Enhancement of Moment Based Painterly Rendering Using Connected Components

M. Obaid, R. Mukundan, T. Bell
Department of Computer Science and Software Engineering
University Of Canterbury
Christchurch, New Zealand.
{mho33@student.canterbury.ac.nz}

Abstract

Moment functions have been used recently to compute stroke parameters for painterly rendering applications. The technique is based on the estimation of geometric features of the intensity distribution in small windowed images to obtain the brush size, colour and direction. This paper proposes an improvement of this method, by additionally extracting the connected components so that adjacent regions of similar colour are grouped for generating large and noticeable brush stroke images. An iterative coarse-to-fine rendering algorithm is used for painting regions of varying colour frequencies. Performance improvements over the existing technique are discussed with several examples.

Keywords--- Painterly rendering, non-photorealistic rendering, geometric moments, connected component image.

1. Introduction

Artistic rendering has become an important research area in Computer Graphics because of the many challenges posed by the general problem of stylized approximation of an image. This field is inspired by various artistic styles such as paintings [1, 2], drawings [3, 4], animated cartoons [5, 6] and technical illustrations [7, 8]. These artistic styles can be grouped into two categories according to their input data: *3D object-based* [9], which takes 3D model of a scene as their input; and *2D image-based* [10], which takes 2D images as their input.

Interactive artistic rendering techniques are commonly used in digital painting systems and provide the user with a wide range of options and tools. Haeberli [11] introduced such a system that allows the user to place brush strokes manually on a canvas. Each brush stroke is described by its location, colour, size, direction and shape. Haeberli also proposed an automatic way of controlling the brush stroke orientation by using the gradient data of the source image. Non-Interactive methods are a lot more complex to design and implement, as the system needs to extract shape features automatically from input data, and then map these features to the most appropriate brush-stroke parameters for stylized rendering. Hertzmann [1] proposed a method for automatically painting brush strokes using spline curves. Painting is done on several layers, where larger stokes are used in lower layers and thinner strokes in upper layers. The intensity gradient of the image controls the orientation of the spline curves.

Geometric moments [12] are popular shape descriptors for image analysis. They have been employed in non-interactive painterly rendering applications for estimating stroke parameters based on local intensity distributions. In this paper, we present a method that aims to improve upon the previous work using this approach by Shiraishi and Yamaguchi [10]. Shiraishi's method first computes the brush stroke locations from the source image, and then the stroke parameters such as the location, length, width, and angle are calculated using image moment functions. Strokes are then painted using alpha blending in the order of largest to smallest. Our method extracts the connected components from the image to identify the shape of larger brush strokes. This greatly enhances the painterly appearance of the image. Nehab and Velho [13, 18] extended the work of Shiraishi and Yamagushi by using a multi-resolution technique and introducing parametrized stroke positions image. They also proposed a stroke placement method with the addition of a stroke normalization and a dithering method for stroke positioning using a variance filter and blue-noise dithering.

This paper is organised as follows. Section 2 describes the use of geometric moment functions as shape descriptors. Section 3 gives an overview of the moment based painterly rendering algorithm. Section 4 describes how connected colour components could be easily computed. Section 5 explains how the proposed method uses connected components for generating more expressive strokes for artistic rendering. Section 6 presents some of the results obtained. Finally, section 7 concludes the paper and outlines some future directions in this area.

2. Geometric Moments

Several pattern recognition applications use geometric moments as shape descriptors. The low-order geometric moments can be easily computed from an image segment, and used to identify that segment's essential shape features. Given an image intensity function $f(i, j)$, i=0,1,…M–1, j=0,1,…N–1, the $(p+q)^{th}$ order geometric moments m_{pq} are computed as

$$m_{pq} = \sum_{i=0}^{M-1}\sum_{j=0}^{N-1} i^p j^q f(i, j) \quad (1)$$

The principal shape features are computed as follows:

$$x_c = \frac{m_{10}}{m_{00}}, \quad y_c = \frac{m_{01}}{m_{00}} \quad (2)$$

$$a = \frac{m_{20}}{m_{00}} - x_c^2$$

$$b = \frac{m_{02}}{m_{00}} - y_c^2$$

$$c = 2\left(\frac{m_{11}}{m_{00}} - x_c y_c\right) \quad (3)$$

In the above equations, x_c, y_c denote the image centroid, and the zeroth order moment m_{00} gives the sum of intensity values. Using the values of a, b and c, the parameters of the equivalent rectangle of the shape can be computed (Figure 1).

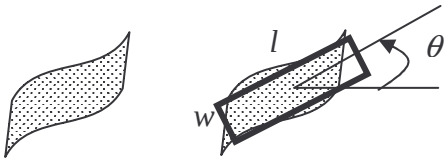

Figure 1: A shape and its equivalent rectangle

The length l, width w, and the angle of orientation θ of the equivalent rectangle are given by the following equations:

$$w = \sqrt{6(a + c - \Delta)}$$

$$l = \sqrt{6(a + c + \Delta)}$$

$$\theta = \frac{1}{2}\tan^{-1}\left(\frac{b}{a - c}\right)$$

$$\Delta = \sqrt{b^2 + (a - c)^2} \quad (4)$$

In the context of painterly rendering, the above values can be used to map a brush stroke image onto the equivalent rectangle computed for an image segment.

3. Moment Based Painterly Rendering

This section reviews the moment based painterly rendering algorithm proposed by Shiraishi and Yamaguchi [10]. Figure 2 shows an image used as an input for this algorithm in our study. The process contains two phases: a preparation phase and a composition phase. In the preparation phase, the stroke attributes are calculated and a list of strokes from largest to smallest is computed. In the composition phase, the strokes are painted over a canvas.

Figure 2: Sample image used for painterly rendering.

Three steps are involved in the preparation phase. The first step is to define a stroke distribution over the canvas. The stroke distribution depends on the level of detail (variations in intensity values) in a region. A measure of variations of intensity values surrounding a pixel can be obtained by taking the sum of squared differences of intensity values between the pixel and its neighbouring pixels in a small window. If k denotes half the window size, we can obtain this measure as the value of $g(i, j)$ at pixel (i, j) using the following equation:

$$g(i, j) = \sum_{u=i-k}^{i+k}\sum_{v=j-k}^{j+k}\left(f(u,v) - f(i, j)\right)^2 \quad (5)$$

If $g(i, j)$ is inverted and normalised to a valid intensity range, then a low value of $g(i, j)$ indicates a large variation of intensity in the neighbourhood of (i, j). The image obtained by assigning this value of $g(i, j)$ to pixel (i, j) is called the stroke area image (Figure 3(A).)

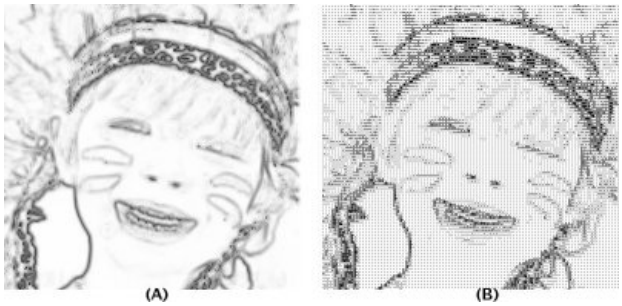

Figure 3: (A) A stroke area image. (B) A stroke locations Image

In the second step of the preparation phase, the stroke area image is used to generate the stroke locations image by applying a dithering algorithm without clustering (see Figure 3(B)). In the dithered image, regions of high detail will correspond to dark regions with a high density of dots. Each dot in the dithered image represents a stroke location. In [10], a modified version of a space filling curve dithering algorithm giving in [14] was used to generate the stroke locations image. The brush stroke parameters are computed from geometric moments (equations (4)) at each point in the stroke locations image.

The final step of the preparation phase is to sort the strokes to be painted into a list in the order of largest to smallest.

Figure 4: The painterly rendered version of the image in Figure 2.

In the composition phase, strokes are painted on a blank canvas using alpha blending one after the other. This process requires scaling, rotating and applying the appropriate colour to the brush stroke template image. Figure 4 shows the painterly rendered version of the image in Figure 2.

4. Connected Components

The primary limitation of the method described above is that large regions of nearly constant colour are not painted by correspondingly large brush strokes. The dither function imposes a constraint on the maximum distance between two stroke locations, which in turn limits the size of the brush strokes used. It may also be noted that a large region of nearly white colour in the stroke area image represents only a collection of regions of similar colours, and not necessarily a single region of constant colour.

For a more stylised rendering of images, it is imperative to have large brush strokes visible in regions of constant colour. This requirement has motivated us to consider the use of connected components as an important image feature for painterly rendering. In other words, the painterly appearance of an image can be further enhanced, if regions of similar colour can be identified and grouped as a single component and then used for mapping brush stroke images.

Even though a recursive 8-point connected component algorithm is the simplest to implement, it may require a large stack space for images of size larger than 100x100 pixels. As pixels are scanned from top to bottom and left to right, the algorithm needs to be initiated for every unlabelled pixel. Due to the systematic way in which pixels are scanned, it is necessary to check only four neighbours (instead of eight) for similar colours (Figure 5).

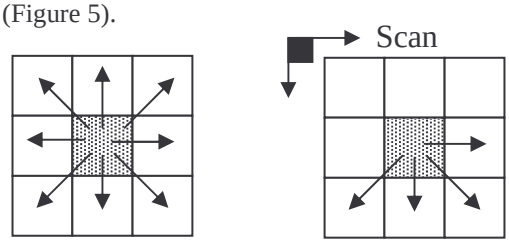

Figure 5: 8-point and 4-point connectivity

Since connected components are used only for mapping brush stroke images, we can also use a simple one-pass iterative connected component algorithm with 4-point connectivity as shown in Figure 5. With this process, connected components of certain shapes as shown in Figure 6 will get split into two components, but this is acceptable since such an image would require brush strokes in more than one direction anyway.

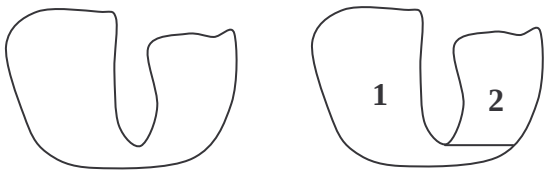

Figure 6: Splitting of connected components

5. Improved Algorithm

The ideas outlined in the previous section can be implemented in a moment-based painterly rendering algorithm for further enhancing the expressiveness of the rendering style. Regions where large brush strokes are to be used are first identified by extracting the connected components. This process is illustrated in Figure 7.

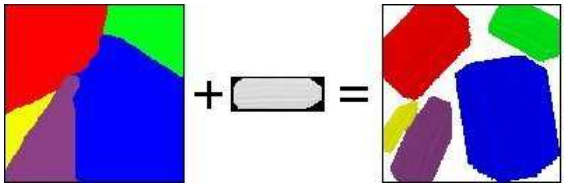

Figure 7: Brush image mapped to connected components

The painterly rendered image is composed by painting brush strokes on a blank canvas. This process requires identifying connected colour regions iteratively from the input image, computing the stroke attributes for connected regions and painting the brush strokes on

canvas. Figure 8 illustrates the process of producing the painterly rendered output image.

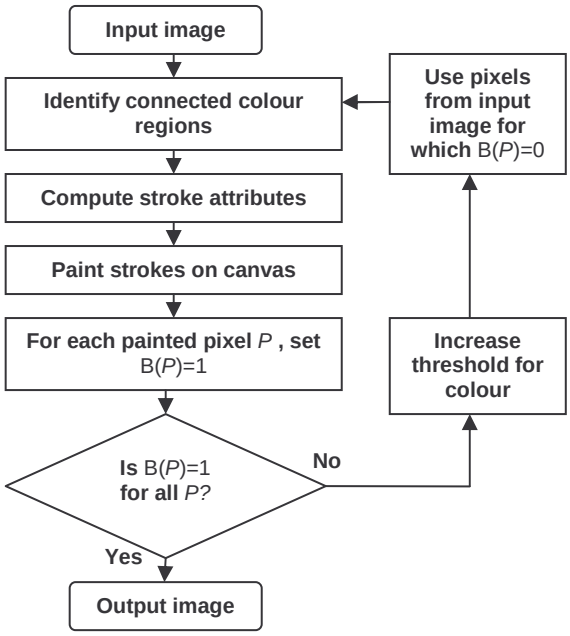

Figure 8: Flow chart for the modified painterly rendering algorithm

The process is further explained below.

5.1 The Painting Process

An artist usually produces paintings by applying large brush strokes for constant colour areas, and finer strokes for detailed areas in an image. The connected component colour regions use different sizes of brush strokes depending on each colour region's size.

The algorithm starts with a blank canvas and a bit array B(*P*) of the same size as the input image. The array, which is initialised to zero, is used to check if a pixel P has already been painted or not. Connected regions of similar colour are then identified as explained in Section 4. Regions whose size is smaller than a given threshold are ignored (leaving gaps as shown in Figure 10), and later merged into the surrounding regions by increasing the threshold. This will cause regions of large colour variations to appear smudged, giving them a painterly appearance.

Brush stroke parameters are computed from geometric moments (equations (4)) for each connected component. Each stroke is painted on the canvas by applying an inverse transformation (equations (6)) to the brush stroke template image with the appropriate colour of the stroke. Figure 9 shows a brush image with length *L*, and width *W*.

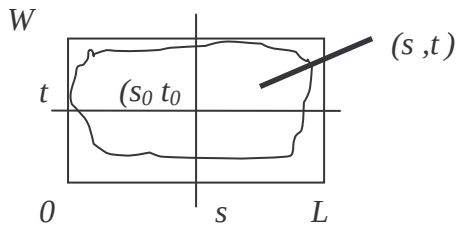

Figure 9: Inverse transformation parameters

The inverse transformation that maps a point (*x*, *y*) in an equivalent rectangle with parameters x_c, y_c, *l*, *w*, θ (equations 2, 4) to an axis-aligned brush image coordinates (*s*, *t*) is given below:

$$x' = (x - x_c)\cos\theta + (y - y_c)\sin\theta$$
$$y' = -(x - x_c)\sin\theta + (y - y_c)\cos\theta$$
$$s = \left(\frac{x'}{l} + \frac{1}{2}\right)L$$
$$t = \left(\frac{y'}{w} + \frac{1}{2}\right)W \qquad (6)$$

After obtaining the coordinates (*s*, *t*), the colour value at an image pixel (*x*, *y*) is set to that of the brush image at (*s*, *t*) and the buffer value for pixel P=(*x*, *y*) is set to one, i.e., B(*P*)=1. Figure 10 shows the brush images applied to equivalent rectangles of connected components obtained from Figure 2.

Figure 10: The initial connected component brush strokes for the image in Figure 2.

As seen in Figure 10, the initial painting of large brush strokes partially covers the painting canvas. Running the process iteratively for the unpainted regions with B(*P*)=0, and simultaneously increasing the colour similarity threshold for each iteration, leads to painting the whole image with different sizes of brush strokes. The progressive rendering algorithm uses a coarse-to-fine approach for selecting stroke regions. The final painterly rendered image is shown in Figure 11 below.

Figure 11: The painterly rendered version of the image in Figure 2.

6. Results

This section shows some of the results obtained using the connected component explained in previous section.

Figure 12: An image of an African safari tree

Figure 12 shows a test image used in our experimental analysis. It contains both regions of constant colour requiring large brush strokes (applied by first extracting connected components), and regions with large colour variations requiring finer brush strokes. The processed image is shown in Figure 13.[1]

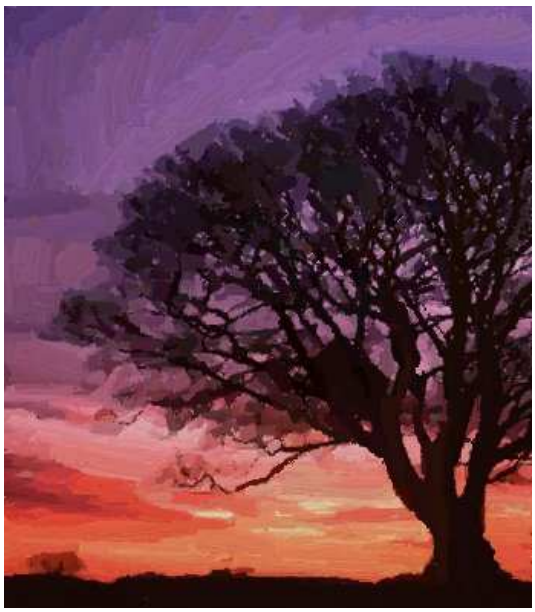

Figure 13: The painterly rendered version of the image in Figure 12.

Another image which demonstrates painterly rendering using different sizes of brush strokes depending on the connected component's size is shown below in Figure 14.

Figure 14: An image of a Siberian Tiger

The result obtained by applying the method presented in this paper is shown in Figure 15.

[1] Images can be viewed at www.cosc.canterbury.ac.nz\research\PG\mho33

Figure 15: The painterly rendered version of the image in Figure 14.

7. Conclusion

This paper has presented a moment-based algorithm for artistic rendering of images, where brush stroke parameters are computed using geometric moments of local intensity distributions. Instead of using small windowed regions on locations identified by a dither function, our algorithm uses connected components for stroke placement. Connected components also allow the rendering of large brush stroke images in regions of nearly constant colour. With this modification, the quality of stylized rendering of the painted images could be significantly improved. Some performance improvements over existing techniques could also be achieved by eliminating the need for the computation of the stroke area image and the dither image.

Further research in this area is directed towards enhancing the edges between two features of different colours in an image.

Acknowledgment

Some of the test images used in our research are taken from [15, 16, 17, 19].

References

[1] Aaron Hertzmann. Painterly Rendering with Curved Brush Strokes of Multiple Sizes. SIGGRAPH 98 Conference Proceedings, pages 453-460, July 1998.

[2] Cassidy J. Curtis, Sean E. Anderson, Joshua E. Seims, Kurt W. Fleischer, and David H. Salesin. Computer-Generated Watercolor. SIGGRAPH 97 Conference Proceedings, pages 421-430, August 1997.

[3] Mario Costa Sousa and John W. Buchanan. Observational models of graphite pencil materials. Computer Forum, 19(1), pages 27-49, March 2000.

[4] Teresa W. Bleser, John L. Sibert, and J. Patrick McGee. Charcoal sketching: Returning control to the artist. ACM Transactions on Graphics, 7(1), January 1988.

[5] Adam Lake, Carl Marshall, Mark Harris, and Marc Blackstein. Stylized rendering techniques for scalable real-time 3d animation. NPAR 2000: First International Symposium on Non Photorealistic Animation and Rendering.

[6] Barbara Meier. Painterly Rendering for Animation. In Proceedings of SIGGRAPH 96, pages 477-484, 1996.

[7] M. PAGES Salisbury, M. T. Wong, J. F. Hughes, and D. Salesin. Orientable Textures For Image-Based Pen-And-Ink Illustration. SIGGRAPH 1997 Conference Proceedings, pages 401-406, 1997.

[8] Victor Ostromoukhov. Digital facial engraving. SIGGRAPH 99 Conference Proceedings, pages 417-424, August 1999.

[9] A. Hertzmann, D. Zorin. Illustrating smooth surfaces. SIGGRAPH 2000 Conference Proceedings. New Orleans, Louisiana. Pages 183-.192, July 2000.

[10] Michio Shiraishi and Yasushi Yamaguchi. An algorithm for automatic painterly rendering based on local source image approximation. NPAR 2000 : First International Symposium on Non Photorealistic Animation and Rendering,

[11] Paul E. Haeberli. Paint By Numbers: Abstract Image Representation. SIGGRAPH 90 Conference Proceedings, 24(4), pages 207-214, August 1990

[12] R. Mukundan and K.R. Ramakrishnan. Moment Functions in Image Analysis - Theory and Applications. World Scientific Publishing Co. Pte Ltd, Singapore, September 1998.

[13] Diego Nehab and Luiz Velho. Multiscale Moment-Based Painterly Rendering. SIBGRAPI'02 Conference Proceedings, pages 244-251, IEEE, 2002.

[14] Luiz Velho and Jonas de Miranda Gomes. Digital halftoning with space filling curves. Computer Graphics, 25(4), pages 81-90, July 1991.

[15] Waltman, Jason. Implementation of: An Algorithm for Automatic Painterly Rendering Based on Local Source Image Approximation. 10 December 2001. 5 May 2005 http://www.jasonwaltman.com/graphics/paint-local-source.html

[16] Oliveira, Danilo. Nehab, Diego. Diogo, Andrade. Automatic painterly rendering of digital pictures. 12 December 2000. 6 April 2005 http://www.visgraf.impa.br/Courses/ip00/Projects.html

[17] Wasilewski, Michael. Painterly Rendering. 1 November 2005 http://www.cgl.uwaterloo.ca/~mmwasile/cs798/as1/

[18] Diego Nehab. Moment Based Painterly Rendering. 24 January 2003. 6 April 2005. http://www.cs.princeton.edu/~diego/academic/phd/526/final.pdf

[19] We For Animals. 22 February 2006. http://www.weforanimals.com/free-pictures/wild-animals/tigers/1/tiger-1.htm