Painterly rendering with content-dependent natural paint strokes

Hua Huang, Tian-Nan Fu & Chen-Feng Li

The Visual Computer
International Journal of Computer Graphics

ISSN 0178-2789
Volume 27
Number 9

Vis Comput (2011) 27:861-871
DOI 10.1007/s00371-011-0596-5

Springer

Vis Comput (2011) 27:861–871
DOI 10.1007/s00371-011-0596-5

ORIGINAL ARTICLE

Painterly rendering with content-dependent natural paint strokes

Hua Huang · Tian-Nan Fu · Chen-Feng Li

Published online: 4 May 2011

Abstract We present a new *painterly rendering* method that simulates artists' content-dependent painting process and the natural variation of hand-painted strokes. First, a new stroke layout strategy is proposed to enhance the contrast between large and small paint strokes, which is an important characteristic of hand-painted paintings. Specifically, the input image is partitioned into nonuniform grids according to its *importance map*, and determined by the grid size, an individually constructed paint stroke is applied in each grid. Second, an anisotropic digital brush is designed to simulate a real paint brush. In particular, each bristle of the digital brush has an individual color, so that strokes rendered by the new brush can have multiple colors and naturally varied textures. Finally, we present a novel method to add lighting effects to the canvas. This lighting imitation method is robust and very easy to implement, and it can significantly improve the quality of rendering. Comparing with traditional painterly rendering approaches, the new method simulates more closely the real painting procedure, and our experimental results show that it can produce vivid paintings with fewer artifacts.

Keywords Non-photorealistic rendering · Painterly rendering · Stroke · Lighting imitation

H. Huang (✉) · T.-N. Fu
School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an, China
e-mail: huanghua@mail.xjtu.edu.cn

T.-N. Fu
e-mail: ftnmail@gmail.com

C.-F. Li
School of Engineering, Swansea University, Swansea, UK
e-mail: c.f.li@swansea.ac.uk

1 Introduction

Non-photorealistic rendering (NPR) transforms realistic visual media to achieve various artistic styles. In contrast to *photorealistic rendering* whose focus is on the expression of realistic details, NPR aims to send abstract and emotional information in a similar way as real art works. NPR techniques are widely used to imitate hand-painted paintings, drawings and illustrations [2, 6, 20, 24]. These computer-based techniques reduce significantly the cost of expensive manual creation. In addition, with the rapid growth of Internet, the acquisition of high quality images and videos is becoming increasingly easier. Thus, the demand of changing images or videos into different artistic styles is also increasing exponentially which provides a strong motivation for the study of NPR.

As a typical NPR technique, painterly rendering transforms realistic images or videos in an artistic way such that the result looks like hand-painted. Many different methods have been developed for this task. Litwinowicz [16] proposed a method to add impressionistic effects to images and video clips. Using curved brush strokes of different sizes, Hertzmann [6] rendered a digital painting on multiple layers. Hays and Essa [5] presented an image-based painterly rendering approach and extended it to video painterly rendering. All these works show impressive rendering results.

Similar to hand painting, images produced by painterly rendering are created by painting strokes on a canvas. The strokes are usually painted on several layers with each layer using a different stroke size. Specifically, paint strokes have larger sizes on lower layers and smaller sizes on upper layers, and correspondingly the painting procedure is a refining process. However, this traditional painting approach has two problems. First, for regions containing details, the large strokes initially painted on the lower layer contribute nothing to the final result, and they may hinder the small strokes

Fig. 1 "Portrait of Caspar Goodrich" (1887), by John Singer Sargent. (**a**) The original painting. (**b**) The face region, in which small paint strokes are used to depict details. (**c**) The background region, in which large paint strokes are used to emphasized the abstract impression

from painting the details on the upper layer. Second, using small strokes to refine the result painted by large strokes will damage the abstract impression generated by the large strokes, which is often the desired artistic effect for smooth regions. For hand painting, strokes with different sizes are applied not only on different layers but also in different regions on the canvas. Paint strokes often have a larger size in smooth regions and a smaller size in regions full of details. As shown in Fig. 1, there is a clear contrast between large and small paint strokes, and using different stroke sizes for different regions adds a distinct characteristic to hand-painted paintings.

Traditional painterly rendering methods usually paint a stroke as an anti-aliased line (straight or curved) of a solid color without any texture. Although the real stroke texture can be mapped onto the stroke path, the result still looks more or less mechanical due to limitation of predefined stroke textures. On the contrary, for hand painting, paint strokes always have multiple colors and naturally varied textures.

To overcome the above problems, this paper presents a novel painterly rendering algorithm. The new method uses a new strategy for arranging digital strokes, such that strokes with different sizes are used in different regions on the canvas. Furthermore, an anisotropic brush is designed to simulate natural strokes with multiple colors and varied textures. Finally, we propose a simple and robust method to effectively imitate the lighting effect. After a brief review of related work in Sect. 2, the algorithm is explained in details in Sect. 3. Some experimental results are given in Sect. 4 to demonstrate the effectiveness of the new method.

2 Related work

Painterly rendering can be traced back to Haeberli's work [4], where a natural scene was converted into an impressionistic image by using an ordered collection of brush strokes. In particular, each stroke had several attributes including location, color, size, direction and shape, and most of these attributes were interactively controlled.

Litwinowicz [16] presented an automatic method to transform images and video clips into impressionistic style. In this automatic method, brush strokes were arranged on a regular grid with fixed spacing. The length and radius of strokes were predefined, and the color of each stroke was sampled from the original image. When the input is a video clip, the optical flow was used to guide the movement of strokes between adjacent frames so that the temporal coherence could be retained.

A new method with multiple painting layers was presented by Hertzmann [6], where the image was rendered on a sequence of layers and each layer had a different brush size. Larger brushes were used to draw a rough sketch on lower layers and smaller brushes were used to refine the details on upper layers. Curved brush strokes were also introduced to imitate curved strokes in real paintings. Each stroke was assigned with a single color and rendered as a solid cubic B-spline. Later, Hertzmann and Perlin [8] further extended this work to cope with videos, where each frame was painted as a still image. To improve temporal coherence, the difference between frames was calculated, after which the painting process was performed only in the regions with significant changes.

Gooch et al. [3] proposed a new strategy for stroke layout. While Hertzmann [6] arranged strokes on a symmetrical grid, Gooch et al. [3] used a segmentation-based method. First, the input image was segmented into features, whose approximate medial axes were extracted to form a set of tokens. Then, these feature tokens were used to determine stroke properties such as path, width and color. In this paper, we also present a new stroke layout strategy. Unlike previous works, our method arranges strokes on a nonuniform grid to preserve fine features of the image. The details will be explained in Sect. 3.1.

Hertzmann [7] proposed a special technique to add lighting effects to painterly rendered images. Specifically, a pair of opacity and height maps were assigned to each stroke to construct the height map of the whole image, after which the lighting effect was added to the image by *bump-mapping* the original painting based on the height map. In this paper, we present a new method that achieves a similar effect and is very easy to implement. The details are described in Sect. 3.4.

Hays and Essa [5] proposed a uniform painterly rendering technique for images and videos. Different from most previous methods, strokes in this paper were placed on an arbitrarily large canvas. The orientation of each stroke was computed by using RBF interpolation of key-point orientations. When processing video clips, the properties of strokes were constrained over time, and edges at various frequencies were used to guide the creation and refinement of strokes.

Computation of stroke orientation is an important issue for a painterly rendering algorithm, and it has been widely studied [17, 22, 23]. Recently, Lee et al. [14] presented a novel method to determine the orientation of strokes for videos. In their work, the motion information was extracted from image sequences of a scene and then used to determine the stroke orientation in regions with significant motions. In regions with little motion, the orientation of strokes was determined by the image gradient field. The stroke orientation determined by this composite approach shows an advantage in expressing the movement of objects.

Several example-based methods were proposed to produce painting results through analyzing and learning features of real paintings [9, 11, 19]. In these methods, one or more example paintings are given as templates. Then some key features (e.g., color and texture) are extracted from the templates and transferred to the input realistic image to make it have a painting appearance.

Recently, several object-based painterly rendering techniques were proposed [12, 13, 15, 21, 26]. Based on image parsing techniques, Zeng et al. [21] presented a semantics-driven painting approach. The input image was first decomposed into a parse tree representation consisting of constituent components. Then each image component was painted by selecting appropriate brushes from a pre-defined brush dictionary and placing them on the canvas. The selection of brushes was guided by the image semantics described in the parse tree so that each image component has an individual painting style. Zhao and Zhu [26] presented an interactive abstract painting system which is guided by the psychological principle. They further improved the method introduced by Zeng et al. [21] with a few adaptations and applied it to render input photographs. In addition, a numerical ambiguity analysis method was proposed to assess and control the abstract level of the result painting. Kagaya et al. [13] proposed an novel video painting system. Similar to the object-based image painting technique, the input video clip was segmented into the background and the temporally moving (or deforming) objects. The user could specify style parameters for each object in some keyframes. Then these parameters will automatically propagate to the entire video and each object will be painted according to its specific style parameters. Our new method is also content dependent, but follows a different strategy. First, while object-based methods try to paint different objects with different styles, we focus on the arrangement of brushes with different sizes and naturally varied textures. Second, our approach is more flexible in that the painting result can be simply adjusted by modifying the *importance value* of each pixel. For instance, using edge detection result as the *importance map*, our method can produce results with sharp edges and clear contrast between large and small brushes. Third, our approach is easier to implement and more robust because the content object is a relatively high-level concept and is sometimes difficult to accurately extract from input images and videos.

3 Algorithm

The new algorithm consists of four major parts addressing respectively the layout of strokes (i.e., position and size), the orientation of strokes, the rendering method and the lighting effect.

3.1 Layout of strokes

Similar to many existing methods, multiple stroke sizes are used in this paper and they are also applied to multiple layers. However, instead of using a fixed stroke size for each layer, we use different stroke sizes for different parts of the image and, on the same layer, strokes can have different sizes depending on the image content. The idea is to design an *importance driven* stroke layout strategy [18], which paints smooth regions with larger strokes while painting details with smaller strokes. Given an input image, this is achieved via three steps: computing the importance map, partitioning the canvas into grids according to local importance and assigning different stroke sizes to different grids.

In order to obtain a reliable importance map, a bilateral filtering is first performed on the input image to remove noises and to keep edge features. The result of bilateral filtering is a *reference image*, which is then used to compute the importance map. Areas close to edges are often considered more important, and consequently they require more attention when painting. Thus, we compute the importance energy $E(x, y)$ of a pixel (x, y) in the reference image with the following equation

$$E(x,y) = \sqrt{\left(\frac{\partial I(x,y)}{\partial x}\right)^2 + \left(\frac{\partial I(x,y)}{\partial y}\right)^2}, \tag{1}$$

where $I(x, y)$ denotes the grayscale intensity and the gradients are calculated using Sobel operator. Depending on content features, different measures can be adopted to adjust the importance map. For instance, *face detection* can be used to increase the energy in the face area [25].

After the importance map is constructed, we divide the canvas into a set of rectangular grids by cutting repeatedly along the x and y directions (see Fig. 2(b)). At the beginning, the whole canvas is one grid. Then, cutting through the longer edge, the canvas is divided into two smaller grids. For each new grid, this division process is repeated until the total importance energy in the resulting grid is less than a given threshold T_e. A grid is divided into two parts to maximize the difference between the average energies of the two

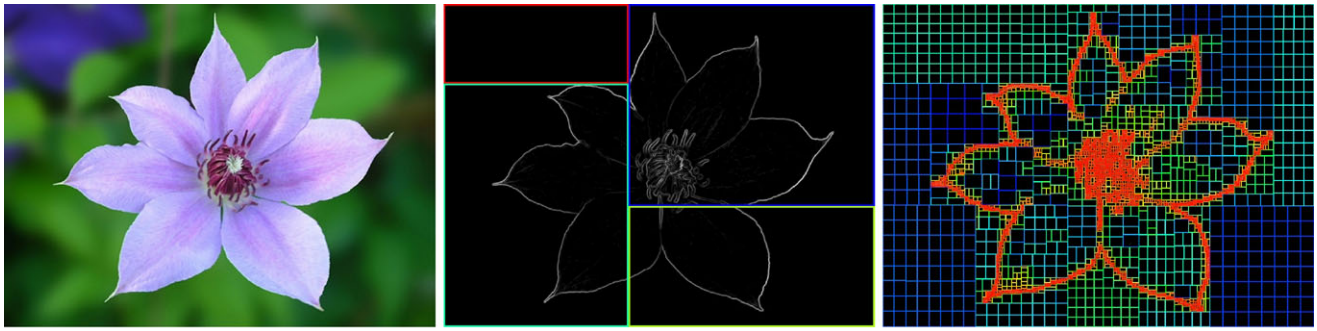


Fig. 2 Importance map and canvas partition. (**a**) Input image. (**b**) Importance map with canvas partitioned twice along the x and y directions. (**c**) The final result of canvas partition. Partitions are marked using different colors, and their sizes decrease as the color changing from *blue* to *red*

resulting grids and, at the same time, to minimize the difference between the area of the two grids. To find the optimal cutting point of a partition, we solve the following optimization problem

$$\arg\min_i \left(\frac{\bar{E}_i^{\text{smaller}} + \delta}{\bar{E}_i^{\text{bigger}} + \delta} \right) \cdot \left(\frac{S_i^{\text{bigger}}}{S_i^{\text{smaller}}} \right)^{\gamma}, \tag{2}$$

where i is the index of valid cutting points, $\bar{E}_i^{\text{smaller}}$ and $\bar{E}_i^{\text{bigger}}$ denote respectively the smaller and larger average energy of the two resulting grids, S_i^{smaller} and S_i^{bigger} denote respectively the smaller and larger areas of the two resulting grids, δ is a small positive constant which is set to 0.01 in our implementation, and γ is used to adjust the weight between the two terms in the equation, which is set to 1.0 for all examples in this paper. In addition, a maximum and a minimum threshold are defined to control the size of final grids. If the area of a grid is larger than the maximum size, it must be divided into two smaller grids. If the area of a grid is smaller than the minimum size, it will not be divided any further. In our implementation, the maximum and minimum sizes are set to the maximum and minimum stroke sizes specified by the user in terms of area.

As shown in Fig. 2(c), an nonuniform rectangular mesh is formed from canvas partition. The importance energy of pixels in smooth regions of a reference image are usually small. Hence, the partition operation in these regions quickly ends after a few divisions reaching the threshold T_e, which leads to larger grids in smooth regions. On the contrary, for regions containing fine features, more divisions are performed before reaching the threshold T_e and consequently the resulting grids in these regions are relatively small.

Strokes are distributed to the partitioned canvas according to the size of each grid. We sort the grids in descending order according to their area and loop from top to bottom the grid list. Within each grid, two potential anchor points are defined. Denoted by $p_{\min}$, the first anchor point is defined as the pixel with the minimum coverage (see Sect. 3.3 for detailed definition of the coverage map). Denoted by $p_{\max}$, the second anchor point is the pixel whose color has the greatest difference from the corresponding pixel on the reference image. If the coverage of $p_{\min}$ is smaller than a given threshold T_c, a stroke will be placed at this point. Otherwise, the color difference between the canvas and the reference image at $p_{\max}$ is checked and, if it is greater than a threshold T_d, a stroke will be placed at $p_{\max}$. For a specific grid, if none of these two conditions is met, then no stroke will be placed. The size of the stroke is selected from a list containing user-specified standard values and, according to the area of the grid, it is set to the closest standard value. The layout of strokes is completed for a layer when the loop ends and every grid is checked. In our approach, 3 to 5 layers of strokes are usually sufficient to produce a good result.

3.2 Orientation of strokes

Haeberli [4] suggested using the gradient direction to guide strokes, and the stroke direction was set perpendicular to the local gradient. This approach works well in regions containing fine features, but the stroke directions become chaotic in smooth regions where local gradients get close to singular. Hays and Essa [5] proposed an alternative method to overcome this problem. A few points with large gradient magnitudes were selected to form a key-point set and at each key point, the stroke direction was defined perpendicular to the local gradient. For other points on the canvas, the stroke directions were computed through an RBF interpolation (see, e.g., [1]). This method can produce consistent orientation fields for smooth regions, but in regions with details, the strokes with interpolated directions often fail to capture the edges. As a result, the strokes may go across the edge and cause damage to the content structure. Litwinowicz [16] proposed a method that falls in between. Specifically, for points with large gradient magnitudes, the orientation was set to the normal of the local gradient, and

for other points with small gradient magnitudes, the orientation was interpolated using thin plate spline.

In this paper, we use a method similar to Litwinowicz [16] to generate the stroke orientation field. Based on the importance map, the orientation of each point is computed as

$$O(x,y) = \widetilde{E}(x,y) \cdot O_{GN}(x,y) + \left(1 - \widetilde{E}(x,y)\right) \cdot O_{RBF}(x,y), \quad (3)$$

where O_{GN} denotes the orientation field obtained from the normals of local gradients, O_{RBF} denotes the orientation field computed from RBF interpolation and $\widetilde{E}(x,y)$ is the importance energy normalized to interval [0, 1]. This simple approach ensures that in smooth regions, the stroke direction is mainly determined by the RBF-interpolated orientation, and in the regions with fine features, the stroke direction is mainly determined by the local gradient.

3.3 Stroke rendering

A widely used method for stroke rendering is drawing anti-aliased lines. Although easy to implement, the strokes rendered by anti-aliased lines differ significantly from real hand-painted strokes. As shown in Fig. 3(b), an anti-aliased line has only a solid color without any texture. To simulate hand-painted strokes, we introduce an anisotropic brush model as shown in Fig. 3(a), which is also used in our previous work [10]. Each visible pixel, whose intensity is greater than 0 in the brush map, represents a bristle on the brush. The intensity of pixel indicates the contact status between the bristle and the canvas. White means completely touched and black means completely separated. A stroke is rendered by specifying a color to every bristle and dragging the brush map on the canvas. As shown in Fig. 3(c), the stroke rendered by this method has multiple colors and varied textures along the stroke path.

The brush map (i.e., Fig. 3(a)) is predefined and for each stroke, the digital brush described above is initialized through three steps: (1) The brush map is uniformly resized (using bicubic interpolation) to the specified stroke size; (2) The intensity of each pixel in the brush map is multiplied by a user-specified factor to adjust the transparency of the stroke; and (3) Pigment colors are assigned individually to visible pixels whose intensities are greater than 0 in the brush map (the details of pigment color assignment are described in the next paragraph). After initializing the brush, the stroke is painted along a path consisting of a set of ordered points. Specifically, the center of the brush map is placed in turn at each point on the stroke path and, for every brush position, the corresponding pixels on the canvas are painted with color

$$C_{\text{canvas}} = \frac{I_{\text{bristle}}}{255} \cdot C_{\text{bristle}} + \left(1 - \frac{I_{\text{bristle}}}{255}\right) \cdot C_{\text{canvas}}, \quad (4)$$

where C_{bristle} represents the color of each bristle, C_{canvas} is the color of the corresponding point on the canvas and I_{bristle} is the intensity value of the associated bristle. In the above equation, the transparency of the stroke is taken into account through the intensity of the brush map.

The painting process starts with filling the canvas with a solid color similar to the color of a real canvas. Then, following the strategy described in Sect. 3.1, strokes are distributed to the canvas and rendered immediately upon creation. For every stroke, the color of each bristle is sampled from the reference image, followed by an adjustment simulating the natural color variation of hand-painted strokes. First, the center of the brush map is placed at the anchor point of the stroke, and the color of each bristle is sampled from the corresponding point on the reference image. Secondly, denoting the individual bristle of the brush with subscript $_{bristle}$, the color of each bristle is adjusted by

$$C_{\text{bristle}} = \alpha \cdot C_{\text{bristle}} + (1-\alpha) \cdot C_{\text{anchor}} + C_{\text{perturbation}}, \quad (5)$$

where C_{anchor} is the color sampled from the reference image at the anchor point of the stroke, $\alpha \in [0,1]$ is an adjustable parameter, and $C_{\text{perturbation}}$ is an optional random color specific to each stroke. To obtain the path of a stroke, we use the method proposed by Hertzmann [6] to search for several *control points* starting from the anchor point, which are then connected together to form a smooth curve (the curve can go cross grid boundaries). The number of control points is determined by the maximal length of a stroke, a user-specified parameter. From one end to the other, all points on the curve are checked in order to form the stroke path, and the checking procedure is terminated when either of the following two conditions is satisfied. (1) The color difference between the reference image and the canvas at the current point is smaller than that between the reference image and the average color of the brush. (2) The orientation difference between the current point and the anchor point is greater than a given threshold T_o. Those points that have not been checked are discarded, and the checked points form the stroke path.

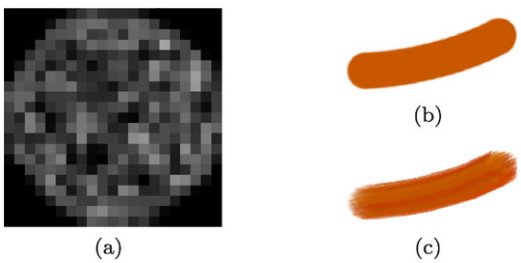


Fig. 3 Brush map and the associated strokes. (**a**) Magnified brush map. Each block represents a pixel. (**b**) Stroke rendered by an anti-aliased line. (**c**) Stroke rendered by the anisotropic brush (**a**)

Till now, the stroke size, the transparency (specified by the user), the brush color (up to each individual bristle) and the stroke path are all determined, and the stroke can be accordingly rendered on the canvas.

A coverage map is also prepared to record the painting process and indicate the coverage extent of each pixel on the canvas. The coverage map is initialized to black, which means no pixel on the canvas is covered by pigment. After rendering the image canvas with the colored brush, the coverage map is updated by setting all the bristle colors to white and painting with the same stroke. As described in Sect. 3.1, the coverage map is used in the determination of the anchor point of each stroke.

3.4 Lighting effect

Adding lighting effects to the painted canvas can significantly improve the rendering quality and makes the result more similar to a hand-painted painting [7]. We present an effective method to imitate the lighting effect which is very robust and easy to implement. A canvas height map is first constructed to indicate the height of each pixel on the canvas. Unlike Hertzmann's method [7], in which a height and an opacity map are assigned to each stroke, we use instead the brush map and the anti-aliased line to render the height and the opacity maps of each stroke. The pixel intensity on the height map of a stroke is assigned as the average intensity of bristles that pass through the corresponding point on the canvas. The opacity map of a stroke is obtained by rendering the stroke path on a black background using a white anti-aliased line (see Fig. 4), and the width of the anti-aliased line is set to a value slightly less than the diameter of the brush (i.e., the width or height of the brush map). Then, using the opacity maps of individual strokes as the weight function, the canvas height map can be readily constructed as the weighted sum of the stroke height maps.

Instead of using the Phong shading model to perform bump-mapping, we imitate the lighting effect by simply adjusting the intensity of each pixel on the canvas. For each pixel (x, y), its intensity adjustment is proportional to height difference $D(x, y)$

$$\begin{cases} d = O(x, y) - \pi/2, \\ D(x, y) = h(x + \cos d, y + \sin d) - h(x, y), \end{cases} \tag{6}$$

(a) (b)

Fig. 4 The height and opacity maps of a stroke. (**a**) The height map. (**b**) The opacity map

where d represents the direction perpendicular to the stroke orientation $O(x, y)$, h denotes the pixel height, i.e., the corresponding pixel intensity on the canvas height map. The intensity of pixel (x, y) is increased if $D(x, y)$ is positive and vice versa. The intensity adjustment also depends on the original intensity of the pixel. The adjustment has to be a relatively small value if the original intensity is too large or too small (i.e., close to 0 or 255); and a relatively large amount of adjustment could be allowed if the original intensity is neutral, i.e., close to the mean 127. Specifically, for a pixel (x, y) on the canvas, the intensity adjustment can be expressed as

$$I(x, y) = I(x, y) + a \cdot D(x, y) \times (\min\{I(x, y), 255 - I(x, y)\}/127.5), \tag{7}$$

where a is a tuning parameter to control the intensity of lighting effects. Increasing the value of a will enhance the effect of specular reflection. Since the canvas is a color image, the intensity adjustment is performed separately on the R, G and B channels.

4 Results

A sequence of experimental results are presented in this section to demonstrate the performance of the new method. For comparison, we also give results produced by other painterly rendering algorithms, which includes Litwinowicz [16], Hertzmann [6, 7], Hays and Essa [5] and Zhao and Zhu [26]. While the results of Litwinowicz [16] and Hertzmann [6] are generated by our own implementations, the results of Hertzmann [7] and Hays and Essa [5] are obtained from the corresponding paper and the project page, respectively. Zhao and Zhu [26] result is produced by the software provided on their project page.

Figure 5 shows the rendering results of Fig. 2(a) generated respectively by the proposed method and two other methods. From the results, it is clear that our method produces a better contrast between larger and smaller strokes. In our method, the obscure background is painted with larger brushes, and the foreground containing details is painted with smaller strokes. The clear contrast between the sophisticated foreground and the flat background emphasizes the message of the painting. In Hertzmann [6], the areas painted by large strokes on the bottom layer are almost completely covered by smaller strokes painted on upper layers and as a result, the foreground and background objects are blurred together giving a dull appearance. Meanwhile, Hertzmann [6] does not distinguish image content at different regions and on the same layer, a fixed stroke size is used for the entire canvas. Thus, some local features such as edges are often rendered by strokes that are too large to capture the details

Fig. 5 Images rendered from Fig. 2(a). (**a**) Our result. (**b**) The result rendered by Hertzmann [6]. (**c**) The result rendered by Zhao and Zhu [26]. When producing this result, the input image is interactively segmented into foreground and background, with the abstract level of the foreground set to the lowest and the abstract level of the background set to medium

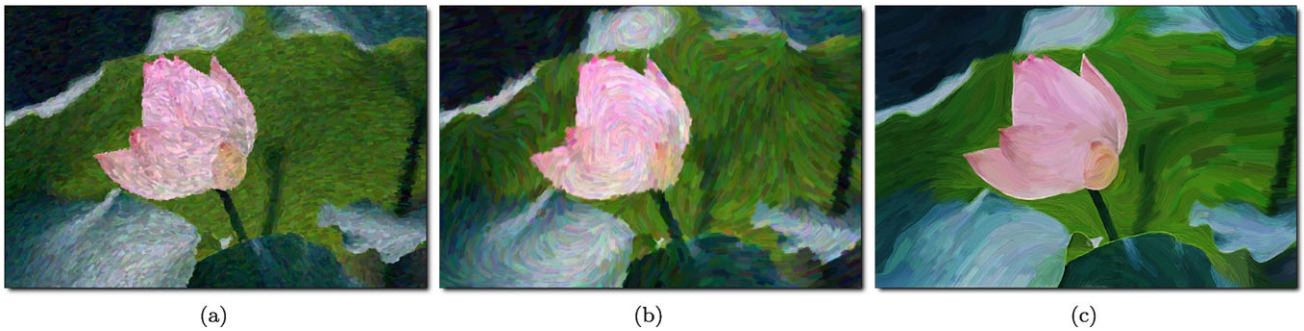


Fig. 6 Examples rendered respectively by (**a**) Litwinowicz [16], (**b**) Hays and Essa [5], and (**c**) our approach

and correspondingly the results at these areas appear rough and bumpy. In Zhao and Zhu [26], the input image is interactively segmented into a hierarchy of components, after which the painterly rendering algorithm is performed within each component. Benefiting from image segmentation, large contours are well preserved in the output image (see the edge of the flower in Fig. 5(c)). However, this approach often fails to reproduce the small details that are not trivial to segment (see the stamen region in Fig. 5(c)). It is observed from our limited experiments that Zhao and Zhu [26] tends to give more abstract results, even when the abstract level has been set to the lowest, and the contrast between large and small strokes rarely exhibits in their results. Similar disadvantages are also found in other traditional painterly rendering algorithms. In Fig. 6, the contrast between larger and smaller strokes is not obvious in paintings produced by Litwinowicz [16] and Hays and Essa [5]. The stroke sizes used in the entire image seems to be similar, and edges are not well represented. In contrast, edges in our result are successfully preserved, which benefits from our new stroke layout strategy and also the use of the importance map generated by edge detection.

In the traditional painterly rendering method, the rendering results produced by larger strokes on lower layers are usually refined by smaller strokes painted on upper layers. However, for smooth regions in the input image, it is completely unnecessary to paint them again with smaller strokes since artificial noises could be added to the image. Also, it is often a waste of work to paint the sophisticated regions with large strokes, which are to be completely covered and refined by smaller strokes. By using different stroke sizes in different regions, our new method has a potential to significantly reduce the number of paint strokes while achieving a better quality. As shown in Table 1, the number of strokes

Table 1 Comparison of stroke numbers between our new approach and Hertzmann [6]. Six different input images are rendered by Hertzmann [6] and our approach, respectively (our rendering results are shown in Fig. 9). The dimensions of input images are listed in the second column. The total number of strokes used in the two approaches are listed respectively in the third and the fourth columns

Input image	Dimension	Our approach	Hertzmann [6]
Dog	1024 × 842	11232	48116
Kiwi fruit	800 × 773	8368	38325
Windmill	1024 × 768	18729	67851
Deer	900 × 783	9194	40735
Man	664 × 999	16860	40681
Street	900 × 900	26072	94468

used by the new method is far less than that used in Hertzmann [6].

To investigate the performance of the new method for adding lighting effects, a comparison with Hertzmann [7] is performed. In order to avoid the influence of other painterly rendering operations, we add the lighting effect to the same input image and use the same height field with Hertzmann [7]. Figure 7(a) shows a painting rendered by Hertzmann [6] from a realistic source image, and Fig. 7(b) shows the lighting effect added by using bump-mapping [7]. Shown in Fig. 7(c) is the lighting effect added to the same input image Fig. 7(a) using our approach and the same hight field as Hertzmann [7]. Comparing Fig. 7(b) and Fig. 7(c), it can be observed that the two lighting effects appear to be very similar except that Hertzmann [7] makes the original painting darker while our approach makes it brighter. However, our approach is easier to implement. Furthermore, the height field generated by Hertzmann [7] seems to be exaggerated. The feedback from a few artists confirms that our height field generated by the anisotropic brush is closer to real paintings. The final painterly rendering result produced by our new algorithm for the same realistic source image is shown in Fig. 7(d).

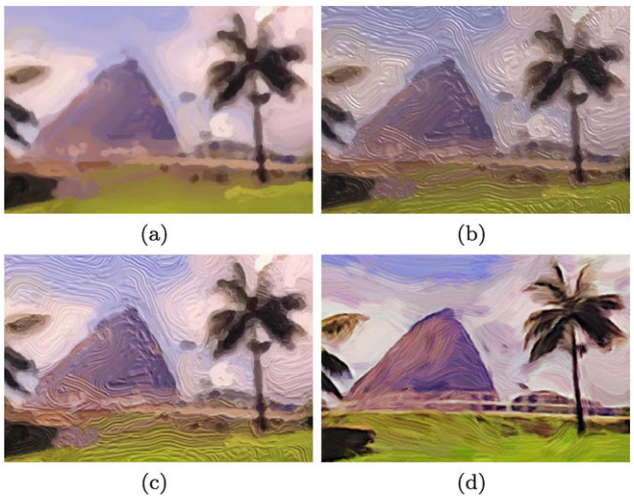


Fig. 7 Lighting effect comparison between Hertzmann [6, 7] and our approach. (**a**) A painting rendered from a realistic source image, using Hertzmann [6]. (**b**) Lighting effect produced by Hertzmann [7] for image (**a**). (**c**) Lighting effect produced by our approach, using the same height field generated by Hertzmann [7] and added to the same input image (**a**). (**d**) Our rendering result for the same realistic source image

Besides comparing with other painterly rendering algorithms, we also compare the different techniques developed within our new algorithm, including the stroke layout strategy and the integrated stroke rendering and lighting method. Figure 8 shows a group of results painted using our new approach, but with different settings. In particular, Fig. 8(a) is rendered using the full version of the new method. Fig. 8(b) is painted with our new stroke layout strategy switched off and replaced by that proposed by Hertzmann [6]. It can be observed that the edges in the image are not well preserved and many fine features have been lost. In addition, the contrast between large and small strokes is very moderate in Fig. 8(b) giving a flat impression. Figure 8(c) is produced by keeping the proposed stroke layout strategy but replacing the stroke rendering and lighting method with drawing anti-aliased lines. As expected, the details are successfully captured in Fig. 8(c), but the strokes rendered by using anti-aliased lines appear to be very mechanical.

More results rendered by the proposed approach are given in Fig. 9 at the end of the paper. These examples further demonstrate that the new approach has a stable performance and can be successfully applied to a wide range of images to achieve different painting styles.

5 Conclusion

Based on a novel strategy for stroke layout, we proposed in this paper an effective and robust painterly rendering method. First, according to an importance map built from the input image, the canvas is non-uniformly divided into grids with different area. Then strokes with user-specified

Fig. 8 Rendering effects of different techniques developed within the new algorithm. (**a**) is generated using the full version of the new painterly rendering algorithm. (**b**) is generated using the stroke layout strategy proposed by Hertzmann [6]. (**c**) is painted using anti-aliased lines without lighting effect

Fig. 9 Painterly rendering results of our new method. The importance maps of some reference images have been slightly adjusted by interaction, e.g. the energy in the face region of *Man* has been enhanced

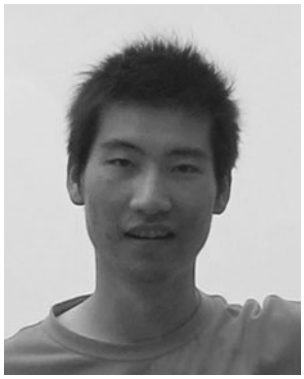
standard sizes are distributed to the partitioned canvas according to the size of individual grids. Strokes of larger sizes are usually allocated to smooth regions in the image, while strokes of smaller sizes are placed in the regions full of details. This strategy simulates the clear contrast between large and small strokes, which is often a key characteristic for hand-painted paintings. In addition, an anisotropic brush model with hand-painting features is designed to render the strokes. The paint strokes rendered by our approach has multiple colors and varied textures, which is more similar to the strokes in real paintings. After rendering, an effective and easy-to-implement method is presented to add lighting effects to the result image. Demonstrated by various examples, the proposed painterly rendering method can produce visually pleasant results for a wide range of paintings.

Acknowledgements This work was supported in part by grants from the National Natural Science Foundation of China (No. 60970068) and Program for New Century Excellent Talents in University under Project NCET-09-0635. The author would also like to thank the support from the Royal Society through the Royal Society / NSFC International Joint Project (JP100987).

References

1. Bors, A.G.: Introduction of the radial basis function (rbf) networks. In: Online Symposium for Electronics Engineers. DSP Algorithms: Multimedia, vol. 1 (2001)
2. Curtis, C., Anderson, S., Seims, J., Fleischer, K., Salesin, D.: Computer-generated watercolor. In: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, pp. 421–430. ACM/Addison-Wesley, New York (1997)
3. Gooch, B., Coombe, G., Shirley, P.: Artistic vision: Painterly rendering using computer vision techniques. In: Proceedings of the 2nd International Symposium on Non-photorealistic Animation and rendering, June, Citeseer, pp. 03–05 (2002)
4. Haeberli, P.: Paint by numbers: Abstract image representations. Comput. Graph. **24**(4), 214 (1990)
5. Hays, J., Essa, I.: Image and video based painterly animation. In: Proceedings of the 3rd International Symposium on Non-photorealistic Animation and Rendering, pp. 113–120 (2004). doi:10.1145/987657.987676
6. Hertzmann, A.: Painterly rendering with curved brush strokes of multiple sizes. In: Computer Graphics. Proceedings. SIGGRAPH 98 Conference Proceedings, pp. 453–60. ACM, New York (1998). Proceedings of SIGGRAPH 98: 25th International Conference on Computer Graphics and Interactive Techniques, 19–24 July 1998, Orlando, FL, USA
7. Hertzmann, A.: Fast paint texture. In: Proceedings of the 2nd International Symposium on Non-photorealistic Animation and Rendering, June, pp. 03–05 (2002). doi:10.1145/508530.508546
8. Hertzmann, A., Perlin, K.: Painterly rendering for video and interaction. In: Proceedings of the 1st International Symposium on Non-photorealistic Animation and Rendering, pp. 7–12. ACM, New York (2000)
9. Hertzmann, A., Jacobs, C., Oliver, N., Curless, B., Salesin, D.: Image analogies. In: Proceedings of SIGGRAPH, pp. 327–340 (2001)
10. Huang, H., Fu, T., Li, C.: Anisotropic brush for painterly rendering. In: Proceedings of Computer Graphics International. Computer Graphics Society (2010)
11. Huang, H., Zang, Y., Li, C.: Example-based painting guided by color features. Vis. Comput. **26**(6–8), 933–942 (2010)
12. Huang, H., Zhang, L., Fu, T.N.: Video painting via motion layer manipulation. Comput. Graph. Forum **29**(7), 2055–2064 (2010)
13. Kagaya, M., Brendel, W., Deng, Q., Kesterson, T., Todorovic, S., Neill, P., Zhang, E.: Video painting with space-time-varying style parameters. IEEE Trans. Vis. Comput. Graph. (2010)
14. Lee, H., Lee, C., Yoon, K.: Motion based painterly rendering. Comput. Graph. Forum **28**(4), 1207–1215 (2009)
15. Lin, L., Zeng, K., Lv, H., Wang, Y., Xu, Y., Zhu, S.: Painterly animation using video semantics and feature correspondence. In: Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering, pp. 73–80. ACM, New York (2010)
16. Litwinowicz, P.: Processing images and video for an impressionist effect. In: Computer Graphics Proceedings, SIGGRAPH 97, pp. 407–414. ACM, New York (1997). Proceedings of 24th International Conference on Computer Graphics and Interactive Techniques, 3–8 August 1997, Los Angeles, CA, USA
17. Olsen, S., Maxwell, B., Gooch, B.: Interactive vector fields for painterly rendering. In: Proceedings of Graphics Interface 2005, p. 247 (2005). Canadian Human-Computer Communications Society
18. Streit, L., Buchanan, J., et al.: Importance driven halftoning. Comput. Graph. Forum **17**, 207–218 (1998)
19. Wang, B., Wang, W., Yang, H., Sun, J.: Efficient example-based painting and synthesis of 2d directional texture. IEEE Trans. Vis. Comput. Graph. **10**(3), 266–277 (2004)
20. Winkenbach, G., Salesin, D.: Computer-generated pen-and-ink illustration. In: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques, p. 100. ACM, New York (1994)
21. Zeng, K., Zhao, M., Xiong, C., Zhu, S.C.: From image parsing to painterly rendering. ACM Trans. Graph. **29**(1), 2:1–2:11 (2009). doi:http://doi.acm.org/10.1145/1640443.1640445
22. Zhang, E., Mischaikow, K., Turk, G.: Vector field design on surfaces. ACM Trans. Graph. **25**(4), 1294–1326 (2006). doi:http://doi.acm.org/10.1145/1183287.1183290
23. Zhang, E., Hays, J., Turk, G.: Interactive tensor field design and visualization on surfaces. IEEE Trans. Vis. Comput. Graph. **13**(1), 94–107 (2007). doi:10.1109/TVCG.2007.16
24. Zhang, S., Chen, T., Zhang, Y., Hu, S., Martin, R.: Video-based running water animation in Chinese painting style. Sci. China Ser. F **52**(2), 162–171 (2009)
25. Zhang, S., Li, X., Hu, S., Martin, R.: Online video stream stylization. Tech. rep. (2009). URL http://cg.cs.tsinghua.edu.cn/papers/VideoStylization.pdf
26. Zhao, M., Zhu, S.C.: Sisley the abstract painter. In: NPAR '10: Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering, pp. 99–107. ACM, New York (2010). doi:http://doi.acm.org/10.1145/1809939.1809951

Hua Huang is a professor in School of Electronics and Information Engineering, Xi'an Jiaotong University. He received his B.S. and Ph.D degrees from Xi'an Jiaotong University in 1996 and 2006, respectively. His main research interests include image and video processing, pattern recognition and machine learning.



Tian-Nan Fu is currently a master student in School of Electronics and Information Engineering, Xi'an Jiaotong University. He received his B.S. degree from Xi'an Jiaotong University in 2008. His research interests include image and video processing.

Chen-Feng Li is currently a lecturer at the School of Engineering of Swansea University, UK. He received his B.Eng. and M.Sc. degrees from Tsinghua University, China in 1999 and 2002 respectively and received his Ph.D. degree from Swansea University in 2006. His research interests include computer graphics and numerical analysis.