A dynamic drawing algorithm for interactive painterly rendering

David Vanderhaeghe, Pascal Barla, Joëlle Thollot, François X. Sillion

A dynamic drawing algorithm for interactive painterly rendering (sketch 0387)

David Vanderhaeghe, Pascal Barla, Joëlle Thollot, François X. Sillion
ARTIS GRAVIR/IMAG INRIA

Figure 1: Three different painterly styles produced by our method at interactive rates. Left: long strokes are drawn along silhouettes using surface normals. Middle: strokes have a common global orientation set by the user. Right: Surface principal curvature is used to orient thick strokes. Note that illumination features such as shading and highlights are correctly represented independently of the user's stylistic decisions.

1 Motivation

Painterly rendering is a technique that takes inspiration from traditional paintings, usually focusing on effects similar to those achieved with oil or acrylic paint, where the individual brush strokes are more or less perceived individually. The main idea is to render a scene projected on the image plane by a set of 2D vector paint strokes holding style attributes (color, texture, etc). This representation has the effect of abstracting the rendering by using primitives larger than pixels, and emphasizing the image plane through the 2D nature of the paint strokes.

In the general case, paint strokes simultaneously represent informations about objects in the scene (such as shape or reflective properties of a surface from the current point of view) while following a stroke style provided by the user (straight or curved brush strokes, thick or thin outline, etc). During the animation they also follow the 2D or 3D motion of some scene elements. The main issues in painterly rendering originate from these conflicting goals (as stated by Durand [2002]). **Temporal coherence** of the strokes motion is of primary interest: it comes from the desire to link the motion of a 2D primitive (a stroke) to that of a 3D primitive (e.g. a surface). Another important aspect is the **density** of strokes: when zooming in or out from an object, the number of strokes used to represent it must increase or decrease in order to maintain a uniform density in the picture plane while keeping a constant thickness in image space. Finally, an ideal painterly renderer would let the user fully specify the strokes **style** in a way that is independent of the depicted scene, but at the same time should ensure that some properties of the scene are well-represented, such as object silhouettes or lighting.

2 Contributions

We present an object-space, particle-based system that extends the pioneering work of Meier [1996]. Our main contribution is a fully view- and lighting-dependent behavior that explicitly performs a trade-off between a user-specified stroke style and the faithful representation of the depicted scene. This way, our system offers a broader expressiveness by removing the constraints usually found in previous approaches, while still ensuring **temporal coherence**.

Following Durand's study on computer depiction [2002], our system is composed or three stages: attributes design, particle distribution and strokes drawing. The attributes stage takes as input the 3D scene and computes the scene attributes using G-buffers. Attributes such as normals or curvature can be used to control strokes attributes, e.g. orientation and thickness. We blur color and depth attributes to filter out details that could not be represented by thick strokes during the drawing stage. The distribution stage takes as input the 3D scene, distributes 3D particles over the scene surfaces and for each frame projects a subset of the particles to compute the strokes positions while ensuring a complete coverage of the image. This way our system allows to control stroke **density** by means of a view-dependent particles distribution. Finally the strokes drawing stage takes as input the 2D strokes positions, the strokes attributes computed from the extracted G-buffers and a set of user-defined parameters: length, bending, orientation, texture, etc. Then strokes are drawn starting from their 2D position. As strokes attributes remain constant over the drawing, strokes are cut off whenever the underlying attributes become too far from the starting ones. This ensures that the important details of the scene are kept to obtain a good depiction, while using the user-supplied **style**.

Compared to Meier's work, each stage provides improved functionality: A large freedom for the choice of strokes attributes, a view-dependent density management, and a faithful representation of scene features. Most of the work is done on the GPU allowing to obtain a dynamic painting algorithm that bears some similarities with traditional techniques and runs at interactive frame rates.

References

DURAND, F. 2002. An invitation to discuss computer depiction. In *2nd International Symposium on Non-Photorealistic Animation and Rendering (NPAR'02)*.

MEIER, B. J. 1996. Painterly rendering for animation. In *Siggraph'96, Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ACM Press, 477–484.

Figure 2: Our painting method preserves depth cues in the scene: note how paint strokes stop at objects silhouettes.