Stylized Rendering Using Samples of a Painted Image

Chung-Ren Yan, Ming-Te Chi, Tong-Yee Lee, *Member*, *IEEE*, and Wen-Chieh Lin, *Member*, *IEEE*

Abstract—We introduce a novel technique to generate painterly art maps (PAMs) for 3D nonphotorealistic rendering. Our technique can automatically transfer brushstroke textures and color changes to 3D models from samples of a painted image. Therefore, the generation of stylized images or animation in the style of a given artwork can be achieved. This new approach works particularly well for a rich variety of brushstrokes ranging from simple 1D and 2D line-art strokes to very complicated ones with significant variations in stroke characteristics. During the rendering or animation process, the coherence of brushstroke textures and color changes over 3D surfaces can be well maintained. With PAM, we can also easily generate the illusion of flow animation over a 3D surface to convey the shape of a model.

Index Terms—Nonphotorealistic rendering, stylized rendering, painterly art map (PAM), stroke.

1 Introduction

Artistic rendering has been one of the ultimate goals in nonphotorealistic rendering (NPR) research. A natural way to generate artistic rendering would be learning from artists' works and imitating their painting style in an NPR system. It is, however, very difficult to transfer artists' painting styles since they can manipulate a rich variety of stroke patterns and color changes to deliver their creation and emotion. Although there has been a great advance in NPR in the past decade, real-time stylized rendering on 3D models has been a challenging problem as the rendering system needs not only to generate a variety of brushstroke styles flexibly but also to maintain frame-to-frame coherence of brushstrokes stably. Moreover, the rendering system needs to be computationally efficient so that real-time rendering can be achieved. These difficulties motivate us to develop a real-time NPR system to render 3D models with a gallery of styles presented on actual paintings from talented artists.

Existing approaches to stylized rendering onto 3D models work well on 1D and 2D line-art strokes [1], [2], but an artist-designed brush texture map is required to paint color strokes on 3D models [3], [4]. There are still many limitations to these approaches in rendering 3D models regarding the requirements of stroke variety, stroke coherence, and real-time computation. Praun et al. [1] proposed a Tonal Art Map (TAM) to render hatching strokes over 3D surfaces in real time. These hatch images are generated by adding hatching strokes repeatedly to represent different tones. This is a very good method for handling line-art strokes. However, for non-line-art strokes such as those from the painting examples in Fig. 1, it is very difficult to generate a good TAM using the Praun et al. approach. To solve this difficulty, we present a novel technique, called painterly art map (PAM), from paint samples of a given artwork. PAM can then be used to light 3D models, making them as though they have been painted using the same style and technique as the input painting.

Our system requires users to select several paint samples from an actual painting image (see Fig. 2). PAM consists of a sequence of tone images. Texture synthesis is an ideal mechanism for generating each tone image from each selected sample. Many celebrated texture methods [5], [6], [7], [8], [9] are good candidates for generating good-quality tone maps for our application. However, there are several challenging issues that need to be tackled further. When the existing texture synthesis is used to generate each tonal map independently, it is difficult to guarantee good coherence in stroke textures and color changes as one tonal map is switched to another. Sudden changes in stroke and color are always perceived in this manner. To alleviate this problem in coherence, the synthesis process needs to ensure the similar stroke structures between two consecutive PAMs. However, any overemphasis on this issue can lead to the loss of the details of the original paint samples from the synthesized maps.

To solve the above difficulties, we develop a novel multiresolution patch-based synthesis algorithm to generate each PAM. Our algorithm synthesizes the coarse levels of a map to have a stroke orientation similar to the first PAM, and therefore, the coherent orientation of consecutive PAMs can be well maintained. Our algorithm then synthesizes the finer levels with an emphasis on preserving the stroke textures and colors of the original paint sample. The major contributions of this paper are described as follows: We present a novel NPR system for stylized rendering. Given a sample painting, our NPR system semiautomatically builds PAMs by which it generates rendered results with artistic styles derived from an input painting. The PAM technique

- *C.-R. Yan, M.-T. Chi, and T.-Y. Lee are with the Computer Graphics Group/Visual System Laboratory, Department of Computer Science and Information Engineering, National Cheng-Kung University, No. 1, Ta-Hsueh Road, Tainan 701, Taiwan, R.O.C. E-mail: chongren.yan@gmail.com, dodowell@csie.ncku.edu.tw, tonylee@mail.ncku.edu.tw.*
- *W.-C. Lin is with the Department of Computer Science, National Chiao-Tung University, Hsinchu, Taiwan, 1001 Ta Hsueh Rd., Hsinchu 300, Taiwan 300, R.O.C. E-mail: wclin@cs.nctu.edu.tw.*

Manuscript received 27 June 2007; revised 4 Sept. 2007; accepted 23 Sept. 2007; published online 4 Oct. 2007.
Recommended for acceptance by H. Hauser.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2007-06-0077.
Digital Object Identifier no. 10.1109/TVCG.2007.70440.

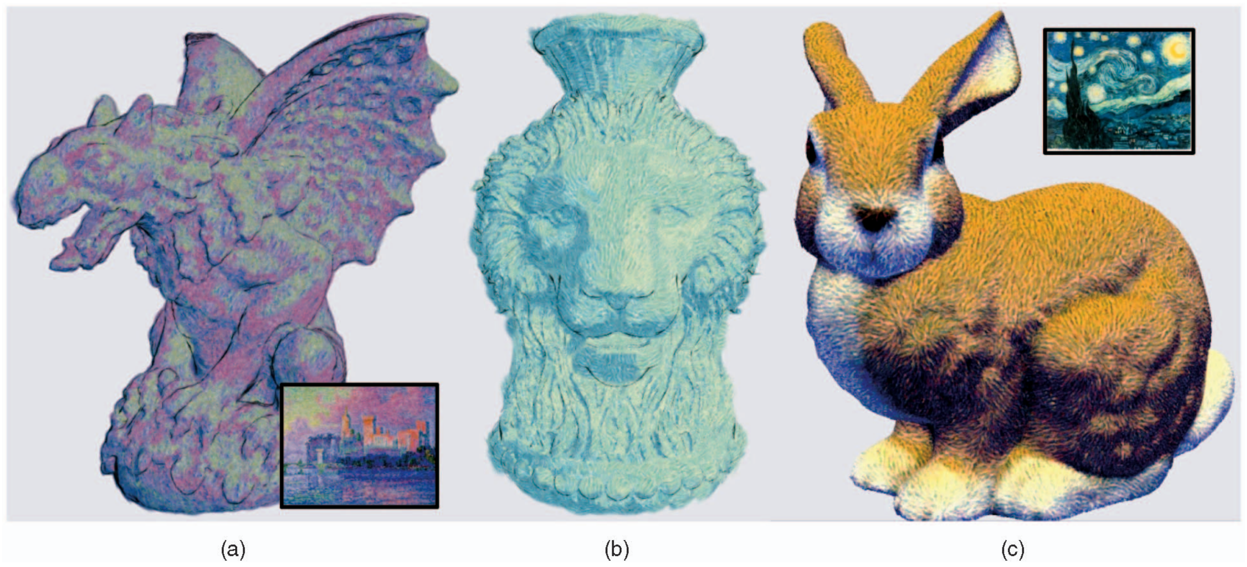


Fig. 1. The input painted images and our results. (a) Rendering with "Palais des Papes Avignon" painted by Paul Signac. (b) and (c) Rendering with "The Starry Night" painted by Van Gogh. Originally, the models in (a) and (b) are uncolored, and the model in (c) is colored.

can maintain the coherence of brushstroke textures and color changes over 3D surfaces. An additional feature of our system is the ability to animate the PAMs, giving the illusion of strokes flowing over the surface. Finally, we also apply the proposed NPR system to render isosurfaces from medical data as an example of its potential applications.

2 Related Works

NPR techniques can generate artistic styles for images. Many NPR approaches concentrate on converting photographic images into artistic images. Hertzmann [10] presents an introductory survey of stroke-based rendering in this 2D NPR category. Example-based NPR techniques [6], [11] are very popular and successful in generating visually pleasing results. The example-based approach transfers the styles of sample paintings to the synthesized paintings. In addition to 2D NPR, several techniques have also been developed to render 3D scenes, for example, the polygonal mesh technique [1], [12], [13], [14], [15], [16], [17], [18] and the 3D points technique [4], [19]. These 3D NPR techniques paint different strokes over the surface of a 3D model, including simple line drawings, decal strokes, paintings, and hatching strokes. The stroke variations on a 3D surface, such as color, orientation, size, density, and texture, can convey features of models and lighting variations. The common objective of NPR-stylized animation from 3D models is to ensure frame-to-frame stroke coherence.

Based on hand-drawn hatching patterns, Praun et al. [1] presented a pioneering TAM technique to render a 3D surface using hatching strokes in real time. TAM is constructed by repeatedly adding hatching strokes to represent the tone

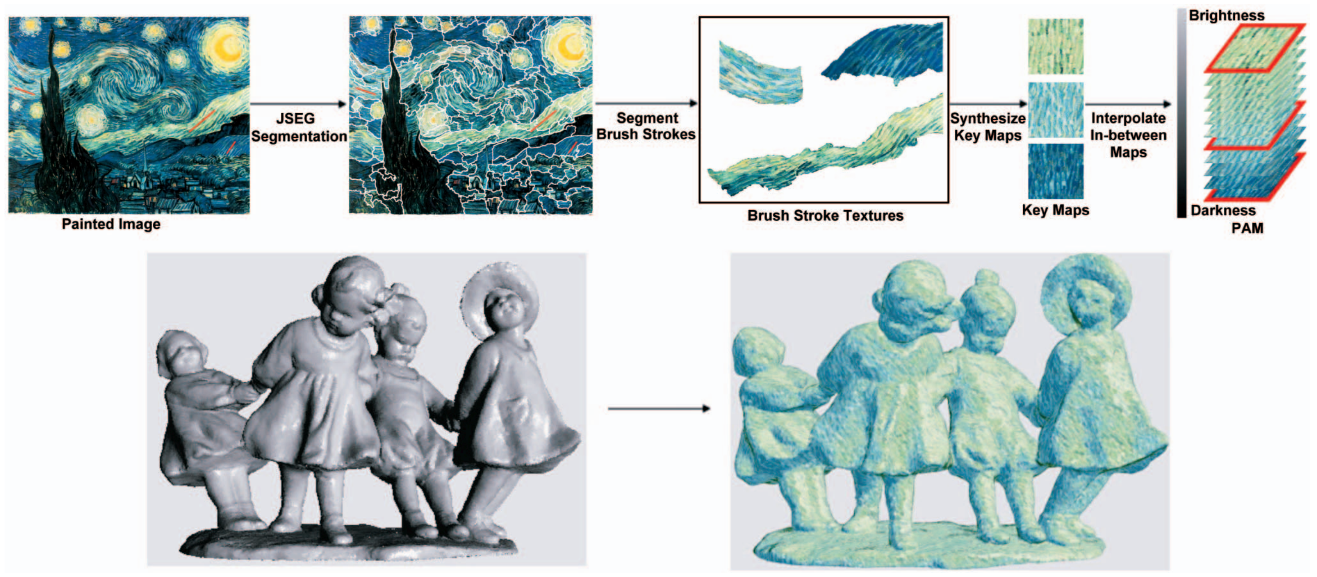


Fig. 2. An overview of our NPR system. The user can sketch guiding lines and then use JSEG and our merging method to segment brushstroke textures. Finally, our texture synthesis method is used to construct the key maps and PAM, and then, PAM is used to render the input model.

variance. Surfaces are then rendered by selecting the textures with the approximate tone in the TAM. Webb et al. [2] later utilized a hardware-assisted 3D volume texture to better control tone (including color and texture) and to avoid blending or aliasing artifacts in the TAM-based approach. Although TAM is an efficient method for rendering line-art strokes, it has difficulty in representing other painting styles since most painting strokes are much more complicated than the line-art style. Kulla et al. [3] shaded 3D models using painted samples by artists to represent dark-to-bright transitions. Color and texture variations in paints were considered in shading 3D models. However, this technique requires an artist to provide a real paint example with shading variations from dark to light. In this paper, we consider a more general and challenging problem in the same vein as [3]. From a painted image, the user arbitrarily selects regions, from which our system generates a stylized image and animation that looks as though it has been painted using the same technique. These segmented regions feature a variety of brushstrokes rather than a single type.

Our work is also related to texture synthesis. Although existing texture synthesis methods [5], [6], [7], [8], [9] are efficient in generating good-quality textures, they synthesize textures independently and without considering the coherence and translation smoothly within different textures. These texture synthesis algorithms are not suitable for PAM synthesis since we need the textures in the PAM to have a similar stroke distribution and to preserve the original stroke characteristics of painted images. In addition, Zelinka and Garland [20] proposed a jump map to create search links for each pixel. These links record the possible adjacent pixels in the same image and make the synthesis process run very quickly. This work can be considered as a prior work of the coherence search space (CSS) algorithm introduced in our paper. Cohen et al. [21] modeled textures into equal-sized squares and assigned a color on each edge. If the adjacent tiles have the same color of abutting edge, they can be arranged side by side. This method can synthesize arbitrary-size textures easily and quickly.

Our PAM synthesis problem is most similar to the transitional texture synthesis problem in which a sequence of transitional textures between two texture samples is generated. These transitional textures form a continuous spectrum on which the geometric structure and color statistics are continuously varied. Several authors have tackled the challenge of transitional texture synthesis in the past. Heeger and Bergen [22] and Bar-Joseph et al. [23] worked on a related problem of mixing textures from multiple samples, but they did not address the problem of generating transitional textures. Zhang et al. [5] synthesized a spatially varying texture from two input textures. Their approach requires a user to manually specify textons and to choose suitable input textures for blending. Matusik et al. [24] proposed a morphable interpolation for textures relying on a large database in which textures are manually rotated and scaled to approximately the same orientation and feature size. In general, these approaches require intensive user interventions and are mostly designed for structural textures as they exploit sharp edge features or strong color contrast to align the texture elements of a structural texture. In PAM synthesis, the edges or colors of paint strokes may not be obvious in actual paintings. These algorithms cannot be directly applied to PAM synthesis. Therefore, we propose a multiresolution texture synthesis algorithm to construct PAMs in which the variations of brushstroke textures are generated to represent different tones.

3 System Overview

Fig. 2 shows an overview of the proposed NPR system. The user starts by selecting a sample painting and interactively selecting representative regions within that image. The system then automatically builds a PAM in two steps: first, by generating a set of "key maps," one for each region selected by the user and, then, by completing the PAM by generating in-between maps that represent a painter's style at intermediate tone values between those associated with the key maps. A novel multiresolution patch-based texture synthesis is used during this stage. Our rendering method is based on Chi and Lee's work [4], which uses a splat representation. Each splat is textured using an image from the PAM, selected according to the computed tone at that surface location. PAM textures have an apparent stroke direction; thus, the stroke textures are aligned with directions computed to follow the principal curvature directions on the surface. Furthermore, the system can animate the rendered textures, giving the illusion of strokes flowing over the surface. This effect can be used for animation (for example, simulating the flow of water or clouds) and can also aid in the perception of a shape.

4 Painterly Art Map Construction

4.1 Painting Image Segmentation Using Scribbles

From a painting, our PAM technique constructs a sequence of tone images from segmented irregular regions. We apply the JSEG algorithm [25] to segment the sample painting. JSEG introduces a J value to measure local image inhomogeneities. However, JSEG was originally designed for natural scenes rather than painted images and does not consider brushstroke directions, which are a very important attribute of paintings. Therefore, a modified JSEG algorithm is used and described as follows: In the spatial segmentation and merging stage of JSEG, we set a high threshold on the J value to obtain an oversegmented result, as shown in Fig. 2. Next, the user scribbles some lines on the painting to indicate the regions of interest (ROIs) (one line at each ROI). For each segmented region by JSEG, we then compute the average intensity $\overline{I}_r$, standard deviation of intensity σ_r, and average edge direction $\overline{D}_r$ in the region r. $\overline{D}_r$ is obtained by averaging the gradient directions at the pixel locations with the highest 20 percent gradient magnitudes. Finally, region growing is applied to merge oversegmented regions in which the $\overline{I}_r$, σ_r, and $\overline{D}_r$ values are similar. In the region-growing process, seeds are put on those regions with scribbled lines. The scribbled lines serve as the user's hints for specifying ROIs. Thus, our system allows the user to flexibly select brushstroke textures from a sample painting.

In general, the local variation of brush textures is not significant within each segmented region, but the global

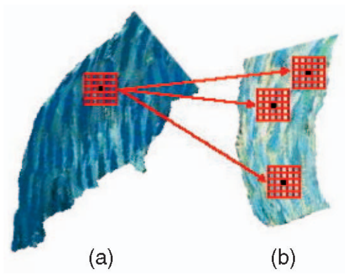


Fig. 3. CSS construction from brushstroke texture A to B. Each pixel in A has multiple links to the pixels in B whose neighborhoods are similar in stroke color and stroke orientation.

variation of brush textures across all segmented regions can be significant. To reduce the difficulty of PAM synthesis in later steps, we roughly align the stroke orientation and normalize the stroke density in all segmented regions. We rotate segmented regions from their original directions $\overline{D}_r$ to a vertical direction. To normalize the stroke density, the user scribbles two line segments on each region. The distance between these two line segments tells us the approximate stroke density in this region. We compute this distance information for all segmented regions and pick the one with the median distance value. Using this median value, we resize all segmented regions so that their new distance values are almost identical in all resized regions.

4.2 Coherence Search Space

A PAM consists of several tone images synthesized from extracted brushstroke textures. As brushstroke textures represent strokes with different luminance in painted images, the coherence among different synthesized textures is very important if we want to use a PAM to shade a 3D model. In Section 4.1, we adjust all segmented brush textures to approximately the same stroke density, orientation, and stroke size. This adjustment is helpful for the subsequent synthesis procedure that will be used to maintain the coherence of textures in PAM.

Coherence means that the distribution of the strokes must be similar in all the textures in PAM and that the shapes of the strokes in the original brush textures must also be preserved. However, it is difficult to place strokes in similar relative positions in each synthesized texture. An intuitive way to solve this problem is to select feature points and align them in the synthesis process. This is a convenient method of achieving coherence in PAM if the structures of the textures are almost regular, such as brick textures. However, in most paintings, the strokes have irregular structures and distribution. It is difficult to determine the feature points automatically, and it is also too tedious for a user to select all the features in brushstroke textures. Hence, the first step of our texture synthesis algorithm is to automatically create a CSS between two brushstroke textures. The CSS establishes the correspondence of patches between two brushstroke textures in terms of feature coherence. Fig. 3 illustrates an example of a CSS from A to B.

Let A and B be two brushstroke textures that we want to construct a CSS from A to B. N_p is a neighborhood centered at pixel p. We define F_p as the feature map of a texture obtained by applying the Gaussian smooth filter and edge detection operator to N_p and H_p, the hue variations of pixels in N_p. To measure the similarity between two patches centered at pixels p and q, we define a distance function $S_{css}(p, q)$:

$$S_{css}(p, q) = SSD(F_p, F_q) + SSD(H_p, H_q), p \in A, q \in B, \quad (1)$$

$$H_p = \frac{h_p - \mu}{\sigma}, \quad (2)$$

where SSD is the sum of the squared differences between two image patches, h_p is the hue value of pixel p, and μ and σ are the mean and the standard deviation of the hue values of all pixels, respectively. In the CSS, each pixel in one brushstroke texture has multiple links to the pixels in the other brushstroke texture whose neighborhoods are most similar. We compute S_{css} for all pixel pairs between two brushstroke textures, and a link is built between pixels p and q if S_{css} is less than a threshold H_{css}. We use $M_{A\rightarrow B}(p)$ to denote the set of all pixels in B linked to a pixel p in A:

$$M_{A\rightarrow B}(p) = \{q \mid S_{css}(p, q) < H_{css}, p \in A, q \in B\}. \quad (3)$$

4.3 Key Map Generation Using Multiresolution Synthesis

We synthesize a set of rectangular textures called key maps from brushstroke textures extracted from a painted image (Section 4.1). There are two reasons for generating rectangular key maps. First, extracted brushstroke textures usually have irregular shapes. It is easier to construct PAM based on stroke textures in a rectangular shape. Second, we can control the stroke orientation and stroke size in the key map construction process more finely. The basic idea is to disassemble a brushstroke texture into small patches and align the stroke orientation of the small patches when reassembling them in the texture synthesis process.

To control the stroke orientation and stroke size in key maps, we develop a multiresolution patch-based synthesis algorithm in which cross-level patch selection and dynamic programming patch stitching are used to synthesize key maps. Our synthesis algorithm forces the coarser levels to have structures similar to the first key map, whereas the finer levels have their own detailed contents. In this way, the global orientations of all strokes are aligned, and the detailed characteristics of individual strokes are also preserved. In the followings, we will describe our cross-level patch selection approach and will not address the patch stitching problem as there have already been several mature techniques developed in the texture synthesis field, such as dynamic programming [6], graph cut [8], and image feathering [26]. Readers who are interested in patch stitching can refer to these papers.

Our cross-level patch selection process is illustrated in Fig. 4. Given two brushstroke textures A and B extracted from a painted image, we construct their Gaussian pyramids $A_L, A_{L-1}, A_{L-2}, \ldots, A_1$ and $B_L, B_{L-1}, B_{L-2}, \ldots, B_1$, respectively. Here, $A_L = A$ and $B_L = B$ denote the brushstroke textures at the finest resolution level. Larger subscript values correspond to finer resolution levels. Our cross-level patch selection process (Fig. 4) consists of three major steps: 1) we first synthesize the first key map A'_L using a multiresolution texture synthesis algorithm modified from Wei and Levoy's

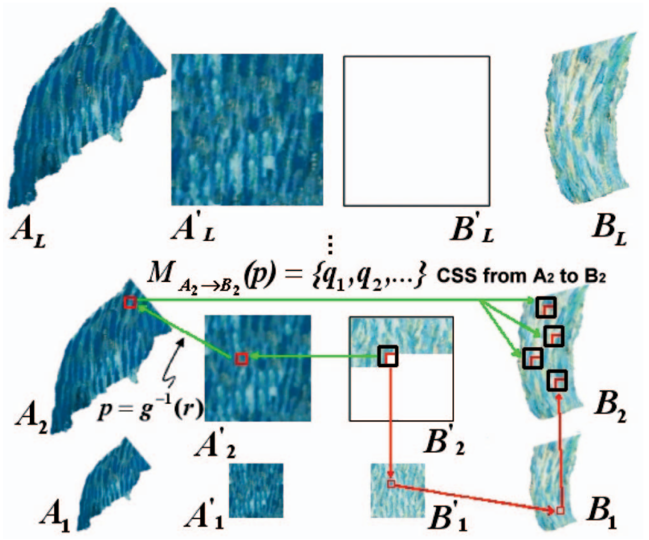


Fig. 4. Cross-level patch selection in multiresolution key map synthesis (levels 2 to L).

method [27].[1] The intermediate results from the multiresolution synthesis process of A_L are denoted as $A'_{L-1}, A'_{L-2}, \ldots, A'_1$. 2) Then, the coarsest level key map B'_1 is generated according to the CSS described in Section 4.2. 3) Finally, when synthesizing the finer level B'_i, the best matched patches are chosen based on the CSS and the correspondence of the prior level B'_{i-1}. The cross-level patch selection ensures that the stroke orientation in the synthesized B'_L is aligned with that in A'_L and the local color characteristics are similar to the brush texture B_L. More details for the second and third steps are described in the following paragraphs.

4.3.1 Key Map Synthesis at the Coarsest Level

We synthesize B'_L starting from the coarsest level. At the coarsest level, the patch selection process is shown in Fig. 5, where p and q denote a pixel in A and B, respectively, and r represents a pixel in A' and B' at the same location. We first trace where the patch $A'_1(N_r)$ was picked from A_1 based on the mapping $g(p) = r$ recorded during the synthesis process of A'_1 from A_1. Here, $g(p) = r$ denotes that N_r in A'_1 is originally copied from N_p in A_1. As the stroke orientation at the pixel r in B'_1 should match the stroke orientation at the pixel r in A'_1, we pick a set of candidate patches in B_1 whose stroke orientations are similar to $A_1(N_p) \approx A'_1(N_r)$ according to the CSS result between A_1 and B_1. The center of the best matched patch in B_1 is determined by

$$q^* = \arg\min_q d_{overlapping}(B_1(N_q), B'_1), \tag{4}$$

where $d_{overlapping}$ is the sum of the squared differences at the overlapping region between the current synthesized texture B'_1 and candidate patch $B_1(N_q)$, and $q \in M_{A_1 \to B_1}(g^{-1}(r))$. In $d_{overlapping}$ computation, we enforce L-shape neighbors as a constraint and use the feature-weighted function [28] to assign weights to each pixel in L-shape neighbors to calculate the similarity. In addition, their prioritized pixel

1. The modification we made is to change a pixel-based synthesis process to a patch-based one, which improves the speed and better preserves the global structure of a texture.

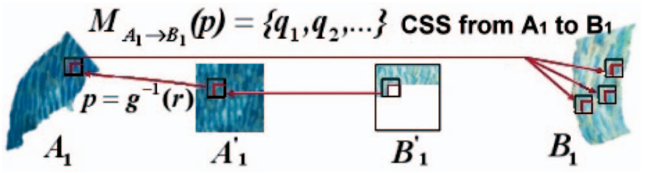

$M_{A_1\to B_1}(p) = \{q_1, q_2, \ldots\}$ CSS from A1 to B1
$p = g^{-1}(r)$
A_1
A'_1
B'_1
B_1

Fig. 5. Patch selection in key map synthesis at the coarsest level.

resynthesis method [28] is also applied to the overlapping regions of the patches to reduce the artifacts.

4.3.2 Key Map Synthesis at Other Levels

We will only describe the synthesis process at level 2 to simplify the notations since the synthesis process is the same from levels 2 to L. To synthesize at pixel r of B'_2, we first find the best matched patch $B_2(N_{q^*})$ in B_2 by applying the same patch selection process in the coarsest level to A_2, A'_2, and B_2:

$$q^* = \arg\min_q d_{overlapping}(B_2(N_q), B'_2). \tag{5}$$

In addition, we obtain another patch $B_2(N_{q^{**}})$ via the route of B'_1 and B_1. This can be done straightforwardly since the pixel correspondence from B'_2 to B'_1 and from B_1 to B_2 are simply downsampling and upsampling, respectively. Moreover, we can also trace where a patch in B'_1 is copied from B_1 by bookkeeping the synthesis process of B'_1 (similar to the way we obtain the mapping g).

$B_2(N_{q^*})$ and $B_2(N_{q^{**}})$ can be interpreted as the best matched patches in terms of the similarity to the first key map in a finer and coarser level, respectively. The best patch is then determined based on the following equation:

$$\begin{aligned} &min(d_{overlapping}((1 - w_2)B_2(N_{q^*}), B'_2), \\ &d_{overlapping}(w_2 B_2(N_{q^{**}}), B'_2)), \end{aligned} \tag{6}$$

where w_2 is used to adjust the influence of the first key map at a finer level relative to a coarser lever. Since the finer levels correspond to more localized features, we set a larger weight in finer levels to preserve the localized features of the stroke texture B. The weight w at level l is defined as $w_l = \frac{l}{l}$.

To construct the remaining key maps, we replace the stroke texture B by other extracted brushstroke textures one by one. In this way, we can generate a set of key maps whose stroke orientations are aligned with that of the first key map. Additionally, we make all synthesized PAMs tileable to increase the rendering speed at runtime. This can be easily done by setting the left and top boundaries of the textures as L-shape constraints when synthesizing the right and bottom boundaries of another texture.

4.4 In-Between Map Synthesis

After synthesizing the key maps in PAM, we use a linear blending method to generate in-between maps and make the variation between key maps smoother. The number of in-between maps N is decided by computing the difference of the average intensities of two adjacent key maps:

$$N = \frac{|\mu_A - \mu_B|}{k}, \tag{7}$$

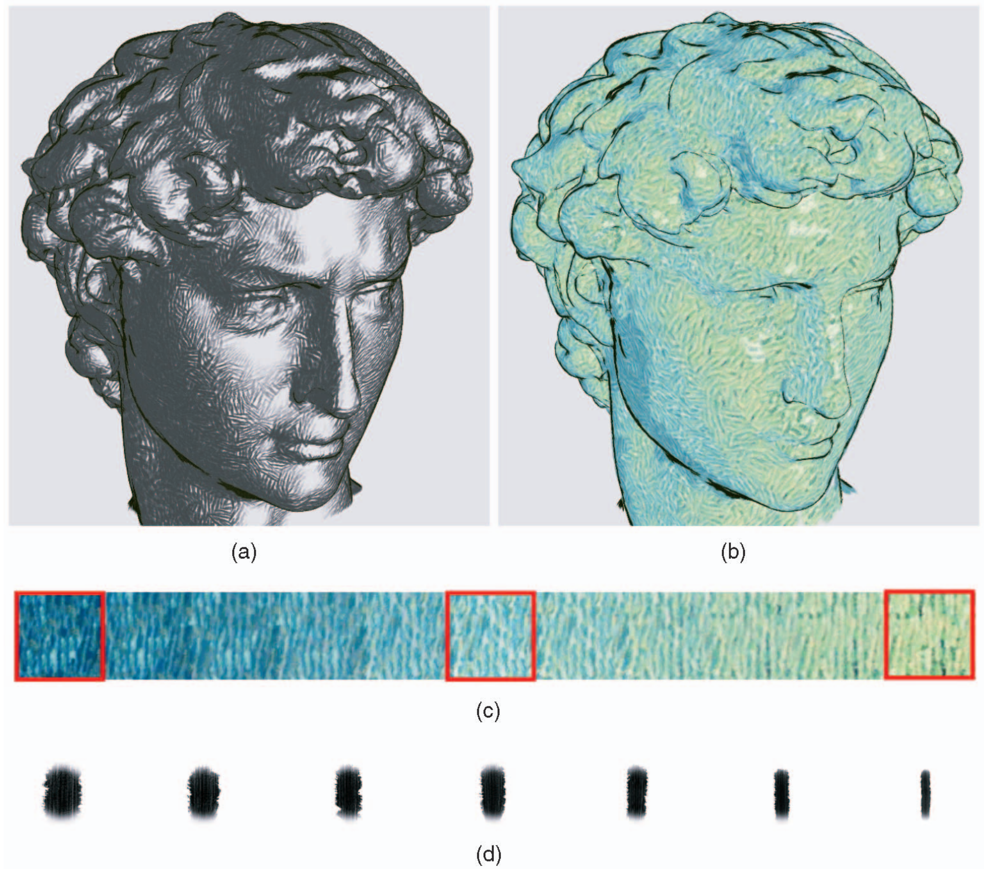


Fig. 6. Rendering comparison. (a) Rendering result by Chi and Lee's method [4]. (b) Our rendering result by PAM. (c) Synthesized results of PAM used in (b). (d) Stroke patterns specified by a user, from dark to bright, used in Chi and Lee's work [4].

where μ_A and μ_B denote the average intensities of two key maps I_A and I_B, respectively. The coefficient k is used to control the intensity incremental level between PAMs. $k = 5$ is used in this paper. The in-between maps IM are computed as follows:

$$IM = \alpha I_A + (1 - \alpha) I_B, \quad \alpha = 1, \frac{N-1}{N}, \frac{N-2}{N}, \ldots, 0. \quad (8)$$

An example result of in-between maps is shown in Fig. 6c.

5 Stylized Nonphotorealistic Rendering Using Painterly Art Maps

In this section, we apply PAMs to render a 3D model using the 3D painterly splat rendering framework proposed by Chi and Lee [4]. Their framework offers great flexibility in many aspects of NPR, including stylization and abstraction. Later, they apply their framework to anatomic visualization [29]. In the preprocessing stage, this framework converts the input mesh model into a multiscale segmented 3D point hierarchy and uses curvature information to compute smooth direction fields over a 3D surface. At rendering time, each 3D point is drawn as a splat primitive textured by a series of 2D brushstroke shapes, shown in Fig. 6d. In the spirit of Chi and Lee's framework, splats of varied sizes rendered on the screen are analogous to the multiple-size strokes drawn on the canvas. Our NPR is based on their framework with the following extensions: 1) replacing the stroke shape texture by PAMs, 2) enhancing the color transfer from PAMs to a colored model, and 3) animating the PAMs to generate strokes flowing over the surface.

5.1 Rendering with Painterly Art Maps

We render each splat as a single texture-mapped quad with a Gaussian-shaped alpha. Packing the PAMs into a 3D texture will make the texture and color transitions in PAMs vary continuously with the graphic hardware interpolation. Therefore, during rendering with PAMs, neighboring splats can potentially be drawn with varying brush textures and colors. We implement our splat rendering in three passes similar to surfels [30]: 1) the first pass renders splats into a depth buffer to obtain visibility information, 2) the second pass renders a splat color additively weighted with a Gaussian-kernel into a color buffer and accumulates the weights, and 3) the third pass normalizes the pixel colors using the accumulated weights in the color buffer. Moreover, to enrich our rendering, our system allows the user to specify different sets of PAMs for different parts of a model to mimic varying brush textures and colors in different areas of interest drawn by the artist. When zooming in the models, we simply enlarge the strokes instead of changing the stroke density in the current implementation.

5.2 Rendering with Luminance and Chromatic Variance of Painterly Art Maps

Luminance and chromatic variance are important characteristics of brushstrokes in PAMs. In this section, we present an interesting alternative to rendering models using PAMs. When rendering a colored model, the color of the rendered

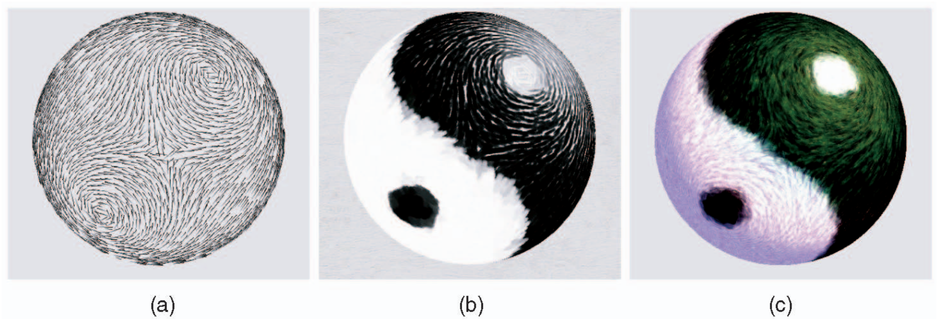


Fig. 7. (a) Direction fields. (b) Painterly rendering of a 3D Tai-Chi Ball [4]. (c) Animating the strokes along the direction fields in (a) to illustrate the shape of a Tai-Chi Ball. Animation results can be seen in the accompanying videos.

model can mainly be determined by the original color of the model, and both the luminance and chromatic variance of PAMs are used to modify the model colors, that is, adding the fine details of brushstrokes in PAMs on top of the model colors. The color of a splat is obtained by composing the model color and the PAMs in the YUV color space, which is defined in terms of one luminance and two chrominance components. Given a set of PAMs, all PAM images are converted from the RGB color space to the YUV color space and stored as a 3D texture in floating-point format. Each entry of this 3D texture is computed by

$$texel(u, v, w) = texel(u, v, w) - \mu, \tag{9}$$

where (u, v) are texture coordinates on the wth image of PAMs, and w is the tone coordinate. Each $texel(u, v, w)$ value is a YUV tuple (Y, U, V), and $\mu = (\mu_Y, \mu_U, \mu_V)$ are the mean values of each channel in the YUV color space for a set of PAMs. In rendering, the RGB color S_{RGB}^{model} of a splat S is first converted into its YUV color S_{YUV}^{model}, and the rendering YUV color of a splat S_{YUV} is then composed by

$$\begin{aligned} S_{YUV} = \{&S_Y^{model} + \alpha \cdot texel(u, v, w)_Y, \\ &S_{UV}^{model} + \beta \cdot texel(u, v, w)_{UV}\}, \end{aligned} \tag{10}$$

where α and β are parameters to control the strength of luminance and chromatic of PAMs independently, and w is determined by the lighting condition (that is, the dot product of the lighting vector and the splat normal vector). Finally, S_{YUV} is converted to S_{RGB} again to render splat S.

5.3 Animating Stroke Flow with Painterly Art Maps

In practice, artists orient strokes along some directions to express varied artistic effects. Many cognitive scientists and visual artists have suggested motion as visual cues for the perception of shape and depth [31]. In Chi and Lee's work [4], they orient strokes along the direction field to represent varied artistic effects. However, their framework only presents still stylized illustration of shapes without the capability to animate their textured splats. With the advantage of PAMs, we can easily generate an illusion of flow motion in two steps: 1) rotate the coordinates of a splat along its direction field and 2) update the texture coordinates of a splat at time t by the following:

$$\begin{aligned} u_t &= u_0 + NoiseOffsetU, \\ v_t &= v_0 + NoiseOffsetV + t \times FlowOffset, \end{aligned} \tag{11}$$

where (u_0, v_0) is the texture coordinate of a splat before flow animation, *NoiseOffsetU* and *NoiseOffsetV* are small offset constants of u and v and computed using Perlin noise. The flow speed is controlled by a parameter *FlowOffset*. Our PAM is tileable, as mentioned in Section 4.3; therefore, moving and scaling the texture coordinate will not induce artifacts. In addition, we construct the PAM to make strokes aligned along the v direction in the synthesis step. Therefore, as texture coordinates move along a direction (that is, v direction in (11)), an illusion of flow animation is generated. The splat was oriented along its direction field, and therefore, this modification makes a splat look as if it is flowing along its direction field.

Fig. 7 shows an example for expressing a 3D Tai-Chi Ball using our technique (see our accompanying video for animating the flow on a 3D Tai-Chi Ball). In Figs. 7b and 7c, we demonstrate our method's advantages over that used in [4]. In Fig. 7c, this example is rendered with the stroke texture and color variation from the paint samples and, therefore, appears more NPR stylized than that in Fig. 7b. With flow motion, the result becomes more interesting and helps in the perception of the shape of this object. More animation results are demonstrated in Section 6.

6 Results

There are two major improvements in our point-based stylized rendering framework compared to Chi and Lee's work [4]. First, we use PAMs to transfer the brushstroke (shape and color) of actual paintings from an example painting to 3D models. In contrast to their method, our approach can generate NPR results with richer styles. Fig. 6 shows a visual comparison with Chi and Lee's work [4]. More importantly, our approach does not require a user to specify an artistically and carefully designed brush texture map like that in Fig. 6d, which is usually a very tedious and trial-and-error task. Second, our NPR splat rendering approach using PAM can control the stroke pattern direction and generate the illusion of stroke flowing over a surface and, therefore, can easily create interesting NPR animations. In contrast, the approaches relying on user-specified brush texture maps [3], [4] cannot generate such effects easily. In the following, we present many results obtained from the proposed methods. More results and our accompanying videos can be found at http://graphics.csie.ncku.edu.tw/Paper_Video/TVCG_NPR_2007/IEEE_TVCG_NPR_demo.mov.

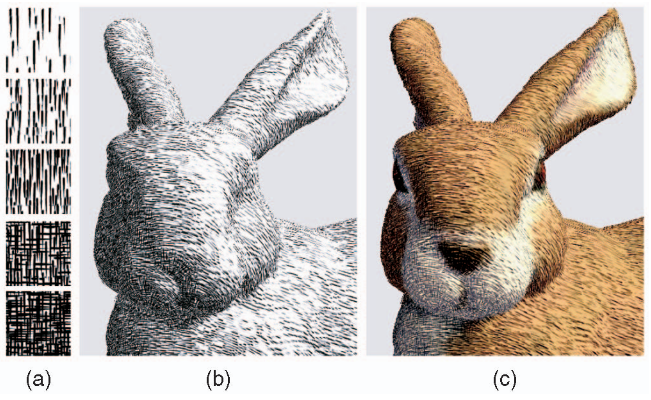


Fig. 8. (a) Our synthesized PAM results of hatching patterns. (b) Rendering result without model color. (c) Rendering result with model color.

6.1 Generating a Line-Art Painterly Art Map

Although real-time hatching [1] can generate very good line-art TAMs, it cannot handle other painting styles because the shapes of strokes are irregular and the textures of strokes are complex. In contrast, our method uses multiresolution synthesis to preserve the shape of a stroke and to maintain the coherence in PAM as well. We do not only generate good hatching strokes as the real-time hatching [1] but also can handle other stylized paintings well, such as oil paintings. Figs. 6c and 8a show synthesized PAM results of oil paintings and hatching patterns, respectively. In Figs. 8b and 8c, we show our rendering results without and with model color using hatching strokes.

6.2 Stylization of Different Paintings

In Fig. 13, we show four different sets of PAMs constructed from four paintings to render the same 3D model. In these four sets of PAMs, their strokes vary significantly in color, length, density, and regularity. As a result, we can generate varied NPR stylizations of a 3D model.

6.3 Multipass Rendering with Painterly Art Maps

Fig. 9 imitates paintings with multiple brushstrokes of multiple sizes. At runtime, we can simply use the curvature information to identify if a splat represents a coarser (Fig. 9a) or a finer (Fig. 9b) detail. A large splat size is then chosen to generate an image (Fig. 9a) as though the model is painted using a large brushstroke. A small brushstroke size is then used to draw (Fig. 9b) to obtain another image. These two images are then blended to create the final image (Fig. 9c).

6.4 2D and 2.5D

We can also treat a 2D image as a 3D point set on a 2D plane and then render it using PAMs. Fig. 10 shows an example of a 2D image rendered with different types of image abstraction. Figs. 10b, 10c, and 10d are rendered using the PAMs in Figs. 13a, 13b, and 13c, respectively. In Fig. 11, we demonstrate an interesting 2.5D stylized rendering. The 2.5D model is obtained using a sketch-based modeling method [32].

6.5 Stroke Flow Animation

We show stroke flow results in Figs. 7 and 12. As demonstrated in the accompanying video, the strokes are animated and flowing along their direction fields. For example, in Fig. 7, with animated flows, a Tai-Chi movement can be better interpreted as a 3D ball rather than a 2D circle. In the example of "Starry Night" by Van Gogh, we divide it into several regions in Fig. 12 and assign them with different attributes, such as direction fields, PAMs, and animation controls (for example, making the moon and the star twinkle). In the accompanying video, we also demonstrate a brushing effect in which the user can interact with the bunny by brushing its furs. This effect is generated by orienting stroke direction in response to the user's mouse control. The animation in the accompanying video demonstrates that the strokes rendered by the PAM technique exhibit frame-to-frame coherence when we animate models and vary the light and viewing angles.

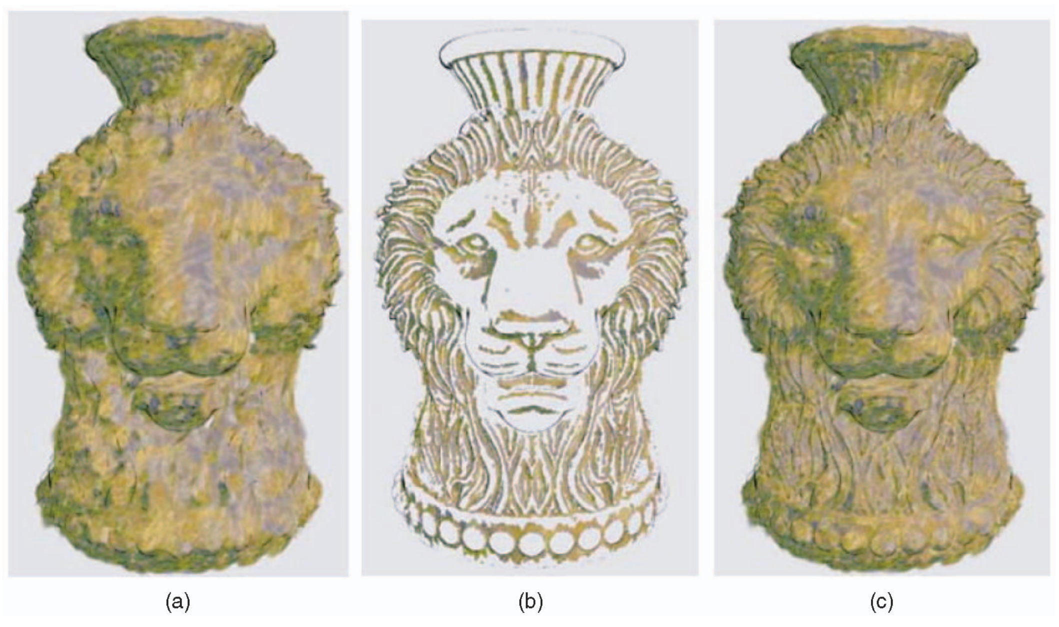


Fig. 9. Multipass rendering with PAMs. (a) Brushstrokes with a large size. (b) Brushstrokes with a small size. (c) The result by blending (a) and (b).

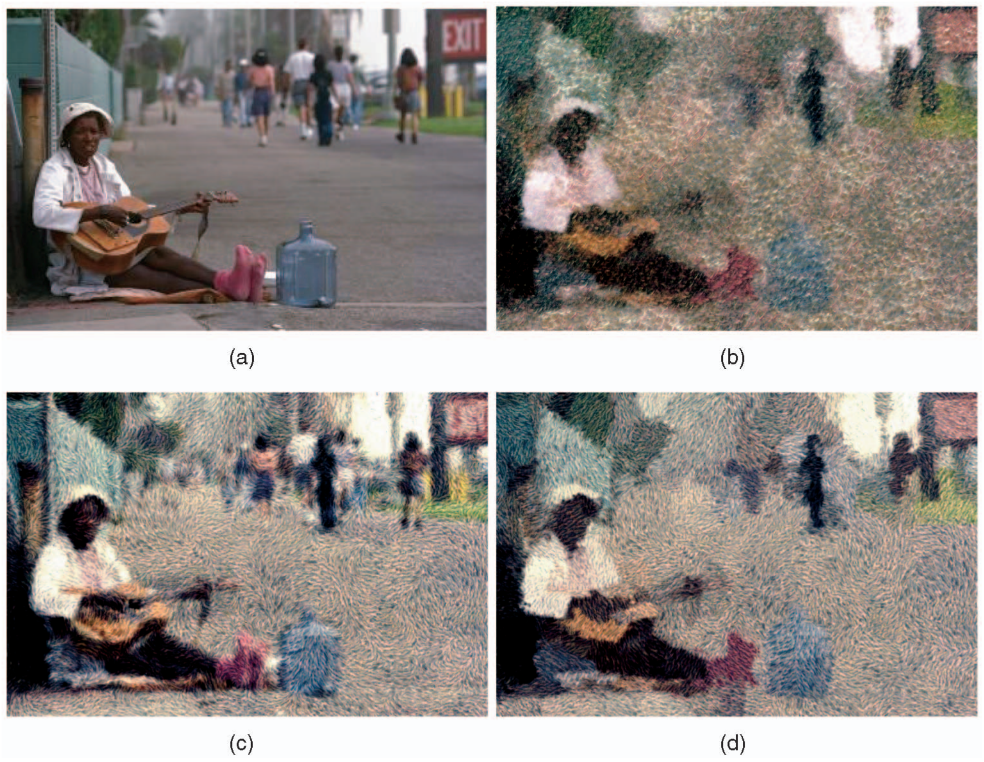


Fig. 10. (a) Input image (Photo courtesy of http://philip.greenspun.com). (b)-(d) Using the varied types of PAMs in Fig. 13 to render a 2D image in a different image abstraction.

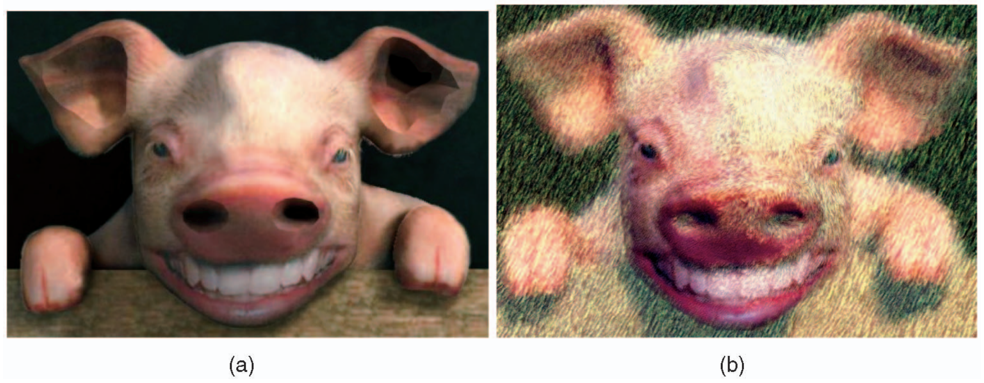


Fig. 11. Example of 2.5D painterly rendering. (a) 2.5D models. (b) PAM rendering results.

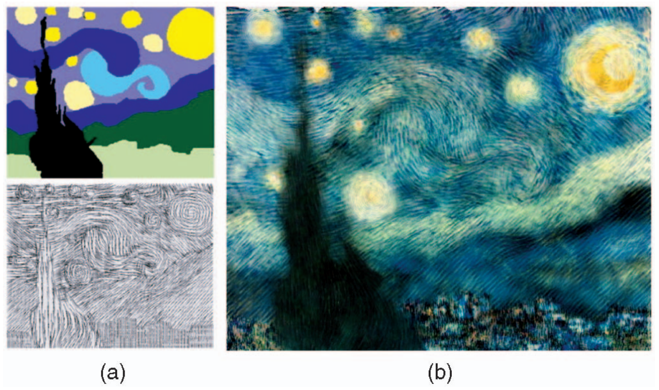


Fig. 12. Animating Van Gogh's "Starry Night." (a) Region map and vector fields. (b) A frame in the result video (see the accompanying video (http://graphics.csie.ncku.edu.tw/Paper_Video/TVCG_NPR_2007/IEEE_TVCG_NPR_demo.mov) for full animation).

6.6 Rendering Performance

Our rendering system was implemented using OpenGL on the Nvidia GeForce 8800GTS graphics card. It renders all examples interactively on a Pentium D 2.8 GHz processor, running on Microsoft Windows XP. Table 1 shows the rendering speed and the number of rendered splats in our experiments. The number of splat nodes displayed on the screen is a major factor that affects the rendering speed, that is, the fewer splats, the higher the performance in frames per second. The silhouette is an important element in NPR that expresses the shape of an object attractively. In Figs. 1a, 1b, and 6, we display silhouette splats using Chi and Lee's method [4]. Detecting and drawing silhouette splats will degrade the rendering speed substantially. For example, when rendering silhouette splats for Figs. 1a and 1b, the rendering speed will be degraded from 9.7 and 15.6 fps to 6.4 and 10 fps, respectively. Finally, we also demonstrate

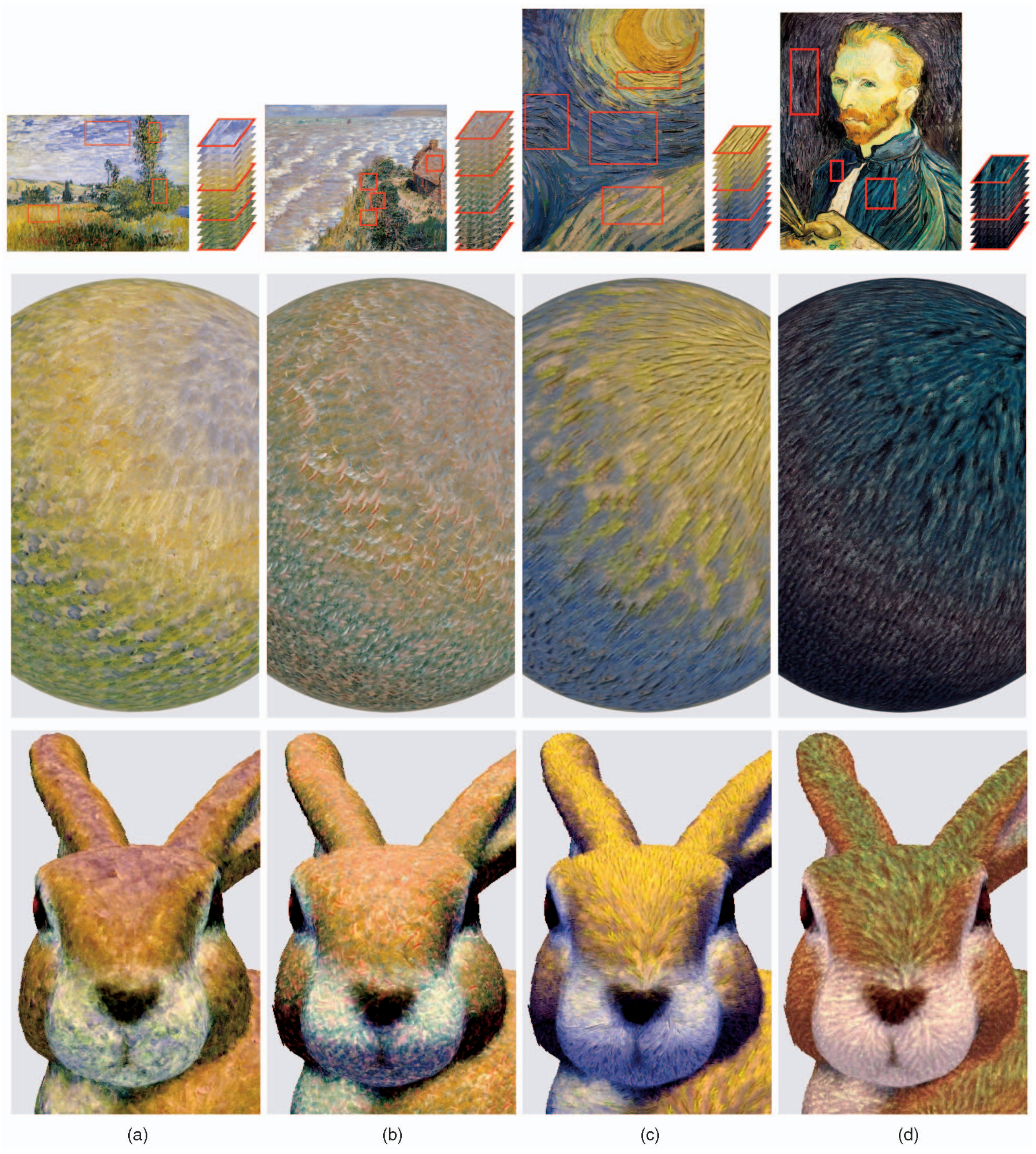


Fig. 13. Using varied types of PAMs to shade 3D models. The top row shows the painted image and synthesized PAM. The red rectangles in the painted images are the brush regions specified by the user. The second and last rows are the rendering results on a sphere and bunny, respectively.

more still results in the supplementary image file (which can be found on the Computer Society Digital Library at www.computer.org/tvcg/archives.htm).

6.7 Comparison and Application

The lit sphere approach [18] used painterly samples to render 3D models, and it is the most relevant work to the proposed method. The comparison between these two approaches is described as follows: First, the lit sphere is a triangle-based NPR system, whereas ours is a point-based NPR system. We render each point or splat as a stroke drawn on the canvas. Their method aims at transferring the shading of the painterly samples to 3D models. However, the characteristics of texture strokes are not maintained well. In contrast, our method can preserve the stroke characteristics of input samples. Second, to illuminate 3D models, we encode several stroke textures on a 3D volume texture, whereas their approach encodes an input sample on a lit sphere. With several stroke textures, our approach potentially yields a richer variety of stroke patterns and color changes than theirs. In addition, the users can simply scribble several lines to select stroke

TABLE 1
Performance Statistics

Figure	Splats	Silhouette Splats	Resolution	Fps
Fig. 7 taichi	3311	0	640x480	72.0
Fig. 8 bunny	8549	0	1280x1024	28.1
Fig. 1c bunny	9941	0	1280x1024	21.4
Fig. 1b lion vase	17972	6848	1280x1024	10.0
Fig. 6 david	9776	23232	1280x1024	7.9
Fig. 1a gargoyle	22248	14240	1280x1024	6.4

patterns, and then, our system can automatically build PAMs for stylized rendering. In contrast, the lit sphere requires several steps to interactively select triangular regions of a 2D image and place them on the corresponding hemisphere of a lit sphere. Good understanding of a shading study on the sphere is required to create a good lit sphere. Therefore, our method seems easier to use than theirs with respect to the issue of human intervention. Third, as mentioned in the lit sphere work [18], their method works well for static scenes only. Whenever a scene is transformed and the texture features of a lit sphere are very prominent, the surface will appear to "swim." This "swim" artifact is quite annoying to a viewer. Fig. 14a is excerpted from the lit sphere approach [18], and the "swim" is perceived as "rings" in this illustration. In contrast, our approach does not have this artifact, as seen in our accompanying video. Fig. 14b shows our result in contrast to theirs in Fig. 14a, and our result preserves the characteristics of input stroke textures well. Finally, our approach can easily create the illusion of stroke flowing over the surface of 3D models. However, their approach does not support this capability.

The NPR techniques can be potentially useful to many applications such as medical and scientific visualization. Recently, Bruckner and GrLoller [33] extend the lit sphere approach to volume rending and create very nice volume rendered results in NPR stylization. Here, two examples (Figs. 15a and 15b) of visualizing isosurfaces of medical data are demonstrated as a potential application of our proposed techniques. In the near future, we would like to explore more applications of our algorithm such as rendering the stream surface of flow visualization or visualizing molecular surface. Also, we would like to enhance the capabilities of our system to be more sophisticated for specific applications [33] in volume rendering.

7 Conclusion and Future Work

In this paper, we present a semiautomatic system to segment and extract stroke patterns from painted images. From the extracted stroke patterns, we construct a PAM to generate NPR stylized images and animation. Our PAM rendering approach demonstrates many properties that are advantageous for NPR applications, including 1) the maintenance of frame-to-frame stroke coherence—the distribution of splat nodes maintains the coherence of the stroke pattern in translation, scaling, and rotation, and PAM helps maintain the coherence in lighting; 2) brushstroke textures and color changes can be automatically transferred to 3D models with the stylization of input paintings; and 3) PAM can be easily used to animate stroke flow over the surface to convey the shape of a static model or the motion of objects in a static image.

Some limitations of the proposed approach will be resolved in the near future. First, chromatic variance is limited to shifting from the mean chromatic color while rendering a splat using PAMs. A better treatment may be required as the stroke pattern is intermixed with very different chromatic colors or a splat's model color is very different from a PAM's color. Considering this problem, a statistical analysis model for color transfer [34] can improve the color transfer quality. Second, we equalize the stroke density by simply resizing the width of the strokes in different stroke textures in the segmentation stage. We will

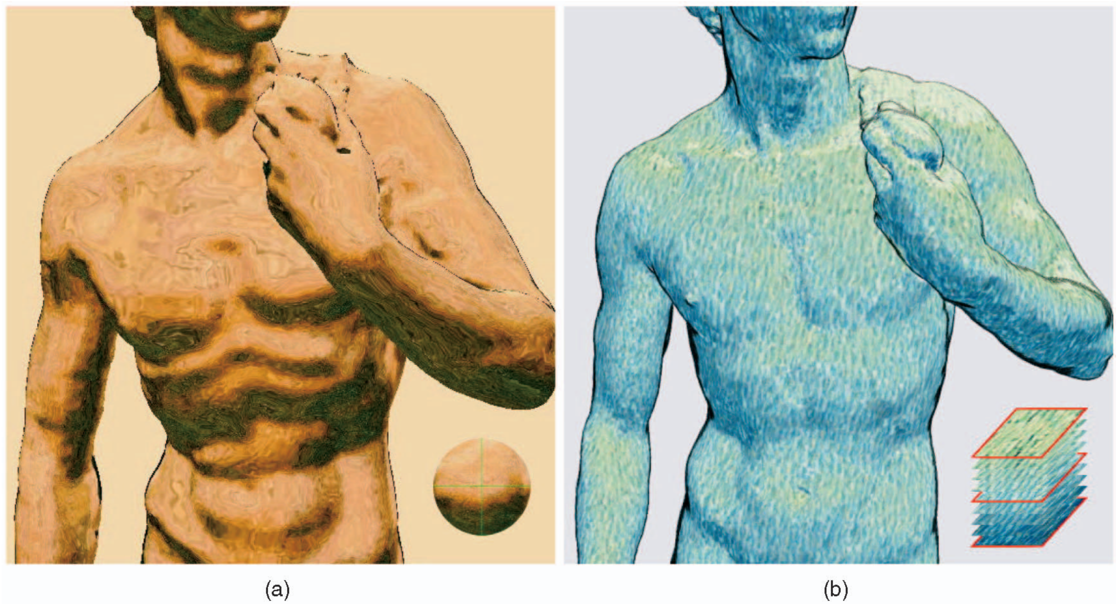

(a) (b)

Fig. 14. Visual comparisons of our technique versus the lit sphere approach. (a) Image excerpted from the lit sphere approach in [18]. (b) Image rendered by the proposed method.

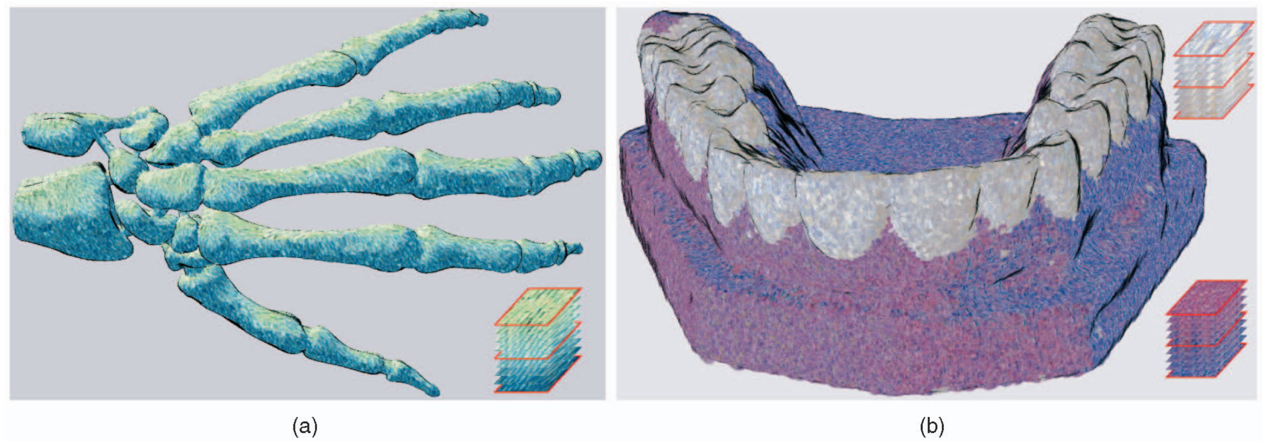


Fig. 15. Medical visualization. (a) A hand skeleton is rendered by a set of PAMs. (b) Teeth and gums are rendered by different sets of PAMs.

improve our key map and in-between map synthesis approaches so that the stroke density can also be controlled in the synthesis process. Third, from our experience, it is difficult to automatically detect the stroke density on most paintings without any user's interaction. We would like to explore automatic methods to detect the stroke density. Although we can easily control the stroke direction and tone variance on each splat through splat rendering, it still has some blurring artifacts on the overlapped area of neighbor splats. Instead of blending the overlapped area, we may be able to fix the problem by using an optimization method to synthesize the overlapped area of neighbor splats. Finally, we also plan to consider better camera-sampling analysis [35] to improve the point-based rendering quality.

Acknowledgments

The authors thank the reviewers for their insightful comments. They also thank the AIM@SHAPE Shape Repository, Stanford 3D Scanning Repository, Stanford Digital Michelangelo Project, and Large Geometric Models Archive, Georgia Institute of Technology (Georgia Tech), for sharing their various models. Fig. 14a is courtesy of the authors of the lit sphere approach [18]. This work is supported by the Landmark Program of the NCKU Top University Project under Contract B0008 and is supported in part by the National Science Council under Contracts NSC-96-2628-E-006-200-MY3 and NSC-96-2221-E-009-152-MY3. For correspondence, please contact Professor Tong-Yee Lee at tonylee@mail.ncku.edu.tw

References

[1] E. Praun, H. Hoppe, M. Webb, and A. Finkelstein, "Real-Time Hatching," *Proc. ACM SIGGRAPH '01,* pp. 579-584, 2001.
[2] M. Webb, E. Praun, A. Finkelstein, and H. Hoppe, "Fine Tone Control in Hardware Hatching," *Proc. Second Int'l Symp. Non-Photorealistic Animation and Rendering (NPAR '02),* pp. 53-58, 2002.
[3] C.D. Kulla, J.D. Tucek, R.J. Bailey, and C.M. Grimm, "Using Texture Synthesis for Non-Photorealistic Shading from Paint Samples," *Proc. 11th Pacific Conf. Computer Graphics and Applications (PG '03),* pp. 477-481, 2003.
[4] M.-T. Chi and T.-Y. Lee, "Stylized and Abstract Painterly Rendering System Using a Multiscale Segmented Sphere Hierarchy," *IEEE Trans. Visualization and Computer Graphics,* vol. 12, no. 1, pp. 61-72, Jan./Feb. 2006.
[5] J. Zhang, K. Zhou, L. Velho, B. Guo, and H.-Y. Shum, "Synthesis of Progressively Variant Textures on Arbitrary Surfaces," *ACM Trans. Graphics,* vol. 22, no. 3, pp. 295-302, July 2003.
[6] A.A. Efros and W.T. Freeman, "Image Quilting for Texture Synthesis and Transfer," *Proc. ACM SIGGRAPH '01,* pp. 341-346, 2001.
[7] Q. Wu and Y. Yu, "Feature Matching and Deformation for Texture Synthesis," *ACM Trans. Graphics,* vol. 23, no. 3, pp. 364-367, Aug. 2004.
[8] V. Kwatra, A. SchLodl, I. Essa, G. Turk, and A. Bobick, "Graphcut Textures: Image and Video Synthesis Using Graph Cuts," *ACM Trans. Graphics,* vol. 22, no. 3, pp. 277-286, July 2003.
[9] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra, "Texture Optimization for Example-Based Synthesis," *ACM Trans. Graphics,* vol. 24, no. 3, pp. 795-802, Aug. 2005.
[10] A. Hertzmann, "A Survey of Stroke-Based Rendering," *IEEE Computer Graphics and Applications,* vol. 23, no. 4, pp. 70-81, 2003.
[11] A. Hertzmann, C.E. Jacobs, N. Oliver, B. Curless, and D.H. Salesin, "Image Analogies," *Proc. ACM SIGGRAPH '01,* pp. 327-340, 2001.
[12] O. Deussen and T. Strothotte, "Computer-Generated Pen-and-Ink Illustration of Trees," *Proc. ACM SIGGRAPH '00,* pp. 13-18, 2000.
[13] A. Hertzmann and D. Zorin, "Illustrating Smooth Surfaces," *Proc. ACM SIGGRAPH '00,* pp. 517-526, 2000.
[14] J.D. Northrup and L. Markosian, "Artistic Silhouettes: A Hybrid Approach," *Proc. First Int'l Symp. Non-Photorealistic Animation and Rendering (NPAR '00),* pp. 31-37, 2000.
[15] B.J. Meier, "Painterly Rendering for Animation," *Proc. ACM SIGGRAPH '96,* pp. 477-484, 1996.
[16] D. Cornish, A. Rowan, and D. Luebke, "View-Dependent Particles for Interactive Non-Photorealistic Rendering," *Proc. Graphics Interface (GI '01),* pp. 151-158, 2001.
[17] R.D. Kalnins, L. Markosian, B.J. Meier, M.A. Kowalski, J.C. Lee, P.L. Davidson, M. Webb, J.F. Hughes, and A. Finkelstein, "WYSIWYG NPR: Drawing Strokes Directly on 3D Models," *ACM Trans. Graphics,* vol. 21, no. 3, pp. 755-762, July 2002.
[18] P.-P.J. Sloan, W. Martin, A. Gooch, and B. Gooch, "The Lit Sphere: A Model for Capturing NPR Shading from Art," *Proc. Graphics Interface (GI '01),* pp. 143-150, 2001.
[19] H. Xu, N. Gossett, and B. Chen, "Pointworks: Abstraction and Rendering of Sparsely Scanned Outdoor Environments," *Proc. 15th Eurographics Symp. Rendering (EGSR '04),* pp. 45-52, 2004.
[20] S. Zelinka and M. Garland, "Interactive Texture Synthesis on Surfaces Using Jump Maps," *Proc. 14th Eurographics Symp. Rendering (EGSR '03),* pp. 90-96, 2003.
[21] M.F. Cohen, J. Shade, S. Hiller, and O. Deussen, "Wang Tiles for Image and Texture Generation," *ACM Trans. Graphics,* vol. 22, no. 3, pp. 287-294, July 2003.
[22] D.J. Heeger and J.R. Bergen, "Pyramid-Based Texture Analysis/Synthesis," *Proc. ACM SIGGRAPH '95,* pp. 229-238, 1995.
[23] Z. Bar-Joseph, R. El-Yaniv, D. Lischinski, and M. Werman, "Texture Mixing and Texture Movie Synthesis Using Statistical Learning," *IEEE Trans. Visualization and Computer Graphics,* vol. 7, no. 2, pp. 120-135, Apr.-June 2001.

[24] W. Matusik, M. Zwicker, and F. Durand, "Texture Design Using a Simplicial Complex of Morphable Textures," *ACM Trans. Graphics,* vol. 24, no. 3, pp. 787-794, Aug. 2005.

[25] Y. Deng, B.S. Manjunath, and H. Shin, "Color Image Segmentation," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR '99),* pp. 2446-2451, 1999.

[26] L. Liang, C. Liu, Y.-Q. Xu, B. Guo, and H.-Y. Shum, "Real-Time Texture Synthesis by Patch-Based Sampling," *ACM Trans. Graphics,* vol. 20, no. 3, pp. 127-150, July 2001.

[27] L.-Y. Wei and M. Levoy, "Fast Texture Synthesis Using Tree-Structured Vector Quantization," *Proc. ACM SIGGRAPH '00,* pp. 479-488, 2000.

[28] T.-Y. Lee and C.-R. Yan, "Feature-Based Texture Synthesis," *LNCS 3482,* pp. 1043-1049, 2005.

[29] T.-Y. Lee, C.-R. Yan, and M.-T. Chi, "Stylized Rendering for Anatomic Visualization," *Computing in Science and Eng.,* vol. 9, no. 1, pp. 13-19, 2007.

[30] H. Pfister, M. Zwicker, J. van Baar, and M. Gross, "Surfels: Surface Elements as Rendering Primitives," *Proc. ACM SIGGRAPH '00,* pp. 335-342, 2000.

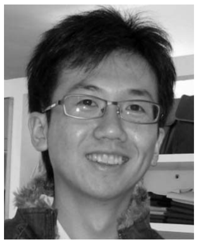
[31] E.B. Lum, A. Stompel, and K.-L. Ma, "Using Motion to Illustrate Static 3D Shape-Kinetic Visualization," *IEEE Trans. Visualization and Computer Graphics,* vol. 9, no. 2, pp. 115-126, Apr.-June 2003.

[32] T.-Y. Lee, C.-H. Lin, S.-W. Yen, and H.-J. Chen, "A Natural Pen-and-Paper Like Sketching Interface for Modeling and Animation," *Proc. Int'l Conf. Computer Animation and Social Agents (CASA '07),* pp. 87-92, 2007.

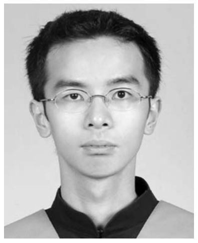
[33] S. Bruckner and M.E. GrLoller, "Style Transfer Functions for Illustrative Volume Rendering," *Computer Graphics Forum,* vol. 26, pp. 715-724, 2007.

[34] E. Reinhard, M. Ashikhmin, B. Gooch, and P.S. Shirley, "Color Transfer between Images," *IEEE Computer Graphics and Applications,* vol. 21, no. 5, pp. 34-41, Sept./Oct. 2001.

[35] P.-H. Lin and T.-Y. Lee, "Camera-Sampling Field and Its Applications," *IEEE Trans. Visualization and Computer Graphics,* vol. 10, no. 3, pp. 241-251, May/June 2004.



Chung-Ren Yan received the BS and MS degrees from the Department of Computer Science and Information Engineering, Chung-Hua University, Hsinchu, Taiwan, in 2000 and 2002, respectively. He is currently working toward the PhD degree in the Department of Computer Science and Information Engineering, National Cheng-Kung University, Tainan, Taiwan. His research interests include computer graphics, texture synthesis, and image processing.



Ming-Te Chi received the BS degree in geography from the National Taiwan University, Taipei, in 2000 and the MS degree from the Department of Computer Science and Information Engineering, National Cheng Kuang University, Tainan, Taiwan, in 2003. He is currently working toward the PhD degree in the Department of Computer Science and Information Engineering, National Cheng-Kung University. His research interests include computer graphics and nonphotorealistic rendering.

Tong-Yee Lee received the PhD degree in computer engineering from Washington State University, Pullman, in May 1995. Now, he is a professor in the Department of Computer Science and Information Engineering, National Cheng-Kung University, Tainan, Taiwan. He is an associate editor of the *IEEE Transactions on Information Technology in Biomedicine* from 2007 to 2010. He is also on the editorial advisory board of the *Journal Recent Patents on Engineering*, an editor of the *Journal of Information Science and Engineering* and a region editor of the *Journal of Software Engineering*. His current research interests include computer graphics, nonphotorealistic rendering, image-based rendering, visualization, virtual reality, surgical simulation, medical visualization and medical system, and distributed and collaborative virtual environments. He leads the Computer Graphics Group, Visual System Laboratory, National Cheng-Kung University (http://graphics.csie.ncku.edu.tw/). He is a member of the IEEE and the ACM.

Wen-Chieh Lin received the BS and MS degrees in control engineering from the National Chiao-Tung University, Hsinchu, Taiwan, in 1994 and 1996, respectively, and the PhD degree in robotics for dynamic near-regular texture tracking and manipulation from Carnegie Mellon University, Pittsburgh, in 2005. Since 2006, he has been with the Department of Computer Science and the Institute of Multimedia Engineering, National Chiao-Tung University, as an assistant professor. His current research interests include computer vision, computer graphics, and computer animation. He is a member of the IEEE and the IEEE Computer Society.