Real-time Watercolor Simulation with Fluid Vorticity within Brush Stroke

Roman Ďurikovič and Zuzana Páleníková
Faculty of Mathematics, Physics and Informatics, Comenius University Bratislava
842 48 Bratislava, Slovakia
Email: roman.durikovic@fmph.uniba.sk

***Abstract*—We investigate the watercolor brush strokes from the perspective of fluid dynamics and the pigments movement within the flowing water. We propose a method to plausibly simulate process of watercolor painting on rough paper including the diffusion effects on graphics processor. We present a method based on 2D fluid simulation using Navier-Stokes equations to simulate the ink diffusion. Intricate details of water flow are preserved by taking into account the vorticity, too. Paper roughness and structure are considered in our approach when evaluating viscous term. Finally, to compose multiple colored semi-transparent layers of brush strokes Kubelka-Munk theory is used.**

***Keywords*-Ink simulation, ink diffusion, pigment rendering**

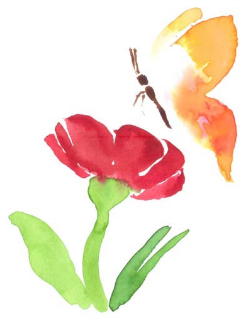
Figure 1. Example of real life watercolor painting. Notice uneven spread of paint over paper's surface.

I. Introduction

Watercolor is a challenging artistic technique that relies heavily on physical phenomena such as behavior of water on paper of different properties. To achieve desired effects it is necessary to plan out individual brush strokes and order in which they will be carried out beforehand, since watercolor behaves differently on wet paper and on dry paper. Any mistakes are practically irreversible. A visually plausible computer simulation of such effect is therefore an attractive alternative. There are many applications which give existing image the look and feel of watercolor, yet there is a shortage in application which simulate the process and allow user to intervene in the process. Having such a system would also be a great training tool for beginner artists. Being able to precisely control amount of water on paper and balance ink concentration without the risk of ruining expensive material enables the artist to express himself more freely.

Simulating watercolor painting has been an interesting challenge in computer graphics. We aim at producing a model that covers pigment and fluid motion that can be utilized in real-time application with instant feedback.

Another motivation to our research is to preserve the artistic process of painting the cultural artifact. Many excellent painters producing the cultural heritage artifacts are getting old and their skills have not been passed to younger generation, we aim to preserve the process and the skills by capturing them with the virtual brush.

So far the research in watercolor visualization has focused mostly on off-line rendering of a scene in painterly style. Fluid motion and effects caused by ink diffusion in water on paper's surface are modeled either by ad-hoc determined diffusion equations [1] or by solving Navier-Stokes equations for incompressible flow, mostly using Eulerian approach. Pigment amounts are represented as concentrations in water cells [2] or as individual particles taking their velocity from the water movement [3].

Ink diffusion simulation has mostly been used for monochromatic paintings such as Oriental calligraphy [1]. However when multiple pigments are needed, it is necessary to correctly compose different ink layers. For this purpose, Kubelka-Munk optical compositing equations have been used [2].

We propose a method for real-time simulation of watercolor painting with emphasis on diffusion effects seen in real life, see Fig. 1. First, we present an overview of methods used for simulating fluids in digital watercolor painting. We look into different approaches to fluid representation and behavior and discuss their pros and cons.

Next we propose our method which uses Navier-Stokes equations for incompressible flow to simulate fluid motion and particle-based method to model ink movement. We identify useful ideas found in other works and incorporate them into our own system. We describe details of implementation and stumbling blocks we've encountered due to hardware limitation, design choices and other reasons. Finally, we evaluate results of our system compared to real world samples. We analyze system's performance both from perspective of visual fidelity and responsiveness and provide examples of our system's output.

II. Proposed Theoretical Model

We first focus on visualization of brush strokes, since it does not depend on chosen method of ink diffusion and fluid motion. We represent water and ink amounts as scalar values of 2D grid.

A. Rendering overlapping brush strokes

We chose to simulate the stacking of pigmented layers with Kubelka and Munks theory [4]. This allows us to take into account the local influence of subsurface scattering on the pigmented layer applied upon the primer layer. Kubelka and Munks theory is considered as a good evaluation of the reflectance in the normal direction to the surface while it assumes that the incident and the outgoing light flux are diffuse, see Fig. 2. To compose two layers of colored pigments with reflectance and transmittance values R_1, R_2 and T_1, T_2 respectively, we utilize the following Kubelka-Munk equations:

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

where R represents overall reflectance and T stands for overall transmittance. Reflectance and transmittance for individual layers are computed as follows:

$$\begin{aligned} R_1 &= \sinh(bSh_1), & T_1 &= \frac{b}{c}, \\ b &= \sqrt{a^2 - 1}, & a &= 1 + \frac{K}{S}, \\ c &= a \sinh(bSh_1) &+& \; b \cosh(bSh_1), \end{aligned} \tag{2}$$

where K and S in these equations represent absorption and scattering coefficients of the evaluated layer and h_1 denotes thickness of the layer. We assume layer thickness equals ink amount in fluid simulation. In our work we use absorption and scattering coefficients for pigments, K and S values, respectively determined by measurements in [2]. Both the coefficients as well as reflectance and transmittance are computed as three-dimensional vectors corresponding to the respective RGB components. Figure 3 on top raw demonstrates the use of yellow and blue RGB components for coefficients.

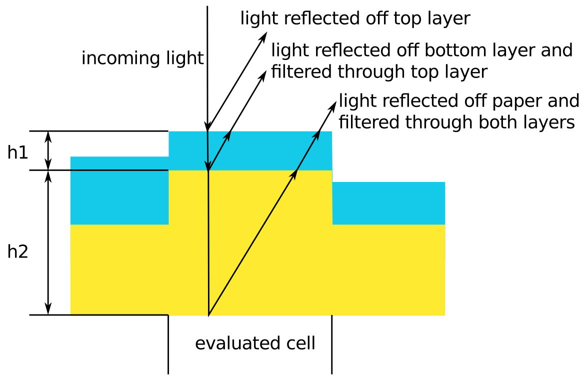
light reflected off top layer
incoming light
light reflected off bottom layer and filtered through top layer
light reflected off paper and filtered through both layers
h1
h2
evaluated cell

Figure 2. Decomposition of optical phenomena contributing to final color of evaluated cell. *h1* and *h2* represent thickness of respective pigment layers at evaluated cell.

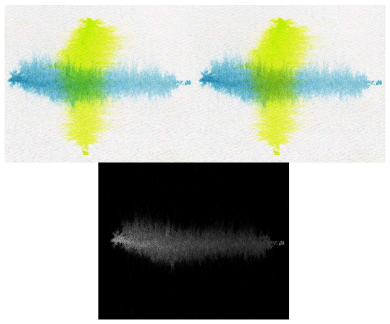

Figure 3. Images in the top row represent two dried overlapping strokes rendered with our system. On the left side, the blue stroke is placed on top, while on the right image the yellow stroke is the top-most. Bottom image shows layer thickness used in rendering, it was obtained by inverting lightness channel from scanned image of a dry stroke in LAB color model.

B. Water dynamics simulation

The watercolor strokes can be separated into fluid dynamics layer simulating the water solvent movement and the pigment layer simulating the diffusion effects. The continuous form of standard Navier-Stokes equations for incompressible flow is given by:

$$\begin{aligned} &\frac{\partial \vec{u}}{\partial t} = -(\vec{u} \cdot \nabla)\vec{u} - \frac{\nabla p}{\rho} + \nu \nabla^2 \vec{u} + \vec{F}, \\ &\nabla \cdot \vec{u} = 0, \end{aligned} \tag{3}$$

where $\vec{u}$ is fluid velocity, ρ is fluid density, p stands for pressure, ν and $\vec{F}$ represent viscosity and external forces, respectively. To solve the equations we discretize the space into a uniform 2D grid and employ the standard projection method, where we first, advect the fluid velocity and quantity. Then we compute pressure at grid centers by solving Poisson equation and subtract pressure gradient from advected velocity, see [5] for more details. In our implementation we omitted viscous term, its role in damping the system will be replaced by diffusion term described later. We use no-slip boundary conditions on the borders of water puddles.

C. Fluid vorticity within brush stroke

Vorticity [3] is responsible for uneven distribution of pigments within the water solvent, the effect that can be seen when the pigment density is very low. We employ the vorticity in the form of external force $\vec{f_{vort}}$:

$$\begin{aligned} \vec{f_{vort}} &= \epsilon(\vec{\psi} \times \vec{w})\Delta x, \\ \vec{\psi} = \frac{\vec{\eta}}{|\vec{\eta}|}, \quad \vec{\eta} &= \nabla|\vec{w}|, \quad \vec{w} = \nabla \times \vec{u}, \end{aligned} \tag{4}$$

where ϵ is a parameter for controlling the magnitude of vorticity force and Δx represents the size of simulation grid.

Vorticity force is finally added to velocity as the last step of simulation.

D. Taking paper structure into account

Paper's uneven surface has significant influence on water flow. To simulate inhibition of velocity caused by dragging the fluid across uneven surface we employ following viscosity term:

$$\vec{u_{new}} = \vec{u} - \nabla\vec{u} \cdot h, \tag{5}$$

where h stands for paper height at evaluated point. Equation 5 is applied at the final step after the velocity calculation from Navier-Stokes Equations 3. By this step we can also achieve the global damping of system to guarantee the stability during the user interaction.

III. Implementation

The method we propose is a combination of grid-based and particle-based approach. We use a 2D grid to store information about fluid amount and velocity as well as intermediate computations such as pressure, curl etc. We also use grid to store information about the paper. Our paper model consists of 3-channel color texture which represents what paper looks like and single channel texture which represents irregularities on paper surface as a height map. In most of our experiments, paper height map was obtained as lightness channel from color texture converted to La*b* color model, see Fig. 4. On the other hand, we use particles to store information about pigment distribution. For visualization purpose these particles are then rendered into 2D texture which represents thickness of pigment layer at particular grid cell. This approach proves to be more flexible when dealing with multi-colored paintings. Particle-based pigment model makes mixing multiple pigments in one water puddle easier. Determining order in which strokes were placed is easy and does not require creating a new grid for each stroke. Adding new particles to simulation is trivial, whereas grid-based approach to pigment storage would require to create new grid in places where one pigment overlaps with another. Using particle buffer is more efficient in terms of memory consumption compared to grid storage and is easy to implement on GPU, which is crucial to achieve application with real-time feedback.

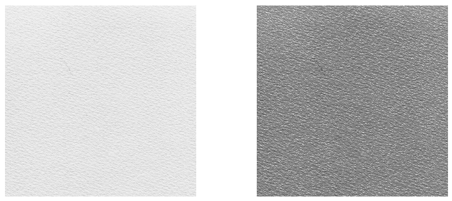
Figure 4. Images we used as our paper model. On the left side is RGB texture of paper's appearance, on the right side is grayscale texture of paper's structure.

Our method consists of three main stages. First, we compute fluid movement by solving Navier-Stokes equations with added vorticity confinement and some minor modifications to incorporate paper's influence on fluid movement. Then we integrate particle movement, according to water velocity in grid cell where particle is located. Finally, we visualize pigment particles using Kubelka-Munk optical compositing equations to resolve overlapping strokes.

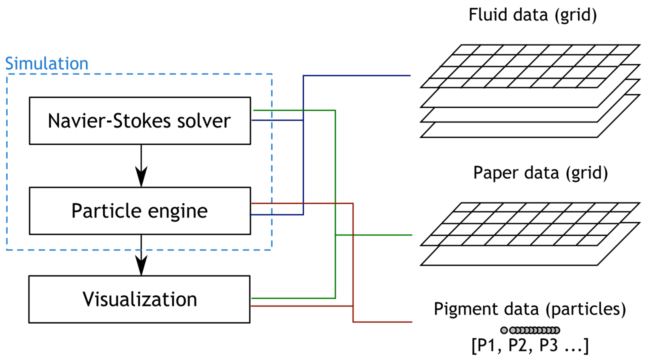

Figure 5. Diagram showing high-level logic of our method, colored lines depict where particular data structure is used.

IV. Navier-Stokes Solver

Change in fluid velocity consists of four terms: advection, diffusion, pressure and external forces. Helmholtz-Hodge decomposition allows us to compute divergence-free velocity as a difference of divergent velocity and gradient of pressure. Is useful to first compute advection, diffusion and external forces terms, then compute pressure out of resulting divergent velocity field and finally, subtract pressure gradient from velocity to achieve divergence-free flow.

A. Advection

We trace the trajectory of fluid in a given grid cell back in time along direction opposite to its current velocity and copy values in traced cell to currently computed cell. Mathematically this process can be described by following equation:

$$q(\vec{x}, t + \delta t) = q(\vec{x} - \vec{u}(\vec{x}, t)\delta t, t), \tag{6}$$

where q represents property we are advecting, in our case fluid quantity and velocity, $\vec{x}$ stands for position of evaluated cell on grid, $\vec{u}$ is velocity at evaluated cell and finally, t and δt represent time and time step, respectively.

B. Diffusion

To compute diffusion in time we use backward method of integration again, resulting in following implicit equation:

$$(\mathbf{I} - \nu\delta t\nabla^2)\vec{u}(\vec{x}, t + \delta t) = \vec{u}(\vec{x}, t), \tag{7}$$

where $\vec{u}$, $\vec{x}$ and t are velocity, cell position and time respectively, δt stands for time step , $\mathbf{I}$ represents identity matrix and ∇^2 is the discrete form of Laplacian operator. ν is viscosity coefficient used to control amount of diffusion, higher values model fluid more resistant to flow. Implicit integration formulated above is a Poisson equation for velocity we solve Poisson equation with Jacobi iterative method.

C. External forces as paper surface

In our implementation we model paper influence as an external force that decelerates fluid's flow according to irregularities on paper's surface. This can be formulated in general as $\vec{u}_{new} = \vec{u} - k\vec{u}$ where k is a coefficient which takes paper's structure into account. Choosing appropriate coefficient is not an easy task, because by adding external forces we introduce instability into simulation. To keep velocity divergence-free, we have to perform computation of external forces before computing and subtracting pressure gradient. We have tried different forms of k, in the end we settled on following solution:

$$\vec{u}_{new} = \vec{u} - clamp(\nabla\vec{u}, 0, d)h\vec{u}, \tag{8}$$

where h represents paper height at evaluated position and $\nabla\vec{u}$ is divergence of velocity. Since we calculate paper's influence before computing and subtracting pressure gradient, divergence can be non-zero. It describes spots where quantity is most likely to leave the area. We clamp divergence to avoid negative values since we are only interested in inhibition of flow not acceleration. We also set upper bound to divergence values to d. To keep simulation stable, we set $d = 0.05$. Unfortunately, this means that in order to maintain stability of system, paper's influence is much weaker than in real life.

D. Pressure computation

Since formula to calculate pressure is a Poisson equation in form of $\nabla^2 p = \nabla \cdot \vec{w}$ we utilize Jacobi iterative method like in the case of viscous diffusion. Performing this we are able to simulate spreading of water from areas with high amounts of water to areas with small amounts of water.

E. Vorticity confinement

Finally, we add vorticity confinement to computed velocity to preserve fine rotational details of fluid flow, that would be lost due to numerical imprecision. First, we compute curl value $\vec{w}$ out of velocity $\vec{u}$ with following formula:

$$\vec{w} = \nabla \times \vec{u} = \left(\frac{\partial u_z}{\partial y} - \frac{\partial u_y}{\partial z}, \frac{\partial u_x}{\partial z} - \frac{\partial u_z}{\partial x}, \frac{\partial u_y}{\partial x} - \frac{\partial u_x}{\partial y}\right). \tag{9}$$

Curl is a vector value, however since we consider 2D simulation its first two coordinates are zero. Finally, we end up with Equation 4. We add vorticity force $\vec{f_{vort}}$ directly to velocity field $\vec{u}$. Thanks to vector calculus identity $\nabla \cdot \nabla \times = \nabla \times \nabla \cdot$ velocity stays divergence-free.

F. Drying

As the last step in the simulation loop we perform drying process. We simply subtract small constant amount of fluid from every wet cell. We clamp resulting value to zero. Along with globally uniform drying we also perform adhoc heuristic to simulate spreading of water puddle when fluid quantity surpasses critical value. General idea is that if height of water column (water quantity + paper height) is higher than water column of its dry neighbor we move fixed amount of water from wet cell to dry cell according to drying rate of paper.

G. Boundary conditions

To correctly solve any differential equation we need to pose appropriate boundary conditions, that is how method behaves on borders of the simulation domain. In our case, simulation domain is represented by wet areas of paper, therefore its border consists of dry cells directly adjacent to wet cells. For velocity we use no-slip condition, which means that velocity goes to zero at the border of domain. Therefore, velocity of any dry cell automatically equals zero. However, correct solution of Poisson equation for pressure requires it to satisfy pure Neumann boundary conditions, expressed as $\frac{\partial p}{\partial \vec{n}} = 0$ where $\vec{n}$ is vector in direction normal to domain boundary. This means that change in pressure along direction perpendicular to borders of brush stroke equals zero. We achieve this when we approximate pressure derivative with finite difference method that we set pressure of adjacent dry border cells to value at evaluated cell.

V. Pigment Particle Engine

We separated pigment movement from fluid movement by simulating pigment as particles. To move particles we look up velocity value in corresponding fluid grid cell and use it to determine changes in particle's position. We found second order Runge-Kutta integration to give acceptable results. New position $\vec{x}$ of particle p is determined by following equations:

$$\begin{aligned} \vec{u}_p^{n+1} &= \vec{f}_{advection} \cdot \Delta t = \rho \cdot \vec{u}_{fluid}, \\ \vec{x}_p^{n+\frac{1}{2}} &= \vec{x}_p^{n} + \tfrac{1}{2}\Delta t' \cdot \vec{u}_p^{n}, \\ \vec{x}_p^{n+1} &= \vec{x}_p^{n+\frac{1}{2}} + \Delta t' \cdot \vec{u}_p^{n+\frac{1}{2}}, \end{aligned} \tag{10}$$

where $\vec{x}$ and $\vec{u}$ denote position and velocity, respectively. Subscript indicates particle for which we are evaluating position and superscripts indicate time iteration. $\vec{f}_{advection}$ is velocity field of pigment particle, ρ stands for fluid density and Δt with $\Delta t'$ represent time steps for fluid and particle simulations, respectively. The particles do not interact with each other, therefore clumping of ink particles may occur. Further improvement of particle simulation can be achieved by taking into account repulsive forces between particles. Since, we implement particle simulation on GPU which processes each particle individually without access to other particles we do not take repulsive forces into account.

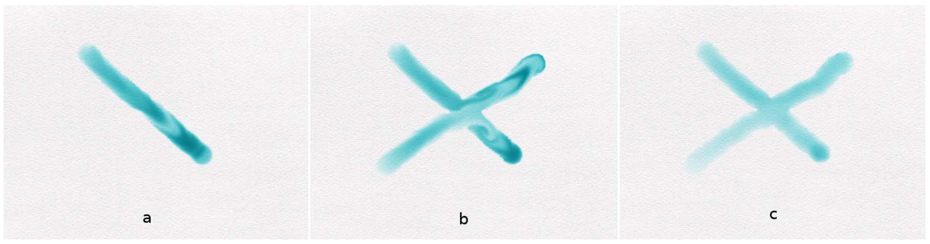

Figure 6. Brush strokes created by our system. Color represents the amount of fluid within the watercolor, darker shades mean higher values. a) Single water brush strokes with swirls within the fluid. b) Second brush stroke crossing the previous one with captured fluid flow. c) After a while the fluid reached its equilibrium.

Simulation grid size (pixels)	Grid size ratio 1.0	1.5	2.0
320x240	57.93 fps	54.85 fps	56.19 fps
400x300	52.38 fps	54.99 fps	54.62 fps
512x384	39.87 fps	38.45 fps	27.48 fps
640x480	25.73 fps	24.83 fps	16.07 fps

Table I
TABLE SHOWING PERFORMANCE OF OUR IMPLEMENTATION MEASURED AS AVERAGE FRAMES PER SECOND.

VI. VISUALIZATION

To properly render pigmented layers, see Section II-A, we need 2D grid that represents pigment's thickness on paper. To create such grid, we render particles into a 2D texture which then serves as thickness layer. We render particles as point sprites whose intensity falls off with distance from particle's position. We also assign each particle random coefficient which can be interpreted as staining power, with which we then multiply given base alpha value. Result is single channel texture where higher intensity means thicker pigment layer.

A. *Fixation of particles on dry paper*

Thickness of individual pigmented layers is updated every frame by rendering corresponding particles into thickness texture. When pigment particles end up on dry location, we want to fix those particles onto paper permanently. Because we separated visualization from simulation it is not so straightforward. In the simulation loop we delete particles that end up on dry paper. Then we render reflectance and transmittance values of pigment composite we stored from previous frame onto paper. In dry cells which contain some pigment we overwrite paper's reflectance value with composite of paper's old reflectance and pigment composite from previous frame. This procedure assumes that water in almost dry cells does not move very much.

VII. RESULTS AND EVALUATION

We implemented both fluid simulation and Kubelka-Munk optical compositing on GPU using GLSL fragment shaders. The amount of fluid and velocity are stored in 2D textures, with simulation domain using smaller grid resolution than the visualization grid. Application runs at interactive frame rate, performance varies depending on size of simulation domain. Table I shows results of experiments with different grid sizes. The *grid size ratio* between visualization grid size and simulation grid size determines how much bigger is the visualization domain compared to simulation domain. All tests were performed on HP Pavilion dv7 with NVidia GeForce 9600M GT graphics card. Scaling visualization domain allows for higher resolution image while maintaining reasonable image quality and responsiveness. Figure 6 has been rendered with grid ratio 2.0 in other word the visualization grid size was twice as large as simulation grid. Speed performance is shown on Table I, frame rate lower than 30fps displays noticeable lag in user feedback.

We compare our results with real watercolor samples. Unless noted otherwise, the following simulation parameters were used:

- Global parameters: simulation grid resolution 320x240, visualization grid resolution 640x480, splat radius 20pixels, fluid per splat = 0.02
- Fluid simulation parameters: fluid viscosity = 1.0, vorticity strength = 5.0, number of Jacobi iterations in pressure computation = 40 (20 in diffusion computation), paper influence = 0.1, drying rate = 0.00001
- Particle engine parameters: particles per splat = 1500, particle sprite radius = 0.5

A. *Dark border around strokes*

First, we demonstrate darkening border of strokes, attribute typical to watercolor. It occurs when stroke is painted with thin solution of watercolor and large amount of water. Stroke slowly dries, borders drying faster than center, and fluid as well as pigment travels from center of stroke to border where it dries up and leaves darker mark on paper. Figure 7 shows result of our system side by side with real sample.

B. *Evolution of feathery patterns*

Paper's influence on watercolor flow exhibits itself mainly when ink spreads over area of wet paper. We experimented with dropping large blot of pigment on boundary of wet area on paper, Figure 8 top images. Figure 8 shows simulation

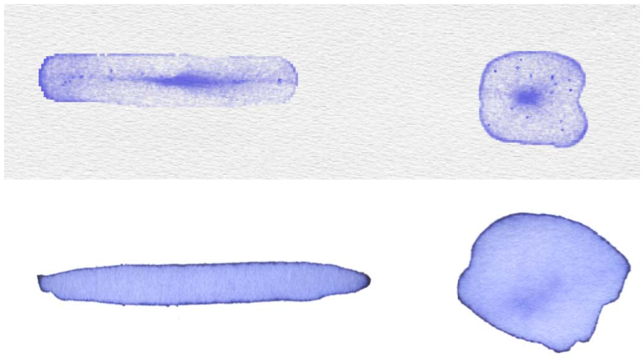
Figure 7. Example of stroke with dark border. Top image is produced by our simulation using French Ultramarine pigment. Bottom image is real paint sample.

of this process. We highlighted feathery pattern within fluid flow on third image, this pattern. Because we don't take forces between particles into account, particles tend to clump together in cells with largest change in velocity. In real life pigment particles would repel each other and slowly balance ink concentrations. In the example we used Brilliant Orange pigment.

VIII. Conclusion

The proposed simulation is based on the analysis of factors involved in watercolor painting. The factors taken into consideration by the simulation include how the process of water spreading on paper takes place, how watercolor pigments dissolve into the water through diffusion, how the diffusion process characteristics influence the spreading of pigment particles, and how the characteristics of the paper texture affects the pigment diffusion throughout the water stroke. This computational watercolor technique also allows the combination of strokes of different colors and black ink Japanese calligraphy.

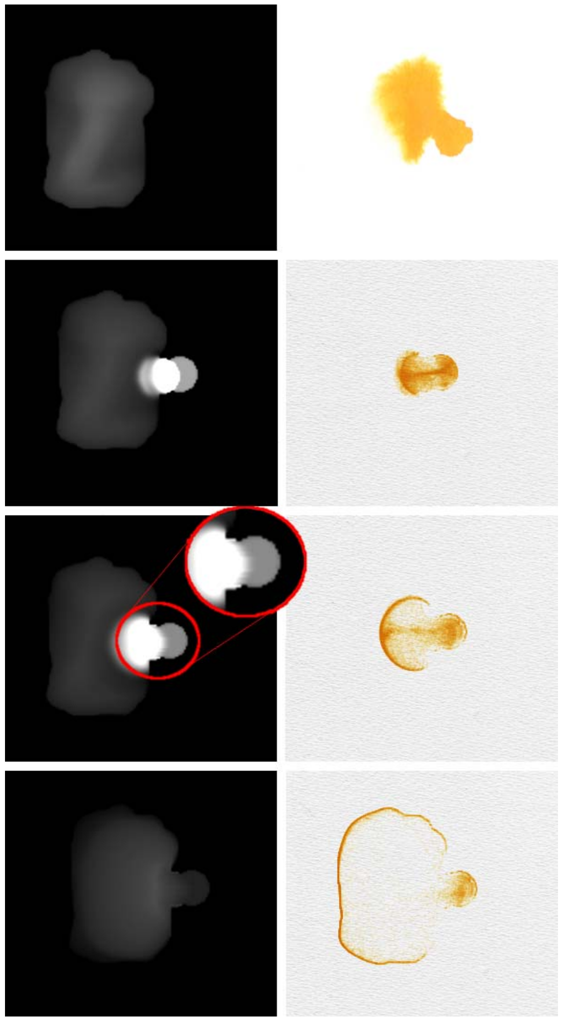
Figure 8. Series of images illustrating diffusion of a blot of Brilliant orange pigment through wet puddle. Top left: clear water puddle before pigment application, Top right: real paint diffusion process. Images on row 2-4 in left column depict fluid distribution over paper through simulation time, right column shows corresponding pigment diffusion simulation.

Acknowledgment

The authors would like to thank the unknown reviewers for their comments that help to improve the manuscript. Our project was partly supported by the Comenius University Bratislava grant.

References

[1] T. Kunii, G. Nosovskij, and T. Hayashi, "A diffusion model for computer animation of diffuse ink painting," in *Computer Animation '95., Proceedings.*, apr 1995, pp. 98 –102.

[2] C. J. Curtis, S. E. Anderson, J. E. Seims, K. W. Fleischer, and D. H. Salesin, "Computer-generated watercolor," in *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '97. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1997, pp. 421–430. [Online]. Available: http://dx.doi.org/10.1145/258734.258896

[3] S. Xu, X. Mei, W. Dong, Z. Zhang, and X. Zhang, "Interactive visual simulation of dynamic ink diffusion effects," in *Proceedings of the 10th International Conference on Virtual Reality Continuum and Its Applications in Industry*, ser. VRCAI '11. New York, NY, USA: ACM, 2011, pp. 109–116. [Online]. Available: http://doi.acm.org/10.1145/2087756.2087770

[4] F. M. P. Kubelka, "Ein beitrag zur optik der farbenstriche," *Zurich Tech. Physik*, vol. 12, p. 593, 1931.

[5] J. Onderik, M. Chládek, and R. Ďurikovič, "Interface reconstruction of multiple immiscible fluids," *Arabian Journal for Science and Engineering*, vol. 40, no. 1, pp. 269–278, 2015. [Online]. Available: http://dx.doi.org/10.1007/s13369-014-1486-8