Computer-Generated Watercolor

Cassidy J. Curtis *Sean E. Anderson*[*] *Joshua E. Seims* *Kurt W. Fleischer*[†] *David H. Salesin*

University of Washington [*]Stanford University [†]Pixar Animation Studios

Abstract

This paper describes the various artistic effects of watercolor and shows how they can be simulated automatically. Our watercolor model is based on an ordered set of translucent glazes, which are created independently using a shallow-water fluid simulation. We use a Kubelka-Munk compositing model for simulating the optical effect of the superimposed glazes. We demonstrate how computer-generated watercolor can be used in three different applications: as part of an interactive watercolor paint system, as a method for automatic image "watercolorization," and as a mechanism for non-photorealistic rendering of three-dimensional scenes.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation; I.6.3 [Simulation and Modeling]: Applications.

Additional Keywords: Fluid simulation, glazing, illustration, Kubelka-Munk, non-photorealistic rendering, optical compositing, painting, pigments, watercolor.

1 Introduction

Watercolor is like no other medium. It exhibits beautiful textures and patterns that reveal the motion of water across paper, much as the shape of a valley suggests the flow of streams. Its vibrant colors and spontaneous shapes give it a distinctive charm. And it can be applied in delicate layers to achieve subtle variations in color, giving even the most mundane subject a transparent, luminous quality.

In this paper, we characterize the most important effects of watercolor and show how they can be simulated automatically. We then demonstrate how computer-generated watercolor can be used in three different applications: as part of an interactive watercolor paint system (Figure 7), as a method for automatic image "watercolorization" (Figure 10), and as a mechanism for non-photorealistic rendering of three-dimensional scenes (Figures 14 and 13).

The watercolor simulator we describe is empirically-based: while it does incorporate some physically-based models, it is by no means a strict physical simulation. Rather, our emphasis in this work has been to re-create, synthetically, the most salient artistic features of watercolor in a way that is both predictable and controllable.

1.1 Related work

This paper follows in a long line of important work on simulating artists' traditional media and tools. Most directly related is Small's groundbreaking work on simulating watercolor on a Connection Machine [34]. Like Small, we use a cellular automaton to simulate fluid flow and pigment dispersion. However, in order to achieve even more realistic watercolor effects, we employ a more sophisticated paper model, a more complex shallow water simulation, and a more faithful rendering and optical compositing of pigmented layers based on the Kubelka-Munk model. The combination of these improvements enables our system to create many additional watercolor effects such as edge-darkening, granulation, backruns, separation of pigments, and glazing, as described in Section 2. These effects produce a look that is closer to that of real watercolors, and captures better the feeling of transparency and luminosity that is characteristic of the medium.

In the commercial realm, certain watercolor effects are provided by products such as Fractal Design Painter, although this product does not appear to give as realistic watercolor results as the simulation we describe. In other related work, Guo and Kunii have explored the effects of "Sumie" painting [13], and Guo has continued to apply that work to calligraphy [12]. Their model of ink diffusion through paper resembles, to some extent, both Small's and our own water simulation techniques.

Other research work on modeling thick, shiny paint [2] and the effects of bristle brushes on painting and calligraphy [30, 36] also bears relation to the work described here, in providing a plausible simulation of traditional artists' tools.[1] The work described here also continues in a growing line of non-photorealistic rendering research [5, 6, 9, 16, 22, 23, 26, 33, 39, 40], and it builds on previous work on animating the fluid dynamics of water [1, 10, 19] and the effects of water flow on the appearance of surfaces [7, 8, 28].

1.2 Overview

The next section describes the physical nature of the watercolor medium, and then goes on to survey some of its most important characteristics from an artist's standpoint. Section 3 discusses how these key characteristics can be created synthetically. Section 4 describes our physical simulation of the dispersion of water and pigment in detail. Section 5 discusses how the resulting distributions of pigment are rendered. Section 6 presents three different applications in which we have used our watercolor simulation and provides examples of the results produced. Finally, Section 7 discusses some ideas for future research.

2 Properties of watercolor

For centuries, ground pigments have been combined with water-soluble binding materials and used in painting. The earliest uses of watercolor were as thin colored washes painstakingly applied to detailed pen-and-ink or pencil illustrations. The modern tradition of watercolor, however, dates back to the latter half of the eighteenth century, when artists such as J. M. W. Turner (1775–1851), John Constable (1776–1837), and David Cox (1783–1859) began to experiment with new techniques such as wiping and scratching out, and with the immediacy and spontaneity of the medium [35].

To simulate watercolor effectively, it is important to study not only the physical properties of the medium, but also the characteristic phenomena that make watercolor so popular to artists. A simulation is successful only if it can achieve many of the same effects. In the

[1]This approach is essentially the same as the "minimal simulation" approach taken by Cockshott *et al.* [2], whose "wet & sticky" paint model is designed to behave like the real medium as far as the artist can tell, without necessarily having a real physical basis.

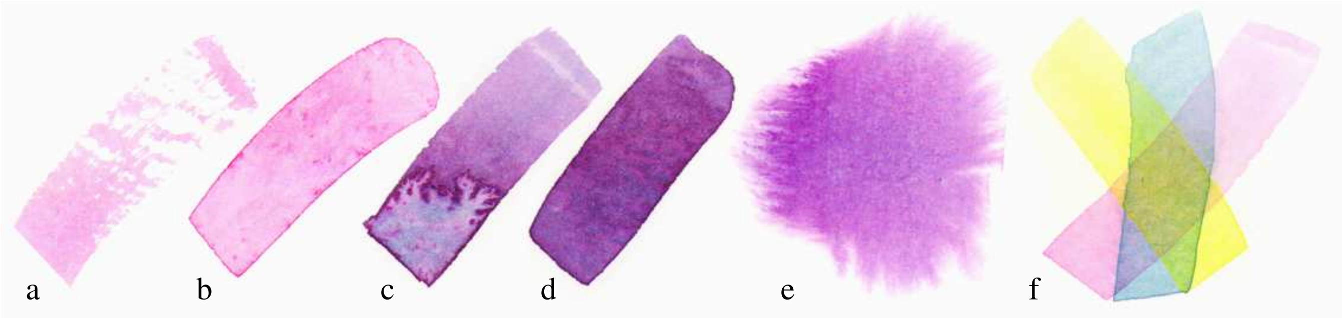

Figure 1 Real watercolor effects: drybrush (a), edge darkening (b), backruns (c), granulation (d), flow effects (e), and glazing (f).

rest of this section, we therefore discuss the physical nature of watercolor, and then survey some of the most important characteristics of watercolor from an artist's standpoint.

2.1 Watercolor materials

Watercolor images are created by the application of watercolor paint to paper. *Watercolor paint* (also called, simply, *watercolor*) is a suspension of pigment particles in a solution of water, binder, and surfactant [17, 25, 35]. The ingredients of watercolor are described in more detail below.

Watercolor paper is typically not made from wood pulp, but instead from linen or cotton rags pounded into small fibers. The paper itself is mostly air, laced with a microscopic web of these tangled fibers. Such a substance is obviously extremely absorbent to liquids, and so the paper is impregnated with *sizing* so that liquid paints may be used on it without immediately soaking in and diffusing. Sizing is usually made of cellulose. It forms a barrier that slows the rate of water absorption and diffusion. For most watercolor papers, sizing is applied sparingly and just coats the fibers and fills some of the pores, leaving the paper surface still rough.

A *pigment* is a solid material in the form of small, separate particles. Watercolor pigments are typically ground in a milling process into a powder made of grains ranging from about 0.05 to 0.5 microns. Pigments can penetrate into the paper, but once in the paper they tend not to migrate far. Pigments vary in *density*, with lighter pigments tending to stay suspended in water longer than heavier ones, and thus spreading further across paper. *Staining power*, an estimate of the pigment's tendency to adhere to or coat paper fibers, also varies between pigments. Certain pigments exhibit *granulation*, in which particles settle into the hollows of rough paper. Others exhibit *flocculation*, in which particles are drawn together into clumps usually by electrical effects. (Since flocculation is similar in appearance to granulation, we discuss the modeling of granulation only in this paper.)

The two remaining ingredients, *binder* and *surfactant*, both play important roles. The binder enables the pigment to adhere to the paper (known as "adsorption of the pigment by the paper"). The surfactant allows water to soak into sized paper. A proper proportion of pigment, binder, and surfactant is necessary in order for the paint to exhibit the qualities desired by artists. (However, as these proportions are controlled by the paint manufacturer and not the artist, we have not made them part of our model.)

The final appearance of watercolor derives from the interaction between the movements of various pigments in a flowing medium, the adsorption of these pigments by the paper, the absorption of water into the paper, and the eventual evaporation of the water medium. While these interactions are quite complex in nature, they can be used by a skilled artist to achieve a wide variety of effects, as described in the next section.

2.2 Watercolor effects

Watercolor can be used in many different ways. To begin with, there are two basic brushing techniques. In *wet-in-wet* painting, a brush loaded with watercolor paint is applied to paper that is already saturated with water, allowing the paint to spread freely. When the brush is applied to dry paper, it is known as *wet-on-dry* painting. These techniques give rise to a number of standard effects that can be reliably employed by the watercolor expert, including:

- *Dry-brush* effects (Figure 1a): A brush that is almost dry, applied at the proper grazing angle, will apply paint only to the raised areas of the rough paper, leaving a stroke with irregular gaps and ragged edges.
- *Edge darkening* (Figure 1b): In a wet-on-dry brushtroke, the sizing in the paper, coupled with the surface tension of water, does not allow the brushstroke to spread. Instead, in a gradual process, the pigment migrates from the interior of the painted region towards its edges as the paint begins to dry, leaving a dark deposit at the edge. This key effect is one that watercolor artists rely upon and that paint manufacturers take pains to ensure in their watercolor paint formulations [17].
- Intentional *backruns* (Figure 1c): When a puddle of water spreads back into a damp region of paint, as often happens when a wash dries unevenly, the water tends to push pigment along as it spreads, resulting in complex, branching shapes with severely darkened edges.
- *Granulation* and *separation* of pigments (Figure 1d): Granulation of pigments yields a kind of grainy texture that emphasizes the peaks and valleys in the paper. Granulation varies from pigment to pigment, and is strongest when the paper is very wet. Separation refers to a splitting of colors that occurs when denser pigments settle earlier than lighter ones.
- *Flow patterns* (Figure 1e): In wet-in-wet painting, the wet surface allows the brushstrokes to spread freely, resulting in soft, feathery shapes with delicate striations that follow the direction of water flow.

One other very important technique in watercolor is the process of *color glazing* (Figure 1f). Glazing is the process of adding very thin, pale layers, or *washes*, of watercolor, one over another, to achieve a very clear and even effect. Each layer of watercolor is added after the previous layer has dried. More expensive watercolor paints are specially formulated to have a low *resolubility*, which not only allows thin uniform washes to be overlaid, but in fact allows any type of brushing technique to be employed over a dried wash (including dry-brush and wet-on-wet) without disturbing the underlying layers.

Glazing is different from ordinary painting in that the different pigments are not mixed physically, but optically—in their superposition on the paper. Glazes yield a pleasing effect that is often described as "luminous," or as "glowing from within" [4, 32]. We suspect that this subjective impression arises from the edge-darkening effect. The impression is intensified with multiple super-

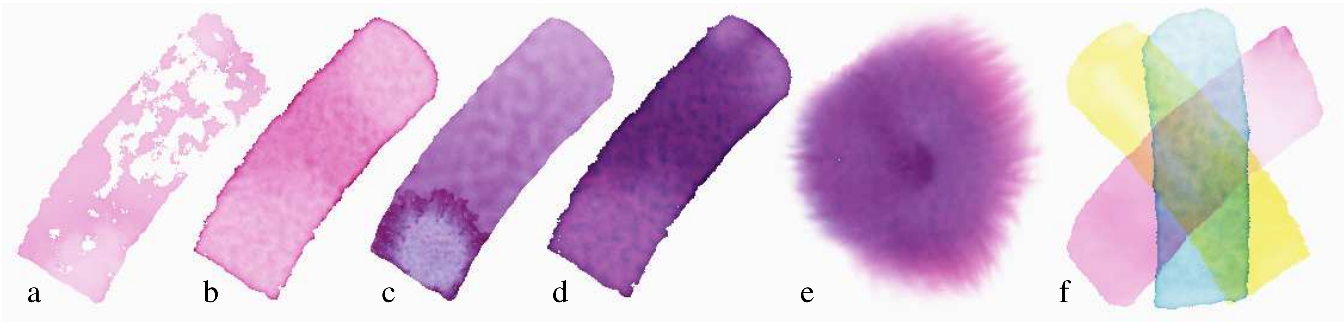

Figure 2 Simulated watercolor effects created using our system.

imposed wet-on-dry washes.

Figure 1 shows scanned-in images of real watercolors. Figure 2 illustrates similar effects obtained from our watercolor simulations.

3 Computer-generated watercolor

Implementing all of these artistic effects automatically presents an interesting challenge, particularly given the paucity of available information on the physical processes involved.[2] In this section, we propose a basic model for the physical and optical behavior of watercolors. The details of this model are then elaborated in the next two sections.

We represent a complete painting as an ordered set of washes over a sheet of rough paper. Each wash may contain various pigments in varying quantities over different parts of the image. We store these quantities in a data structure called a "glaze."

Each glaze is created independently by running a fluid simulation that computes the flow of paint across the paper. The simulation takes, as input, parameters that control the physical properties of the individual pigments, the paper, and the watercolor medium. In addition, the simulation makes use of *wet-area masks*, which represent the areas of paper that have been touched by water. These masks control where water is allowed to flow by limiting the fluid flow computation. The next section describes this fluid simulation in detail.

Once the glazes are computed, they are optically composited using the Kubelka-Munk color model to provide the final visual effect, as described in Section 5.

4 The fluid simulation

In our system, each individual wash is simulated using a three-layer model (Figure 3). From top to bottom, these three layers include:

- The *shallow-water layer* — where water and pigment flow above the surface of the paper.
- The *pigment-deposition layer* — where pigment is deposited onto ("adsorbed by") and lifted ("desorbed") from the paper.
- The *capillary layer* — where water that is absorbed into the paper is diffused by capillary action. (This layer is only used when simulating the backrun effect.)

[2] Indeed, As Mayer points out in his 1991 handbook [25, p. 13]: "The study of artists' materials and techniques is hampered by the lack of systematic data of an authentic nature based on modern scientific laboratory investigations with which to supplement our present knowledge—the accumulation of the practical experience of past centuries, necessarily quite full of principles which rest on the shaky foundations of conjecture and consensus. . . . We await the day when a sustained activity, directed from the viewpoint of the artists, will supply us with more of the benefits of modern science and technology."

In the shallow-water layer (Figure 3a), water flows across the surface in a way that is bounded by the wet-area mask. As the water flows, it lifts pigment from the paper, carries it along, and redeposits it on the paper. The quantities involved in this simulation are:

- The wet-area mask M, which is 1 if the paper is wet, and 0 otherwise.
- The velocity u, v of the water in the x and y directions.
- The pressure p of the water.
- The concentration g^k of each pigment k in the water.
- The slope ∇h of the rough paper surface, defined as the gradient of the paper's height h.
- The physical properties of the watercolor medium, including its *viscosity* μ and *viscous drag* κ. (In all of our examples, we set $\mu = 0.1$ and $\kappa = 0.01$.)

Each pigment k is transferred between the shallow-water layer and the pigment-deposition layer by adsorption and desorption. While pigment in the shallow-water layer is denoted by g^k, we will use d^k for any deposited pigment. The physical properties of the individual pigments, including their density ρ, staining power ω, and granularity γ—all affect the rates of adsorption and desorption by the paper. (The values of these parameters for our examples are shown in the caption for Figure 5.)

The function of the capillary layer is to allow for expansion of the wet-area mask due to capillary flow of water through the pores of the paper. The relevant quantities in this layer are:

- The *water saturation* s of the paper, defined as the fraction of a given volume of space occupied by water.
- The *fluid-holding capacity* c of the paper, which is the fraction of volume not occupied by paper fibers.

All of the above quantities are discretized over a two-dimensional grid representing the plane of the paper.

We will refer to the value of each quantity, say p, at a particular cell using subscripts, such as $p_{i,j}$. We will use bold-italics (such as $\boldsymbol{p}$) to

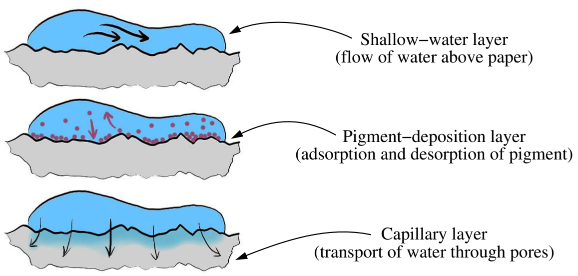

Figure 3 The three-layer fluid model for a watercolor wash.

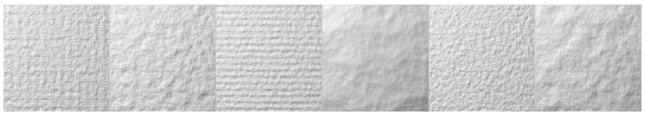

Figure 4 Example paper textures.

denote the entire array of discretized values.

4.1 Paper generation

In real watercolor, the structure of the paper affects fluid flow, backruns, and granulation. The mechanics underlying these effects may be quite complex, and may depend on the precise connections among the individual fibers, as well as the exact slopes of the fine-scale peaks and valleys of the paper. We use a much simpler model in our system. Paper texture is modeled as a height field and a fluid capacity field. The height field $\boldsymbol{h}$ is generated using one of a selection of pseudo-random processes [29, 41], and scaled so that $0 < h < 1$. Some examples of our synthetic paper textures can be seen in Figure 4. The slope of the height field is used to modify the fluid velocity $\boldsymbol{u}$, $\boldsymbol{v}$ in the dynamics simulation. In addition, the fluid capacity $\boldsymbol{c}$ is computed from the height field $\boldsymbol{h}$, as $c = h * (c_{max} - c_{min}) + c_{min}$.

4.2 Main loop

The main loop of our simulation takes as input the initial wet-area mask $\boldsymbol{M}$; the initial velocity of the water $\boldsymbol{u}$, $\boldsymbol{v}$; the initial water pressure $\boldsymbol{p}$; the initial pigment concentrations $\boldsymbol{g}^k$; and the initial water saturation of the paper $\boldsymbol{s}$. The main loop iterates over a specified number of time steps, moving water and pigment in the shallow-water layer, transferring pigment between the shallow-water and pigment-deposition layers, and simulating capillary flow:

proc *MainLoop*($\boldsymbol{M}, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{p}, \boldsymbol{g}^1, \ldots, \boldsymbol{g}^n, \boldsymbol{d}^1, \ldots, \boldsymbol{d}^n, \boldsymbol{s}$):
  for each time step **do**:
    MoveWater($\boldsymbol{M}, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{p}$)
    MovePigment($\boldsymbol{M}, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{g}^1, \ldots, \boldsymbol{g}^n$)
    TransferPigment($\boldsymbol{g}^1, \ldots, \boldsymbol{g}^n, \boldsymbol{d}^1, \ldots, \boldsymbol{d}^n$)
    SimulateCapillaryFlow($\boldsymbol{M}, \boldsymbol{s}$)
  end for
end proc

4.3 Moving water in the shallow water layer

For realism, the behavior of the water should satisfy the following conditions:

1. The flow must be constrained so that water remains within the wet-area mask.
2. A surplus of water in one area should cause flow outward from that area into nearby regions.
3. The flow must be damped to minimize oscillating waves.
4. The flow must be perturbed by the texture of the paper to cause streaks parallel to flow direction.
5. Local changes should have global effects. For example, adding water in a local area should affect the entire simulation.
6. There should be outward flow of the fluid toward the edges to produce the edge-darkening effect.

The first two conditions are satisfied directly by the basic shallow-water equations using appropriate boundary conditions [24, 38]:

$$\frac{\partial u}{\partial t} = -\left(\frac{\partial u^2}{\partial x^2} + \frac{\partial uv}{\partial y^2}\right) + \mu \nabla^2 u - \frac{\partial p}{\partial x} \qquad (1)$$

$$\frac{\partial v}{\partial t} = -\left(\frac{\partial v^2}{\partial y^2} + \frac{\partial uv}{\partial x^2}\right) + \mu \nabla^2 v - \frac{\partial p}{\partial y} \qquad (2)$$

These equations are implemented in the *UpdateVelocities*() subroutine. Conditions 3 and 4 are met by adding terms to the fluid flow simulation involving the viscous drag κ and the paper slope ∇h, as shown in the *UpdateVelocities*() pseudocode. Conditions 5 and 6 are accomplished by two additional subroutines, *RelaxDivergence*() and *FlowOutward*(). All three of these routines are used to implement the movement of water in the shallow-water layer:

proc *MoveWater*($\boldsymbol{M}, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{p}$):
  UpdateVelocities($\boldsymbol{M}, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{p}$)
  RelaxDivergence($\boldsymbol{M}, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{p}$)
  FlowOutward($\boldsymbol{M}, \boldsymbol{p}$)
end proc

4.3.1 Updating the water velocities

To update the water velocities, we discretize the equations (1) and (2) spatially on a staggered grid (as in Foster [10]). An effect of this discretization is that our solution is resolution-dependent. Generalizing to a resolution-independent model is an important goal for future work.

The staggered grid represention stores velocity values at grid cell boundaries and all other values (pressure, pigment concentrations, etc.) at grid cell centers. We use the standard notation for staggered grids, referring to quantities on cell boundaries as having "fractional" indices. For example, the velocity u at the boundary between the grid cells centered at (i,j) and $(i+1,j)$ is called $u_{i+.5,j}$. Furthermore, we will use the shorthand notation $(uv)_{i,j}$ to denote $u_{i,j} v_{i,j}$. We will also use indices to denote quantities that are not represented directly, but computed implicitly from their two immediate neighbors instead. For instance,

$$p_{i+.5,j} \equiv (p_{i,j} + p_{i+1,j}) / 2$$
$$u_{i,j} \equiv (u_{i-.5,j} + u_{i+.5,j}) / 2$$

In the pseudocode below, we discretize equations (1) and (2) in time and solve forward using Euler's Method with an adaptive step size. The step size Δt is set to ensure that velocities do not exceed one pixel per time step:

proc *UpdateVelocities*($\boldsymbol{M}$, $\boldsymbol{u}$, $\boldsymbol{v}$, $\boldsymbol{p}$):
  $(\boldsymbol{u}, \boldsymbol{v}) \leftarrow (\boldsymbol{u}, \boldsymbol{v}) - \nabla h$
  $\Delta t \leftarrow 1 / \lceil \max_{i,j} \{|\boldsymbol{u}|, |\boldsymbol{v}|\} \rceil$
  for $t \leftarrow 0$ **to** 1 **by** Δt **do**
    for all cells (i,j) **do**
      $A \leftarrow u^2_{i,j} - u^2_{i+1,j} + (uv)_{i+.5,j-.5} - (uv)_{i+.5,j+.5}$
      $B \leftarrow (u_{i+1.5,j} + u_{i-.5,j} + u_{i+.5,j+1} + u_{i+.5,j-1} - 4u_{i+.5,j})$
      $u'_{i+.5,j} \leftarrow u_{i+.5,j} + \Delta t\,(A - \mu B + p_{i,j} - p_{i+1,j} - \kappa u_{i+.5,j})$
      $A \leftarrow v^2_{i,j} - v^2_{i,j+1} + (uv)_{i-.5,j+.5} - (uv)_{i+.5,j+.5}$
      $B \leftarrow (v_{i+1,j+.5} + v_{i-1,j+.5} + v_{i,j+1.5} + v_{i,j-.5} - 4v_{i,j+.5})$
      $v'_{i,j+.5} \leftarrow v_{i,j+.5} + \Delta t\,(A - \mu B + p_{i,j} - p_{i,j+1} - \kappa v_{i,j+.5})$
    end for
    $(\boldsymbol{u}, \boldsymbol{v}) \leftarrow (\boldsymbol{u}', \boldsymbol{v}')$
    EnforceBoundaryConditions($\boldsymbol{M}$, $\boldsymbol{u}$, $\boldsymbol{v}$)
  end for
end proc

The *EnforceBoundaryConditions*() procedure simply sets the velocity at the boundary of any pixel not in the wet-area mask to zero.

4.3.2 Relaxation

Following Foster et al. [10], we also relax the divergence of the velocity field $\partial u / \partial x + \partial v / \partial y$ after each time step until it is less than some tolerance τ by redistributing the fluid into neighboring grid cells. In our implementation of the following pseudocode, we have used $N = 50$, $\tau = 0.01$ and $\xi = 0.1$:

proc *RelaxDivergence*($\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{p}$):
  $t \leftarrow 0$
  repeat
    $(\boldsymbol{u}', \boldsymbol{v}') \leftarrow (\boldsymbol{u}, \boldsymbol{v})$
    $\delta_{max} \leftarrow 0$
    for all cells (i, j) **do**
      $\delta \leftarrow \xi(u_{i+1/2,j} - u_{i-1/2,j} + v_{i,j+1/2} - v_{i,j-1/2})$
      $p_{i,j} \leftarrow p_{i,j} + \delta$
      $u'_{i+.5,j} \leftarrow u'_{i+.5,j} + \delta$
      $u'_{i-.5,j} \leftarrow u'_{i-.5,j} - \delta$
      $v'_{i,j+.5} \leftarrow v'_{i,j+.5} + \delta$
      $v'_{i,j-.5} \leftarrow v'_{i,j-.5} - \delta$
      $\delta_{max} \leftarrow \max(|\delta|, \delta_{max})$
    end for
    $(\boldsymbol{u}, \boldsymbol{v}) \leftarrow (\boldsymbol{u}', \boldsymbol{v}')$
    $t \leftarrow t + 1$
  until $\delta_{max} \leq \tau$ **or** $t \geq N$
end proc

4.3.3 Edge darkening

In a wet-on-dry brushstroke, pigment tends to migrate from the interior towards the edges over time. This phenomenon occurs in any evaporating suspension in which the contact line of a drop is pinned in place by surface tension [3]. Because of this geometric constraint, liquid evaporating near the boundary must be replenished by liquid from the interior, resulting in outward flow. This flow carries pigment with it, leading to edge darkening as the water evaporates. In our model, we simulate this flow by decreasing the water pressure near the edges of the wet-area mask.

The *FlowOutward*() routine removes at each time step an amount of water from each cell according to the cell's distance from the boundary of the wet-area mask, with more water removed from cells closer to the boundary. The distance to the boundary is approximated by first performing a Gaussian blur with a $K \times K$ kernel on the wet-area mask $\boldsymbol{M}$. Then an amount of water is removed from each cell according to the value of the resulting Gaussian-blurred image $\boldsymbol{M}'$:

$$\boldsymbol{p} \leftarrow \boldsymbol{p} - \eta (1 - \boldsymbol{M}') \boldsymbol{M} \tag{3}$$

In our examples, $K = 10$ and $0.01 \leq \eta \leq 0.05$.

An example of the edge-darkening effect is shown in Figure 2b.

4.4 Moving pigments

Pigments move within the shallow-water layer as specified by the velocity field $\boldsymbol{u}, \boldsymbol{v}$ computed for the water above. In this part of the simulation, we distribute pigment from each cell to its neighbors according to the rate of fluid movement out of the cell:

proc *MovePigment*($\boldsymbol{M}, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{g}^1, \ldots, \boldsymbol{g}^n$):
  $\Delta t \leftarrow 1 / \lceil \max_{i,j} \{|\boldsymbol{u}|, |\boldsymbol{v}|\} \rceil$
  for each pigment k **do**
    for $t \leftarrow 0$ **to** 1 **by** Δt **do**
      $\boldsymbol{g}' \leftarrow \boldsymbol{g} \leftarrow \boldsymbol{g}^k$
      forall cells (i, j) **do**
        $g'_{i+1,j} \leftarrow g'_{i+1,j} + \max(0,\ u_{i+.5,j}\, g_{i,j})$
        $g'_{i-1,j} \leftarrow g'_{i-1,j} + \max(0,\ -u_{i-.5,j}\, g_{i,j})$
        $g'_{i,j+1} \leftarrow g'_{i,j+1} + \max(0,\ v_{i,j+.5}\, g_{i,j})$
        $g'_{i,j-1} \leftarrow g'_{i,j-1} + \max(0,\ -v_{i,j-.5}\, g_{i,j})$
        $g'_{i,j} \leftarrow g'_{i,j} - \max(0,\ u_{i+.5,j}\, g_{i,j}) + \max(0,\ -u_{i-.5,j}\, g_{i,j})$
          $+ \max(0,\ v_{i,j+.5}\, g_{i,j}) + \max(0,\ -v_{i,j-.5}\, g_{i,j})$
      end for
      $\boldsymbol{g}^k \leftarrow \boldsymbol{g}'$
    end for
  end for
end proc

4.5 Pigment adsorption and desorption

At each step of the simulation, pigment is also adsorbed by the pigment-deposition layer at a certain rate, and desorbed back into the fluid at another rate (in a process similar to the one described by Dorsey *et al.* [8] for weathering patterns due to fluid flow.) The *density* ρ^k and *staining power* ω^k are scalars that affect the rate at which each pigment k is adsorbed and desorbed by the paper. The *granulation* γ^k determines how much the paper height h affects adsorption and desorption.

proc *TransferPigment*($\boldsymbol{g}^1, \ldots, \boldsymbol{g}^n, \boldsymbol{d}^1, \ldots, \boldsymbol{d}^n$):
  for each pigment k **do**
    for all cells (i, j) **do**
      if $M_{i,j} = 1$ **then**
        $\delta_{down} \leftarrow g^k_{i,j}(1 - h_{i,j}\gamma^k)\rho^k$
        $\delta_{up} \leftarrow d^k_{i,j}(1 + (h_{i,j} - 1)\gamma^k)\rho^k / \omega^k$
        if $(d^k_{i,j} + \delta_{down}) > 1$
          then $\delta_{down} \leftarrow \max(0,\ 1 - d^k_{i,j})$
        if $(g^k_{i,j} + \delta_{up}) > 1$
          then $\delta_{up} \leftarrow \max(0,\ 1 - g^k_{i,j})$
        $d^k_{i,j} \leftarrow d^k_{i,j} + \delta_{down} - \delta_{up}$
        $g^k_{i,j} \leftarrow g^k_{i,j} + \delta_{up} - \delta_{down}$
      end if
    end for
  end for
end proc

4.6 Backruns: diffusing water through the capillary layer

Backruns occur only when a puddle of water spreads slowly into a region that is drying but still damp [37]. In a *damp* region, the only water present is within the pores of the paper. In this situation, flow is dominated by capillary effects, not by momentum as in the shallow water equations.

In the backrun simulation, water is absorbed from the shallow-water layer above at the absorption rate α, and diffuses through the capillary layer. Each cell transfers water to its four neighbors until they are saturated to capacity c. If any cell's saturation exceeds a threshold σ, then the wet-area mask is expanded to include that cell. In this way, capillary action within the paper can enable a puddle to spread. The variation in cell capacity from pixel to pixel results in an irregular branching pattern. Other parameters affecting this process are ϵ, the minimum saturation a pixel must have before it can diffuse to its neighbors, and δ, a saturation value below which a pixel will not receive diffusion.

proc *SimulateCapillaryFlow*(s, M):
  forall cells (i, j) **do**
    if $(M_{i,j} > 0)$ **then**
      $s_{i,j} \leftarrow s_{i,j} + \max(0, \min(\alpha,\ c_{i,j} - s_{i,j}))$
  end for
  $\boldsymbol{s}' \leftarrow \boldsymbol{s}$
  for all cells (i, j) **do**
    for each cell $(k, \ell) \in$ *neighbors*(i, j) **do**
      if $s_{i,j} > \epsilon$ **and** $s_{i,j} > s_{k,\ell}$ **and** $s_{k,\ell} > \delta$ **then**
        $\Delta s \leftarrow \max(0, \min(s_{i,j} - s_{k,\ell},\ c_{k,\ell} - s_{k,\ell})/4)$
        $s'_{i,j} \leftarrow s'_{i,j} - \Delta s$
        $s'_{k,\ell} \leftarrow s'_{k,\ell} + \Delta s$
      end if
    end for
  end for
  $\boldsymbol{s} \leftarrow \boldsymbol{s}'$
  for all cells (i, j) **do**
    if $s_{i,j} > \sigma$ **then**
      $M_{i,j} \leftarrow 1$
  end for
end proc

4.7 Drybrush effects

The drybrush effect occurs when the brush is applied at the proper angle and is dry enough to wet only the highest points on the paper surface. We model this effect by excluding from the wet-area mask any pixel whose height is less than a user-defined threshold. An example of simulated drybrush is shown in Figure 1a.

5 Rendering the pigmented layers

We use the Kubelka-Munk (KM) model [14, 20] to perform the optical compositing of glazing layers. (The same model was also used by Dorsey and Hanrahan to model the transmission of light through layers of copper patina [7].)

In our use of the KM model, each pigment is assigned a set of *absorption coefficients* K and *scattering coefficients* S. These coefficients are a function of wavelength, and control the fraction of energy absorbed and scattered back, respectively, per unit distance in the layer of pigment. In our implementation, we use three coefficients each for K and S, representing RGB components of each quantity.

5.1 Specifying the optical properties of pigments

In typical applications of KM theory, the K and S coefficients for a given colorant layer are determined experimentally, using spectral measurements from layers of known thicknesses. However, in our application we have found it to be much more convenient to allow a user to specify the K and S coefficients interactively, by choosing the desired appearance of a "unit thickness" of the pigment over both a white and a black background. Given these two user-selected RGB colors R_w and R_b, respectively, the K and S values can be computed by a simple inversion of the KM equations:

$$S = \frac{1}{b} \cdot \coth^{-1}\left(\frac{b^2 - (a - R_w)(a-1)}{b(1-R_w)}\right)$$
$$K = S(a-1)$$

where

$$a = \frac{1}{2}\left(R_w + \frac{R_b - R_w + 1}{R_b}\right), \qquad b = \sqrt{a^2 - 1}$$

The above computations are applied to each color channel of S, K, R_w, and R_b independently. In order to avoid any divisions by zero, we require that $0 < R_b < R_w < 1$ for each color channel. This restriction is reasonable even for opaque pigments, since the user is specifying reflected colors through just a thin layer, which should still be at least partially transparent. While for most valid combinations of specified colors the computed K and S values fall in the legal range of 0 to 1, for certain very saturated input colors the absorption or scattering coefficients computed by this method may actually exceed the value of 1 in some color channels. Though such a large value of K or S is clearly not possible for any physical pigment, we have not noticed any ill effects in our simulation from allowing such "out-of-range" values. The situation is somewhat analogous to allowing an "alpha" opacity to lie outside the range 0 to 1, another non-physical effect that is sometimes useful [15].

We have found this method of specifying pigments to be quite adequate for creating a wide range of realistic paints (see Figure 5). In addition, the method is much easier than taking the kind of extremely careful measurements that would otherwise be required. By specifying the colors over black and white, the user can easily create different types of pigments. As examples:

- *Opaque paints*, such as Indian Red, exhibit a similar color on both white and black. Such paints have high scattering in the same wavelengths as their color, and high absorption in complementary wavelengths.
- *Transparent paints*, such as Quinacridone Rose, appear colored on white, and nearly black on black. Such paints have low scattering in all wavelengths, and high absorption in wavelengths complementary to their color.
- *Interference paints*, such as Interference Lilac, appear white (or transparent) on white, and colored on black. Such paints have high scattering in the same wavelengths as their color, and low absorption in all wavelengths. Such pigments actually get their color from interference effects involving the phase of light waves, which have been modeled accurately by Gondek *et al.* [11]. While our simple model does not simulate phase effects, it nevertheless manages to produce colors similar in appearance to the interference paints used in watercolor painting.

Our method also makes it easy to simulate real paints that exhibit slightly different hues over black than white, such as Hansa yellow. Figure 5(i) shows a simulated swatch of this pigment over both black and white backgrounds.

5.2 Optical compositing of layers

Given scattering and absorption coefficients S and K for a pigmented layer of given thickness x, the KM model allows us to compute reflectance R and transmittance T through the layer [20]:

$$\begin{aligned} R &= \sinh bSx / c \\ T &= b/c \qquad \text{where} \qquad c = a \sinh bSx + b \cosh bSx \end{aligned}$$

We can then use Kubelka's optical compositing equations [20, 21] to determine the overall reflectance R and transmittance T of two abutting layers with reflectances R_1, R_2 and T_1, T_2, respectively:

$$R = R_1 + \frac{T_1^2 R_2}{1 - R_1 R_2} \qquad\qquad T = \frac{T_1 T_2}{1 - R_1 R_2}$$

This computation is repeated for each additional glaze. The overall reflectance R is then used to render the pixel.

For individual layers containing more than one pigment of thicknesses $x^1, \ldots, x^n$, the S and K coefficients of each pigment k are weighted in proportion to that pigment's relative thickness x^k. The overall thickness of the layer x is taken to be the sum of the thicknesses of the individual pigments.

In our fluid simulation (see Section 4), we use g^k to denote the concentration of pigment in the shallow-water layer, and d^k for the concentration of pigment deposited on the paper. These values are summed to compute the thickness parameter x^k used by the Kubelka-Munk equations.

5.3 Pigment examples

Figure 5 shows the palette of colors used in the examples, with each pigment shown as a swatch painted over a black stripe. The colors we chose for these pigments were based on fairly casual observations of the colors of the actual paints over black and white backgrounds. The K and S coefficients were then derived from these colors by the procedure outlined in Section 5.1. As the thickness of a layer of pigment increases, its color traces a complex curve through color space. For example, Figure 6 shows the range of colors obtainable by glazing "Hansa Yellow" over both white and black backgrounds. Note the difference in hue between the two curves, and the change in both hue and saturation along each curve. This complexity is one of the qualities that gives these pigments their rich appearance.

5.4 Discussion of Kubelka-Munk model

The KM model appears to give very plausible and intuitive results in all the cases we have tried. On the one hand, these results are not very surprising, considering that the KM model was specifically designed for situations akin to watercolor in which there are multiple pigmented layers that scatter and absorb light. It is, however,

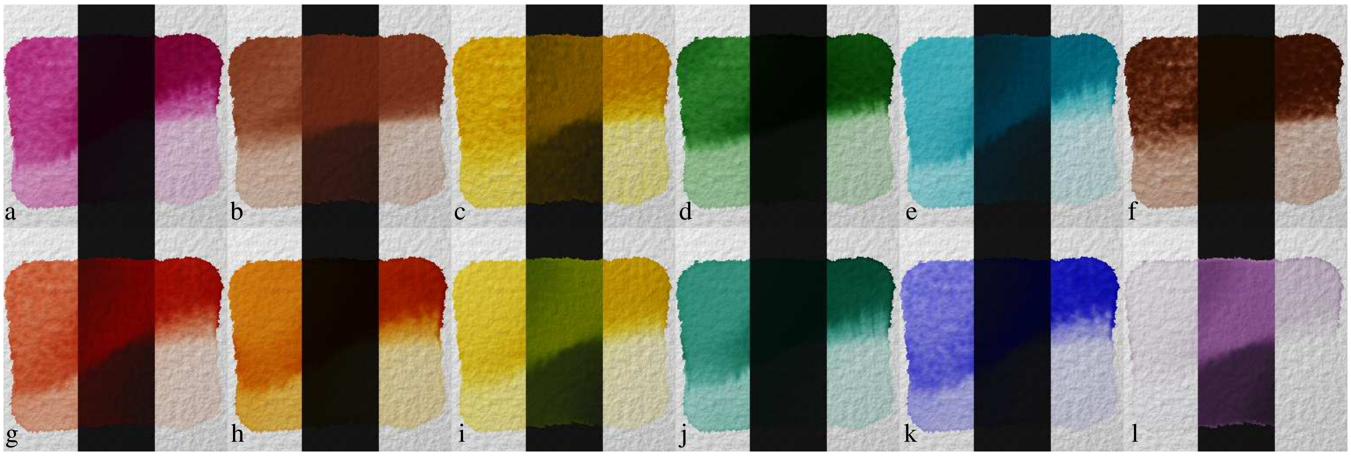

Figure 5 Various synthetic pigments. The swatches were all created using identical initial conditions, with thicker pigment in the top half, and extra water in the upper left and lower right corners. The only changes from swatch to swatch are the pigments' optical and physical parameters, shown at right. The swatches are painted over a black stripe to distinguish the more opaque pigments such as "Indian Red" (b) from the more transparent ones such as "Brilliant Orange" (h).

	Pigment	K_r	K_g	K_b	S_r	S_g	S_b	ρ	ω	γ
a	"Quinacridone Rose"	0.22	1.47	0.57	0.05	0.003	0.03	0.02	5.5	0.81
b	"Indian Red"	0.46	1.07	1.50	1.28	0.38	0.21	0.05	7.0	0.40
c	"Cadmium Yellow"	0.10	0.36	3.45	0.97	0.65	0.007	0.05	3.4	0.81
d	"Hookers Green"	1.62	0.61	1.64	0.01	0.012	0.003	0.09	1.0	0.41
e	"Cerulean Blue"	1.52	0.32	0.25	0.06	0.26	0.40	0.01	1.0	0.31
f	"Burnt Umber"	0.74	1.54	2.10	0.09	0.09	0.004	0.09	9.3	0.90
g	"Cadmium Red"	0.14	1.08	1.68	0.77	0.015	0.018	0.02	1.0	0.63
h	"Brilliant Orange"	0.13	0.81	3.45	0.005	0.009	0.007	0.01	1.0	0.14
i	"Hansa Yellow"	0.06	0.21	1.78	0.50	0.88	0.009	0.06	1.0	0.08
j	"Phthalo Green"	1.55	0.47	0.63	0.01	0.05	0.035	0.02	1.0	0.12
k	"French Ultramarine"	0.86	0.86	0.06	0.005	0.005	0.09	0.01	3.1	0.91
l	"Interference Lilac"	0.08	0.11	0.07	1.25	0.42	1.43	0.06	1.0	0.08

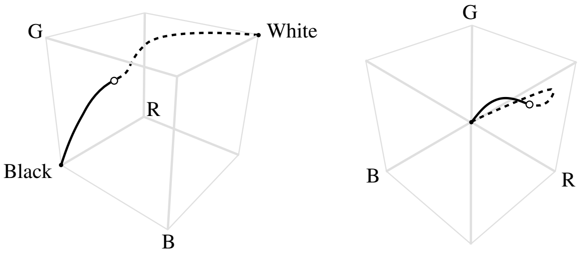

Figure 6 The range of colors obtainable by compositing varying thicknesses of "Hansa Yellow" over black (solid curve) and over white (dashed curve). The point where the two curves meet is R_∞, the color of an infinitely thick layer. At left, the RGB cube is viewed in perspective; at right, we look directly down the luminance axis, showing the difference in hue between the two curves.

worth noting that there are a number of fine points in the basic KM assumptions that are satisfied, at best, only partially in our situation:

1. *All colorant layers are immersed in mediums of the same refractive index.* This assumption is in fact violated at both the "air to pigment-layer" and "pigment-layer to paper" boundaries (although a fairly simple correction term has been proposed [18], that could be used to increase accuracy).

2. *The pigment particles are oriented randomly.* This assumption is satisfied for most watercolor paints, although not all. For example, metallic paint pigments have mostly horizontal flakes.

3. *The illumination is diffuse.* Our simulated watercolors will obviously not look entirely correct under all lighting and viewing conditions. Duntley [20] has a more general theory with four parameters instead of two that can account for more general lighting conditions.

4. *The KM equations apply only to one wavelength at a time.* Fluorescent paints violate this assumption.

5. *There is no chemical or electrical interaction between different pigments, or between the pigment and medium, which would cause clumping of pigment grains and a non-uniform particle size.* These assumptions are violated for most watercolor pigments.

In summary, the fact that the KM model appears to work so well could actually be considered quite surprising, given the number of basic assumptions of the model violated by watercolor. We suspect that while the results of the model are probably not very physically accurate, they at least provide very plausible physical approximations, which appear quite adequate for many applications.

6 Applications

In this section, we briefly discuss three different applications of computer-generated watercolor: interactive painting, automatic image "watercolorization," and 3D non-photorealistic rendering.

6.1 Interactive painting with watercolors

We have written an interactive application that allows a user to paint the initial conditions for the watercolor simulator. The user sets up one or more glazes for the simulator, where each glaze has sub-layers for pigments, water, and a wet-area mask. Common to all glazes are a reference image and a shaded paper texture. There are slider controls to adjust the physical parameters for each glaze (including viscous drag κ, edge darkening η and kernel size K) as well as the number of times to iterate the simulation.

The glaze's pigment channels are represented by colored images in the glaze. Each pigment is painted independently using a circular brush with a Gaussian intensity drop-off. The brush size, penumbra, and overall intensity parameters are adjustable. A palette of pigments associated with a glaze may be defined by specifying the color of each pigment over black and over white, as described in Section 5.1, using an HSV color picker—or by loading predefined pigments from files. For each pigment, the density ρ, staining power ω, and granulation γ may also be controlled using sliders.

The wet-area mask can be painted directly using a similar brush, or by selecting regions from the reference image using "intelligent scissors" [27]. The user can achieve drybrush effects by setting the

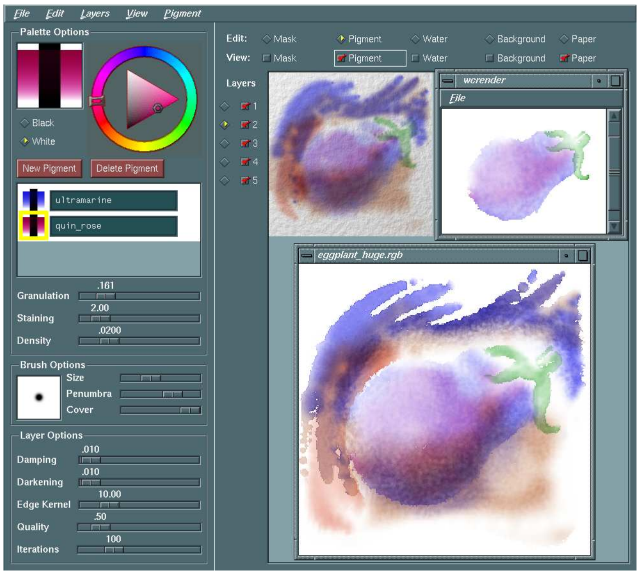

Figure 7 An interactive painting application. At top center are the initial conditions painted by the user; at top right, a watercolor simulation in progress, showing two of the painting's five glazes. The large image is the finished painting.

depth to which the brush is allowed to touch the paper.

Although our watercolor simulator runs too slowly for interactive painting, compositing many glazes of pigments using the KM model is feasible in real-time, and provides the user with valuable feedback about the colors resulting from the simulation. The reference image, paper texture, and set of glazes and their sublayers can be independently toggled on and off, and displayed in any combination. Another helpful feature is a rendering window that displays the progress of running the simulator on a (possibly scaled-down) set of glazes. Lower-resolution simulations are enlarged for display in the viewing window, and each frame of the simulator's animation is typically displayed in a fraction of a second to a few minutes. A screenshot from the application appears in Figure 7, showing several simulations at different stages of completion.

6.2 Automatic image "watercolorization"

Another application we have built allows a color image to be automatically converted into a watercolor illustration, once mattes for the key elements have been extracted and an ordered set of pigments has been chosen, with one pigment per glaze. In our tests, we generated these regions quickly using a commercial paint program. We have also chosen the pigments by hand in our examples, although for further automation, the choice of pigments could instead be computed through an optimization process [31].

The conversion is executed in two stages: *color separation* (Figure 8), in which the ideal distribution of pigment in each glaze is calculated to produce the desired image; and *brushstroke planning* (Figure 9), in which each glaze is painted in an attempt to re-create the desired pigment distributions by adding brushstrokes of water and pigment, taking into account the behavior of the medium. The result is an image that approximates the original but has the flow patterns and texture of a watercolor painting.

Color separation. Color separations are calculated using a brute-force search over a discrete set of thicknesses for each pigment. Given an ordered list of n pigments, the thickness range for each pigment is first divided into m steps, using a binary subdivision by Manhattan distance in the six-dimensional space of the R_i and T_i Kubelka-Munk parameters. Using the KM optical compositing model, a composite color is then computed for each of the m^n combinations, and the color is stored in a 3d-tree according to its *RGB* color values. (The tree is pruned so that the difference between

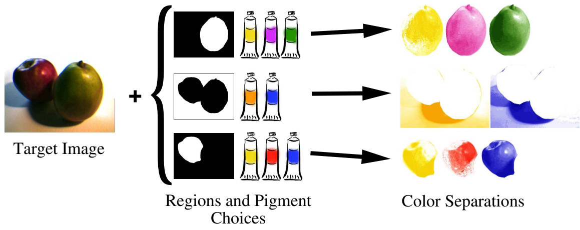

Figure 8 Overview of the color separation process.

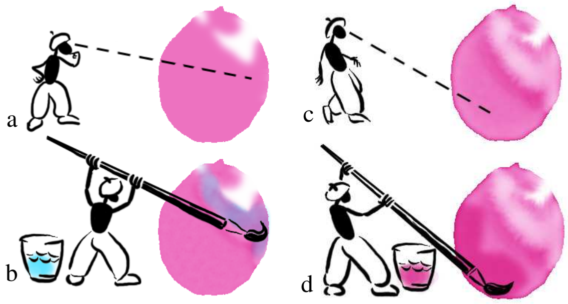

Figure 9 Brushstroke planning. At given intervals, the planner identifies regions containing too much pigment (a) and thins them out by adding plain water (b). The planner can also compensate for a lack of pigment (c) by adding a pigmented wash (d).

colors is less than 1/255.) For the separations used for Figures 10 and 14, $m = 20$, and $n = 3$. Color separations are computed by searching the tree for each pixel to find the pigment combination yielding the closest color to the desired color. These colors are stored in an image called the *target glaze*.

Brushstroke planning. A painter can somewhat control the concentration and flow of pigment in a wash by carefully monitoring the relative wetness of brush and paper, knowing that spreading water carries pigment with it and tends to thin it out. Similarly, we control a glaze by adding incremental brushstrokes of pigment, and we control the direction of flow by increasing or decreasing water pressure wherever pigment is added.

The overall process works by repeatedly querying and manipulating the state of the glaze at a user-specified interval (30 to 100 steps in our examples) during the simulation, using user-controlled parameters δ_g, ϕ_g, and ϕ_p. In our examples, the steps below were repeated between 2 and 5 times, and we used the following values: $0.01 \leq \delta_g \leq 0.2$, $\phi_g = -\delta_g$, and $\phi_p = 1.0$. At each step, the current pigment distribution is compared to the target glaze, ignoring high frequency details, by performing a low-pass filter on the difference between the two. Then one of two actions is performed:

1. In areas where the current glaze does not have enough pigment (by more than δ_g), increment g by δ_g, and increment p by ϕ_g.

2. In areas where the current glaze has too much pigment (by more than δ_g), increment p by ϕ_p.

As a final step, highlights are created by removing paint from areas defined by the user in the form of mattes. This step is analogous to the "lifting out" technique used by artists for similar effects. Figure 10 shows the final results, and Figure 11 shows the appearance of the painting in progress as glazes are added.

Figure 10 An automatic watercolorization (left) of a low resolution image captured using a poor-quality video camera (above). The finished painting consists of 11 glazes, using a total of 2750 iterations of the simulator, rendered at a resolution of 640 by 480 pixels in 7 hours on a 133 MHz SGI R4600 processor.

6.3 Non-photorealistic rendering of 3D models

A straightforward extension of the automatic watercolorization of the previous section is to perform non-photorealistic watercolor rendering directly from 3D models.

Given a 3D geometric scene, we automatically generate mattes isolating each object. These mattes are used as input to the watercolorization process, along with a more traditional "photorealistic" rendering of the scene as the target image. The pigment choices and brushstroke planning parameters are supplied by the user. As shown in Figure 12, even a very primitive "photorealistic" image can thus be converted into a richly textured painting (seen in Figure 14). Figure 13 shows several frames from a painterly animation of clouds generated using only a few dozen spheres.

7 Future Work

Other effects. There are several techniques we do not model, such as spattering and some aspects of the drybrush technique. One way to simulate the appearance of bristle patterns in drybrush would be to integrate hairy brushes [36] with the watercolor simulator. The integration of watercolor with other media such as pen-and-ink would also be interesting.

Automatic rendering. We would like to explore further the idea of automatic watercolorization of images in a more general sense. An algorithm to automatically specify wet areas so that hard edges are placed properly would be especially useful, as well as a color-separation algorithm to calculate the optimal palette of pigments to use for various regions of the image [31]. Other possibilities include automatic recognition and generation of textures using drybrush, spattering, scraping, and other techniques.

Generalization. Our model treats backruns and wet-in-wet flow patterns as two separate processes. In real watercolor, however, they are just two extremes of a continuum of effects, the difference between them being simply the degree of wetness of the paper. A model that could integrate these two effects, parametrized by wetness, would be a significant improvement.

Animation issues. When an animated sequence is converted to watercolor one frame at a time, the resulting animation exhibits certain temporal artifacts, such as the "shower door" effect [26]. In the future we would like to develop a system that takes into account the issue of coherency over time and allows the user to control these artifacts.

Acknowledgements

The 3D target animation for Figures 12–14 was created by Siang Lin Loo. We would also like to thank Ron Harmon of Daniel Smith Artists' Materials for connecting us to reality; John Hughes, Alan Barr, Randy Leveque, Michael Wong, and Adam Finkelstein for many helpful discussions; and Daniel Wexler and Adam Schaeffer for assistance with the images.

This work was supported by an Alfred P. Sloan Research Fellowship (BR-3495), an NSF Presidential Faculty Fellow award (CCR-9553199), an ONR Young Investigator award (N00014-95-1-0728) and Augmentation award (N00014-90-J-P00002), and an industrial gift from Microsoft.

References

[1] Jim X. Chen and Niels da Vitoria Lobo. Toward interactive-rate simulation of fluids with moving obstacles using navier-stokes equations. *Graphical Models and Image Processing*, 57(2):107–116, March 1995.

[2] Tunde Cockshott, John Patterson, and David England. Modelling the texture of paint. *Computer Graphics Forum (Eurographics '92)*, 11(3):217–226, September 1992.

[3] Robert D. Deegan, Olgica Bakajin, Todd F. Dupont, Greg Huber, Sidney R. Nagel, and Thomas A. Witten. Contact line deposits in an evaporating drop. *James Franck Institute (University of Chicago) preprint*, October 1996.

[4] Jeanne Dobie. *Making Color Sing*. Watson-Guptill, 1986.

[5] Debra Dooley and Michael F. Cohen. Automatic illustration of 3D geometric models: Lines. *Computer Graphics*, 24(2):77–82, March 1990.

[6] Debra Dooley and Michael F. Cohen. Automatic illustration of 3D geometric models: Surfaces. In *Proceedings of Visualization '90*, pages 307–314. October 1990.

[7] Julie Dorsey and Pat Hanrahan. Modeling and rendering of metallic patinas. In *SIGGRAPH '96 Proceedings*, pages 387–396. 1996.

[8] Julie Dorsey, Hans Køhling Pedersen, and Pat Hanrahan. Flow and changes in appearance. In *SIGGRAPH '96 Proceedings*, pages 411–420. 1996.

[9] Gershon Elber. Line art rendering via a coverage of isoparametric curves. *IEEE Transaction on Visualization and Computer Graphics*, 1(3):231–239, September 1995.

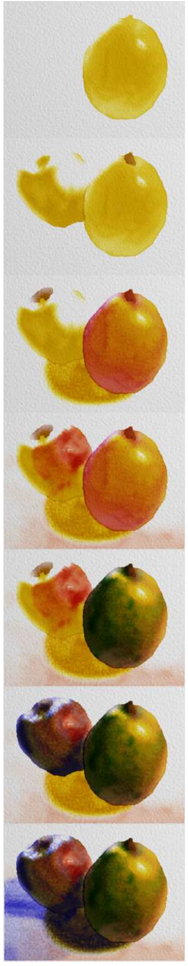

Figure 11 Steps in the rendering of Figure 10.

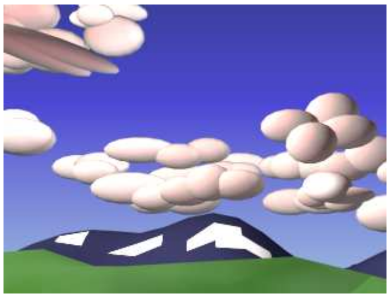

Figure 12 The target image for Figure 14.

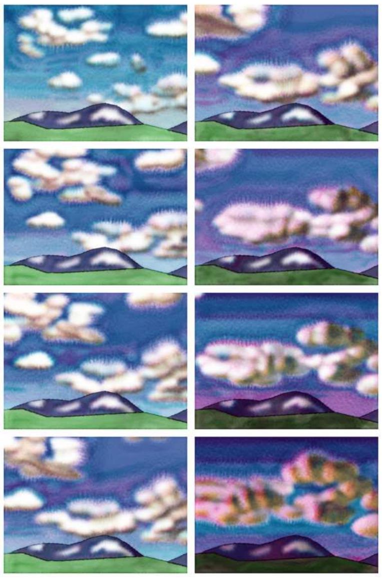

Figure 13 Several frames from a non-photorealistic animation of moving clouds.

Figure 14 Detail of one frame from Figure 13.

[10] Nick Foster and Dimitri Metaxas. Realistic animation of liquids. In *Graphics Interface '96*, pages 204–212. 1996.

[11] Jay S. Gondek, Gary W. Meyer, and Jonathan G. Newman. Wavelength dependent reflectance functions. In *SIGGRAPH '94 Proceedings*, pages 213–220. 1994.

[12] Qinglian Guo. Generating realistic calligraphy words. *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, E78A(11):1556–1558, November 1996.

[13] Qinglian Guo and T. L. Kunii. Modeling the diffuse painting of sumie. In T. L. Kunii, editor, *IFIP Modeling in Comnputer Graphics*. 1991.

[14] Chet S. Haase and Gary W. Meyer. Modeling pigmented materials for realistic image synthesis. *ACM Trans. on Graphics*, 11(4):305, October 1992.

[15] Paul Haeberli and Douglas Voorhies. Image processing by linear interpolation and extrapolation. *IRIS Universe Magazine*, (28), Aug 1994.

[16] Paul E. Haeberli. Paint by numbers: Abstract image representations. In *SIGGRAPH '90 Proceedings*, pages 207–214. 1990.

[17] Ron Harmon. *personal communication*. Techical Manager, Daniel Smith Artists' Materials, 1996.

[18] D. B. Judd and G. Wyszecki. *Color in Business, Science, and Industry*. John Wiley and Sons, New York, 1975.

[19] Michael Kass and Gavin Miller. Rapid, stable fluid dynamics for computer graphics. In *SIGGRAPH '90 Proceedings*, pages 49–57. 1990.

[20] G. Kortum. *Reflectance Spectroscopy*. Springer-Verlag, 1969.

[21] P. Kubelka. New contributions to the optics of intensely light-scattering material, part ii: Non-homogeneous layers. *J. Optical Society*, 44:330, 1954.

[22] John Lansdown and Simon Schofield. Expressive rendering: A review of nonphotorealistic techniques. *IEEE Computer Graphics and Applications*, 15(3):29–37, May 1995.

[23] Wolfgang Leister. Computer generated copper plates. *Computer Graphics Forum*, 13(1):69–77, 1994.

[24] James A. Liggett. Basic equations of unsteady flow. *Unsteady Flow in Open Channels, Vol. 1*, eds: K. Mahmood and V. Yevjevich, Water Resources Publications, Fort Collins, Colorado, 1975.

[25] Ralph Mayer. *The Artist's Handbook of Materials and Techniques*. Penguin Books, 5 edition, 1991.

[26] Barbara J. Meier. Painterly rendering for animation. In *SIGGRAPH '96 Proceedings*, pages 477–484. 1996.

[27] Eric N. Mortensen and William A. Barrett. Intelligent scissors for image composition. In *SIGGRAPH '95 Proceedings*, pages 191–198. 1995.

[28] F. Kenton Musgrave, Craig E. Kolb, and Robert S. Mace. The synthesis and rendering of eroded fractal terrains. In *SIGGRAPH '89 Proceedings*, pages 41–50. 1989.

[29] Ken Perlin. An image synthesizer. In *SIGGRAPH '85 Proceedings*, pages 287–296. July 1985.

[30] Binh Pham. Expressive brush strokes. *CVGIP: Graphical Models and Image Processing*, 53(1), 1991.

[31] Joanna L. Power, Brad S. West, Eric J. Stollnitz, and David H. Salesin. Reproducing color images as duotones. In *SIGGRAPH '96 Proceedings*, pages 237–248. 1996.

[32] Don Rankin. *Mastering Glazing Techniques in Watercolor*. Watson-Guptill, 1986.

[33] Takafumi Saito and Tokiichiro Takahashi. Comprehensible rendering of 3D shapes. *Computer Graphics*, 24(4):197–206, August 1990.

[34] David Small. Simulating watercolor by modeling diffusion, pigment, and paper fibers. In *Proceedings of SPIE '91*. February 1991.

[35] Ray Smith. *The Artist's Handbook*. Alfred A. Knopf, 1987.

[36] Steve Strassmann. Hairy brushes. In *SIGGRAPH '86 Proceedings*, pages 225–232. August 1986.

[37] Zoltan Szabo. *Creative Watercolor Techniques*. Watson-Guptill, 1974.

[38] C. B. Vreugdenhil. *Numerical Methods for Shallow-Water Flow*. Kluwer Academic Publishers, 1994.

[39] Georges Winkenbach and David H. Salesin. Computer–generated pen–and–ink illustration. In *SIGGRAPH '94 Proceedings*, pages 91–100. 1994.

[40] Georges Winkenbach and David H. Salesin. Rendering free-form surfaces in pen and ink. In *SIGGRAPH '96 Proceedings*, pages 469–476. 1996.

[41] Steven P. Worley. A cellular texturing basis function. In *SIGGRAPH '96 Proceedings*, pages 291–294. 1996.