Pen-and-ink textures for real-time rendering

Jennifer Fung
New Media Innovation Centre

Oleg Veryovka
Electronic Arts

Abstract

Simulation of a pen-and-ink illustration style in a real-time rendering system is a challenging computer graphics problem. Tonal art maps (TAMs) were recently suggested as a solution to this problem. Unfortunately, only the hatching aspect of pen-and-ink media was addressed thus far. We extend the TAM approach and enable representation of arbitrary textures. We generate TAM images by distributing stroke primitives according to a probability density function. This function is derived from the input image and varies depending on the TAM's scale and tone levels. The resulting depiction of textures approximates various styles of pen-and-ink illustrations such as outlining, stippling, and hatching.

Key words: Non-photorealistic rendering, real-time shading, texture mapping, filtering, animation

1 Introduction

Pen-and-ink drawing is an important and effective form of pictorial representation [5]. It is often used for technical, architectural, and medical illustrations, most of which are still produced manually. The problem of automatically generating pen-and-ink style images from 3D models was first addressed by Winkenbach and Salesin [17] in 1994 and was further investigated by numerous non-photorealistic rendering research [4].

The focus of our work is approximation of the pen-and-ink rendering style in a real-time interactive system. Previous real-time NPR techniques can be grouped into two categories: image-space methods and object-space methods.

Image-space methods render a 3D model and attempt to approximate the resulting image by strategically placing drawing primitives on the screen. Secord et al. [12] developed a real-time algorithm that places drawing primitives according to a probability density function derived from the rendered image. While image-space techniques are able to approximate pen-and-ink styles well, the resulting animations suffer from the lack of frame-to-frame coherence, i.e. dots and lines do not "stick" to object surfaces and may "swim around" as the object's shading and scale changes.

Object-space methods achieve better frame coherence and often higher frame rates and are the more common choice for interactive applications. In object-space methods, the drawing primitives are attached directly to the surface of the 3D geometry. Many object-space algorithms model drawing primitives with specialized geometry such as particle systems [10, 6, 2] or graftals [7, 9]. Instead of using additional geometry, texture-based methods control local shading using textures [11, 15, 8, 16]. Texture approaches are often more efficient than geometry-based methods and result in rendering at higher frame rates.

Our work is based on a texture-based method that uses tonal art maps (TAMs) introduced by Praun et al. [11]. This previous research addressed the problem of real-time hatching by pre-rendering lines onto mipmapped textures of various grayscale tones. We extend the technique introduced by Praun et al. [11] and introduce an algorithm that creates TAMs from a texture image.

In the following section, we review the relevant previous research and discuss its limitations. Further, we present an overview of our algorithm for distributing stroke primitives according to a probability function. We control the display of texture features at multiple resolutions by changing the probability functions computed from the input image. We describe how our technique differentiates between hatching and outlining strokes and chooses their direction and scale. In the conclusion, we present our results and discuss advantages and limitations of our approach.

2 Previous related work

Praun et al. [11] developed TAMs for real-time hatching. Hatching lines are pre-rendered onto textures of various resolutions and tones. The density of the hatching lines controls the overall grayscale tone of the texture. Shading is produced by blending textures of various tones in a real-time rendering system. The algorithm maintains frame-to-frame coherence using a "stroke nesting property", where strokes on a TAM image appear in all the darker images of the same resolution and in the higher resolution images of the same tone. This algorithm only addresses the issue of hatching and is not suitable for the depiction of subtle surface properties such as shape and texture.

The stylistic depiction of textures is a challenging artistic problem [5]. In computer graphics, this problem was first addressed by Winkenbach and Salesin [17], who suggested the use of pre-rendered textures for pen-and-ink renderings. A similar technique was developed by Freudenberg et al. [3] for a real-time system. Their work relies on textures created by an artist and does not present a solution for automatic texture generation.

The automatic stylistic depiction of still images is well investigated [4, 14]. Most previous research deals with placing drawing primitives at a single resolution and tone. Thus, these previous algorithms are not able to produce multiple textures that satisfy the nesting properties of the TAMs.

The importance-driven halftoning of Streit and Buchanan [13] is a multi-resolution technique that distributes illustration primitives according to an abstract importance function. Like previous image-based algorithms, it is targeted for the production of a single image and does not control the placement of primitives at multiple resolutions. In addition, the algorithm is suitable for distributing dots and short lines only, and the resulting images often suffer from grid-like artifacts.

Veryovka's threshold texturing technique [15] addresses the problem of texture rendering at multiple resolutions and tones, but is suitable mainly for a cartoon style rendering. It can approximate hatching effects, but it is based on pixel level operations and is not able to control multi-pixel primitives such as strokes.

In this work, we present an automatic algorithm that extends previous image-based NPR algorithms by accounting for the distribution of strokes at multible resolutions and tones simultaneously.

3 Our technique

3.1 Overview

Our algorithm for the automatic generation of TAMs from an input image is based on the distribution of strokes according to a probability function. We follow the approach introduced by Secord et al. [12] and extend their technique to handle multiple resolutions and tone levels. We compute distribution functions at multiple resolutions according to some importance functions. The importance functions may differ from the input image, as suggested by Streit and Buchanan [13]. We discuss the creation of the importance functions in the following section.

The target tone of each TAM level is computed in a pre-processing step. We fill the TAM images with a series of randomly distributed pen strokes using the Halton sequence as in Secord et al. [12]. The direction and length of each stroke is computed dynamically using local image features. Thus, the number of necessary drawing primitives depends on the image features and cannot be pre-computed. We match the target tone by periodically checking if the average intensity of the image is achieved by the already rendered strokes.

We generate images in a coarse-to-fine, light-to-dark order, similar to Praun et al. [11]. This ensures that the nesting property of the TAM is satisfied.

3.2 Multi-resolution importance functions

When given enough visual information, people can infer what the rest of an image looks like. It is sometimes preferable to leave certain details to the imagination, because it personalizes the experience and the human imagination is capable of inventing sights that we can't render on a computer. Thus, the challenge is to provide a sufficient amount of visual information at each rendering level and to emphasize the important features.

The features we want to outline differ depending on the tone and resolution of the TAM image. We may have enough area in the finer images to highlight minor features effectively, but minor features will just obscure the main features in the coarse images. In the lighter images, we only have a few strokes. We need all these strokes to outline the main shapes and make the texture recognizable. We have plenty of strokes in the darker tones, so we can use the strokes to highlight additional features, creating a more complete image.

Figure 1: The strokes in the lighter images are used to outline the elephant. Additional strokes in the darker TAMs are used to add eye and ear detail, and shade in the background. Image target intensities from left to right are 0.125, 0.375, 0.625, and 0.875.

We control the features depicted at the current resolution and tone with an importance function. This function identifies the presence and strength of features over an image region. Importance functions were introduced by Streit and Buchanan [13] and are derived from the input image. We generate a tone importance function $T_\ell(x, y)$ and a variance importance function $V_\ell(x, y)$ for each mipmap level ℓ and combine them according to a user defined weight ω. Stronger importance functions may overwhelm the others. We apply a histogram equalization to each function beforehand, so all the functions have a roughly equal distribution.

Our fine level tone function T_0 is the relative inten-

sity of the base image. We recursively calculate T_ℓ at the smaller mipmap levels by averaging the points in $T_{\ell-1}$, as in Streit and Buchanan [13].

The variance function used by Streit and Buchanan [13] measures the local variance of the image at each pixel. This function is too sensitive to noise. Smoothing the noise using a Gaussian filter may result in the loss of some poorly-defined features. The Canny edge detector [1] is not as sensitive to noise and does not blur the edges together, unlike other edge detectors, but does not detect Y-junctions well. We compute our variance function using a method similar to the Canny edge detector, but compensate for the loss of Y-junctions by avoiding thresholding points with a gradient magnitude that are close, both distance-wise and value-wise, to a local maximum.

A high ω is best for images with poorly defined features or images with a low range of intensity values. A low ω is best for images with little noise and well defined features. TAMs generated with a high ω tend to be more "cartoon-like", and TAMs with a low ω are more realistic.

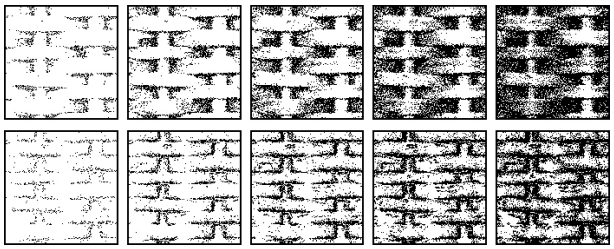

Figure 2: The top stippled images were generated with $\omega = 0$ and the bottom images were generated with $\omega = 1$. Image target intensities from left to right are 0.1, 0.3, 0.5, 0.7, and 0.9.

Our coarse distribution functions favour areas with sharp edges, so we define a new level-based variance weight

$$\omega\prime(\ell) = 1 - (1 - \omega) \cdot \frac{levels - \ell}{levels}, \qquad (1)$$

where $\ell = 0$ is the finest mipmap level.

Our lighter and coarser TAM images favour areas with stronger features. We calculate the final importance function

$$D_{\ell,t}(x, y) = (1 - \omega\prime) \cdot T_\ell(x, y) + \omega\prime \cdot V_\ell, \qquad (2)$$

where t is the target tone ($t = 0$ is white.)

Our darker TAM images favour hatches over outlines, so we scale down $D_{\ell,t}(x, y)$ by

$$s(t) = 1 - t. \qquad (3)$$

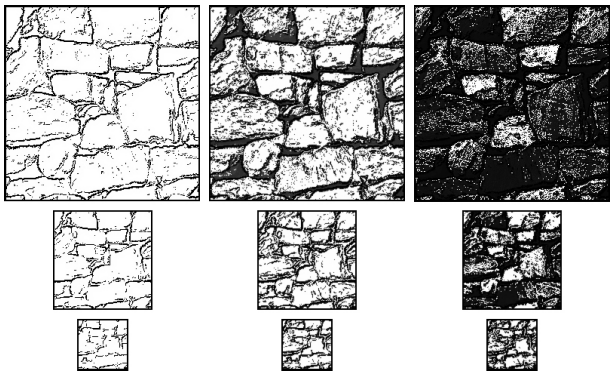

Figure 3: The importance function with $\omega = 0.5$ at different tones and resolutions. The lighter and coarser importance functions favour strong features. The coarser importance functions favour strong edges. Image tones from left to right are 0.1, 0.5, 0.9.

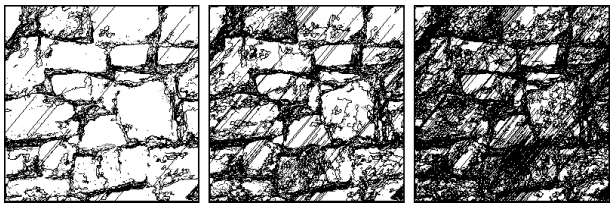

Figure 4: The probability of a stroke becoming a hatching stroke increases as the tone darkens. Image target intensities from left to right are 0.3, 0.5, and 0.7.

3.3 Drawing outlines and hatches

Pen-and-ink artists use outlines to define shapes and suggest texture features [5]. Outlines can occur almost anywhere, but generally trace paths between points with a similar value.

We draw outlines perpendicular to the direction of the gradient of the input image. We stop drawing the outline if the difference in intensity between two adjacent pixels is greater than a threshold, similar to Streit's and Buchanan's importance-driven halftoning algorithm [13].

Hatches are individual lines laid "side by side, or crossed in series" [5]. Hatching does not convey shape, texture, or any other illustrative information except for tone. Hatches occur in dark areas or areas with almost constant tone.

We draw hatches in two directions, θ_0 and θ_1. The two angles should be almost, but not quite, perpendicular to one another, or else they will look forced. We start with the hatches all in the same direction, change direction when the tone becomes darker than a threshold tone

τ_0, and choose a hatching direction randomly when the tone becomes darker than another threshold tone τ_1.

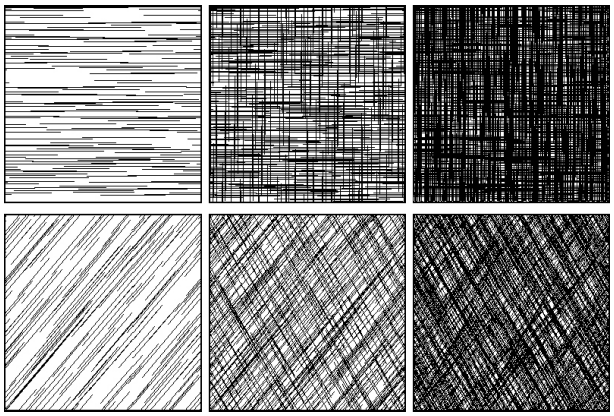

Figure 5: Hatching strokes look more natural when θ_0 and θ_1 are not perpendicular. Top: $\theta_0 = 0$ and $\theta_1 = 90$. Bottom: $\theta_0 = 50$ and $\theta_1 = 120$. These images were generated with $\tau_0 = 0.45$ and $\tau_1 = 0.65$. Image target tones from left to right are 0.3, 0.5, and 0.7

If the gradient direction at the start of the line is uncertain, i.e. the gradient magnitude is less than some threshold, the stroke is drawn as a hatch. Otherwise, the stroke is drawn as an outline.

We must be able to seamlessly tile the TAM images. We assume the base image can be tiled in a seamless manner, so when a stroke being drawn falls off one side of the image, it is continued at the corresponding position on the opposite side.

Using the light-to-dark, coarse-to-fine order of drawing TAM images has additional benefits when using outlines and hatches. Copying the image in light-to-dark order ensures the darker TAM images retain some of the detail present in the lighter images. Carefully placing strokes on coarse images and copying them onto the finer images ensures there are always a few well-placed strokes on the finer resolution images.

3.4 Scaling pen strokes

Praun et al. [11] redraw straight strokes on the higher resolution images by magnifying only the length of the stroke, because people expect strokes to have the same width in pixel space, and the same length in object space.

We cannot apply this same technique to strokes such as outlines that are not straight, because the magnified stroke will look more like a series of line segments instead of a smooth and natural pen stroke. If we want to draw a stroke on levels 0 to ℓ, we first draw the outline on the finest level, basing the direction of the outline on the importance function at level ℓ, then we shrink the stroke to fit on coarser levels by scaling it down lengthwise.

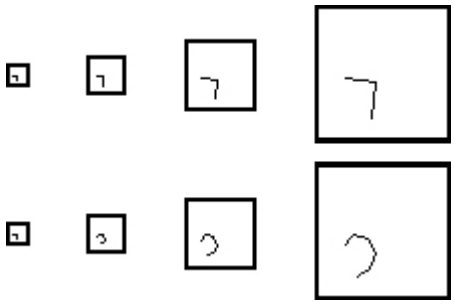

Figure 6: Top: the stroke is drawn on the coarsest image and scaled to fit the finer images. Bottom: the stroke is drawn on the finest image and scaled to fit the coarser images.

4 Conclusion.

In this paper, we introduced a method for representing surface textures in a pen-and-ink style suitable for a real-time rendering. We used the previously developed tonal art maps approach and pre-rendered simulated pen-and-ink representations of the input image into textures of various resolutions and tonal intensities. Drawing primitives are distributed using a probabilistic algorithm according to the importance function derived from the input texture. We guarantee frame-to-frame coherence by copying drawing primitives from the light tonal textures into dark ones. We simulate such pen-and-ink techniques as outlining, texture detailing, hatching, and stippling by varying importance function, the length of strokes and their directions.

The main advantage of our technique over the previous methods is the use of multi-scale importance functions that control the depiction of features at various tones and resolutions.

The resulting real-time rendering effectively depicts surface textures and shading at various resolutions and illumination levels. However, the TAM-based approach to simulation of pen-and-ink style has a number of limitations. Due to texture blending, drawing primitives may appear blurred at intermediate mip-map levels. Also, the primitives appear "stuck"' to object surfaces, thus approximation of loose pen drawing styles is difficult.

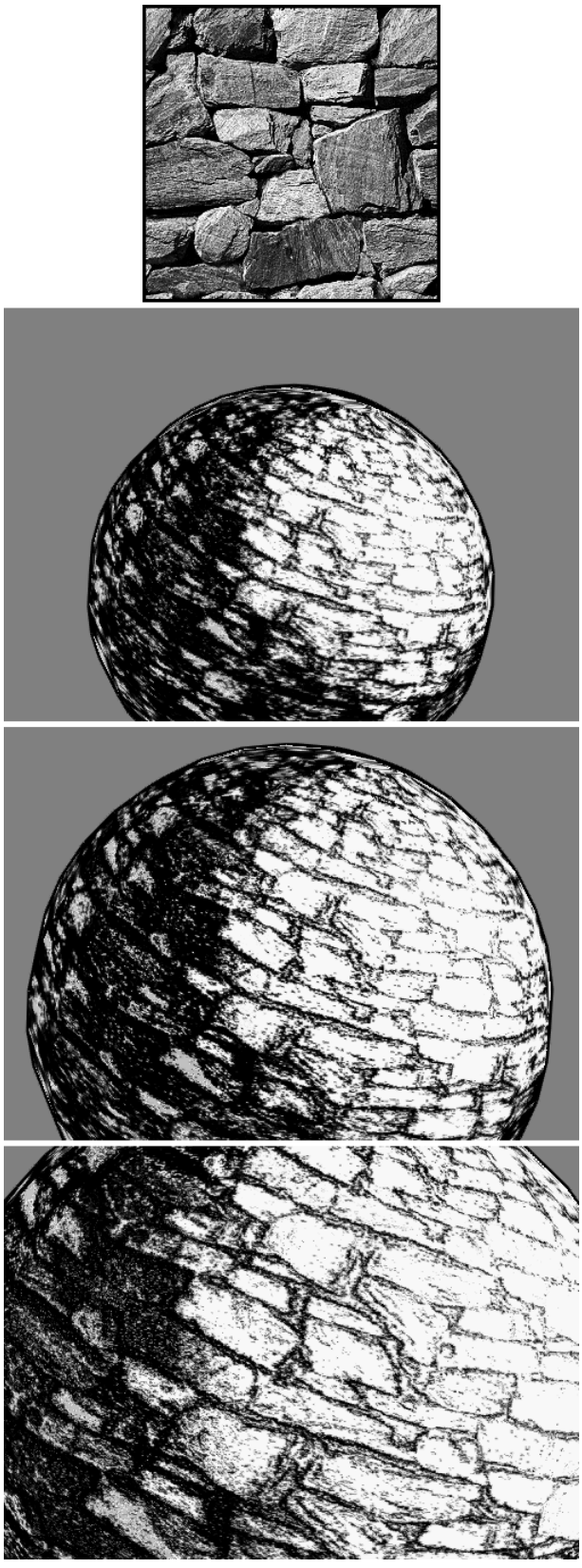

Figure 7: A sphere with a stippled rock TAM at different resolutions.

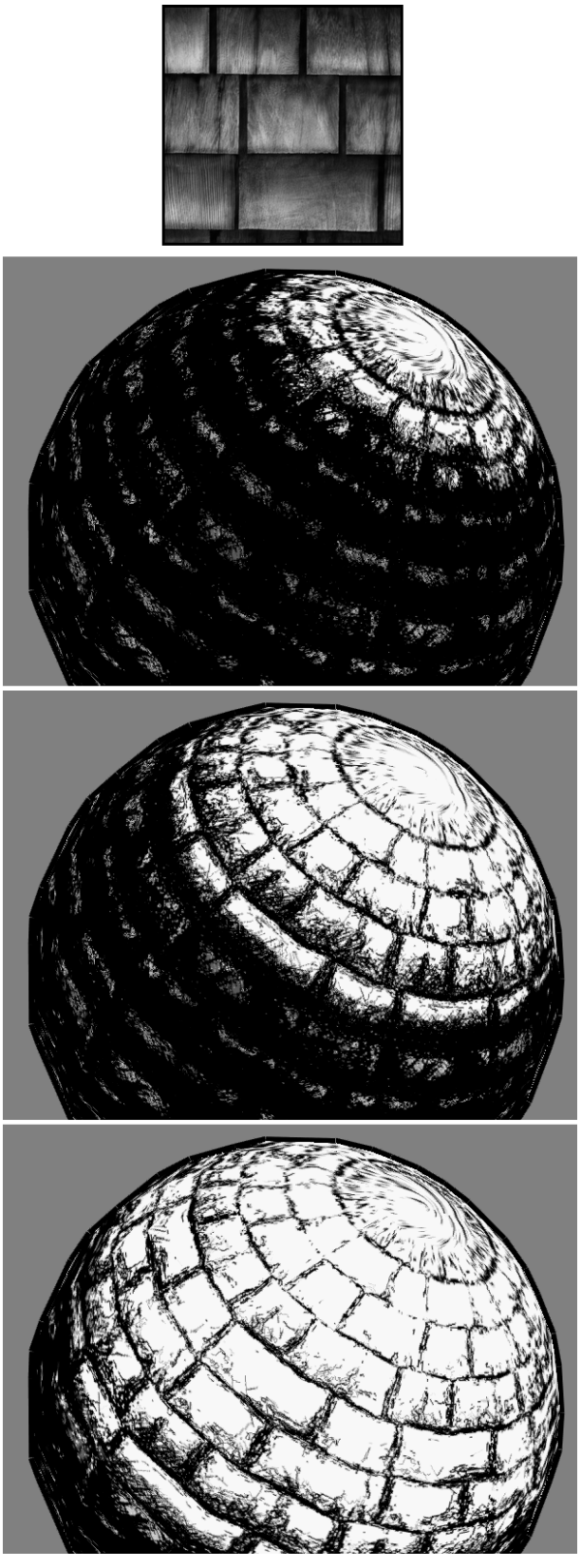

Figure 8: A sphere with hatched shingles TAM at different light settings.

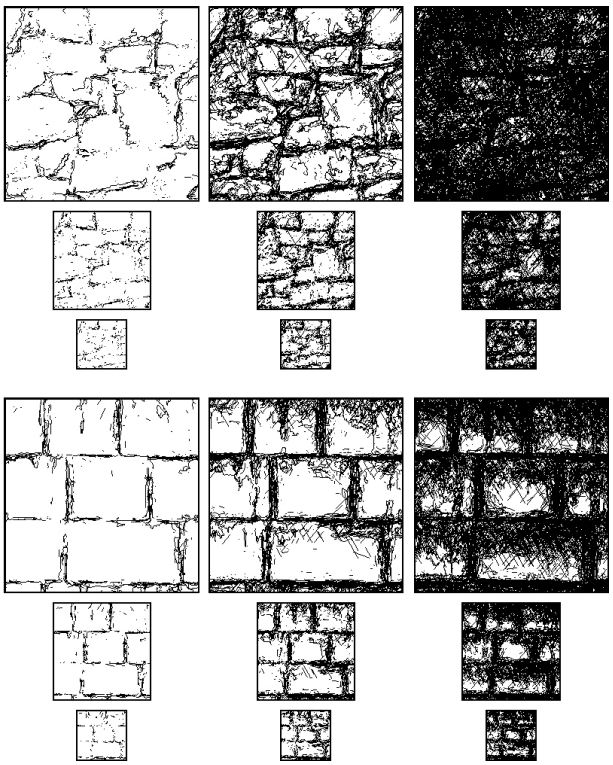

Figure 9: Top: A rock TAM generated with $\omega = 0.5$. Bottom: A shingle TAM generated with $\omega = 0.25$. Image target intensities from left to right are 0.1, 0.5, and 0.9.

References

[1] John F. Canny. A computational approach to edge detection. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 679–698, 1986.

[2] Derek Cornish, Andrea Rowan, and David Luebke. View-dependent particles for interactive non-photorealistic rendering. In *Graphics Interface 2001*, pages 151–158, 2001.

[3] Bert Freudenberg, Maic Masuch, and Thomas Strothotte. Real-time halftoning: A primitive for non-photorealistic shading. In *Thirteenth Eurographics Workshop on Rendering*. Springer-Verlag Wien New York, 2002.

[4] Bruce Gooch and Amy Ashurst Gooch. *Non-Photorealistic Rendering*. A K Peters Ltd, 2001.

[5] Arthur L. Guptill. *Rendering in Pen and Ink*. Watson-Guptill Publications, New York, 1977.

[6] Matthew Kaplan, Bruce Gooch, and Elaine Cohen. Interactive artistic rendering. In *Proceedings of NPAR 2000*, pages 67–74. ACM Press, 2000.

[7] Michael A. Kowalski, Lee Markosian, J. D. Northrup, Lubomir Bourdev, Ronen Barzel, Loring S. Holden, and John Hughes. Art-based rendering of fur, grass, and trees. In *Proceedings of SIGGRAPH 1999*, pages 433–438. Addison Wesley Longman, 1999.

[8] Aditi Majumder and M. Gopi. Hardware-accelerated real time charcoal rendering. In *Proceedings of NPAR 2002*, pages 59–66. ACM Press, 2002.

[9] Lee Markosian, Barbara J. Meier, Michael A. Kowalski, Loring S. Holden, J.D. Northrup, and John F. Hughes. Art-based rendering with continuous levels of detail. In *Proceedings of NPAR 2000*, pages 59–66. ACM Press, 2000.

[10] Barbara J. Meier. Painterly rendering for animation. In *Proceedings of SIGGRAPH 1996*, pages 477–484. ACM Press, 1996.

[11] Emil Praun, Hugues Hoppe, Matthew Webb, and Adam Finkelstein. Real-time hatching. In *Proceedings of SIGGRAPH 2001*, pages 579–584. ACM Press, 2001.

[12] Adrian Secord, Wolfgang Heidrich, and Lisa Streit. Fast primitive distribution for illustration. In *Thirteenth Eurographics Workshop on Rendering*, pages 215–226. Springer-Verlag Wien New York, 2002.

[13] L. M. Streit and J. Buchanan. Importance driven halftoning. In *Proceedings of Eurographics 1998*, pages 207–217, 1998.

[14] Thomas Strothotte and Stefan Schlechtweg. *Non-Photorealistic Computer Graphics: Modeling, Rendering, and Animation*. Morgan Kaufmann Publishers, 2002.

[15] Oleg Veryovka. Animation with threshold textures. In *Graphics Interface 2002*, pages 9–16, 2002.

[16] Matthew Webb, Emil Praun, Adam Finkelstein, and Hugues Hoppe. Fine tone control in hardware hatching. In *Proceedings of NPAR 2002*, pages 53–58. ACM Press, 2002.

[17] Georges Winkenbach and David H. Salesin. Computer-generated pen-and-ink illustration. In *Proceedings of SIGGRAPH 1994*, pages 91–100. ACM Press, 1994.