Edge- and substrate-based effects for watercolor stylization

S E. Montesdeoca
Nanyang Technological University
Interdisciplinary Graduate School, MAGIC

H. S. Seah
Nanyang Technological University, School of Computer Science and Engineering

P. Bénard
Univ. Bordeaux, CNRS, LaBRI
Inria Bordeaux Sud-Ouest

R. Vergne & J. Thollot
Univ. Grenoble Alpes, CNRS, LJK
Inria Grenoble Rhône-Alpes

H.-M. Rall & D. Benvenuti
Nanyang Technological University
School of Art, Design and Media

Figure 1: Our methods allow new and improved edge- and substrate-based effects for watercolor stylization: edge darkening (red), gaps (blue), overlaps (green) and dry-brush (yellow). Still Life, model by Dylan Sisson © Pixar Animation Studios.

ABSTRACT

We investigate characteristic edge- and substrate-based effects for watercolor stylization. These two fundamental elements of painted art play a significant role in traditional watercolors and highly influence the pigment's behavior and application. Yet a detailed consideration of these specific elements for the stylization of 3D scenes has not been attempted before. Through this investigation, we contribute to the field by presenting ways to emulate two novel effects: dry-brush and gaps & overlaps. By doing so, we also found ways to improve upon well-studied watercolor effects such as edge-darkening and substrate granulation. Finally, we integrated controllable external lighting influences over the watercolorized result, together with other previously researched watercolor effects. These effects are combined through a direct stylization pipeline to produce sophisticated watercolor imagery, which retains spatial coherence in object-space and is locally controllable in real-time.

NPAR'17, July 29–30, 2017, Los Angeles, CA

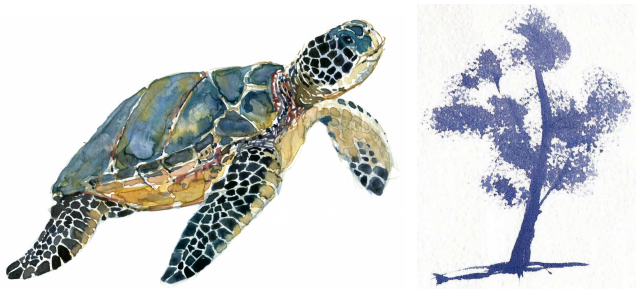
KEYWORDS

watercolor, dry-brush, gaps & overlaps, direct stylization

ACM Reference format:
S. E. Montesdeoca, H. S. Seah, P. Bénard, R. Vergne, J. Thollot, H.-M. Rall and D. Benvenuti. 2017. Edge- and substrate-based effects for watercolor stylization. In *Proceedings of Expressive, Los Angeles, CA, July 29–30, 2017 (NPAR'17)*, 10 pages.
https://doi.org/10.1145/3092919.3092928

1 INTRODUCTION

Watercolors have been subject of extensive research in computer graphics since the beginning of the 1990s, when super computers were used to physically approximate the behavior of the traditional medium [Small 1991]. Thanks to an exponential growth in computing resources and the adoption of programmable graphics hardware, research in watercolor has advanced significantly in finding different ways to reproduce various characteristic watercolor effects. Recent advancements by Montesdeoca et al. [2016; 2017] have extended previous work for watercolor stylization in object-space and contributed to localized art-direction by implementing a direct stylization pipeline in a common digital content creation software. However, several watercolor effects are still missing or need to be improved, for the simulated medium to mature in the computer graphics industry.



(a) Sea Turtle in Watercolor © Frits Ahlefeldt, 2014.

(b) Dry-brush Tree © Joe Cartwright, 2011.

Figure 2: Traditional watercolor paintings.

Among them, this paper addresses edge- and substrate-based (i.e., paper) effects, tackling previously unexplored characteristic watercolor effects in object-space – such as *dry-brush* application and *gaps & overlaps* – and improving upon other existing effects within these two groups. The edges in a painting and the substrate profile are widely used to influence essential artistic techniques that are readily available to the traditional watercolor painter. For instance, the sophistication of the color palette is increased by combining colors at different levels of transparency at the edges through overlaps. Conversely, introducing elements of striking luminance by revealing the substrate at edges through gaps, adds further options for artistic refinements. Edge darkening serves as a reminder of familiar organic qualities of traditional materials and add tonal contrast for the definition of forms. These effects are commonly observable in traditional watercolors (Figure 2) and portrayed in the literature [Brown 2007; Soan 2014]. Applying a combination of edge-based techniques to excellent artistic effects is demonstrated in the 2014 painting by Frits Ahlefeldt depicted in Figure 2a. A dry-brush application of pigments is another technique that is often used in watercolor painting [Franks 1988], but it also finds wide use beyond watercolor, e.g., in comic-book inking [Janson 2003, 15, 104, 107] and fashion illustration [Morris 2010, 61]. An excellent use of a dry-brush application in watercolors is demonstrated in the 2011 painting by Joe Cartwright depicted in Figure 2b. In essence, the dry-brush technique consists in lightly applying a more viscous concentration of watercolors over the rough substrate (emulating its oil counterpart), to achieve a textured appearance.

In computer graphics (CG), a dry-brush application effect has never been attempted before in object-space, whereas gaps & overlaps have been superficially explored by Luft and Deussen [2006a]. These effects present themselves as special challenges, because of their reliance on the physical substrate and the relevant edges that a watercolorist would stylize. To replicate these effects, we take advantage of accurately acquired substrates, RGBD (Red, Green, Blue and linear Depth) edge detection and a direct stylization framework. In this pursuit, we have also enabled the possibility to control external lighting influences on top of the watercolor imagery and found ways to improve further substrate-based effects like granulation. The body of work presented in this paper supports both the 3D scene geometry and the use of texture mapping, which the majority of CG work nowadays relies on.

The effects developed in this paper, together with other low-level localized characteristic watercolor stylization effects, are combined to create believable watercolorized imagery from object-space data, as illustrated in Figure 1. Our algorithms are designed and optimized to maintain real-time performances, as creative interaction is crucial for artistic design choices.

In the remaining of this paper, related work is introduced in Section 2, the proposed edge-based watercolor effects are presented in Section 3, which is followed by substrate-based effects in Section 4 and implementation details in Section 5. The results and discussion of their combination are presented in Section 6. This is followed by the conclusion and future work in watercolor stylization.

2 RELATED WORK

Synthesizing images with a watercolor appearance has been widely explored by the computer graphics community. The common goal of all previous approaches is to reproduce the result of the interaction between water, pigments and the substrate on which the paint is deposited. This interaction produces a number of visual effects, which artists take advantage of and integrate into the overall aesthetic of their artwork. These characteristic effects distinguish watercolors from other natural media and are commonly addressed in the literature under the following terms: color bleeding (wet-in-wet technique), dry brush (dry-on-dry technique), distortions (fingering), edge darkening, pigment turbulences, backruns and gaps & overlaps.

To replicate (usually a subset of) those effects, a large variety of approaches have been proposed, going from simple image filters [Bousseau et al. 2006; Johan et al. 2004] to physical simulations [Curtis et al. 1997; Small 1991]. In this paper we focus on the interactive stylization of 3D animations, for which an artist cannot draw each image manually neither with traditional techniques nor using physical [Chu and Tai 2005; Van Laerhoven and Van Reeth 2005; You et al. 2013] or procedural [DiVerdi et al. 2013; Lu et al. 2013] simulations. In this context, two families of approaches have been proposed: those working in image-space and those working in object-space.

2.1 Image-space methods

Image-space approaches mimic the visual effects that are specific to watercolor by combining several filters and textures that are applied onto the whole image. Most of the explored techniques have been devised to stylize a single image [Johan et al. 2004; Lü and Chen 2014; Wang et al. 2014], but they can still be applied to animations by independently filtering each frame of a video. Low-level art-directed localization of these image-space methods has been successfully incorporated by Semmo et al. [2016] using parameter masks. However, since those masks are assigned in image-space, the stylization suffers from spatial and temporal incoherences when in motion.

Bousseau et al. [2007] solve part of the latter problem using texture advection to make composited textures change according to the optical flow of the video. They also introduce space-time morphological filters to coherently simplify the input animation. This approach produces convincing results for noise-like textures, but it is limited in the amount of effects that it can reproduce. It also requires the knowledge of the full animation and does not offer any local control to the artist. In the spirit of previous work on

painterly rendering [Collomosse et al. 2005; Kagaya et al. 2011], spatio-temporal segmentation could allow such controls, but it would still not be applicable for interactive applications, such as games and virtual reality, where the point of view is not predefined.

Based on the Image Analogy framework [Hertzmann et al. 2001], several methods propose to reproduce watercolor effects from exemplars and make them evolve in time. Fišer et al. [2016] describe a method to turn a photorealistically-rendered 3D model into a painted image. Taking advantage of global illumination information, their method produces believable interactive results for static imagery, but it has artifacts and temporal coherence issues when the 3D scene is animated. Bénard et al. [2013] also extend Image Analogies but they allow interpolation between painted keyframes, by using velocity and orientation fields rendered from object-space data. This is probably the only image-space system that could embed watercolor stylization with spatial and temporal art-direction. However, textures are required to be painted every few frames, making this approach unsuitable for interactive applications.

Finally, neural networks have been used to stylize images [Gatys et al. 2015] and videos [Chen et al. 2017; Selim et al. 2016], but they are not controllable by an artist.

2.2 Object-space methods

In object-space, each 3D object can be individually stylized, and the effects are directly connected to the object surface. This ensures perfect spatial coherence with the object motion when animated. In addition, any image-space post-process can be added at the end of the rendering pipeline to extend the range of possible effects.

Lum and Ma [2001] proposed a first attempt to procedurally generate wash textures by performing line integral convolution of Perlin noise along a 3D object curvature. Burgess et al. [2005] extend this approach to take advantage of the GPU and to incorporate darkened edges. Lei and Chang [2005] describe a similar GPU pipeline that produces darkened edges, granulation and distortion as a post-process in image-space. While generating some appealing results, these methods are still not reproducing several key watercolor effects and do not provide local control to the user.

To improve on the range of replicated watercolor effects, Luft and Deussen [2006a; 2006b] use object IDs to decompose a 3D scene into layers that are rendered during multiple passes. The layers are processed by image-space filters, whose parameters can be independently art-directed, and later recomposed into a final image. The system also improves on edge darkening effects and produces gaps & overlaps between layers, since they are separately distorted. However, this approach does not scale well with scene complexity and still misses some fundamental characteristic effects like pigment turbulence.

Luft et al. [2008] implement watercolor stylization into a CAD system, proposing a tone-based lighting model and stylistic means of abstraction based on ambient occlusion. Even though this method only allows a very specific look to be achieved, integrating such a watercolor rendering engine into a full modeling solution provides more controls to the user than previous automatic approaches. Along those lines, Montesdeoca et al. [2016; 2017] integrate a full real-time watercolor pipeline into Autodesk Maya®. They further increase art-directability by providing a 3D painting interface that allows the user to draw various parameters directly on the object surface. Those are then rendered into 2D buffers and serve as localized controls for 2D and 3D watercolor effects. They incorporate the object-space control to previously studied effects, improving upon them along the way and introducing hybrid-space color bleeding. However, this versatile system does not allow to emulate gaps & overlaps and dry-brush.

In this paper, we focus on investigating edge- and substrate-based effects of watercolors that have barely been explored in object-space before. By doing so, we also found ways of improving existing effects, while preserving the key feature of local direct artistic control – which is highly beneficial in expressive rendering. With these newly developed and improved effects, we can extend the palette of characteristic effects for 3D artists and watercolorists.

3 EDGE-BASED EFFECTS

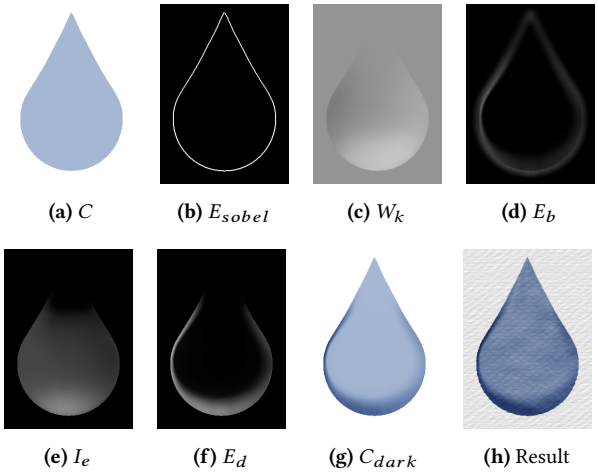
In any type of imagery, edges are imperative as they delineate shape, define form and create contrast. Because of these properties, edges are used in many computer science applications (e.g., image processing, computer vision and image synthesis). Edge-based effects are common in most natural painting media and are widely used for stylization purposes. However, in watercolor imagery, edges present characteristic phenomena due to the fluidity of the medium. This paper focuses on two edge-based effects commonly found in watercolor paintings: *edge darkening* and *gaps & overlaps*.

To be able to perform any edge-based stylization, edges first need to be detected in the image. Many algorithms are available [Papari and Petkov 2011], but the most common and computationally efficient ones involve the use of filters to compute local differentials such as the Sobel, Canny or difference of Gaussians (DoG) filters. However, applying them to regular RGB images only detects discontinuities in the color gradients. They miss edges in 3D space that share similar tonalities – but are perceived through our stereo vision. To detect them, we run a Sobel filter on an RGBD (Red, Green, Blue and Linear Depth) buffer to generate meaningful edges [Nienhaus and Döllner 2005; Saito and Takahashi 1990]. We further control the contribution of the depth channel by a parameter, increasing its influence on the edge detection and generating more uniform edge intensities, regardless of their color tonalities. Once the edges have been extracted, they are used to recreate edge darkening and gaps & overlaps.

3.1 Edge darkening

Edge darkening, which refers to the gradual pigment accumulation towards the edges of painted areas due to surface tension, has been widely studied in previous work [Bousseau et al. 2006; Burgess et al. 2005; Lei and Chang 2005; Luft and Deussen 2006a; Montesdeoca et al. 2017]. Yet, our approach offers necessary improvements by taking advantage of the previously mentioned RGBD edge detection and offering individual local control over the darkened edge width and intensity.

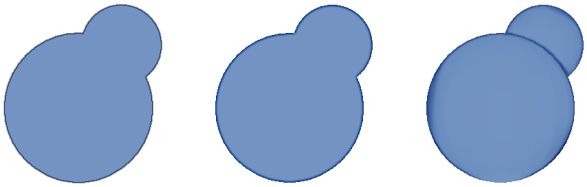
Through the RGBD edge detection, we can also detect and darken edges that share similar tonalities. This enables a more uniform and coherent edge darkening even under motion, when neighboring colors change. We then proceed to use these edges to create edges that emulate gradient pigment accumulation.



(a) C **(b)** E_{sobel} **(c)** W_k **(d)** E_b

(e) I_e **(f)** E_d **(g)** C_{dark} **(h)** Result

Figure 3: Breakdown of an art-directed edge darkening.



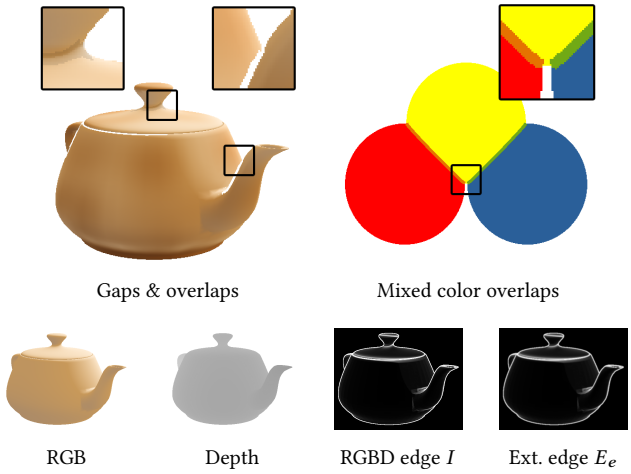
[Lei and Chang 2005] [Montesdeoca et al. 2016] Our approach

Figure 4: Different approaches towards edge darkening.

Although the Sobel filter produces sharp edges (Figure 3b), we generate gradient edge darkening by subsequently blurring them. The convolution is controlled and performed through a separable Gaussian blur kernel, which is established at each individual pixel by an edge width parameter W_k, painted on the surface of the object (Figure 3c). The resulting blurred edges ($E_b = G_{W_k} * E_{sobel}$, Figure 3d) are then amplified by a global edge darkening value and painted edge intensity parameters I_e (Figure 3e). This operation results in the final edge density E_d (Figure 3f – actual density reduced for illustration purposes) to darken the color through the color modification model $C_{dark} = C^{1+E_d}$, where C is the original color (Figure 3a). The edge darkening contribution on the final watercolorized result can be seen in Figure 3h. The resulting edge darkening effect is fully controllable and can be coherently art-directed by an artist for any viewpoint, taking advantage of the object-space painted parameters seen in Figures 3c and 3e. The versatility of our edge darkening approach can be seen and compared in Figure 4. In this comparison, previous approaches only provide a uniform width and struggle to darken edges with similar colors.

3.2 Gaps & overlaps

Gaps & overlaps are characteristic effects of watercolor that have barely been studied in object-space. Luft and Deussen [2006a] explored the effect by rendering each group of elements in individual layers, distorting these independently and compositing them together afterwards. In this way, the distorted layers may partially overlap or show gaps between them. The biggest limitation in their approach is that objects within a common layer will not present any gaps & overlaps. Additionally, relying solely on image-space computations to emulate this effect introduces spatial and temporal incoherences under animation. These problems are also present in pure image-space approaches such as the technique of Wang et al. [2014].



Gaps & overlaps Mixed color overlaps

RGB Depth RGBD edge I Ext. edge E_e

Figure 5: Results of gaps & overlaps in object-space and a breakdown of the edge extraction and edge extension.

Gaps & overlaps naturally happen at edges of watercolor illustrations, when artists loosely paint adjacent areas. Gaps are generally produced when the artist does not want the colors of neighboring regions to touch each other, either because they are both wet and might bleed into one another, or for aesthetic preferences. Conversely, overlaps happen when the artist does not mind these colored areas from slightly overlapping each other, mostly for aesthetic reasons or accidental deviations when painting. Small gaps & overlaps may also appear as a natural byproduct of hand-tremors, which are involuntary reflexes in the human nervous system. To reproduce these two effects, we propose to distort the rendered image at the vicinity of its edges either making the substrate appear between two adjacent regions, or picking and blending neighboring colors. This process is guided by parameters painted by the user on the surface of the stylized meshes. In addition, to generate gaps & overlaps from hand tremors, a procedural object-space tremor value can offset these painted parameters.

As gaps & overlaps are located on edges, these effects and their spatial coherence highly depend on a robust edge detection. The previously detected edges have a general width of two pixels – one pixel on both side of the color or depth discontinuity. Since gaps & overlaps may be wider than this, we extend the detected edges to a maximum constant width m using a linear filtering kernel ($m = 5$ in our experiments). Marching along the gradient of those linear edges, we find the adjacent colors for either overlapping colors, or generating gaps. Depending on the linear edge values, some gradients might not converge towards neighboring contrasting pixels. Fortunately, these cases are found at immediate edge boundaries, which can be identified with the original un-widened edge boundaries. This algorithm is described with pseudo-code in Algorithm 1 and the results presented in Figures 5 and 6.

Algorithm 1 Gaps & overlaps pseudo-code.

Input: RGBD image I, extended edge image E_e, edge image E, substrate color C_s, gaps & overlaps parameter $p \in [-m, m]$
Output: Gaps & overlaps image I_{go}

```
for all pixel I(x, y) do
    I_go(x, y) = I(x, y)
    // Check if not substrate color
    if (I(x, y) ≠ C_s) then
        // Check if in extended edge
        if (E_e(x, y) > 0.2) then
            // Get gradient by fetching neighbor pixels
            G = normalize(Δ_u, Δ_v)
            // OVERLAPS
            if (p > 0) then
                // Check up to m pixels along G
                for i ∈ [1 : m] do
                    // Check if enough overlap param.
                    if (p < i) then break
                    // Get neighboring color
                    C_n = P(x + iG_x, y + iG_y)
                    // Check for difference in RGBD space
                    if (‖C_n − I_go(x, y))‖ > 0.5) then
                        // If not substrate, mix
                        if (C_n ≠ C_s) then
                            // Mix colors in RYB space
                            I_go(x, y) = mixRYB(I_go(x, y), C_n)
                        break
                // Loop reached max → direct edge
                if (i == m) then
                    // Gradient didn't converge to another color
                    // I(x,y) at E edge → negate G
                    C_n = P(x − G_x, y − G_y)
                    I_go(x, y) = mixRYB(I_go(x, y), C_n)
            // GAPS
            if (p < 0) then
                // Check if direct edge
                if (E(x, y) > 0.7) then
                    I_go(x, y) = C_s
                else
                    // Check up to m pixels along G
                    for i ∈ [1 : m] do
                        //Check if enough gap param.
                        if (|p| < i) then break
                        // Get neighboring color
                        C_n = P(x + iG_x, y + iG_y)
                        // Check for difference in RGBD space
                        if (‖C_n − I_go)‖ > 0.5) then
                            // Assign substrate color
                            I_go = C_n
                            break
```

Note: Transparent gaps are obtained by zeroing the output color alpha channel instead of using the substrate color.

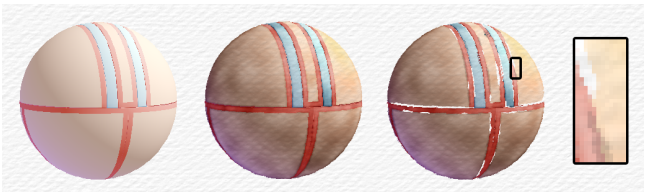


Figure 6: Gaps & overlaps to achieve a sketchier watercolor look. From left to right: original colors, result without and with gaps & overlaps.

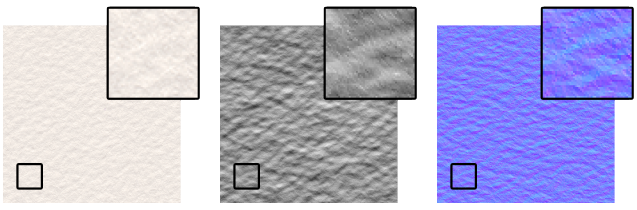
To enhance the overlapping effect, the mixing of the colors is done in RYB (Red, Yellow, Blue) space [Gossett and Chen 2004], following the brightness-preserving color mixing model described by Chen et al. [2015] (see Figure 5b). The gaps can show either the substrate or the color of the revealed 3D models, if the effect is implemented such that the gaps modify the alpha channel of transparent objects, showing any color underneath.

4 SUBSTRATE-BASED EFFECTS

Be it the canvas on which oil paint is placed, or the paper where watercolor pigments are settling in, the substrate has a significant role in natural painted media. The substrate could shine through, distort, accumulate pigments or shade the surface in ways that alter the actual painting. Watercolors, by being a translucent fluid medium, is especially sensitive to the substrate it is being painted on, and presents many substrate-based effects.

Ideally, an accurately measured paper profile would provide the required information to emulate these characteristic effects. However, profilometers for accurate surface height measurement, either optical or stylus-based, are not easily accessible – the equipment is expensive and scanning may take a significant amounts of time. For lack of a profilometer, we resorted to scan three watercolor papers with a flatbed scanner along multiple lighting directions, and acquired their profile through a shape from shading technique [Barmpoutis et al. 2010]. While the results might not be as accurate as the those from an actual profilometer, most irregularities within the extracted 3D profile geometry can be fixed later in object-space. These include a relative unevenness in the geometry that would not produce a uniform heightmap over the entire surface.

Once the 3D substrate data is extracted, a heightmap H and normal map N can be easily produced (see Figure 7). We use them to mimic the novel object-space dry-brush effect and to enhance the substrate granulation and distortion. They also enable the possibility to generate interactive substrate lighting.



Original scan Heightmap H Normal map N

Figure 7: Scanned substrate and extracted profile data.

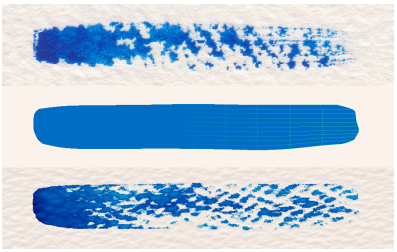


Figure 8: Results of dry-brush and granulation in object-space. From top to bottom: scanned real watercolor, 3D render, watercolorized 3D render.

Tree mesh | Dry-brush variant 1 | Dry-brush variant 2

Figure 9: Stylizing a tree mesh with dry-brush to imitate Figure 2b with different substrates and parameters.

4.1 Dry-brush and granulation

The dry-brush technique is commonly found in natural painting media. However, it has not yet been explored in object-space, which may be owing to its strong dependency on accurate substrate data. Dry-brush is a painting technique that is applied by watercolor artists with the purpose of indicating a textured appearance. Its uses vary substantially from artist to artist, but it is often employed to depict rough textures such as those produced by leaves, clouds or reflections. The rough appearance is caused when viscous pigment ("dry") only reaches the peaks of the substrate. This allows its valleys to remain unpainted, showing their color (or previously painted pigment). As a consequence, the appearance of the dry-brush application varies strongly depending on the substrate profile, the amount of pigments deposited and the pressure, direction and speed at which the brush-stroke is placed. Our object-space approach won't consider the strokes themselves, but rather the resulting general appearance of a dry-brushed area.

To begin emulating this effect, we first let the artist control the substrate roughness through a global scaling factor r that modulates the depth/height of the original heightmap $H \in [0, 1]$ in Equation 1.

$$H_r = ((H - 0.5) \times r) + 0.5\,. \tag{1}$$

We then proceed to calculate the right pigment application. Since the dry-brush effect is closely related to the substrate granulation, which corresponds to the accumulation of pigment at the valleys of the paper, these two effects are consolidated within a joint pigment application procedure. It is controllable locally through a parameter $a \in [-1, 1]$, which is directly painted by the artist on the 3D geometry. The pigment application P_a is governed by:

$$P_a = \begin{cases} t_{dry}|H_r - a|(C_s - C) + C & \text{if } H_r < a \\ C^{d_a} & \text{if } H_r \geq a \end{cases} \tag{2}$$

If the application parameter a is greater than the elevation of the heightmap H_r (case 1 in Equation 2), a dry brush is applied by linear interpolation between the substrate color C_s and the original color C, with t_{dry} a global dry-brush threshold to smoothen dry edges. Otherwise (case 2 in Equation 2), substrate granulation happens, darkening the color C by the accumulated density d_a. This effect has been thoroughly studied before [Bousseau et al. 2006; Lei and Chang 2005; Luft and Deussen 2006a; Montesdeoca et al. 2017], so we can adapt the substrate data and application parameter to take advantage of this characteristic effect. First, we convert the application parameter into a local granulation density $d_g = |a| + d_{\min}$ with $d_{\min} = 0.2$ the default amount of granulation. Then, to also obtain a darker concentration of pigments for brighter colors, we increase the granulation density (×5) through a linear interpolation of the density contribution with the object color luminance L:

$$d_g = d_g(1 - L) + (d_g \times 5)L\,. \tag{3}$$

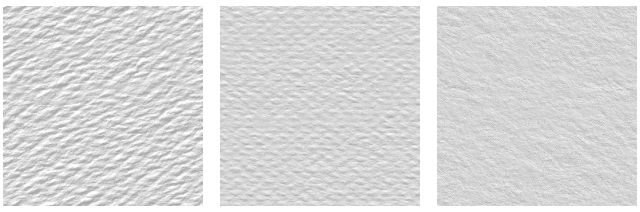
Before darkening the pigment at the valleys of the paper to generate granulation, the heightmap amplitude is reduced, shifted and inverted: $H_{iv} = 1 - ((H_r \times 0.2) + 0.8)$. This modulation is performed to increase the density only at the valleys of the paper and adapting our substrate data to follow the approach of Montesdeoca et al. [2017]. The final pigment density accumulation d_a is therefore given by Equation 4, where D is a global density parameter affecting the entire image:

$$d_a = 1 + (d_g \times H_{iv} \times D)\,. \tag{4}$$

The accumulated density d_a is eventually processed by Equation 2 to obtain the granulated pigment application P_a (see Figures 8 and 9).

4.2 Substrate lighting

A painting is always affected by external lighting conditions, thus it is of importance to consider the substrate lighting that would affect the final watercolorized imagery. Unlike oil paintings [Hertzmann 2002], in the case of watercolors, the profile of the paper tends to remain mostly unchanged. The carrier evaporates and the remaining pigmentation has little effect over the substrate profile – meaning that a shaded watercolor painting will be mostly affected by the roughness of the substrate. Since the normals of the physical substrate were acquired, deferred shading can be easily conducted



Arches-Rough light dir.: top-left | *Fabriano-Cold Press* light dir.: top | *Burgund-Matt* light dir.: top-right

Figure 10: Three watercolor paper specimens lit from different angles. See the accompanying video for their effects on the rendered imagery.

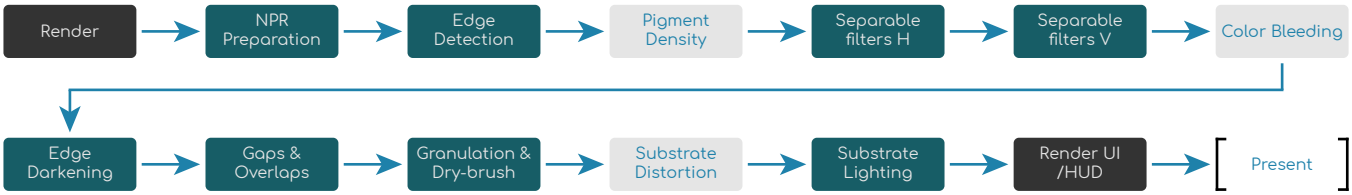


Figure 11: System schematic portraying the rendering pipeline with its different shader operations. Turquoise elements represent the stages at which the necessary data and algorithms are processed.

on top of the painted image, emulating external lighting conditions on the watercolor painting and increasing the tangibility of the digital substrate (see Figure 10).

Due to unavailable substrate profile data, this effect could previously not be controlled and was fixed to pre-existing shading of the scanned substrate textures. With the extracted substrate profile and normals, we implement a simple diffuse illumination model I_d that can easily be customized through a lighting direction $\mathbf{L}$, the extracted normal map $\mathbf{N}$, the substrate roughness r and the diffuse shading contribution d_s through Equation 5.

$$I_d = 1 - ((1 - (\mathbf{L} \cdot (r \times \mathbf{N}))) \times d_s)\,. \tag{5}$$

5 ADDITIONAL EFFECTS AND IMPLEMENTATION

In addition to the watercolor effects proposed and presented in this paper, we have also integrated the effects of pigment density, color bleeding, hand-tremors and substrate distortion to augment the watercolor look. These were incorporated from previous approaches by [Bousseau et al. 2006; Montesdeoca et al. 2017]. The entire rendering pipeline is presented in Figure 11 and was implemented using the direct stylization framework developed in Autodesk Maya by Montesdeoca et al. [2017]. The implementation was substantially extended and is briefly described next.

The first render stage rasterizes the 3D geometry which was previously offset by a tremor value in object-space. Multiple render targets come out of this initial stage, including the color image, the z-buffer, and the control masks with the object-space parameters. During the NPR preparation stage, the z-buffer is transformed into linear depth. Then, the edge detection runs the RGBD Sobel filter, followed by the pigment density stage, which darkens or brightens different parts of the image according to their assigned turbulent density. The separable filter stages perform edge blurs for edge darkening, extended edges for gaps & overlaps, and blurs the image for later bleeding. Afterwards, the color bleeding stage blends part of the blurred image to produce localized bleeding. Additionally, the parameter masks related to edge-based effects are also modified, as darkened edges and gaps & overlaps should not appear in these bled areas.

At the second row of Figure 11, the edge- and substrate-based effects are processed. Edge-based effects begin at the edge darkening stage, concentrating the pigmentation at the previously blurred edges. Gaps & overlaps use the previously extended edges to find their neighboring colors and either mix with them or produce gaps. The substrate-based effects begin at the granulation and dry-brush stage, deriving each effect from the acquired heightmap. Then, the substrate distortion modifies the UVs according to the substrate normals, shifting the sampled color. Once all these effects have been computed, the deferred substrate lighting is performed using the substrate normals. Finally, the remaining user-interface/heads-up-display is rendered and the final watercolorized result is presented to the user. Almost all these stages make extensive use of the parameter masks, so that the effects can be controlled in object space, retaining their spatial coherence and localized art-direction.

The real-time implementation does not present significant issues. However, the parallel nature of shaders requires special consideration compared to off-line approaches, as only the currently evaluated pixel can be altered at each stage. The development of our implementation could also have benefited from a modular, node-based approach. This would have enabled us to iterate quicker and alleviate the complexity of managing multiple rendering targets and several rendering stages only through code.

Three different levels of control are included for most effects. At the highest level of control, global parameters help to set initial stylization values over the entire scene. Then, object-space stylization parameters can be assigned within the "material" shaders of each object. Finally, the lowest level of control is given through the painted parameters, stored in different vertex color sets at each vertex of the meshes. We find that painting parameters in object-space offers versatile control and feels natural, but painting each object in a convoluted scene can take a significant amount of time, especially when considering different viewpoints.

6 RESULTS AND DISCUSSION

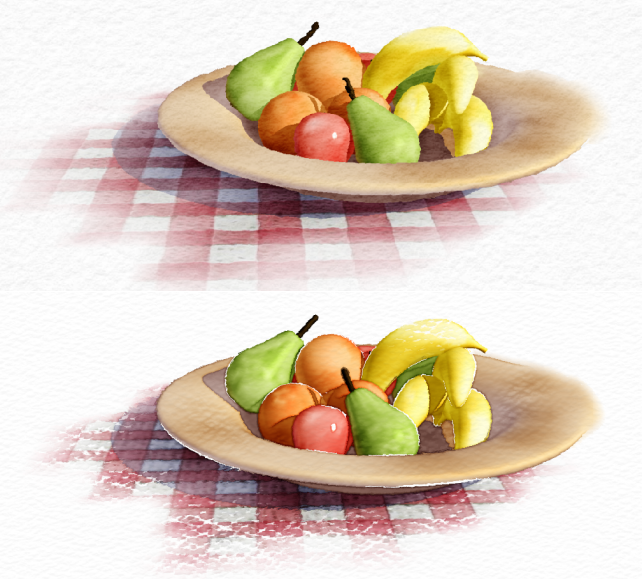
A watercolor look is composed of a series of low-level effects, that together create a defined watercolor style. Although this paper specifically focuses on edge- and substrate-based effects, we incorporated other previously studied watercolor effects into our pipeline. In this way, we can feature the contribution of the developed effects as close to its natural occurrence as possible. Some examples of watercolorized imageries featuring the effects addressed in this paper can be found in Figures 1, 9, 6 and 12.

In Figure 12a, dry-brush was used for the tree leaves, the rooftop and the lamp post to abstract and to provide texture. Some parts of the walls in the house also show a minor dry-brush application to increase the roughness of the watercolor look. Additionally, gaps & overlaps were used extensively to provide the rough and sketchy look. In Figures 12b and 12c, gaps were used to delineate the form of the characters and provide a sketchier look. Figure 12c also

(a) Lowpoly House model, CC BY Rafael Scopel

(b) Spherebot model, CC BY Bastien Genbrugge

(c) Octopus model, CC BY Lukáš Marek



(d) Fruit plate watercolorized with (top) Montesdeoca et al. [2016] method and (bottom) our pipeline with the integrated effects.

Figure 12: Rendered frames featuring edge- and substrate-based effects. Please refer to the accompanying video to see them in motion.

presents some minor dry-brush application where suitable. Finally, Figure 12d provides a comparison with the previous approach by Montesdeoca et al. [2016], demonstrating how gaps & overlaps and dry-brush contribute to the palette of watercolor effects to generate elaborate watercolor renders. To see these examples under motion and animation, please refer to the accompanying video.

The results show an overall pleasant integration of dry-brush and gaps & overlaps to form a sketchier and rougher look, which resembles their usage in traditional watercolors. Other proposed effects such as substrate lighting enable a more realistic and tangible rendered outcome, whereas the RGBD edge detection helps create more consistent and controllable edge darkening, compared to previous approaches.

Though the GPU pipeline (see Figure 11) is quite complex, the render manages to perform in real-time (60+ frames) with ease at full HD (1920 × 1080) resolution, with the following configuration: NVIDIA GeForce GTX 1080, Intel Xeon E5-2609 @2.4Ghz and 16Gb of RAM. As most of the processing is also performed in image-space, the stylization scales well with scene complexity. This is highly beneficial for interactivity and art-direction, especially since most effects are localized.

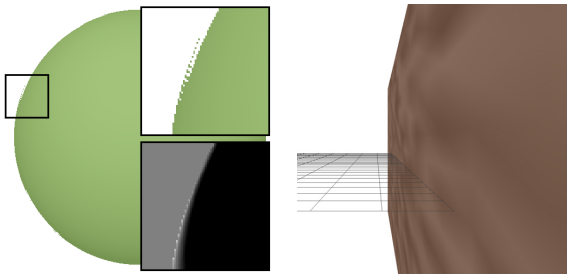
6.1 Limitations

The data from the substrate specimens, acquired with a shape-from-shading technique, was accurate enough for our experimental and aesthetic purposes. Yet a proper physical profile scan could present an increased sharpness and fidelity, which could further improve the modeled effects and the overall watercolor look.

The dry-brush effect is relative to the substrate size, therefore, it will look significantly different under diverse paper profiles and scales, or when the camera moves closer or further away from the subject (if the substrate scale remains static). This behavior is physically correct, but it takes some time to get used to and presents temporal coherence issues in animation. A dynamic paper texture [Bénard et al. 2009; Cunzi et al. 2003; Kaplan and Cohen 2005] might be able to reduce temporal coherence problems caused by the substrate, with a potential penalty in fidelity with the original data. Additionally, the dry-brush technique changes depending on the direction of the brush stroke, and may sometimes leave a trace of the brush bristles (see Figure 8), this was not considered in our object-space approach.

The RGBD edge detection is still prone to noise, especially when using photorealistic textures. A general pre-processing stage for textures [Semmo and Döllner 2015] would create more robust and meaningful edges, especially for effects like gaps & overlaps. Additionally the edge-detection would probably benefit from a deferred rendering pipeline, as the edges would not be influenced by shading conditions (i.e., sharp edges from specular highlights).

A more general limitation of our method comes from the parameter masks painted on the object surface. While this approach helps the spatial coherence of the render and grants higher control over the effects, the 3D surfaces are subject to geometric and perspective distortions, which may be undesirable. In Figure 13a, the spherical mapping of the parameter mask distorts the transition of parameters, increasing its abruptness and potentially creating pixel artifacts. This behavior is especially noticeable when the normals



(a) Geometric distortion **(b)** Perspective distortion

Figure 13: Examples of geometric (a) and perspective (b) distortions affecting the parameter masks.

of the geometric surface are almost perpendicular to the view direction (see the bright pixel parameters at the edge of the sphere). In Figure 13b, a mask with an otherwise uniform noise pattern is foreshortened through perspective distortion, changing the scale of the parametrization relative to its distance from the image plane. While these object-space distortions might not be desirable, they are not too noticeable as long as the parameter masks are well designed.

7 CONCLUSION AND FUTURE WORK

The main contributions presented in this paper include the edge-based effects of gaps & overlaps and the substrate-based effect of dry-brush, which had little study in object-space stylization before. While working to mimic these effects, we extracted edges and acquired substrate data that helped to improve previously studied algorithms such as edge-darkening, substrate granulation and enabled the possibility of shading the finalized watercolorized imagery with external light sources.

The dry-brush effect and gaps & overlaps aid significantly to generate overall sketchier and rougher watercolor imagery, which was not possible before. There is an intrinsic difficulty to achieve naturally occurring imperfections within computer graphics and a volatile natural medium like watercolor is no exception to this. Curiously, these imperfections are believably created by resorting to more robust algorithms and the use of more physically accurate substrates. A physical substrate also enabled the possibility to generate fully controllable external lighting conditions, which have not been given much attention before, but offer a new level of tangibility to the digital creation. These effects, together with several smaller improvements to previously studied effects, enlarge the possibilities and applications that watercolor in object-space offers – bringing the medium forward and offering a stylistic change to otherwise standard rendering procedures and looks.

Future work in object-space watercolor research could focus on layered pigmentation and turbulent pigment mixing. These are unexplored areas that could address aesthetic complexity, commonly found in watercolors. Other features that could bring the stylization forward involve dynamic image-space substrates that communicate with object-space data to remain coherent, and texture painting within a direct stylization pipeline, to enhance the artist-computer interaction. Regarding substrate- and edge-based effects in general, there are other types of expressive rendering which could take advantage of a closer exploration and analysis of these two factors. All natural media, thinly applied, is affected by the substrate it is painted on, and edges are often subject of stylization by artists. While painting parameters in object-space grants extensive control and art-direction, locally painting a complex scene at each vertex can become a cumbersome endeavor. A more volumetric approach towards parameter assignment might present itself as an interesting subject to study from an interactive and aesthetic point of view. Finally, after implementing our contribution in a direct stylization pipeline, we believe that such systems could be generalized to other types of styles in which cross-media parametrization would create an unprecedented versatility and application of expressive rendering techniques.

ACKNOWLEDGMENT

We would like to thank the Multi-plAtform Game and Innovation Centre (MAGIC) and the Maverick team at Inria - Rhône-Alpes for supporting this international collaboration. Our gratitude is also with our colleagues for their comments and suggestions, together with the anonymous reviewers for their constructive feedback. This research is supported by the Interdisciplinary Graduate School (IGS) at the Nanyang Technological University, Singapore.

REFERENCES

Angelos Barmpoutis, Eleni Bozia, and Robert S. Wagman. 2010. A novel framework for 3D reconstruction and analysis of ancient inscriptions. *Machine Vision and Applications* 21, 6 (2010), 989–998. DOI: https://doi.org/10.1007/s00138-009-0198-7

Adrien Bousseau, Matt Kaplan, Joëlle Thollot, and François X. Sillion. 2006. Interactive Watercolor Rendering with Temporal Coherence and Abstraction. In *Proceedings of International Symposium on Non-photorealistic Animation and Rendering (NPAR '06)*. ACM, 141–149. DOI: https://doi.org/10.1145/1124728.1124751

Adrien Bousseau, Fabrice Neyret, Joélle Thollot, and David Salesin. 2007. Video Watercolorization Using Bidirectional Texture Advection. *ACM Transactions on Graphics* 26, 3 (2007), 104. DOI: https://doi.org/10.1145/1276377.1276507

Claire Waite Brown. 2007. *The Watercolor Flower Artist's Bible: An Essential Reference for the Practicing Artist.* Chartwell Books.

Jeremy Burgess, Geoff Wyvill, and Scott A. King. 2005. A system for real-time watercolour rendering. In *Computer Graphics International*. IEEE, 234–240. DOI: https://doi.org/10.1109/CGI.2005.1500426

Pierre Bénard, Adrien Bousseau, and Joëlle Thollot. 2009. Dynamic Solid Textures for Real-Time Coherent Stylization. In *Proceedings of the Symposium on Interactive 3D graphics and games*. ACM, 121–127. DOI: https://doi.org/10.1145/1507149.1507169

Pierre Bénard, Forrester Cole, Michael Kass, Igor Mordatch, James Hegarty, Martin Sebastian Senn, Kurt Fleischer, Davide Pesare, and Katherine Breeden. 2013. Stylizing Animation By Example. *ACM Transactions on Graphics* 32, 4 (2013), 119. DOI: https://doi.org/10.1145/2461912.2461929

Dongdong Chen, Jing Liao, Lu Yuan, Nenghai Yu, and Gang Hua. 2017. Coherent Online Video Style Transfer. *ArXiv e-prints* (2017). http://https://arxiv.org/abs/1703.09211 [accessed: 2017-04-04].

Zhili Chen, Byungmoon Kim, Daichi Ito, and Huamin Wang. 2015. Wetbrush: GPU-based 3D Painting Simulation at the Bristle Level. *ACM Transactions on Graphics* 34, 6 (2015), 200:1–200:11. DOI: https://doi.org/10.1145/2816795.2818066

Nelson S. H. Chu and Chiew-Lan Tai. 2005. MoXi: Real-time Ink Dispersion in Absorbent Paper. *ACM Transactions on Graphics* 24, 3 (2005), 504–511. DOI: https://doi.org/10.1145/1073204.1073221

John P Collomosse, David Rowntree, and Peter M Hall. 2005. Stroke surfaces: temporally coherent artistic animations from video. *IEEE Transactions on Visualization and Computer Graphics* 11, 5 (2005), 540–9. DOI: https://doi.org/10.1109/TVCG.2005.85

Matthieu Cunzi, Joëlle Thollot, Sylvain Paris, Gilles Debunne, Jean-Dominique Gascuel, and Frédo Durand. 2003. Dynamic Canvas for Immersive Non-Photorealistic Walkthroughs. In *Proceedings of Graphics Interface*. A K Peters, LTD., 121–130.

Cassidy J. Curtis, Sean E. Anderson, Joshua E. Seims, Kurt W. Fleischer, and David H. Salesin. 1997. Computer-generated Watercolor. In *Proceedings of the 24th Annual*

Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 1997). ACM, 421–430. DOI: https://doi.org/10.1145/258734.258896

Stephen DiVerdi, Aravind Krishnaswamy, Radomír Měch, and Daichi Ito. 2013. Painting with Polygons: A Procedural Watercolor Engine. *IEEE Transactions on Visualization and Computer Graphics* 19, 5 (2013), 723–735. DOI: https://doi.org/10.1109/TVCG.2012.295

Jakub Fišer, Ondřej Jamriška, Michal Lukáč, Eli Shechtman, Paul Asente, Jingwan Lu, and Daniel Sýkora. 2016. StyLit: Illumination-guided Example-based Stylization of 3D Renderings. *ACM Transactions on Graphics* 35, 4 (2016), 92:1–92:11. DOI: https://doi.org/10.1145/2897824.2925948

Gene Franks. 1988. *Watercolor: Drybrush Technique*. Walter Foster Pub.

Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. 2015. A Neural Algorithm of Artistic Style. *CoRR* abs/1508.06576 (2015). http://arxiv.org/abs/1508.06576 [accessed: 2016-10-28].

Nathan Gossett and Baoquan Chen. 2004. Paint Inspired Color Mixing and Compositing for Visualization. In *IEEE Symposium on Information Visualization*. 113–118. DOI: https://doi.org/10.1109/INFVIS.2004.52

Aaron Hertzmann. 2002. Fast paint texture. In *Proceedings of International Symposium on Non-photorealistic Animation and Rendering (NPAR '02)*. ACM Press, 91. DOI: https://doi.org/10.1145/508530.508546

Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David H. Salesin. 2001. Image Analogies. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. ACM, 327–340. DOI: https://doi.org/10.1145/383259.383295

Klaus Janson. 2003. *The DC Comics Guide to Inking Comics*. Watson-Guptill.

Henry Johan, Ryota Hashimoto, and Tomoyuki Nishita. 2004. Creating Watercolor Style Images Taking Into Account Painting Techniques. *The Journal of the Society for Art and Science* 3, 4 (2004), 207–215. DOI: https://doi.org/10.3756/artsci.3.207

Mizuki Kagaya, William Brendel, Qingqing Deng, Todd Kesterson, Sinisa Todorovic, Patrick J. Neill, and Eugene Zhang. 2011. Video Painting with Space-Time-Varying Style Parameters. *IEEE Transactions on Visualization and Computer Graphics* 17, 1 (2011), 74–87. DOI: https://doi.org/10.1109/TVCG.2010.25

Matthew Kaplan and Elaine Cohen. 2005. A Generative Model for Dynamic Canvas Motion. In *Proceedings of the Eurographics Conference on Computational Aesthetics in Graphics (CAe '05)*. Eurographics Association, 49–56. DOI: https://doi.org/10.2312/compaesth/compaesth05/049-056

Su Ian Eugene Lei and Chun-Fa Chang. 2005. *Real-Time Rendering of Watercolor Effects for Virtual Environments*. Springer Berlin Heidelberg, 474–481. DOI: https://doi.org/10.1007/978-3-540-30543-9_60

Jingwan Lu, Connelly Barnes, Stephen DiVerdi, and Adam Finkelstein. 2013. RealBrush: Painting with Examples of Physical Media. *ACM Transactions on Graphics* 32, 4 (2013), 1–12. DOI: https://doi.org/10.1145/2461912.2461998

Thomas Luft and Oliver Deussen. 2006a. Real-Time Watercolor for Animation. *Journal of Computer Science and Technology* 21, 2 (2006), 159–165. DOI: https://doi.org/10.1007/s11390-006-0159-9

Thomas Luft and Oliver Deussen. 2006b. Real-time Watercolor Illustrations of Plants Using a Blurred Depth Test. In *Proceedings of International Symposium on Non-photorealistic Animation and Rendering (NPAR '06)*. ACM, 11–20. DOI: https://doi.org/10.1145/1124728.1124732

Thomas Luft, Frank Kobs, Walter Zinser, and Oliver Deussen. 2008. Watercolor Illustrations of CAD Data. In *Proceedings of the Eurographics Conference on Computational Aesthetics in Graphics (CAe '08)*. The Eurographics Association. DOI: https://doi.org/10.2312/COMPAESTH/COMPAESTH08/057-063

Eric B. Lum and Kwan-Liu Ma. 2001. Non-photorealistic rendering using watercolor inspired textures and illumination. In *Proceedings of the 9th Pacific Conference on Computer Graphics and Applications*. 322–330. DOI: https://doi.org/10.1109/PCCGA.2001.962888

Chaohui Lü and Xiaolong Chen. 2014. Image watercolorization based on visual weight-map. In *International Congress on Image and Signal Processing (CISP) (CISP)*. 233–237. DOI: https://doi.org/10.1109/CISP.2014.7003783

Santiago E Montesdeoca, Hock Soon Seah, and Hans-Martin Rall. 2016. Art-directed Watercolor Rendered Animation. In *Proceedings of International Symposium on Non-photorealistic Animation and Rendering (NPAR '16)*. The Eurographics Association, 51–58. DOI: https://doi.org/10.2312/exp.20161063

Santiago E. Montesdeoca, Hock Soon Seah, Hans-Martin Rall, and Davide Benvenuti. 2017. Art-directed watercolor stylization of 3D animations in real-time. *Computers & Graphics* 65 (2017), 60 – 72. DOI: https://doi.org/10.1016/j.cag.2017.03.002

Bethan Morris. 2010. *Fashion Illustrator*. Laurence King Publishing.

Marc Nienhaus and Jürgen Döllner. 2005. Blueprint Rendering and Sketchy Drawings. In *GPU Gems II: Programming Techniques for High Performance Graphics and General-Purpose Computation*. Addison-Wesley Professional, Chapter 15, 235–252.

Giuseppe Papari and Nicolai Petkov. 2011. Edge and line oriented contour detection: State of the art. *Image and Vision Computing* 29, 2–3 (2011), 79 – 103. DOI: https://doi.org/10.1016/j.imavis.2010.08.009

Takafumi Saito and Tokiichiro Takahashi. 1990. Comprehensible rendering of 3-D shapes. *Proceedings of the 17th annual conference on Computer graphics and interactive techniques - SIGGRAPH '90* 24, 4 (1990), 197–206. DOI: https://doi.org/10.1145/97879.97901

Ahmed Selim, Mohamed Elgharib, and Linda Doyle. 2016. Painting Style Transfer for Head Portraits Using Convolutional Neural Networks. *ACM Transactions on Graphics* 35, 4 (2016), 129:1–129:18. DOI: https://doi.org/10.1145/2897824.2925968

Amir Semmo and Jürgen Döllner. 2015. Interactive image filtering for level-of-abstraction texturing of virtual 3D scenes. *Computers & Graphics* 52 (2015), 181 – 198. DOI: https://doi.org/10.1016/j.cag.2015.02.001

Amir Semmo, Tobias Dürschmid, Matthias Trapp, Mandy Klingbeil, Jürgen Döllner, and Sebastian Pasewaldt. 2016. Interactive Image Filtering with Multiple Levels-of-Control on Mobile Devices. In *Proceedings of the ACM SIGGRAPH Asia Symposium on Mobile Graphics and Interactive Applications*. ACM, NA. DOI: https://doi.org/10.1145/2999508.2999521

David Small. 1991. Simulating watercolor by modeling diffusion, pigment, and paper fibers. *SPIE Proceedings* 1460, Image Handling and Reproduction Systems Integration (1991), 140–146. DOI: https://doi.org/10.1117/12.44417

Hazel Soan. 2014. *The Artist's Color Guide - Watercolor: Understanding Palette, Pigments and Properties*. F&W Media, Incorporated. https://books.google.com.sg/books?id=GsAUngEACAAJ

Tom Van Laerhoven and Frank Van Reeth. 2005. Real-time simulation of watery paint. *Computer Animation and Virtual Worlds* 16, 3-4 (2005), 429–439. DOI: https://doi.org/10.1002/cav.95

Miaoyi Wang, Bin Wang, Yun Fei, Kanglai Qian, Wenping Wang, Jiating Chen, and Jun-Hai Yong. 2014. Towards Photo Watercolorization with Artistic Verisimilitude. *TVCG* 20, 10 (2014), 1451–1460. DOI: https://doi.org/10.1109/TVCG.2014.2303984

Mi You, Taekwon Jang, Seunghoon Cha, Jihwan Kim, and Junyong Noh. 2013. Realistic paint simulation based on fluidity, diffusion, and absorption. *Computer Animation and Virtual Worlds* 24, 3-4 (2013), 297–306. DOI: https://doi.org/10.1002/cav.1500