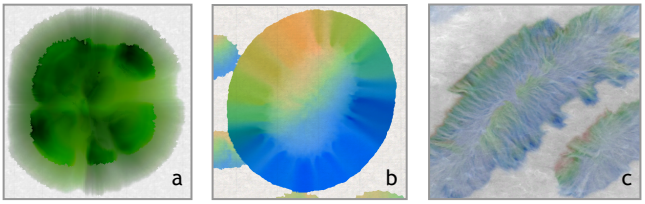


Figure 11: Sample special ink effects. (a) moisture redoubling. (b) shaft-like patterns. (c) stream-like patterns.