Edge- and substrate-based effects for watercolor stylization

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

Hock Soon Seah
Nanyang Technological University
School of Computer Science and Engineering

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

Romain Vergne
Joëlle Thollot
Univ. Grenoble Alpes, CNRS, LJK
Inria Grenoble Rhône-Alpes

Hans-Martin Rall
Davide 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.

KEYWORDS

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

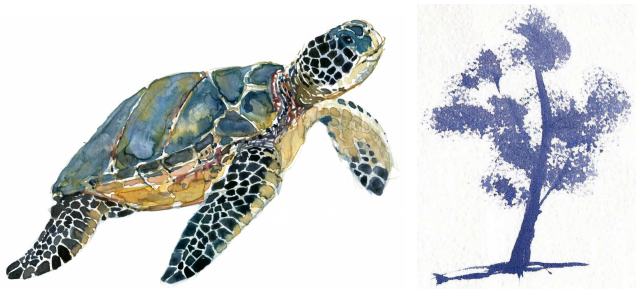
ACM Reference format:
Santiago E. Montesdeoca, Hock Soon Seah, Pierre Bénard, Romain Vergne, Joëlle Thollot, Hans-Martin Rall, and Davide Benvenuti. 2017. Edge- and substrate-based effects for watercolor stylization. In *Proceedings of Expressive, Los Angeles, CA, July 29–30, 2017 (NPAR'17),* 10 pages.
https://doi.org/10.1145/3092919.3092928

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

ACM ISBN 978-1-4503-5081-5/17/07…$15.00
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.



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

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

Figure 2: Traditional watercolor paintings.

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.

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

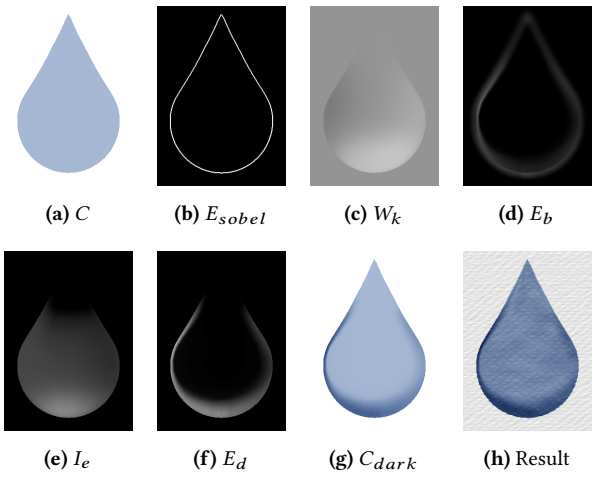


Figure 3: Breakdown of an art-directed edge darkening.

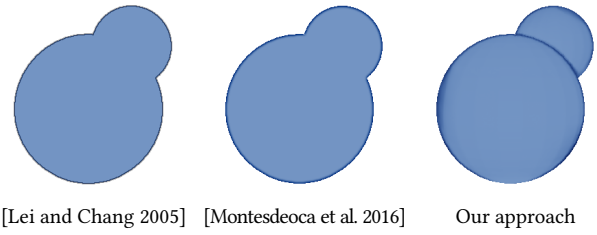


Figure 4: Different approaches towards edge darkening. Compared to previous work, our method darkens edges with similar colors and supports edge width variations.

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.

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

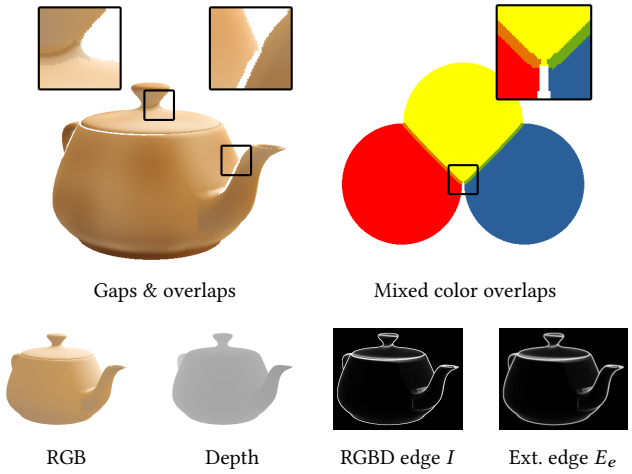


Figure 5: Results of gaps & overlaps in object-space and a breakdown of the edge extraction and edge extension.

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 been barely studied in object-space. Luft and Deussen [2006a] explored this effect by rendering group of 3D scene 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 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.

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.

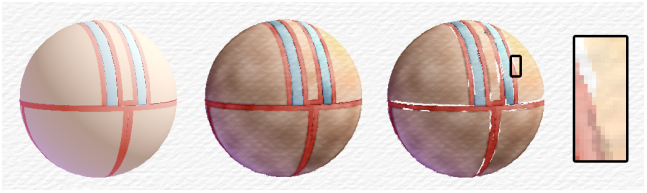


Figure 6: Gaps & overlaps to achieve a sketchier watercolor look. From left to right: original colors, result without and with gaps & overlaps.

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.

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.