Dynamic Painterly Droplets

Ivan Neulander
Google Inc.

We present a straightforward painterly rendering algorithm that composites lightweight models of paint droplets onto a virtual canvas, producing visually pleasing still images and animations. Our method adapts the size, shape, density, and placement of these droplets to the content of a single input image, typically a photograph. The droplets can be rendered dynamically over multiple frames to produce temporally coherent, loopable animations. Our implementation is GPU-accelerated and runs on both desktop and mobile platforms.

Figure 1: A photograph and its rendering using our painterly droplets.

Paint Droplet Model

The basis of our painterly rendering algorithm is the model of a paint *droplet*, which is derived from an arbitrarily sized and oriented ellipse, with radially varying color and opacity. Unlike a true ellipse, a droplet's perimeter is randomized to produce a consistent but noisy outline. This shape variation is implemented in the shader. A droplet's orientation and thickness are governed by the gradient vectors of the input image, and its color is based on the color of the input image over the corresponding region. We implement a low-discrepancy random number generator (RNG) using a multi-dimensional Halton sequence in order to vary the position, shape, and color attributes of each droplet. This reduces clumping and yields an approximately even coverage of randomized values.

Rendering Pipeline

Prior to generating droplets, we construct a gradient image from the input, and then apply blur and gamma adjustment to the gradient vectors. These operations are implemented in Halide [Ragan-Kelley et al. 2012] to portably exploit multithreading and vectorization on our target platforms.

Our rendering pipeline generates droplets in several discrete stages: First, a *base layer* consisting of large and highly opaque droplets is created to efficiently cover the canvas. Second, a group of *uniform droplets* are placed uniformly over the canvas at varying sizes, such that the number of droplets at a given size varies inversely with the size. Finally, a set of *detail droplets* are placed nonuniformly, with density based on the input image gradients and proximity to human faces. The placement of detail droplets follows the idea of [Wexler and Dezeustre 2012] by using an image-based importance sampler. The three droplet types are shown in Figure 1. Each generated droplet is stored in a priority queue that provides rapid access to the largest (longest) droplet.

Once all droplets have been generated, they are composited onto a fixed canvas layer in largest-to-smallest order, as proposed in [Hertzmann 1998]. Prior to compositing, each droplet is optionally perturbed, as described below, to produce temporally coherent animations.

Figure 2: From left: base layer, uniform droplets, detail droplets.

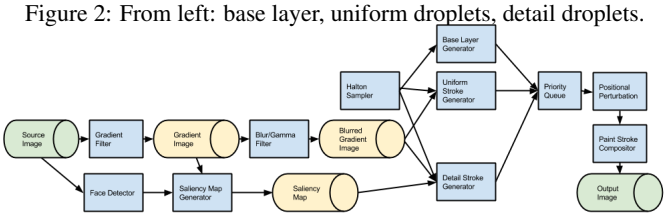

Figure 3: Core components of our painterly rendering pipeline.

Droplet Perturbation

The nominal position of each droplet is obtained either directly from the RNG, or (for detail droplets) indirectly via the importance sampler. In order to generate an animated output, we position the droplet dynamically by perturbing this position over time according to a closed Lissajous curve of user-specified relative frequencies. The size of the curve is based on the droplet's length, producing smaller perturbations for smaller droplets. This helps to preserve detail. While all droplets use the same overall curve shape, their initial phases are uniquely determined by the RNG. We adjust the droplet's colors based on its perturbed position. To improve temporal coherence, we use the size and orientation from the original, unperturbed position.

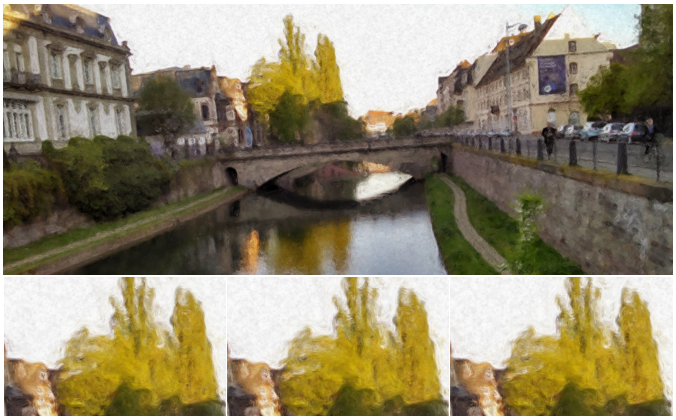

Figure 4: Above: painterly rendering of a scene in Strasbourg; Below: crops of three successive frames from resulting animation.

References

HERTZMANN, A. 1998. Painterly rendering with curved brush strokes of multiple sizes. In *SIGGRAPH '98*, ACM, SIGGRAPH '98.

RAGAN-KELLEY, J., ADAMS, A., PARIS, S., LEVOY, M., AMARASINGHE, S., AND DURAND, F. 2012. Decoupling algorithms from schedules for easy optimization of image processing pipelines. *ACM Trans. Graph. 31*, 4 (July).

WEXLER, D., AND DEZEUSTRE, G. 2012. Intelligent brush strokes. In *ACM SIGGRAPH 2012 Talks*, ACM, SIGGRAPH '12.