Mar. 2006, Vol.21, No.2, pp.159–165 J. Comput. Sci. & Technol.

Real-Time Watercolor for Animation*

Thomas Luft and Oliver Deussen

Department of Computer and Information Science, University of Konstanz, Konstanz, Germany

E-mail: {luft, deussen}@inf.uni-konstanz.de

Revised December 10, 2005.

Abstract We present algorithms that allow for real-time rendering of 3D-scenes with a watercolor painting appearance. Our approach provides an appropriate simplification of the visual complexity, imitates characteristic natural effects of watercolor, and provides two essential painting techniques: the wet-on-wet and the wet-on-dry painting. We concentrate on efficient algorithms based on image space processing rather than on an exact simulation. This allows for the real-time rendering of 3D-scenes. During an animation a high frame-to-frame coherence can be achieved due to a stable segmentation scheme. Finally, we seamlessly integrate a smooth illumination into the watercolor renderings using information from the 3D-scene.

Keywords non-photorealistic rendering, watercolor painting, real-time rendering

1 Introduction

The complex nature of watercolor is characterized by soft, flowing, multifaceted, and random patterns that occur due to the motion of water and color pigments during the painting and drying processes. The interaction of a large number of natural effects make watercolor a unique drawing medium that can be exploited by a skilled artist to achieve a wide variety of artistic results. In the field of non-photorealistic computer graphics, the production of watercolor paintings is one of the most intricate and complex processes.

In contrast to physically based simulations, our aim is to create convincing watercolor renderings of 3D-scenes in real-time. We introduce several algorithms that allow the imitation of the most important drawing techniques and natural effects, which give the appearance of natural watercolor paintings (see Fig.1). The underlying 3D-scenes provide the necessary information for the creation of individual watercolor washes and for the production of a smooth lighting and shadowing. The real-time ability is achieved due to the potentials of hardware accelerated shaders.

We focus on watercolor drawing and also animation as described by the following aspects:

Simplification and Abstraction. We first create a number of abstract watercolor layers that are each based on an individual intensity image. Each intensity image contains the simplified shape of a single object or a group of objects that have a more or less uniform content or color, such as the plate and the particular fruits in Fig.1. Finally, these semi-transparent watercolor layers are composed on the screen.

Watercolor Effects. Our approach imitates significant effects of watercolor, such as the edge darkening, the flow pattern or the pigment granulation that occur due to the paper structure. This allows us to incorporate two import painting techniques: the wet-on-wet painting and the wet-on-dry painting.

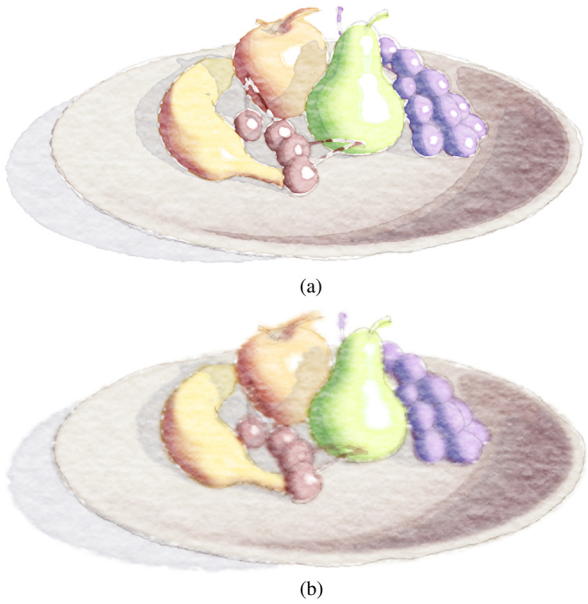

Fig.1. Real-time rendering of a watercolored still life showing different painting techniques. Slight silhouette strokes are included as a sketchy basis of this painting.

Lighting and Shadowing. We incorporate a lighting model that follows the idea of the Phong lighting. This allows us to integrate a smooth illumination that provides very dynamic results. We further describe different methods for the application of the lighting information, such as a color modulation or a masking of the watercolor layers. Additionally, we integrate shadows as an additional watercolor layer.

Real-Time Animation. Since the computation of the watercolor effects is based on an efficient image space processing rather than on an exact simulation, we easily achieve real-time results. During animation a high frame-to-frame coherence can be achieved due to a stable segmentation scheme and the subsequent image space processing.

*A preliminary version of this paper appeared in Proc. Pacific Graphics 2005, Macau.

1.1 Related Work

There are related work that explicitly treat the generation of watercolor paintings. A physically based approach is the work of Curtis *et al.*[1], which is directly related to the cellular automaton approach of Small[2]. Curtis *et al.* give a detailed description of the most important effects that occur during the watercolor painting and drying process. The key idea was to use a three-layer model that consists of a shallow-water layer, a pigment-deposition layer and a capillary layer. While the simulation of these three layers produce very convincing results, it is computationally expensive, and thus cannot be considered as real-time graphics. Another work based on this three-layer model is described by Van Laerhoven *et al.*[3] Their approach allows an interactive simulation of the watercolor painting process. Other work that simulates a virtual canvas is the IMPaSTo framework of Baxter *et al.*[4] and the work of Chu and Tai[5]. Here the user acts as the artist and draws the scene using special input devices.

Some work is produced that does not yield an exact simulation, but rather use image processing algorithms to reproduce a watercolor like finish. The work of Lum and Ma[6] was inspired by watercolor paintings. They describe a raytracing based rendering pipeline that creates procedural textures based on LIC (line integral convolution), which produce a watercolor like appearance. Here the combination of several semi-transparent color layers forms the final result. Their work allows rendering of 3D-scenes, however it does not allow real-time rendering. Burgess *et al.*[7] describe a similar system using Wyvill noise to create a brush like appearance. The work of Lei and Chang[8] is based on simple image space filters to imitate some key effects of watercolor, such as using a Sobel filter to create the edge darkening.

There are also numerous stroke based approaches, such as Strassman[9], or Hertzmann[10]. The approach of Hertzmann places strokes automatically in an iterative process on the basis of an input image. Thereby large strokes are drawn first and then smaller ones are added iteratively, in order to capture the details of the input image. The results are more similar to acrylic or oil than to watercolor painting. Other stroke based approaches that discuss Chinese watercolor paintings are those of Chu and Tai[11] or Su *et al.*[12] A milestone based on particle systems was the *painterly rendering for animation* introduced by Meier[13]. Her work combines a brush stroke like appearance with stable, frame-to-frame coherent 3D-particle sets. However, the results also resemble more acrylic or oil paintings than watercolor.

2 Imitating Watercolor Paintings

The nature of watercolor paintings is characterized by the influence of several essential effects, such as diffusion, backruns, edge darkening or pigment granulation. We concentrate on the imitation of the most significant effects by using simple algorithms instead of an exact, but nevertheless complex simulation. Furthermore, watercolor is a transparent medium, a suspension of water, binder, and color pigments, which results in a transparent and luminous impression.

Our pipeline produces a number of individual watercolor layers that are composited as semi-transparent layers on the rendering device. This procedure follows the natural process of painting with watercolor and also allows for watercolor glazing, which is an important characteristic of watercolor. Glazing describes the composition of several thin washes that are painted on top of each other after each layer has dried thoroughly. As a consequence, the color pigments are blended optically instead through a physical mixing.

In the following, we describe our approach for reproducing a watercolor appearance.

Simplification and Abstraction. The process of drawing is always an abstraction and simplification of the motif: through the use of different brushes, special techniques, and a careful selection of watercolors, the artist is able to create a convincing simplification that is still rich of information.

Following this principle, our final result is a composition of several abstract watercolor layers each containing a more or less uniform content or color, such as the branches and the foliage in Fig.2. To attain this result, the abstraction step of our pipeline begins with the segmentation of the 3D-scene on basis of unique identifiers that are assigned to objects or a group of objects. This segmentation scheme can be improved by using textures containing ID-information, which produce a higher granularity, and thus a higher level of detail. As a result of the segmentation, we obtain a number of intensity images $\rho : \boldsymbol{N}^2 \rightarrow \boldsymbol{R}$ with $\rho(x, y) \in [0, 1]$.

A low-pass filter is then applied to the intensity images using a Gaussian filter kernel. Here the kernel size represents the abstraction level of the watercolor layers. This simplification step results in abstract smooth shapes that directly describe the shape of the individual watercolor layers. Additionally, each watercolor wash is comprised of a user specified color and transparency thereby reproducing the ratio between water and color pigments. Consequently, our watercolor layers $\lambda : \boldsymbol{N}^2 \rightarrow \boldsymbol{R}^4$ are initialized with an overall color vector $\boldsymbol{c}_{rgb}$ and an overall transparency $\boldsymbol{c}_a$. In the following, we outline the imitation of the watercolor effects due to an additional modulation of the alpha channel $\lambda_a(x, y) \in [0, 1]$ of each watercolor layer.

Shape Extraction and Flow Pattern. The shape of a watercolor layer arises from the intensity values of the corresponding intensity image $\rho(x, y)$. To produce a first, hard edged shape of the watercolor layer, we threshold the intensity image by applying a step function and compute an initial transparency (see Fig.3(a)):

$$\lambda_a(x, y) = c_a \cdot step(\kappa_\rho, \rho(x, y)).$$

Here the threshold κ_ρ defines the dilation of the watercolor layer. A high threshold results in a smaller area, while a small threshold increases the color area.

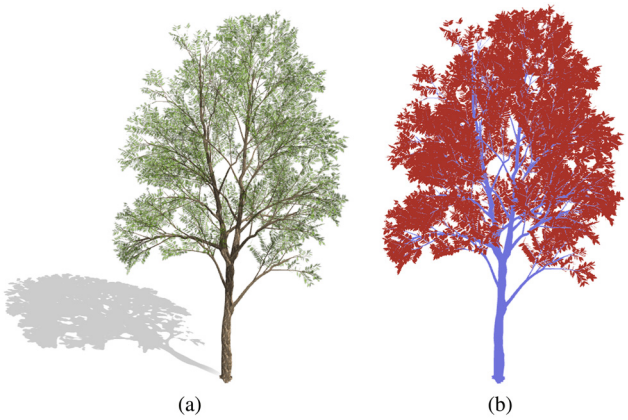

Fig.2. Abstraction and simplification. (a) Photorealistic model. (b) Resulting intensity images after an ID-based segmentation of the foliage and the branches.

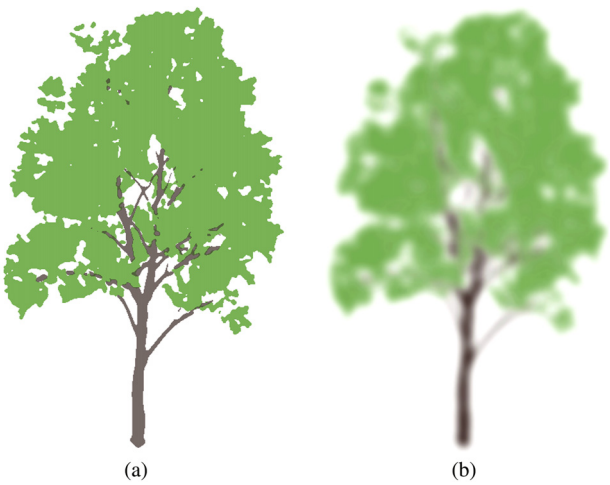

Fig.3. Shape extraction and flow pattern. (a) Shape extracted using a step function for reproducing a wet-on-dry painting, $\kappa_\rho = 0.6$. (b) Using a smooth step function to reproduce a particular flow pattern for a wet-on-wet painting, $\kappa_\delta = 0.7, \kappa_\rho = 0.3$.

The flow pattern mainly describes the pigment behavior at the border of the watercolor layer. While the wet-on-dry painting causes a hard edged border, the wet-on-wet painting causes smooth, feathery patterns at the border that may overlap with the underlying color layers and simultaneously interact with them. We imitate this interaction by providing a blurred border of the watercolor layers that smoothly fade the transparency rather than by simulating an actual diffusion of watercolor pigments. To implement such a border behavior we add another parameter κ_δ and replace the step function with a smooth step function (see Appendix):

$$\lambda_a(x, y) = c_a \cdot \Delta\, step(\kappa_\rho - \kappa_\delta, \kappa_\rho + \kappa_\delta, \rho(x, y)).$$

Here the parameter κ_δ specifies the smoothness of the border and reproduces the wetness of the background. A small κ_δ simulates a harder border, and thus a dry underground, while a large κ_δ produces a smooth color blending, and consequently imitates a wet-on-wet painting (see Fig.3(b)).

The maximum smoothness of the flow pattern is given by the filter kernel size that is used in the simplification step. As a consequence, the application of the Gaussian filter is a condition for creating a smooth flow pattern.

Edge Darkening. During the drying process, the watercolor pigments are transported towards the border due to water flow. This key effect is easily imitated using again our filtered intensity images. The applied Gaussian filter causes a smooth intensity transition following the border of the watercolor layers. To render a darkened border, we additionally modulate the alpha channel $\lambda_a(x, y)$ by the smooth intensity values:

$$\lambda_a(x, y) = \lambda_a(x, y) \cdot (1 + \kappa_\omega \cdot (1 - \rho(x, y))).$$

Here the user can specify the intensity of the edge darkening by changing the attribute κ_ω, which drastically influences the final result (see Fig.4). This attribute equals a particular quality, which usually is determined by the manufacturer of real watercolors.

Fig.4. Edge darkening in combination with a wet-on-dry painting, $\kappa_\omega = 0.8$.

Our imitation of the edge darkening is not an edge filter as utilized by Lei and Chang[8], it rather produces a gradient following the watercolor flow and fading the transparency at the border. Here the Gaussian filtering of the intensity images is again necessary, since it defines the width of the edge darkening.

Pigment Granulation. The underlying paper structure with its valleys and peaks affects the water flowing process significantly and introduces a visible pigment granulation. To imitate this process, we additionally modify the watercolor layer's alpha channel according to an intensity texture $T : \boldsymbol{N}^2 \rightarrow \boldsymbol{R}$ with

$T(x,y) \in [-1,1]$ representing the paper structure (see Fig.5(a)):

$$\lambda_a(x,y) = \lambda_a(x,y) + T(x,y) \cdot \kappa_\tau.$$

Since the paper structure fundamentally influences the watercolor flow, it also changes the shape of the watercolor layer's border. This effect is imitated by modulating the intensity images before the shape extraction using the paper texture:

$$\rho(x,y) = \rho(x,y) + T(x,y) \cdot \kappa_\theta.$$

By using an appropriate texture, this technique can also be utilized to imitate a certain looseness intended by the artist (see Fig.5(b)). The overall intensity of the paper structure and its influence on the border of the watercolor layers is specified by the parameters κ_τ and κ_θ.

A drawback is the spatial texture parameterization of T that causes shower door effects during an animation. This unfavourable effect is addressed in [14, 15]. Using their methods the artefacts can be minimized, however the result is an unusual and unnatural vision.

Fig.5. Pigment granulation. (a) Transparency modulated by the paper structure, $\kappa_\tau = 0.4$. (b) Paper structure modulating the intensity images producing jittered borders, $\kappa_\theta = 0.9$.

Another problem is the interdependency between effect parameters. Here special constraints are useful to avoid a distortion of the results, e.g., the influence of the paper structure during the shape extraction could be constrained by ensuring that $\kappa_\rho - \kappa_\delta > \kappa_\theta$. One solution for this constraint that we have implemented, is the restriction of the border modulation: the intensity image is only modulated in regions containing a gradient due to the low-pass filtering or more formally: $\kappa_\theta \cdot (1 - |2\rho(x,y) - 1|)$. Additionally, this method weights the paper influence depending on the intensity.

The watercolor effects are continuously adjustable, which provides a variety of different styles. Another advantageous effect is the automatic and continuous level of detail that is achieved after applying the low-pass filter for abstraction: Details in which the dimensions drop below the filter kernel size are smoothly faded out.

To increase the performance, we use a combination of an Gaussian filter kernel and a slightly reduced size for the intensity images. Using such low resolution intensity images and scaling them up using a bilinear filter, work similar to a low-pass filter. However, the results have a lesser quality and show temporal artefacts, namely a wobbling of the watercolor shapes during an animation. Consequently, we have to compromise between visual quality and performance.

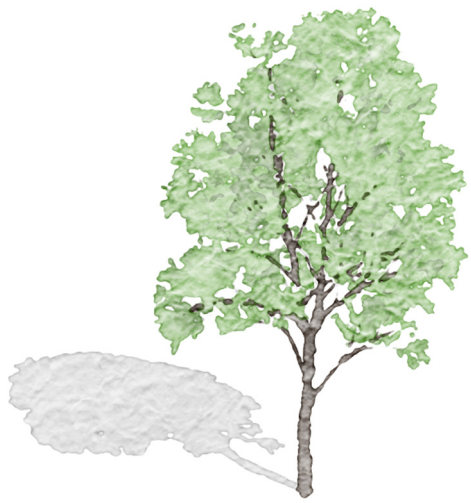

Fig.6. Watercolor effects: Plain watercolor layers are created using a combination of the imitated watercolor effects.

3 Composition of Watercolor Layers

This section describes how an appropriate scene illumination is integrated, how additional watercolor layers are created due to the lighting and shadowing, and how the resulting watercolor layers are finally composed.

3.1 Lighting and Shadowing

The reproduction of a soft and luminous illumination is one of the dominant features of watercolor. This section describes methods that are inspired by painting techniques used to illustrate shading and lighting of a scene.

So far, we have incorporated information from the 3D-scene regarding the shape of the watercolor layers in combination with a static color. However, the applied segmentation scheme that is based on unique identifiers, discards all surface details that occur due to the lighting and shadowing (see Fig.6). As a consequence, we incorporate the lighting information of the 3D-scene to modulate the existing watercolor layers and also to create additional layers.

Lighting. Our lighting is based on the Phong model, which consists of an ambient, a diffuse, and a specular terms. Since the ambient term is constant

and already given by the initial color of the layers, we only consider the diffuse and the specular term. For optimization, these two terms are normalized to $[0, 1]$ and each rendered into an individual lighting map $L_d, L_s : \boldsymbol{N}^2 \rightarrow \boldsymbol{R}$. The resulting lighting maps are then utilized for processing all the watercolor layers.

The specular term reproduces a reflection of light resulting in a highlighted region. Since watercolor is a transparent medium, such regions usually have to be planned at the beginning of the painting process by leaving these areas blank. As a consequence, an object featuring a specular material is created by masking the associated intensity image using the specular lighting map before creating the watercolor layer (see Figs. 7(c), 7(d)):

$$\rho(x, y) = \rho(x, y) \cdot (1 - step(\kappa_s, L_s(x, y))).$$

The parameter κ_s defines the dilation and thus the size of the resulting highlighted area.

Analogous, the diffuse term can be used to generate additional watercolor layers by thresholding the diffuse lighting map and using it as a mask for the intensity images. In this way, especially dark areas can be emphasized by additional, slightly darker watercolor layers (see Fig.7(c)). In combination with a wet-on-dry painting, these watercolor layers reproduce the glazing effect, while the wet-on-wet painting technique results in soft color gradients.

Finally, the diffuse term is used for color modulations of all existing watercolor layers. Therefore, the user specifies two colors for each watercolor layer: one for shaded regions respectively $L_d(x, y) = 0$ and the other for lighted regions respectively $L_d(x, y) = 1$. The color of the watercolor layer is then interpolated using the diffuse lighting map (see Fig.7(b)). This procedure is inspired by the wet-on-wet painting technique in which the artist uses different colors to create a smooth color gradient or a smooth shading due to a pigment diffusion.

Before utilizing the diffuse lighting map, we apply a low-pass filter to avoid untypical sharp edges in the resulting color. Furthermore, this step provides a more abstract and simplified lighting condition. A side effect of this procedure arises at the watercolor layer's border, since here the diffuse lighting term is blended with some *background lighting*. We receive satisfactory results, when initializing the diffuse lighting map with $L_d = 1$ as background lighting before rendering it, resulting in slightly brightened borders in darker regions.

Shadows. Shadows provide another strong depth cue in addition to the existing lighting conditions. We integrate shadows by computing another watercolor layer representing only shadowed regions.

Therefore, we render another lighting map containing the shadow intensity. This so called shadow intensity map is computed using a standard shadow map algorithm providing shadowed and non-shadowed areas. Subsequently, we apply all steps of the watercolor imitation equivalently to the already processed intensity images. Thus, we first apply a low-pass filter to simplify the shadow watercolor layer and then compute all watercolor effects following. The resulting shadow watercolor layer is also modulated by the diffuse lighting term, which further increases the dynamics of the scene. Consequently, the user again defines two different colors for darker and lighter shadow regions.

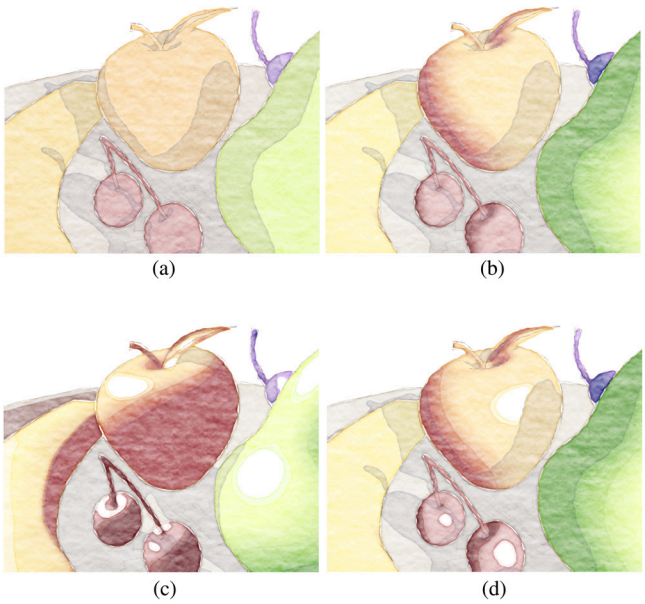

Fig.7. Lighting. (a) Initial ambient lighting. (b) Color modulation by the diffuse term imitates a smooth gradient due to a wet-on-wet painting technique. (c) and (d) Additional watercolor layers by masking the intensity images with the thresholded diffuse term and highlighted regions by masking the intensity images with the specular term. The shadow watercolor layer is additionally included in all images.

3.2 Composition

The individual watercolor layers are finally composed on the screen using the standard blending function for transparent objects. The resulting color R_{rgb} is computed by the color C_{rgb} and transparency C_a of the current watercolor layer and the background color B_{rgb} (at each position (x, y)):

$$R_{rgb} = C_a \cdot C_{rgb} + (1 - C_a) \cdot B_{rgb}.$$

This procedure yields satisfying results, since our approach is not simulation based and all colors are specified by the user. However, another popular approach is the use of the Kubelka-Munk theory[16], which is especially designed for imitating natural color compositions. Here the optical composition is based on individual absorption and scattering coefficients for the color layers.

4 Results

We tested our implementation on a 3.0GHz Pentium IV CPU with a GeForce 6800 graphics board. We use

a multi-pass rendering procedure to render the lighting maps, the intensity images, and the resulting watercolor layers. The described watercolor effects are completely implemented using a fragment shader.

In Table 1 an overview of the scene complexity and the rendering times are given: $\#tri$ is the number of triangles of a scene, $\#\lambda(x,y)$ is the number of watercolor layers, $\rho(x,y)$ gives the computation time to create the intensity images including the lighting maps, $\lambda(x,y)$ gives the rendering time to create the watercolor layers, and *fps* gives the average number of frames per second that we achieve during an animation. All images are rendered at a resolution of 720×720 pixel. The intensity images and the lighting maps are rendered at a resolution of 720×720 for Figs. 8(d) and 8(f), and at a resolution of 370×370 for Figs. 8(a) and 8(e).

Table 1. Comparison of the Scene Complexity and the Rendering Times

Figs.	$\#tri$	$\#\lambda(x,y)$	$\rho(x,y)$ (ms)	$\lambda(x,y)$ (ms)	*fps*
8(a)	67040	5	38.2	8.5	21.4
8(d)	301230	6	65.6	9.3	13.3
8(e)	62592	19	30.3	16.5	21.3
8(f)	55962	11	49.9	12.2	16.1

Although, our approach is completely different from a physically based simulation of watercolor, the results show a convincing imitation of natural watercolor paintings. Naturally, there are still significant differences in comparison to real watercolor paintings due to a missing pigment diffusion that produces spontaneous and subtle textures.

Our approach is not a pure image filter, since we need information that is contained in the given 3D-scene, such as the lighting or the segmentation. Consequently, it cannot be automatically applied to a photography. This would require an alternative separation method, e.g., a mean shift algorithm, and also a completely different color management. On the other hand we obtain 3D-information that allow to create a very dynamic illumination of the scene, although only a Phong like lighting model is used.

The video material (please, visit our web page[17]) show that a high frame-to-frame coherence can be obtained due to the stable ID-based segmentation and the subsequent image space processing.

5 Conclusion

Our algorithms can be used to create real-time graphics of 3D-scenes with a convincing watercolor appearance. We concentrated on image space processing to imitate significant effects of watercolor, such as the edge darkening, the flow pattern or the pigment granulation due to the paper structure. This allowed us to incorporate two import painting techniques: the wet-on-wet painting and the wet-on-dry painting. Nevertheless, our approach is not a simple image space filter, but rather incorporates information of the underlying 3D-scene to imitate a natural painting process with individual watercolor layers and to incorporate a smooth lighting and shadowing of the scene.

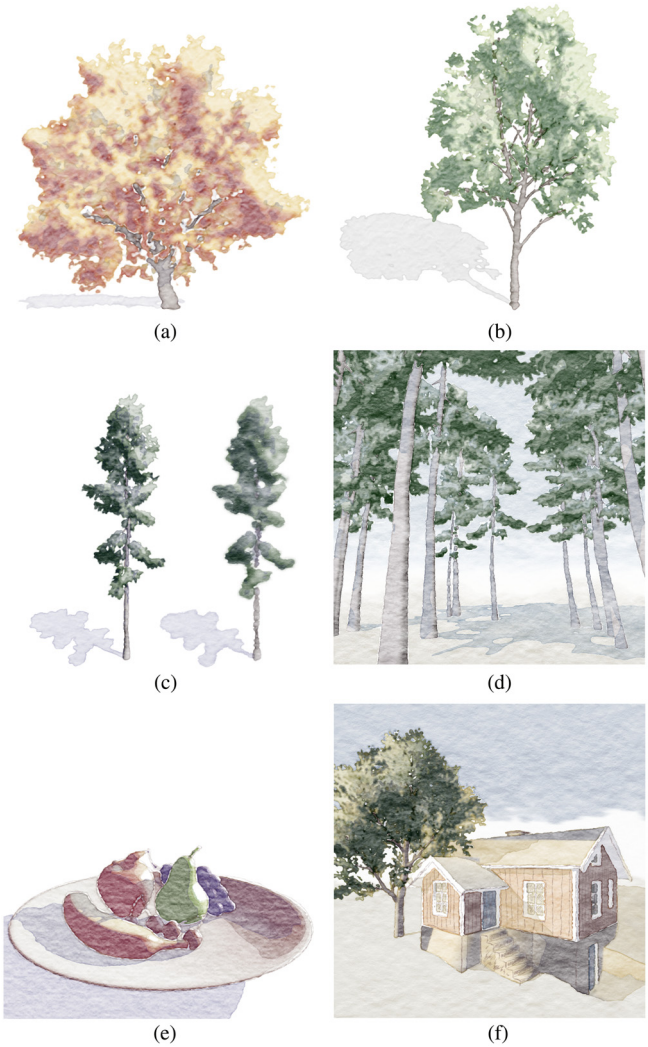

Fig.8. Results: Example scenes showing different effect settings. All images are rendered in real-time.

Future work aims at improving visual quality by treating the wobble effects that may occur during an animation caused by a too low resolution of the intensity images. Here a bicubic instead of a bilinear interpolation for accessing the images could be helpful.

Another subject is to integrate additional depth cues that are based on the depth information itself. It is an often performed painting technique using a wet-on-wet painting for the background, and a dry-on-wet painting for the more detailed foreground. Here, the continuous adjustability of the effect parameters is essential.

References

[1] Cassidy J Curtis, Sean E Anderson, Joshua E Seims *et al.* Computer-generated watercolor. In *Proc. ACM SIGGRAPH*, Los Angeles, US, Aug. 1997, pp.421–430.

[2] David Small. Simulating watercolor by modeling diffusion, pigment, and paper fibers. In *Proc. SPIE '91: Image Handling and Reproduction Systems Integration*, Aug. 1991, volume 1460, pp.140–146.

[3] Tom van Laerhoven, Jori Liesenborgs, Frank van Reeth. Real-time watercolor painting on a distributed paper model. In *Proc. Computer Graphics International*, Crete, Greece, June 2004, pp.640–643.

[4] William V Baxter, Jeremy Wendt, Ming C Lin. IMPaSTo: A realistic model for paint. In *Proc. NPAR*, Annecy, France, June 2004, pp.45–56.

[5] Nelson S-H Chu, Chiew-Lan Tai. MoXi: Real-time ink dispersion in absorbent paper. In *ACM Trans. on Graphics*, 24(3): 504–511.

[6] Eric B Lum, Kwan-Liu Ma. Non-photorealistic rendering using watercolor inspired textures and illumination. In *Proc. Pacific Graphics*, Tokyo, Japan, Oct. 2001, pp.322–330.

[7] Jeremy Burgess, Geoff Wyvill, Scott A King. A system for real-time watercolour rendering. In *Proc. Computer Graphics International*, New York, US, June 2005, pp.234–240.

[8] Su Ian Eugene Lei, Chun-Fa Chang. Real-time rendering of watercolor effects for virtual environments. In *Proc. IEEE Pacific-Rim Conference on Multimedia*, Tokyo, Japan, Nov. 2004, pp.474–481.

[9] Steve Strassmann. Hairy brushes. In *Proc. ACM SIGGRAPH*, Dallas, US, Aug. 1986, pp.225–232.

[10] Aaron Hertzmann. Painterly rendering with curved brush strokes of multiple sizes. In *Proc. ACM SIGGRAPH*, Orlando, US, July 1998, pp.453–460.

[11] Nelson S-H Chu, Chiew-Lan Tai. An efficient brush model for physically-based 3D painting. In *Proc. Pacific Graphics*, Beijing, China, Oct. 2002, pp.413–421.

[12] Sara L Su, Ying-Qing Xu, Heung-Yeung Shum, Falai Chen. Simulating artistic brushstrokes using interval splines. In *Proc. Computer Graphics and Image*, Kauai, Hawaii, Aug. 2002, pp.85–90.

[13] Barbara J Meier. Painterly rendering for animation. In *Proc. ACM SIGGRAPH*, New Orleans, US, Aug. 1996, pp.477–484.

[14] Matthieu Cunzi, Joëlle Thollot, Sylvain Paris *et al.* Dynamic canvas for immersive non-photorealistic walkthroughs. In *Proc. Graphics Interface*, Halifax, Canada, June 2003, pp.121–130.

[15] Matthew Kaplan, Elaine Cohen. A generative model for dynamic canvas motion. In *Proc. the Eurographics Workshop on Computational Aesthetics*, Girona, Spain, May 2005, pp.49–56.

[16] Chet S Haase, Gary W Meyer. Modeling pigmented materials for realistic image synthesis. In *ACM Trans. Graphics*, 11(4): 305–335.

[17] Website Work Group Computer Graphics and Media Design, University of Konstanz. http://graphics.uni-konstanz.de/research/research_en.html.

Thomas Luft graduated in 1997 from Dresden University of Technology. Currently, he works as a Ph.D. student under the supervision of Prof. Oliver Deussen. His research interests encompass non-photorealistic rendering and real-time graphics.

Oliver Deussen graduated in 1991 from Karlsruhe University of Technology where he also received his Ph.D. degree in 1996. He worked as a postdoc researcher from 1996 until 2000 at Otto-von-Guericke University of Magdeburg and was associate professor for computer graphics at the Dresden University of Technology from 2000 until 2003. Since then he is a full professor for computer graphics and media informatics at the University of Konstanz. His research interests encompass non-photorealistic rendering, modeling and rendering of complex botanic objects as well as information visualization.

Appendix. Smooth Step Function

The smooth step function used in our approach is based on a Hermite interpolation. It is specified by three input values: an upper bound u, a lower bound l, and the value x. The first step is a linear interpolation of x in the interval $[l, u]$:

$$t = \frac{x - l}{u - l}.$$

Then t is clamped to $[0, 1]$ and a cubic Hermite interpolation is applied:

$$\Delta step(l, u, x) = 3 \cdot t^2 - 2 \cdot t^3.$$

This corresponds to the standard implementation of the smooth step function that is provided by the OpenGL Shading Language.