

國立交通大學

資訊科學系

碩士論文

水墨工筆畫走獸之合成

The Synthesis of Chinese Fine-Brushwork Painting
on Animals

研究生：湯茜如

指導教授：施仁忠 教授

中華民國九十三年六月

水墨工筆畫走獸之合成
The Synthesis of Chinese Fine-Brushwork Painting on Animals

研究生：湯茜如

Student : Chian-Ru Tang

指導教授：施仁忠

Advisor : Zen-Chung Shih

國立交通大學
資訊科學系
碩士論文

A Thesis
Submitted to Department of Computer and Information Science
College of Electrical Engineering and Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
in

Computer and Information Science

June 2004

Hsinchu, Taiwan, Republic of China

中華民國九十三年六月

水墨工筆畫走獸之合成

研究生：湯茜如

指導教授：施仁忠 教授

國立交通大學資訊科學系



水墨畫是一門歷史悠久的中國傳統藝術。在盛唐時期，除了人物山水已達到極高的境界之外，花鳥走獸畫亦日漸形成獨立的風格。一直到清代中葉，義大利人郎世寧東來，將西方寫生的技法，與中國繪畫主題與意境相融合，呈現出不同於歷代宮廷繪畫與文人繪畫民間繪畫的新穎面貌。在本篇論文中，我們提出一個模擬與合成動物走獸畫的方法。經由一張動物的照片，運用影像處理之技術，模擬出打底的用色，以及分析動物毛髮的方向，以材質貼圖的方式產生刷毛的效果。如此，使用者便能容易地利用現有的動物圖片，不需知悉繁複的水墨畫技巧，在短時間之內描繪出其所欲之中國走獸畫風格。

The Synthesis of Chinese Fine-Brushwork Painting on Animals

Student: Chian-Ru Tang

Advisor: Dr. Zen-Chung Shih

Department of Computer and Information Science

National Chiao-Tung University



ABSTRACT

The Chinese ink painting is a traditional art which has a long history. In Tang Dynasty, in addition to figures and landscapes paintings which have already reached the extremely high realm, flowers, birds and animals also formed the individual style of drawing. In the middle period of Qing Dynasty, Lang Shining brought his skill of European painting and blended it with Chinese subjects and themes. His works presented different and novel appearance from others in the past. In this thesis, we propose a method to synthesize the Chinese ink paintings on animals. Via the photo of an animal, our techniques rely on image processing and image analysis, and generate wash diffusion and animal hairs with textures. Therefore, users may create the desired painting style with ease even if he is not familiar with the painting skills.

Acknowledgements

This thesis would not have been possible without the support and help of many people. It is fortunate for me to have worked with these truly kind and excellent people.

First and foremost, I would like to thank my thesis advisor, Dr. Zen-Chung Shih, for teaching me everything I know about computer graphics and about doing research. His enthusiasm and patience have had huge impact on how I approach my work.

I have learned much from all my collaborators of Computer Graphics and Virtual Reality Laboratory: Der-Lor Way, Yu-Ting Tsai and Yu-Pao Tsai, who are Ph. D candidates, for their constructive suggestions; Wei-Jin Lin, Yen-Lin Lee and Jie-Feng Jian, who are my partners, for their helpful comments. And especially appreciate Wei-Jin Lin for his extraordinary art talent.

I wish to thank my NCTU-friends for the good times: Penny Li, Lyre Wu, Rukawa Wang, Tequila Sun and Widy Liao, each of whom I love to share my joy and sadness with.

Finally, I am grateful to my family for the love and encouragement. And aftermost I feel an immense gratitude to my beloved, James Tsai and Sheena Ringo, who always accompany me when I lose heart.

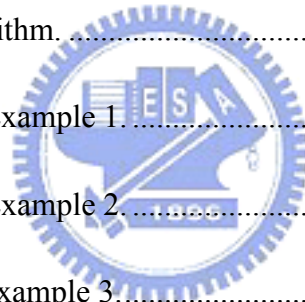
Contents

ABSTRACT (IN CHINESE)	I
ABSTRACT (IN ENGLISH)	II
ACKNOWLEDGEMENTS	III
CONTENTS	IV
LIST OF LISTS	VI
LIST OF FIGURES	VII
CHAPTER 1 INTRODUCTION	1
1.1 MOTIVATION	1
1.2 OVERVIEW	2
1.3 THESIS ORGANIZATION	3
CHAPTER 2 RELATED WORKS	5
2.1 STROKE-BASED RENDERING	5
2.2 CHINESE INK PAINTING	6
CHAPTER 3 BACKGROUND	7
3.1 THE HISTORY ABOUT CHINESE PAINTING ON ANIMALS	7
3.2 FINE-BRUSHWORK PAINTING ON ANIMALS	9
3.2.1 <i>Hair Painted with a Flattened Brush</i>	9
CHAPTER 4 STROKE GENERATION	11
4.1 USER SPECIFIED PRECISION MAP	13
4.2 COLOR FUSION	14

4.3 PRIMITIVES AND STYLE PARAMETERS.....	16
4.3.1 Strokes, Layers and Canvas.....	16
4.3.2 Style Parameters.....	17
4.4 STROKE GENERATION IN DETAIL	19
4.4.1 Initialization	21
4.4.2 Anchor Generation	21
4.4.3 Orientation	24
4.4.4 Edge Clipping.....	29
4.4.5 Color Extraction.....	32
CHAPTER 5 SYNTHESIS OF HAIR AND WASH PAINTINGS	33
5.1 WASH BRUSH RENDERING	34
5.2 INTERPOLATED CONVOLUTION	36
5.3 COLOR WASH EFFECT BY MEAN SHIFT FILTER	38
5.3.1 The Mean Shift Procedure.....	38
5.3.2 The Mean Shift Filter.....	41
5.4 HAIR BRUSH RENDERING	44
5.5 COMPOSITION	45
CHAPTER 6 IMPLEMENTATION AND RESULTS.....	46
CHAPTER 7 CONCLUSIONS AND FUTURE WORKS	53
REFERENCE	55

List of Tables

Table 4.1 The algorithm of color fusion.....	15
Table 4.2 Style parameters used to generate artistic renders.....	18
Table 4.3 Canny edge detection procedure.....	30
Table 4.4 Stroke clipping algorithm.....	31
Table 6.1 Style parameters of Example 1.....	47
Table 6.2 Style parameters of Example 2.....	49
Table 6.3 Style parameters of example 3.....	51



List of Figures

Figure 1.1 The system flowchart.	4
Figure 3.1 (a) Grazing horses by Han Kan. (b) Horse by Xu Beihong [4]. (c) Eight horses by Lang Shining. (d) Rabbits (A year of good harvest) by Chang Ke-Chi (張克齊) [2].	8
Figure 3.2 Examples of hair painted with a flattened brush excerpted from [1].	10
Figure 4.1 Overview of our stroke generator.	12
Figure 4.2 The user specified precision map. (a)(c) Original images. (b)(d) The corresponding precision map defined by the user.	13
Figure 4.3 Example of region adjacency graph (RAG).	14
Figure 4.4 Examples of color fusion. (a) Original image of a cat. (b) $(\sigma_{R}, \minRegion) = (6.5, 20)$ of (a). (c) Original image of a puppy. (d) $(\sigma_{R}, \minRegion) = (6.5, 20)$ of (c).	16
Figure 4.5 Brush stroke object.	16
Figure 4.6 The stroke generation procedure.	20
Figure 4.7 Frequency edge images. (a) Color-fused image. (b)- (e) are different frequency edges, from low to high.	23
Figure 4.8 RBF gradient interpolation. (a) Original image. (b) Local gradients. (c) RBF gradients. Large strokes are basis points	28
Figure 4.9 (a) Two dimensional Gaussian distribution with mean $(0, 0)$ and $\sigma = 1$. (b) Discrete	

approximation to Gaussian function with $\sigma = 1.0$	29
Figure 5.1 Overview of our rendering procedure.....	34
Figure 5.2 Examples of our brush textures.....	35
Figure 5.3 The process of our wash painting in detail.....	35
Figure 5.4 Example of median convolution with size = 3×3	36
Figure 5.5 Example of interpolation. (a) Wash brush image of Gaussian blur. (b) (a) after mean shift filtering. (c) (b) after Alpha blending with value 0.9. (d) Detail image. (e) (d) after median. (f) Interpolation with (c) and (e).....	37
Figure 5.6 Example of a feature space. (a) A 400×276 color image. (b) Corresponding L*u*v color space with 110,400 data points.....	38
Figure 5.7 Example of a 2D feature space analysis. (a) Two-dimensional data set. (b) Seven clusters. (c) Trajectories of the mean shift procedures.....	41
Figure 5.8 Cameraman image. (a) Original. (b) Mean shift filtered (h_s, h_r) = (8, 4).....	42
Figure 5.9 Visualization of mean shift-based filtering and segmentation for gray-level data. (a) Input. (b) Mean shift paths for the pixel on the plateau and on the line. (c) Filtering result (h_s, h_r) = (8, 4).	43
Figure 5.10 Examples of mean shift filtering with (h_s, h_r) = (10, 15).	44
Figure 5.11 Examples of hair brush rendering. (a)(c) Original images. (c)(d) Hair paintings. 45	
Figure 6.1 Example 1.	47
Figure 6.2 Resulting images of example 1.	48
Figure 6.3 Example 2.	49
Figure 6.4 Resulting images of example 2.	50
Figure 6.5 Example 3.	51
Figure 6.6 Resulting images of example 3.	52

Chapter 1

Introduction

1.1 Motivation

The computer graphics community, in addition to concentrating on photorealistic rendering, strives for automatic methods to generate non-photorealistic and expressive renderings. The goal of Non-Photorealistic Rendering (*NPR*) is not to render object realistically but more creatively through simulating various painting styles such as pencil sketch, watercolor, oil painting and Chinese Painting.

Chinese Ink Painting is a traditional art with a long history in China. There are various painting techniques and people are marveled at these artistic works where complex scenes and objects are rendered or certain artistic spirit is conveyed. It is not trivial to simulate the style of Chinese Ink Painting since it usually uses brushes and ink as mediums expressing some concept far beyond the precise appearance of the subjects.

In this thesis, we propose a system to synthesize realistical style of Chinese fine-brushwork painting specific on animals. We focus on illustrating animals because, in life routine, there are various creatures around and it is interesting to make a drawing on them in

such a style. And also this is a novel field which has not yet been explored in the NPR domain. The system flowchart is shown in Figure 1.1.

1.2 Overview

First of all, the user inputs a reference image of the subject, such as a feline or a canine. The user needs to specify the feature regions on the animal, for instance, its eyes and mouth. We provide a friendly user-interface such that it facilitates the user for the designation. The featured regions are great assistance for our system to deal with these regions in various ways. Several parameters need to be defined including brush texture images and other parameters controlling the distribution of strokes.

The next step is the color fusion. Since strokes ought to be put right within the subject, it is necessary to recognize where the background region is and thus it is avoidable to spread strokes in it. We approach it by color fusion and indicating the index of background region. It is simple and more general than indicating the background color since there are probably some regions of the subject with the same color as the background. And it is also easier to the user than extracting the outline off manually since the animal may be furry and fuzzy.

After the color-fused image and its region map are produced, our system does the stroke generation with these input data. The stroke generation relies on image analysis to determine the stroke properties, such as color, orientation, length and position. The user can control the amount of strokes through the input parameters.

Then our system employs these strokes in two different usages. One is the rendering process for color washing, and the other is the hair drawing. These two rendering tasks perform by means of image processing and based on textures which are created from Chinese brushes in reality. Finally these two rendered paintings are compounded into one which is the

final rendering result.

1.3 Thesis Organization

The rest of this thesis is organized as follows. In Chapter 2, we review the related works in non-photorealistic rendering. Then, in Chapter 3, we introduce characterized Chinese fine-brushwork painting on animals. Chapter 4 describes the details of the component of our stroke generator. The rendering procedure is presented in Chapter 5. The experimental results of our technique are demonstrated in Chapter 6. Finally, conclusions and future works are given in Chapter 7.



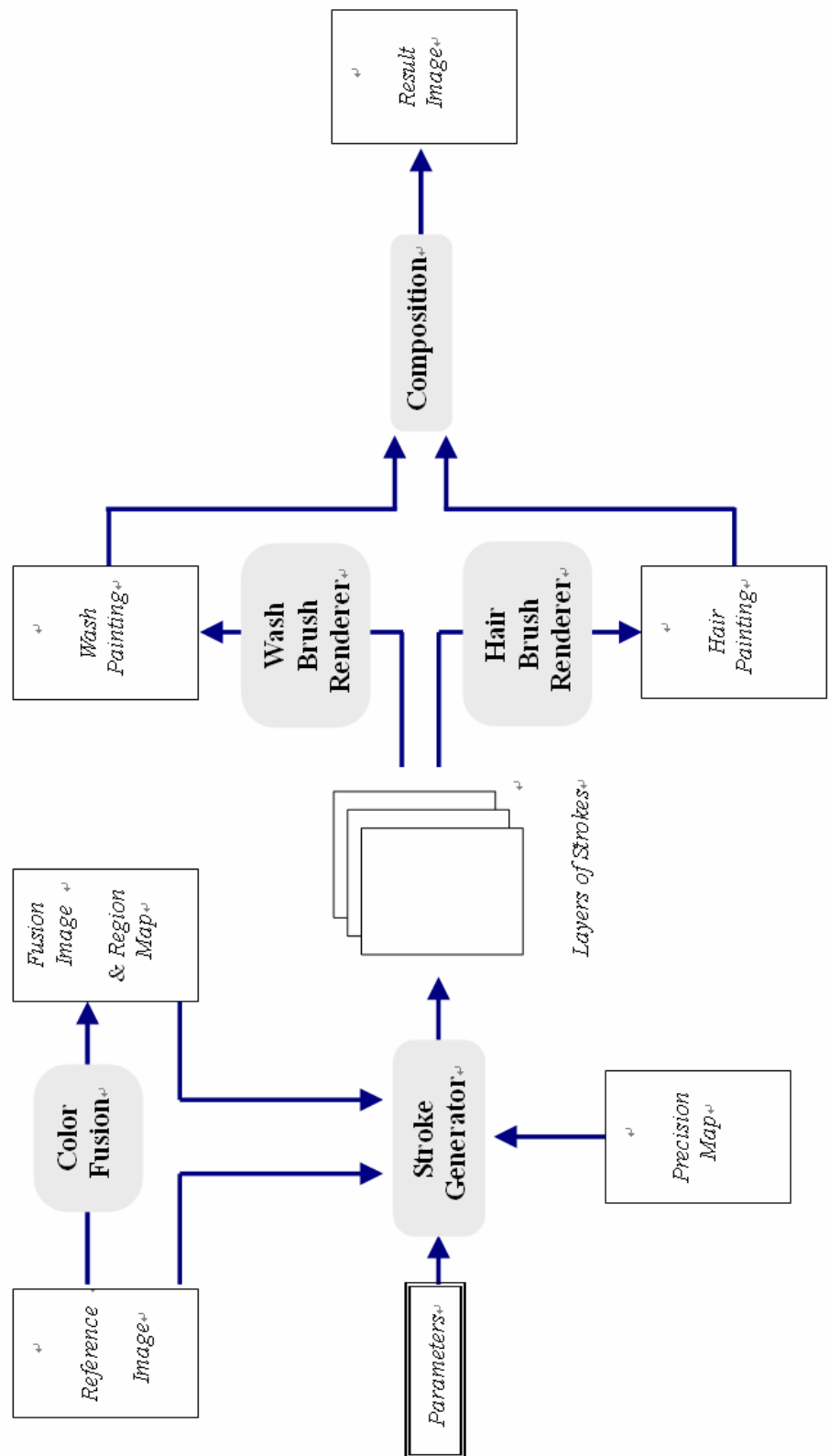


Figure 1.1 The system flowchart.

Chapter 2

Related Works

2.1 Stroke-Based Rendering

Haeberli [11] developed an interactive system, which enables a user to place strokes on a canvas. The stroke distribution is given with an input device such as a mouse. A stroke color is chosen from the colors in the corresponding region of the source image. A stroke orientation as well as size can be controlled by the user. In addition to that, he proposed a method to control the stroke orientation automatically using the intensity gradient of the source image.

Litwinowicz [14] takes a video sequence as its input. This work concentrates on exploiting the frame coherence for a painterly animation sequence. The jittered grid is used as the initial stroke distribution. A stroke color is determined by taking from the source image just as Haeberli's work. The stroke size may vary, because a stroke may be clipped by the edges detected in the source image to preserve the contours of objects. However, the stroke width is constant over the canvas. The orientation is determined by the intensity gradient, which is similar to Haeberli's method [11] as well. The painting order is randomized to obtain a hand-crafted touch.

Hertzmann [12] introduced a method to automatically paint still images with various stroke sizes and shapes and then extended the technique to video [13]. His method composes an image with several layers. Each layer is painted by the strokes with its own constant width. The upper layer is painted with narrower strokes. As a result, the canvas image gradually gets close to the source image. The resulting image contains strokes with different widths. The orientation of spline curves are controlled by the intensity gradient.

2.2 Chinese Ink Painting

The researches about traditional Chinese painting are not as many as the Western painting arts like watercolor [7], pen-and-ink [18][19][20][26][27], oil painting[12] and so on.

In order to simulate stroke characteristics in Chinese Landscape Painting, Weng et al [25] introduced a two-dimensional brush model. The bristles are distributed uniformly as concentric circles, and adding some physics properties such as pressure and ink decline. Later, Huang [15] proposed a new method physically imitating the behavior of ink diffusion in paper structure. He used particles representing carbon and water, simulating the physical phenomenon of diffusion on the paper.

Way and Shih [23] focused on Chinese landscape painting of Tang dynasty (618-907 AD) and Song dynasty (960-1279 AD) and synthesized rock textures by drawing *Hemp-Fiber* texture strokes and *Axe-Cut* texture strokes respectively. Way et al [24] then developed another application to simulate the Chinese painting of trees by using silhouette and texture strokes. Three-dimensional tree models were used as their source models, and they can draw various styles of bark texture by defining the texture patterns.

Borrowing from [12] [14], we intend to develop a semi-automatic rendering system to composite the painting style of the specific Chinese fine-brushwork painting on animals.

Chapter 3

Background

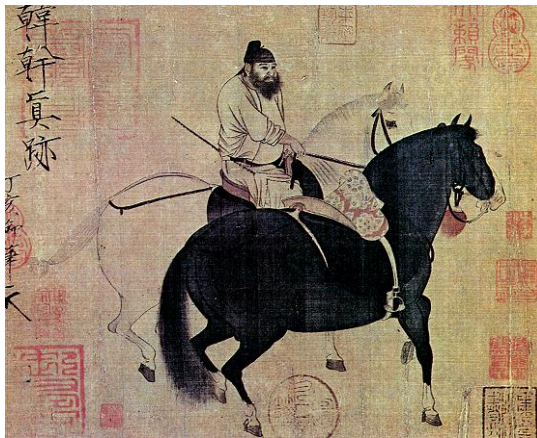
3.1 The History about Chinese Painting on Animals

In the Tang dynasty (618-907 AD), in addition to figures and landscapes paintings that had already reached the extremely high realm, flowers, birds and animals also formed the individual style of drawing. And also many artists were fond of still life paintings on animals. At that time, Ts'ao Pa (曹霸) was one of the most famous master with the divine horse painting. One of his student, Han Kan (韓幹), as a Chinese proverbial saying “the pupil surpasses the master”, was an expert at painting in according to actual live horses and was the first one who promoted still life painting in China.[3]

In the middle period of Qing Dynasty, the animal and flower painter, Lang Shining(郎士寧) (1688~1766), or Giuseppe Castiglione as his Italian name, came to China as a Jesuit missionary and brought his skill of European painting. He turned a Chinese court artist with blending of Chinese and Western painting techniques.

In the twentieth century, more and more drawings of animals came out. Xu Beihong (徐悲鴻) was a celebrated artist and the painted horses are full of passion and power underneath

his brief and semiabstract strokes. Figure 3.1 shows representative works of these artists.



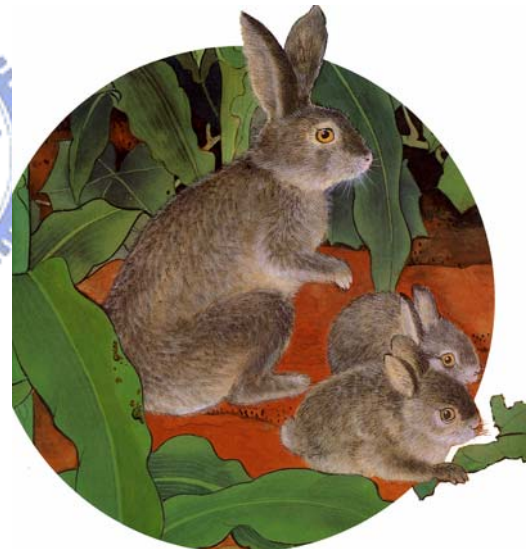
(a)



(b)



(c)



(d)

Figure 3.1 (a) Grazing horses by Han Kan. (b) Horse by Xu Beihong [4]. (c) Eight horses by Lang Shining. (d) Rabbits (A year of good harvest) by Chang Ke-Chi (張克齊) [2].

In this thesis, we focus on highly realistic depiction in describing nature objectivity and catching accurate form on animals as combining Western drawing with Chinese calligraphic strokes. One example is shown in Figure 3.1(d).

3.2 Fine-Brushwork Painting on Animals

There are diverse skills to paint an animal in Chinese fine-brushwork. Referring to [1], this book introduces several skills to paint the head and the hair of cats and tigers. In regard to the hair, we simulate one method which is presented in the book. This method is called “hair painted with a flattened brush” (破鋒絲毛法). This method is described as follows and takes cats for its painting subject.

3.2.1 Hair Painted with a Flattened Brush

When painting the hair of the cat with a flattened brush, the brushwork should be expressive of the texture of the hair.

1. The sketch of the form of the cat starts from the head.
2. Then proceed to the neck, the back, the abdomen, the legs, and the tail.
3. When rendering the hair, the procedure is the same as above.

There is a sense of dryness in the flattened brushwork. Therefore, after the hair is rendered with flattened brushwork, apply wet washes of ink mixed with cyanine to the whole body of the cat to enrich the texture. Examples are illustrated in Figure 3.2.

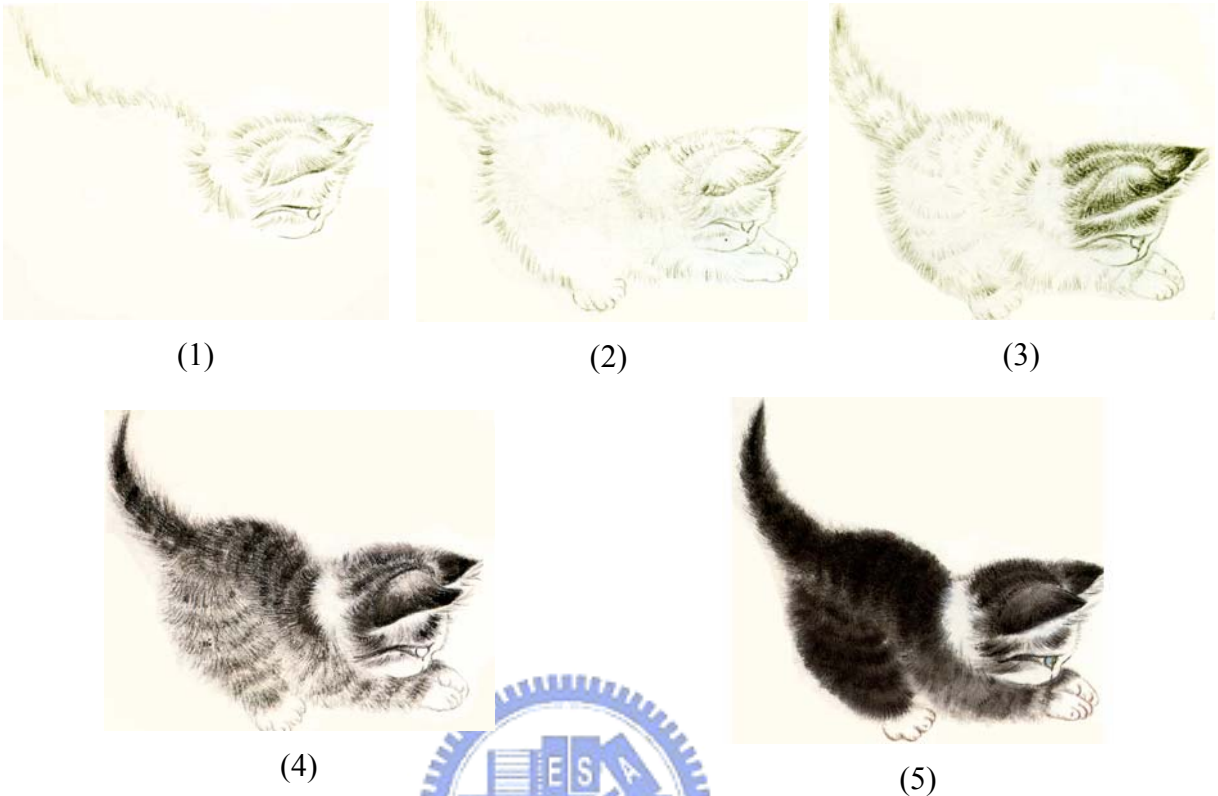


Figure 3.2 Examples of hair painted with a flattened brush excerpted from [1].

We observe the painting style carefully and summarize it as follows. The orientation of brush is greatly related to the contour coherence of the subject figure. The stroke distribution is regularly near the edges. The variation of each stroke length is small. Although the colored wet brushes are used in gradation to flesh out the contour, the hair brushes are still apparent and noticeable. These heuristics are carried out by means of image analysis and processing in our system.

Chapter 4

Stroke Generation

In this chapter, we introduce the stroke generator component of our system. The overview of our *stroke generator* is shown in Figure 4.1.

The *stroke generator* we presented takes four objects as input: 1. the *reference image*; 2. the *precision map*; 3. the *color fusion image* and its *region map*; and 4. the *style parameters*.

The reference image is a photograph of the subject, such as a dog or a cat. In Section 4.1, the precision map specified by the user is to control the tone and the fineness of the stroke distribution. In Section 4.2, the color fusion image and its region map are generated from the reference image used to extract edges. The relevant style parameters used to generate artistic renders are introduced in Section 4.3. Then we describe the kernel of our stroke generator in Section 4.4.

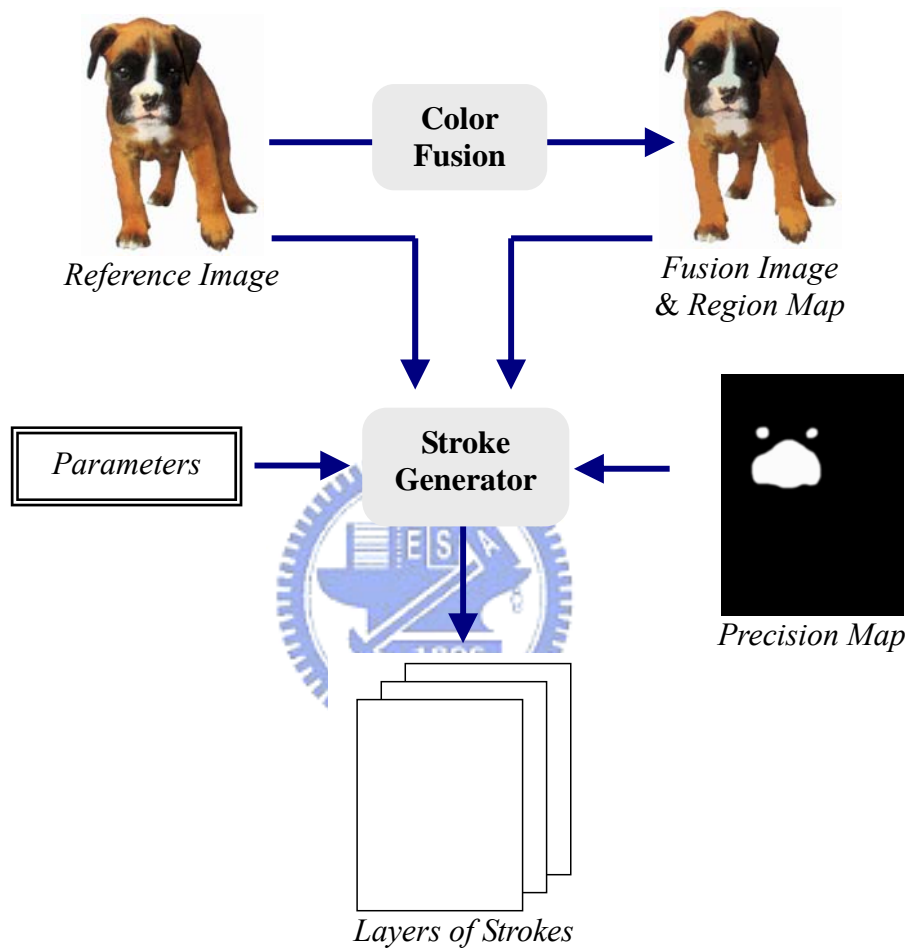


Figure 4.1 Overview of our stroke generator.

4.1 User-Specified Precision Map

Inspired by [9], our system proposes a simple model of precision, controlled spatially through the *precision map*. It is common artistic practice to use coarse tones for regions of little importance, and fine spatial details for regions of focus. Take animals for example, the artist would like to keep the detail information for facial features like the eyes, the nose and the teeth, and leave the other regions, say, the body, in rough.

To make the *precision map* as shown in Figure 4.2, the user first draws the regions of interest (ROI) interactively. By the mouse drawing, the user directly draws the shape of regions over the reference image. To facilitate the designation, we provide three different type of drawing: *freehand*, *rectangular* and *elliptic*. After making ROIs, an intermediate image is created by first filling the background in black and then drawing the pixels within the ROIs in white. In case it causes some aliasing near the boundaries of ROIs, we blur the intermediate image by Gaussian filter and thus the *precision map* is. The precision map is used as a gray-level mask in the stroke generator and the wash renderer.

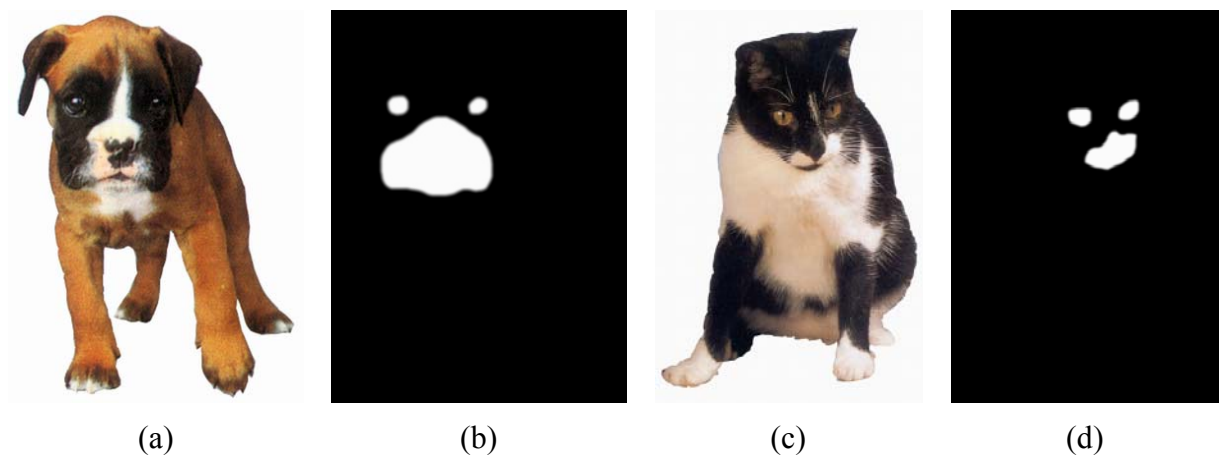


Figure 4.2 The user specified precision map. (a)(c) Original images. (b)(d) The corresponding precision map defined by the user.

4.2 Color Fusion

In the fusion step, extensive use was made of region adjacency graphs (RAG) and graph contraction with a union-find algorithm [21]. The RAG provides a spatial view of the image. A RAG consists of vertices (or nodes) associated with each region and lines joining each pair of adjacency regions. By definition, this region adjacency graph provides a simple-connectivity view of the image as illustrated in Figure 4.3(b).

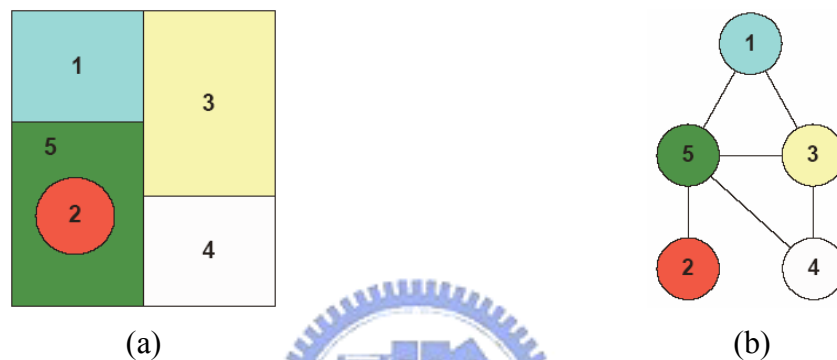


Figure 4.3 Example of region adjacency graph (RAG).

The initial RAG was built from the reference image. The fusion was performed as a transitive closure operation on the graph, under the condition that the color difference between adjacent nodes should not exceed $h_r/2$ where h_r is the user-defined threshold. At convergence, the color of the regions was recomputed and the transitive closure was again performed. After at most three iterations the final labeling of the image (segmentation) was obtained. Small regions (the minimum size, M is defined by the user) were then allocated to the nearest neighbor in the color space. The algorithm is as described in Table 4.1.

Input:	
$LUV_{(p)}$	{ Color Luv at pixel position p }
σR	{ Color merge threshold }
$minRegion$	{ Minimum size of region }
Data Structures:	
$N_{(p)}$	{ 8 connected neighborhood of p }
$Labels(p)$	{ Region label of p }
$RAList(r)$	{ Region adjacency list at region r }
Output:	
$Modes(p)$	{ Updated region color of p }
Algorithm:	
InitializeOutput();	{ Initialize output data structure }
$label \leftarrow 0;$	{ Initialize # of regions }
for each p do $Labels(p) = -1$	{ Initialize labels }
for each p do begin	{ Traverse the image labeling each new region encountered }
if $Labels(p) = -1$	{ If this region has not yet been labeled, label it }
$Labels(p) = ++label;$	
$Modes(p) = LUV_{(p)}$	{ Copy region color into modes }
Fill ($p, label$);	
{ Populate labels with label for this region via an 8-connected fill }	
end	
end	
$iteration = 0; \delta RC = 0;$	{ δRC is difference of region # between $iteration$ }
repeat { Apply transitive closure iteratively to the regions }	
TransitiveClosure ();	
{ Update $Labels(p)$ & $Modes(p)$ until the each neighboring region color $< \sigma R$ }	
until $\delta RC > 0$ or $iteration > 10$	
Prune ($minRegion$);	
{ Prune spurious regions (regions whose area is under $MinRegion$) }	

Table 4.1 The algorithm of color fusion.

After applying the color fusion operation, we will get a color-fused image and its region map. The color-fused image will be used to guide the stroke distribution, and the region map will be used to avoid spreading strokes in the background region while the background region index is defined. Figure 4.4 illustrates examples of color fusion with different parameters.

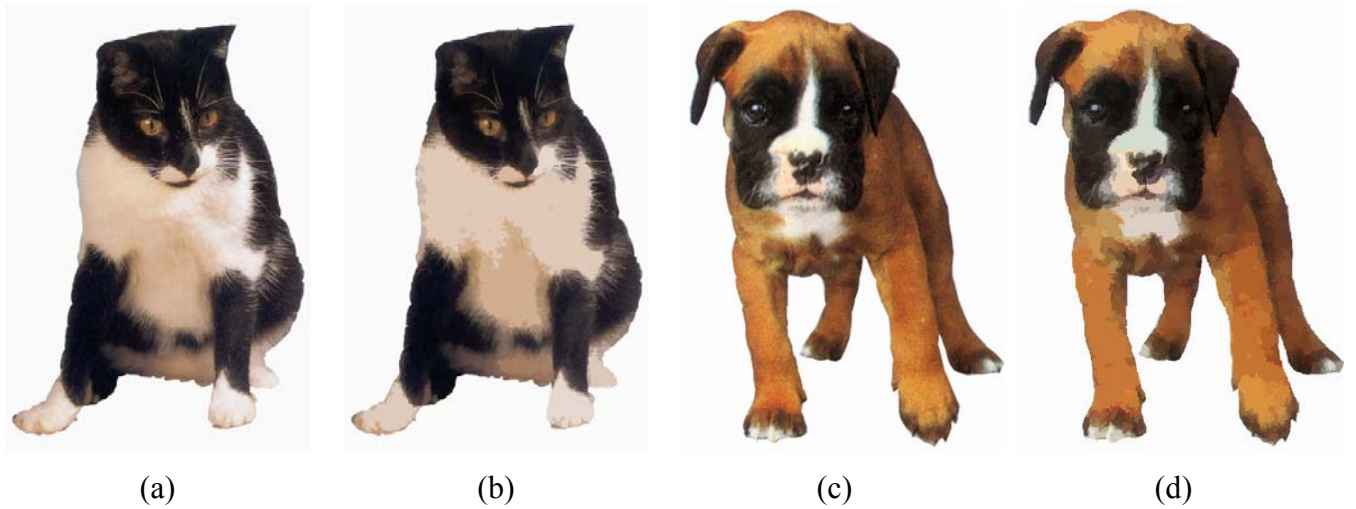


Figure 4.4 Examples of color fusion. (a) Original image of a cat. (b) $(\sigma_R, \text{minRegion}) = (6.5, 20)$ of (a). (c) Original image of a puppy. (d) $(\sigma_R, \text{minRegion}) = (6.5, 20)$ of (c).

4.3 Primitives and Style Parameters

In this section, we introduce the primitives and parameters used in our system.

4.3.1 Strokes, Layers and Canvas

Each brush stroke has several properties that are determined by the analysis of input images. Our brush stroke definition most closely resembles that of [14]. Each brush stroke is an object with the following properties: 1. Anchor point in image coordinates; 2. Angle of orientation in radius; 3. Width in pixel; 4. Lengths in both directions from the anchor point in pixels; 5. Color (R, G, B). Anchor, angle, width, and lengths are kept in floating-point coordinates. A brush stroke object is illustrated in Figure 4.5.

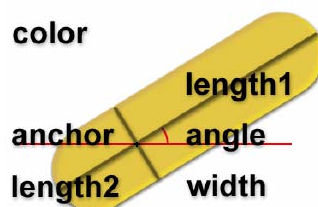


Figure 4.5 Brush stroke object.

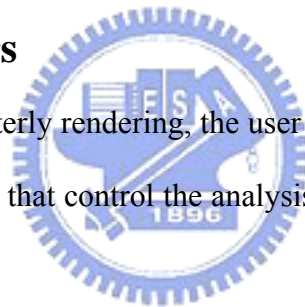
In order to emulate the coarse-to-fine painting process, we extend the concept of *Layers*

which is introduced in [12]. Layers are disjoint groups of brush strokes representing successive passes of refinement in a painting. As used here, brush strokes are created within a Layer only when perceptually necessary as determined by the presence of edges at the frequency band corresponding to each Layer. We set an additional layer of ROI to draw the finest strokes in the region within the precision map in order to keep the greatest detail of the animal features.

Output is rendered onto an arbitrarily large *canvas*. Using a canvas of higher resolution than an input image is analogous to a painter filling an entire canvas according to a small reference photograph. It allows the output rendering to retain the structural coherence of the input image while still displaying considerable artistic styling.

4.3.2 Style Parameters

In order to produce a painterly rendering, the user should first pick a *style*. Here a style is an encapsulation of parameters that control the analysis of input images. Our style parameters are shown in Table 4.2.



Brush Width	w_b	Width of the brush stroke
Stroke Generation Width	w_r	The minimum window size of unoccupied space on the canvas for which a new brush stroke will be created to fill
Color Brush Texture	TC_b	Set of brush texture and mask to use in wash rendering
Hair Brush Texture	TH_b	Set of brush texture and mask to use in hair rendering
Color Fusion	σR $minRegion$	σR is the color difference between neighboring regions, and $minRegion$ is the minimum size of a region
Background Region Index	r_{bg}	Background region index to avoid spreading strokes in the background region
Gradient Threshold	T_g	Threshold of gradient magnitude used at stroke orientation stage.
Basis Point Radius	R_{bpt}	Radius used at stroke orientation stage to control the dense of basis points
Canny Edge Detector	T_{low} T_{up}	Lower threshold and upper threshold used in Canny edge detector
Mean Shift Filter	h_s h_r	h_s is the spatial bandwidth and h_r is the color bandwidth for mean shift filtering
Canvas Size	C_s	Canvas size scale factor
Convolution Size	m_1 m_2	m_1 is Median convolution size used for detail and m_2 is used as Gaussian blur size for non-detail parts of a image after wash brush rendering
Alpha Blending	$alpha$	Alpha blending value used in median interpolation

Table 4.2 Style parameters used to generate artistic renders.

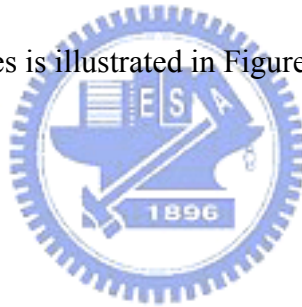
These parameters are defined for each layer. The number of layers is also defined by the user. The wash brush textures and hair brush textures are especially made by a real Chinese painting brush.

4.4 Stroke Generation in Detail

Now we describe in detail each step in our process of the stroke generation. There are five stages of processing:

1. Initialization;
2. Stroke Anchor Generation;
3. Orientation;
4. Edge Clipping;
5. Color Extraction.

An overview of these stages is illustrated in Figure 4.6.



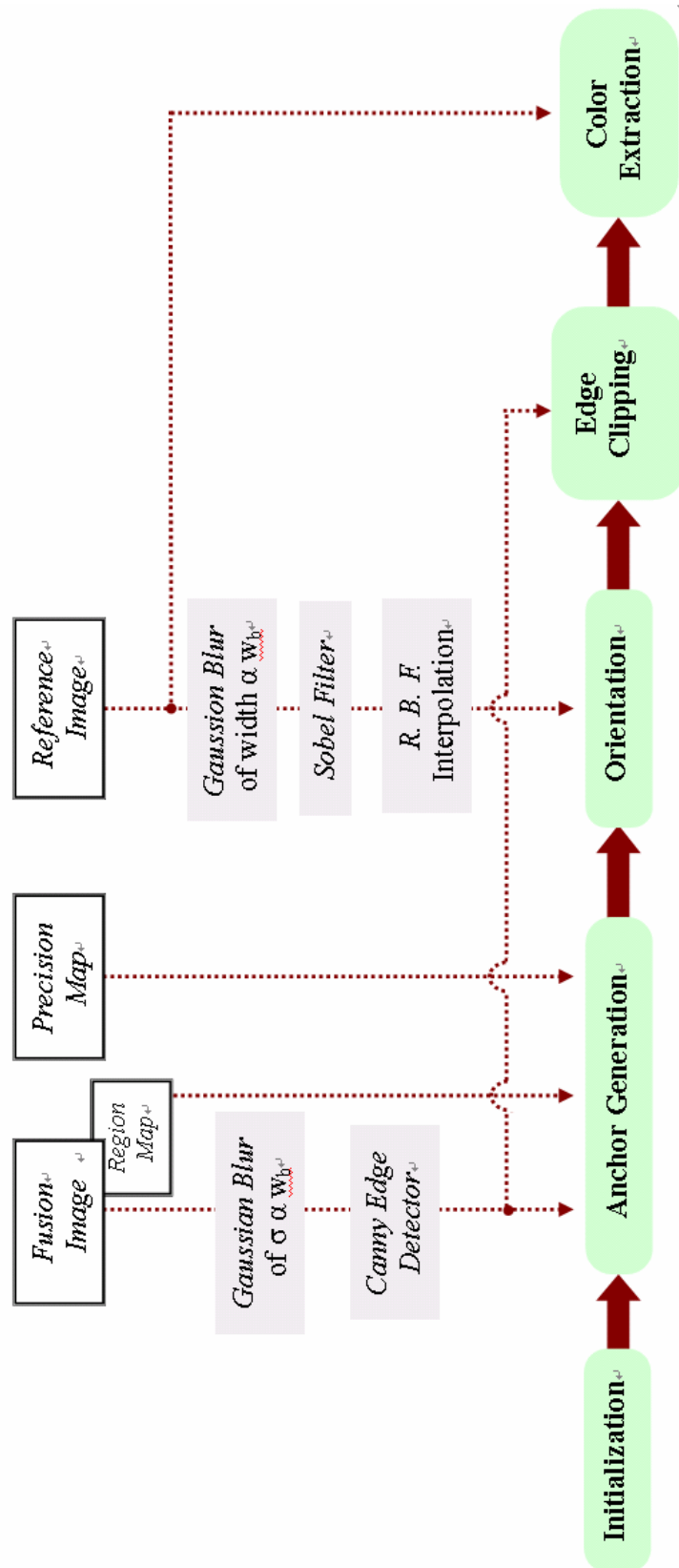


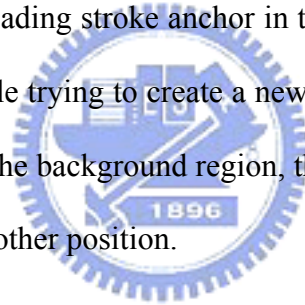
Figure 4.6 The stroke generation procedure.

4.4.1 Initialization

First all layers are initialized. The number of layers depends on the *style* which is determined by the user. Each layer contains an array of brush strokes which is initially empty. The layers will be filled, as necessary, in the next step.

4.4.2 Anchor Generation

In this step, we generate the anchor position of each stroke with the help of the region map. From the color fusion process, we get a color-fused image and its region map which labels where a pixel belongs to. By indicating the index of the background region, it is easy to avoid spreading stroke anchor in the background. We simply look up the region map (table) while trying to create a new stroke position. If the region of the position we picked is not the background region, the creation is success; otherwise it fails and goes on finding another position.

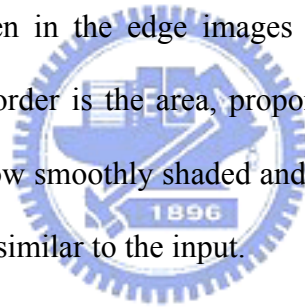


At the bottom layer, the canvas is searched and new brush strokes are created where gaps are found. Gaps are simply defined as area w_r across that contains no brush stroke anchor point. To make strokes less disorder, we restrict the bottom layer strokes to areas “near” the edges but not “on” the edges. If the stroke generation width, w_r , is made larger than the actual width of a brush stroke, it will allow the canvas to show through, that is, the effect of “white- reserving.” Oppositely, making w_r smaller will make the brush strokes appear more crowded. The number of strokes is approximated by

$$\text{Approximated number of strokes} = \frac{\text{areas}(\text{Canny}, w_r)}{w_r}.$$

The anchor point positions are generated randomly and are examined whether it is within areas near edges and is outside the background region. The canvas is searched in pseudo-random order (versus scan-line) in order to avoid certain mechanical looking artifacts that can result if brush strokes are generated in sequence and end up perfectly spaced.

The same process is performed on the higher layers (except the highest), but the generation is restricted to areas near edges of the appropriate frequency (see subsection 4.4.4). That is, we allow strokes generated “on” the edges in these layers. For example, brush strokes in the middle layer are placed on or near the middle frequency edges in the color-fused image. The frequency edge images are illustrated in Figure 4.7. The dark green in the edge images are the detected edges at each frequency. The light green border is the area, proportional to w_r , that brush strokes will be created to fill. Note how smoothly shaded and out of focus regions are refined while remaining perceptually similar to the input.



On the highest layer, called ROI layer, the generation is restricted to areas within the regions of interest (ROIs). ROIs are specified by the user during the precision map definition. The ROIs is searched and new brush strokes are created where gaps are found. Here the w_r of the highest layer is relatively small to make strokes in dense. This is to emulate the painter placing emphasis on some regions as animal facial features.

We believe proximity to edges is a more perceptually accurate description of how an artist might refine a painting than what previous techniques employed. In [12], it creates brush strokes in higher layers when a rendered version of the lower layer

differs too much in color from the original image. In Chinese fine-brushwork painting or other styles, painters worry about refining the structural accuracy of an image rather than correcting color in otherwise empty regions.

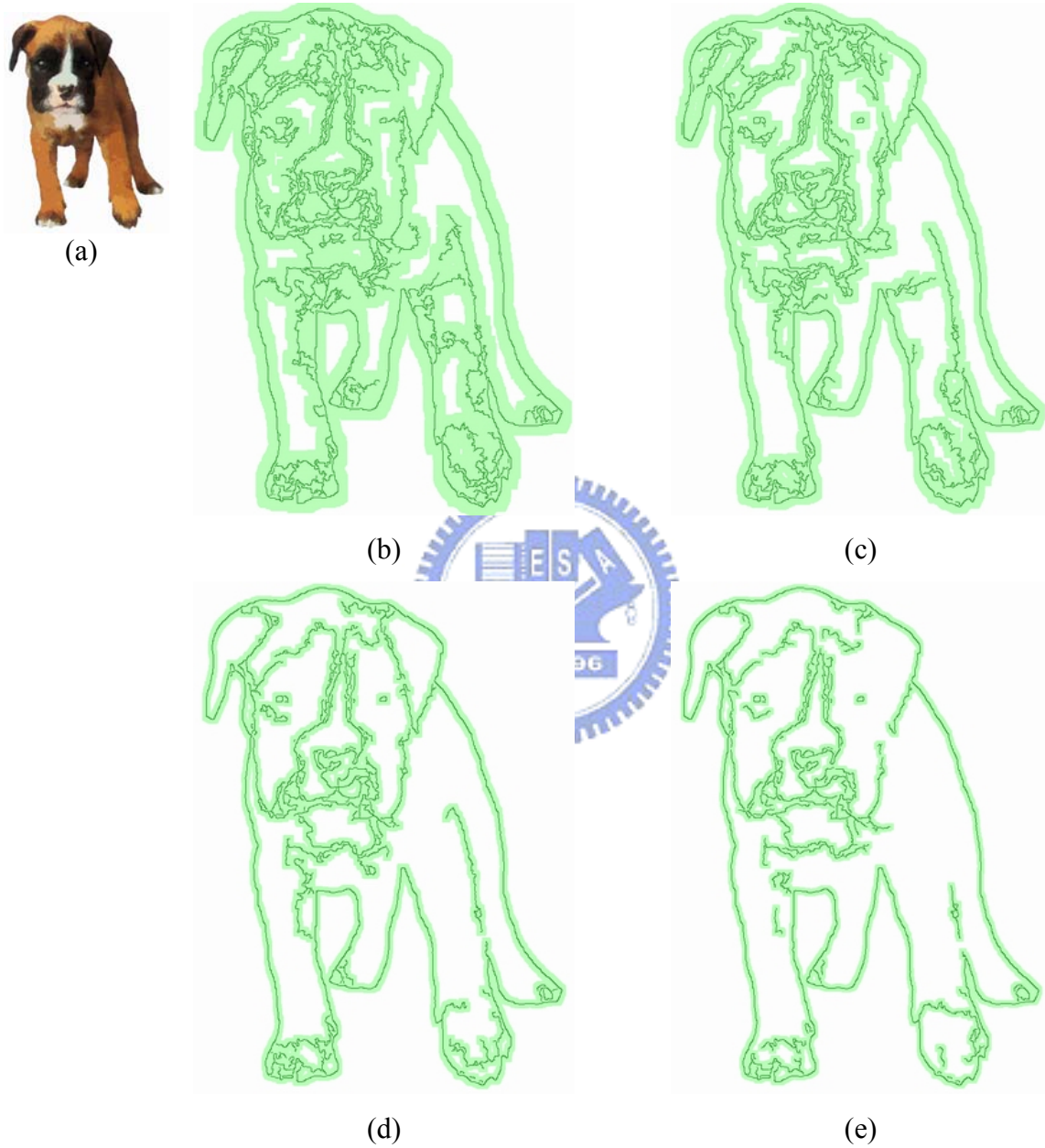
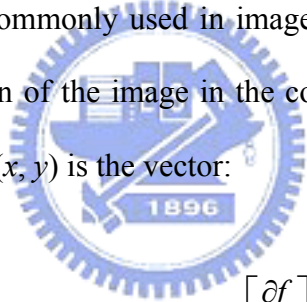


Figure 4.7 Frequency edge images. (a) Color-fused image. (b)- (e) are different frequency edges, from low to high.

4.4.3 Orientation

Painters typically paint by following contours of features in an image. Instead of orienting each brush stroke largely based on the local gradient estimates as suggested by [12][14], we believe that brush stroke orientation should be globally dictated only by the brush stroke lying on the strongest gradients. We achieve this by using radial basis functions (RBFs) to globally interpolate \mathbf{X} and \mathbf{Y} gradients from only the strongest gradients [5].

For each layer, the reference image is blurred by a Gaussian kernel of width proportional to w_b . Gradients on the layer are then estimated across the entire image with a *Sobel* filter which is commonly used in image processing technology [10] to approximate the differentiation of the image in the continuous domain. The gradient of an image $f(x, y)$ at location (x, y) is the vector:


$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

And the magnitude of this vector which generally referred to simply as the gradient and denoted as ∇f_{mag} :

$$\nabla f_{mag} = mag(\nabla f) = \sqrt{G_x^2 + G_y^2}$$

The computation of gradient based on the Sobel kernel is

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \text{ and } S_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}.$$

And the stroke located at (x, y) is determined to be “*Strong*” or not as

$$stroke(x, y).isStrong = \begin{cases} 1 & \nabla f_{mag}(x, y) > T_g \\ 0 & otherwise \end{cases}$$

with a gradient threshold T_g , and not near other “*Strong*” brush strokes. The radius R_{bpt} keeps off the dense and numerous of strong strokes. Once the stroke is “*Strong*”, we set its gradient to be (S_x, S_y) .

These “*Strong*” brush strokes, typically about 10 to 200 per layer, become the basis points for a radial basis function. Typically, all layers contribute “*Strong*” brush strokes to the same radial basis function in order to capture the gradients at different frequency bands.

Radial basis functions (RBF) have originally been used in exact interpolation [16]. In interpolation we have n data points $\mathbf{x}_i \in \mathbf{R}^d$, and n real valued numbers $\mathbf{t}_i \in \mathbf{R}$, where $i = 1, \dots, n$. The task is to determine a function s in linear space such that $s(\mathbf{x}_i) = \mathbf{t}_i$, $i = 1, \dots, n$. The interpolation function is a linear combination of basis functions

$$s(x) = \sum_{i=1}^n w_i b_i(x)$$

as basis functions b_i , radial basis functions of the form are used.

$$b_i(x) = \phi(\|x - x_i\|)$$

where ϕ is a mapping $\mathbf{R}^+ \rightarrow \mathbf{R}$ and the norm is Euclidean distance. There are variant forms of radial basis functions, and we choose our radial basis function as:

$$\phi(r) = r$$

$$\|x - x_i\| = \sqrt{(x_0 - x_{i0})^2 + (x_1 - x_{i1})^2}$$

Next we define a matrix \mathbf{A} such that $A_{ij} = \phi(\|x_i - x_j\|)$, $i, j = 1, \dots, n$. The form of our basis function has a property that \mathbf{A} is non-singular when $n \geq 2$. The singularity plays an important role because the weights are obtained as a solution of system of linear equations. In matrix form the interpolation is simply

$$\mathbf{A}\mathbf{w} = \mathbf{t},$$

where \mathbf{t} consists of n target values, \mathbf{w} is an n -dimensional weighting vector and \mathbf{A} is a $n \times n$ - matrix as defined before. Depending on the singularity of \mathbf{A} , this equation is solved by

$$\mathbf{w} = \mathbf{A}^{-1}\mathbf{t}.$$

For the gradient interpolation, n is the number of basis points whose magnitude of gradient is greater than the specified threshold. We first set up the two equations for gradient \mathbf{X} and \mathbf{Y} respectively:

$$\mathbf{A}\mathbf{w}_x = \mathbf{G}_x \quad \text{and} \quad \mathbf{A}\mathbf{w}_y = \mathbf{G}_y.$$

\mathbf{A} is an $n \times n$ - matrix in which element a_{ij} is the value of the radial function $\phi(\|p_i - p_j\|)$, where p_i and p_j are basis points. \mathbf{G}_x and \mathbf{G}_y are two n -dimensional

vectors in which each element is the gradient \mathbf{X} and \mathbf{Y} of basis point p_i respectively.

After solving the equations by

$$w_x = A^{-1}G_x \quad \text{and} \quad w_y = A^{-1}G_y,$$

We evaluate the gradient of each non-“*Strong*” stroke q by first setting a matrix $Q = \{\phi(\|q - p_i\|)\}$, $i = 1, \dots, n$ and then computing

$$X = Qw_x \quad \text{and} \quad Y = Qw_y$$

where \mathbf{X} and \mathbf{Y} are the interpolated gradients of the stroke q . Each brush strokes angle is set as the arc tangent of the interpolated \mathbf{Y} and \mathbf{X} gradients plus 90 degrees.

By orientating brush strokes based on the strongest gradients, gradients at the edges of objects effectively dictate the internal brush stroke orientation. We believe this produces not only more coherent results, as shown in Figure 4.8, but also more accurately describes the painting style.

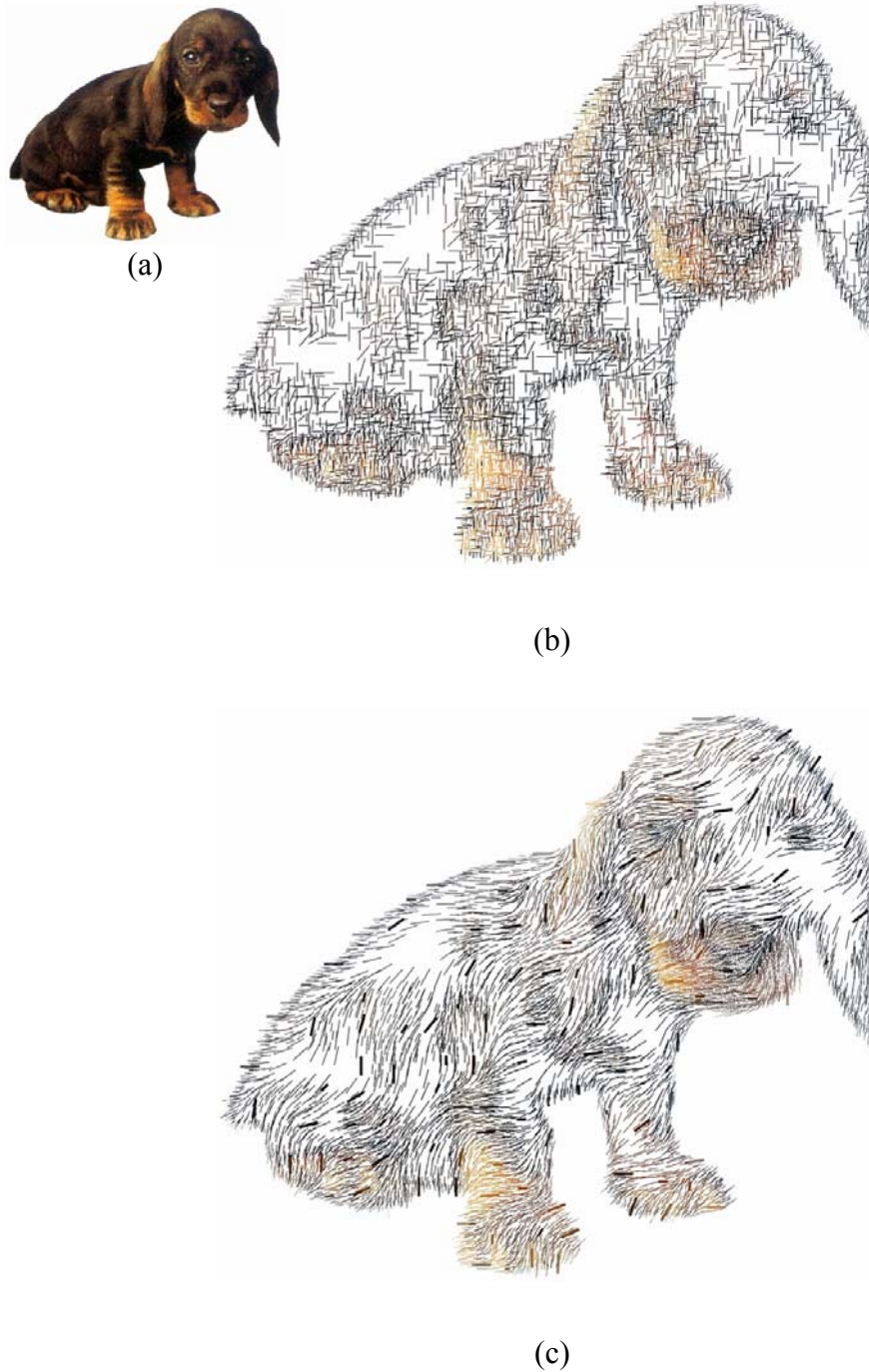


Figure 4.8 RBF gradient interpolation. (a) Original image. (b) Local gradients. (c) RBF gradients. Large strokes are basis points

4.4.4 Edge Clipping

Artist try to maintain the structural coherence of a painting by not allowing brush strokes to cross edges in an image. To emulate this process, we first detect edges in the color-fused image and then the brush strokes are clipped to these edges.

For all layers, the color-fused image is blurred by a *Gaussian* kernel of standard deviation σ proportional to w_b . The Gaussian operator is a two-dimensional convolution that is used to blur images and remove detail and noise. It uses a kernel that represented the shape of a Gaussian (bell-shape) hump. In 2-D, Gaussian has the form:

$$Gaussian(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

This distribution is shown in Figure 4.9(a).

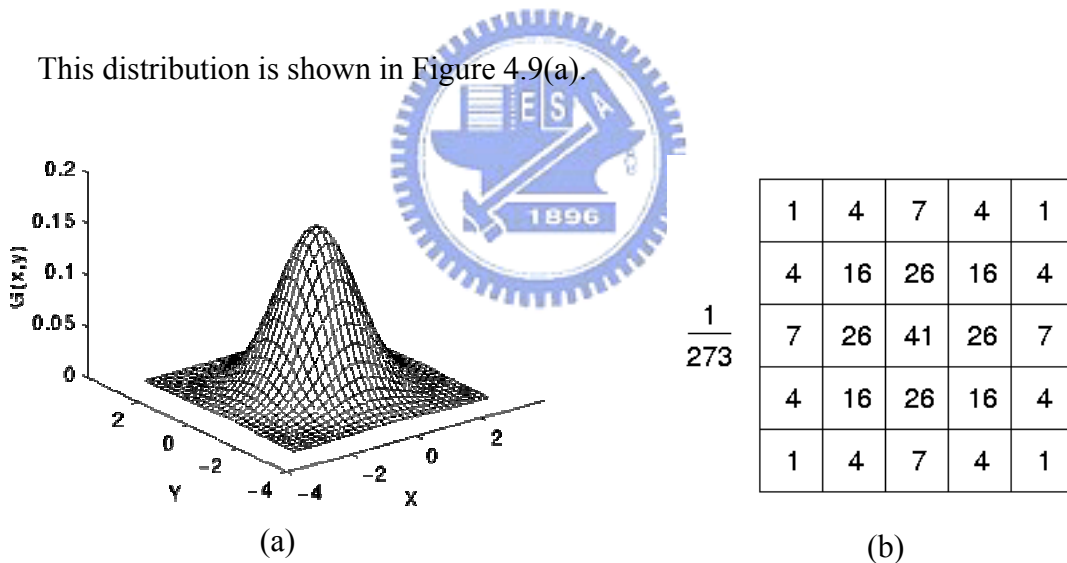


Figure 4.9 (a) Two dimensional Gaussian distribution with mean (0, 0) and $\sigma = 1$. (b) Discrete approximation to Gaussian function with $\sigma = 1.0$

The idea of Gaussian smoothing is to use this 2-D distribution as a point-spread function, and this is achieved by convolution. Since the image is stored as a collection of discrete pixels we need to produce a discrete approximation to the Gaussian function before we can perform the convolution. Figure 4.9(b) shows a suitable integer-valued convolution kernel that approximates a Gaussian with σ of 1.0.

Canny edge detection [6] is then run on the blurred images. The edge images, used to guide stroke anchor generation in subsection 4.4.2, are used here to clip the brush stroke lengths. The Canny edge detection algorithm is known to many as the optimal edge detector. We briefly describe the algorithm as shown in Table 4.3.

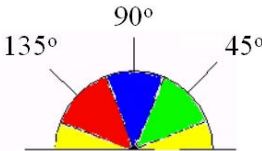
<p><u>Step 1:</u></p> <p>The first step is to filter out any noise in the original image before trying to locate and detect any edges which we have done previously (the Gaussian convolution).</p> <p><u>Step 2:</u></p> <p>The next step is to find the edge strength by taking the gradient of the image. The Sobel operator performs a 2-D spatial gradient measurement on an image.</p> <p><u>Step 3:</u></p> <p>Finding the edge direction is trivial once the gradient in the X and Y directions are known. The formula for finding the edge direction is just $\theta = \arctan\left(\frac{G_y}{G_x}\right)$</p> <p><u>Step 4:</u></p> <p>The next step is to relate the edge direction to a direction that can be traced in an image. There are only four possible directions when describing the surrounding pixels - 0 degrees (in the horizontal direction), 45 degrees (along the positive diagonal), 90 degrees (in the vertical direction), or 135 degrees (along the negative diagonal). So now the edge orientation has to be resolved into one of these four directions depending on which direction it is closest to. Think of this as taking a semicircle and dividing it into 5 regions.</p> <div style="text-align: center;">  </div> <p><u>Step 5:</u></p> <p>Nonmaximum suppression is used to trace along the edge in the edge direction and suppress any pixel value (sets it equal to 0) that is not considered to be an edge. This will give a thin line in the output image.</p> <p><u>Step 6:</u></p> <p>Finally, hysteresis is used as a means of eliminating streaking. The tracking process exhibits hysteresis controlled by two thresholds: T_{up} and T_{low}, with $T_{up} > T_{low}$. Tracking can only begin at a point on a ridge higher than T_{up}. Tracking then continues in both directions out from that point until the height of the ridge falls below T_{low}. This hysteresis helps to ensure that noisy edges are not broken up into multiple edge fragments.</p>
--

Table 4.3 Canny edge detection procedure.

The brush strokes are grown length-wise from their center until they encounter an edge.

The algorithm of stroke clipping is shown in Table 4.4.

Input:	
(x_c, y_c)	{ The anchor position of the stroke }
E	{ Canny edge image }
Data:	
(θ_x, θ_y)	{ The direction of stroke in unit }
$lastSample, newSample$	{ The sampled intensity of Edge image }
Output:	
(x_1, y_1)	{ Endpoint in positive direction of θ }
(x_2, y_2)	{ Endpoint in negative direction of θ }
Algorithm:	
1. set (x_1, y_1) to (x_c, y_c)	
2. bilinearly sample the edge image E at (x_1, y_1) and set $lastSample$ to this value	
3. set (x_{temp}, y_{temp}) to $(x_1 + \theta_x, y_1 + \theta_y)$, taking a unit step in the orientation direction	
4. if (distance[$(x_c, y_c), (x_{temp}, y_{temp})$] > ($\frac{length}{2}$ of stroke)), then stop	
5. bilinearly sample the edge image E at (x_{temp}, y_{temp}) , and set $newSample$ to this value	
6. if ($newSample < lastSample$) then stop	
7. set (x_1, y_1) to (x_{temp}, y_{temp})	
8. set $lastSample$ to $newSample$	
9. go to step 3.	
<p>At the end of this process, the endpoint (x_1, y_1) of the line in one direction has been determined. To find (x_2, y_2), the endpoint in the other direction, set (θ_x, θ_y) to $(-\theta_x, -\theta_y)$ and repeat the above process.</p>	

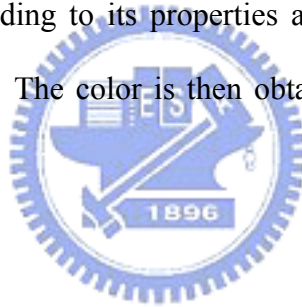
Table 4.4 Stroke clipping algorithm.

Note that the edge image here is the combination of the edge image of blurred color-fused image and that of the precision map. This is because that these strokes should not be across the feature regions for hair brushes. Once the two end points along the positive and negative directions are clipped, we randomly extend these lengths by adding a small amount of value for emulating the handcraft effect made by human beings. Then we look over the distances of them to the ROI center. The ROI center is simply at the center of the bounding box which is computed from the precision map. On account of the orientation of Chinese brush, it is necessary to rotate the stroke if its “tail end” is closer to the ROI center than the

“head end.” By our definition, the “head end” is along the negative direction of the stroke, denoted as e_2 and the “tail end” along the positive direction denoted as e_1 . We compute the Euclidean distance of e_1 to the ROI center and e_2 to the ROI center respectively: $distance(e_1, center)$ and $distance(e_2, center)$. If $distance(e_1, center)$ is less than $distance(e_2, center)$, the stroke is rotated by adding 180 degree to its angle, and also the endpoints e_1 , e_2 and their lengths to the stroke anchor are exchanged.

4.4.5 Color Extraction

Each brush stroke’s color is determined by averaging the color of pixels underneath it in the reference image. The basic rectangle shape of each stroke is known as the stroke length is determined by the previous stage and the thickness is defined by the user. We translate and rotate each stroke shape according to its properties and sample the pixels of the reference image within the stroke shape. The color is then obtained by computing the mean of these sampled values.



Chapter 5

The Synthesis of Hair and Wash

In the previous chapter, we have captured the properties of strokes for each layer. There are two usages of these strokes: wash painting and hair painting. The overview of our *renderer* is illustrated in Figure 5.1. In this chapter, we introduce how to synthesize color diffusion painting and hair painting. From Sections 5.1 to 5.3, we discuss the process of wash painting and in Section 5.4 the process of hair rendering is explained.

There are two painting canvas used in our *renderer*. One is for wash painting, and the other is for hair painting. For rendering, first the canvas is initialized to a solid color. Brush strokes are then rendered according to the drawing order from bottom layer to top layer. These brush stroke textures are created by hand by cropping brush strokes out of real Chinese painting brush. As mentioned in Section 4.3.2, two types of brush textures are used in each layer: texture pair of *wash brush* and texture pair of *hair brush*. The brush stroke texture is a pair of texture and its gray-scale alpha mask.

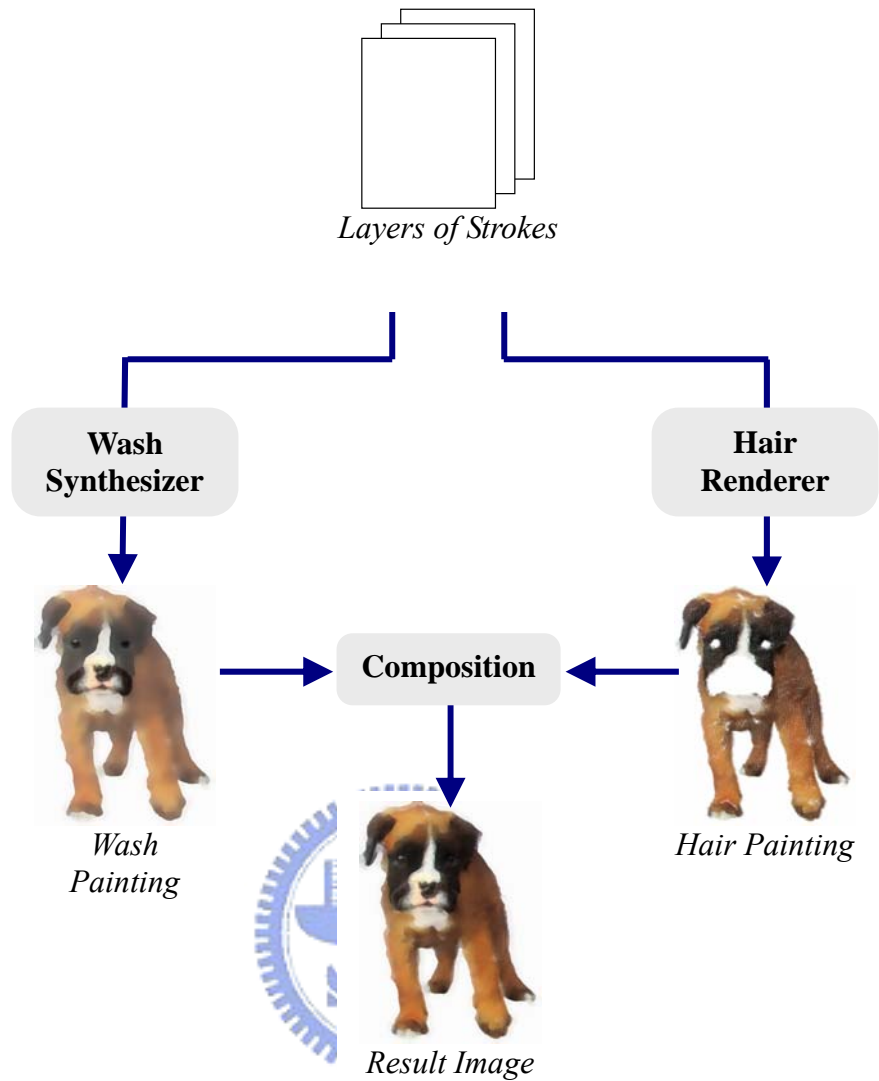


Figure 5.1 Overview of our rendering procedure.

5.1 Wash Brush Rendering

The whole wash painting process is illustrated in Figure 5.3. First we generate wash brush image by render wash brush textures. For each brush stroke to be rendered, the alpha mask and texture are first scaled to the user-defined brush width (thickness) while keeping their width to constant length ratio. Then we scale their length to the length set by the edge clipping. The texture is multiplied by the color of the stroke. The colored texture and alpha mask are then alpha-composited. The composited texture is rotated according to the angle of the stroke and translated to its anchor position on top of the existing strokes. In our

experiment, we found that it is practicable to make the texture of each texture pair to be a pure white image. Some examples of our brush stroke textures are shown in Figure 5.2.



Figure 5.2 Examples of our brush textures.

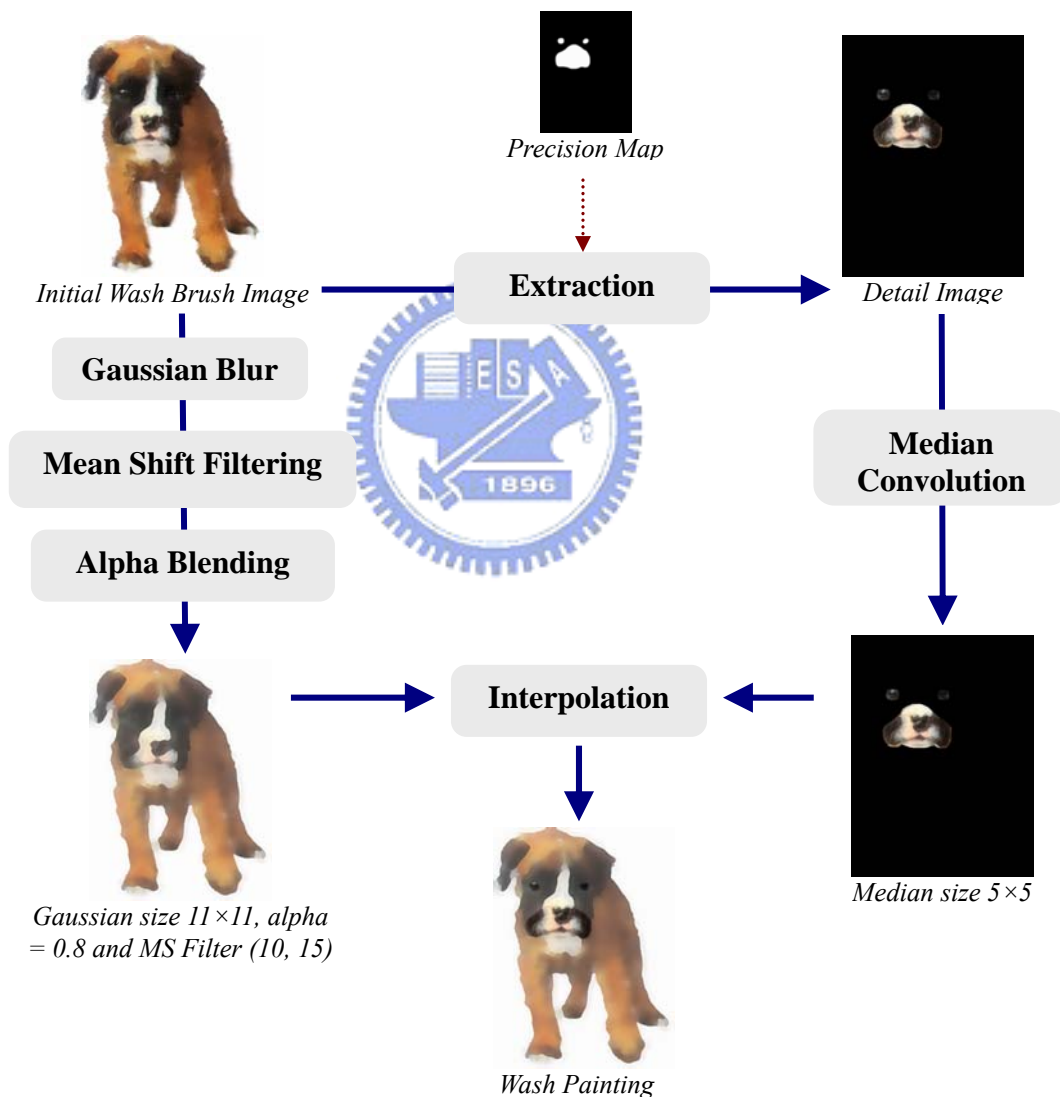


Figure 5.3 The process of our wash painting in detail.

5.2 Interpolated Convolution

After the wash brush strokes are rendered, we obtain the image called *wash brush image* and then apply interpolated convolution to it as shown in Figure 5.3. The purpose of interpolated convolution is to smooth the wash brush image while using different size of convolution to keep the featured regions in detail. Meanwhile a mean shift filter is used to approximate the color diffusion effect.

For median convolution, the *precision map* is used as a gray-level mask to extract parts of the wash brush image. Then two different sizes of convolution, median and Gaussian blur, m_1 and m_2 are applied to extracted parts and the *wash brush image* where $m_1 < m_2$. The median filter computes pixel values in the destination image by computing the median of the input pixels within a specified window. The median is calculating by first sorting all the pixel values from the surrounding neighborhood into numerical order and replacing the pixel being considered with the middle pixel value. The effect of median filtering is the removal of extraneous pixels that do not fit in the image. Figure 5.4 illustrates an example.

Original Intensities

3	3	4
4	87	4
4	5	5

Sorted pixel values: 3, 3, 4, 4, 4, 4, 5, 5, 87

Median filtered value: 4

Figure 5.4 Example of median convolution with size = 3×3 .

After Gaussian blur to the wash brush image, a filter called *mean shift filter* which will be explained in Section 5.3 is applied to it. Then we alpha blend the image with a user-defined value for the reason that the hair over these regions ought to be apparent and noticeable.

Afterwards, the interpolation operation takes these two images as two source images. The interpolation operation is a composite implements the Porter-Duff “over” rule [16], in which the output color of a pixel with source value/alpha tuples (A, a) and (B, b) is given by

$$a * A * (1 - a) * (b * B)$$

The alpha channel of detail parts is the *precision map*, and the alpha channel of wash brush image is *null* which means the wash brush image is considered completely opaque. Figure 5.5 shows an example.

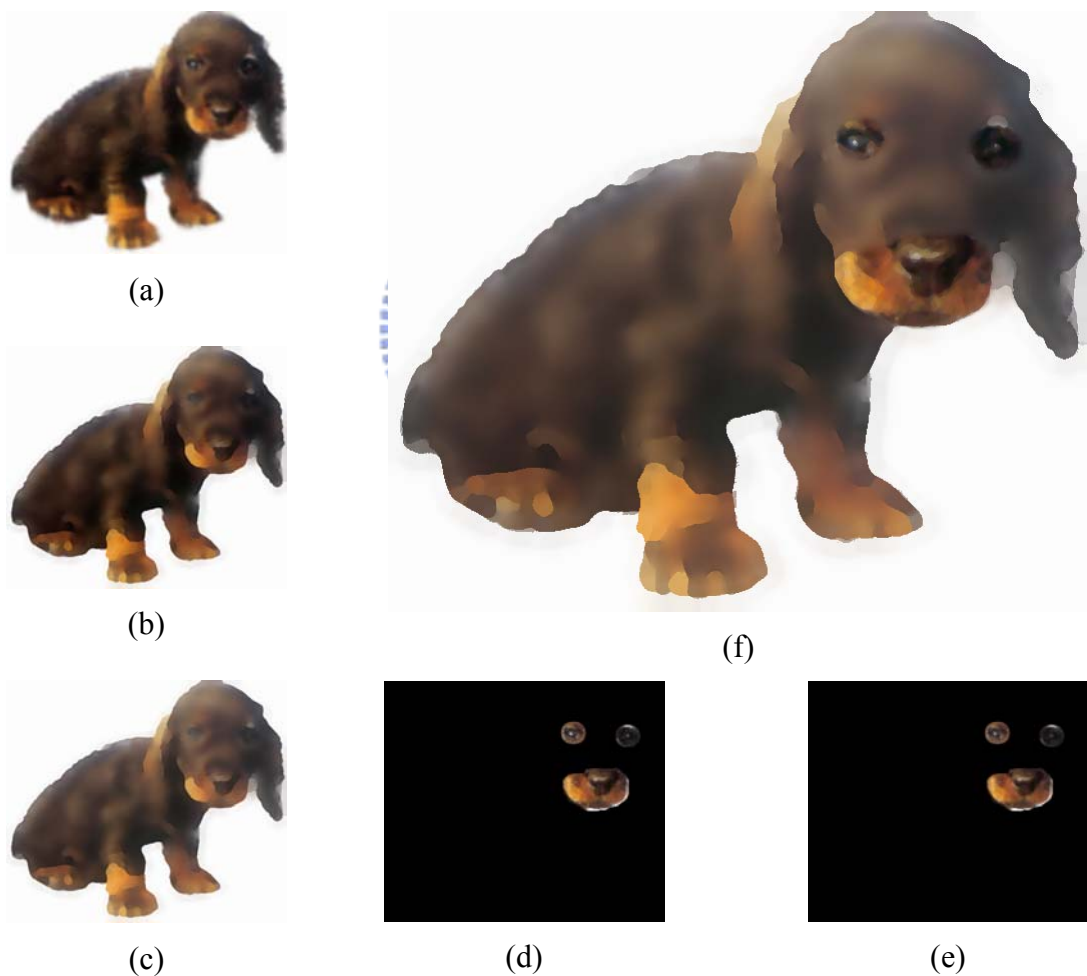


Figure 5.5 Example of interpolation. (a) Wash brush image of Gaussian blur. (b) (a) after mean shift filtering. (c) (b) after Alpha blending with value 0.9. (d) Detail image. (e) (d) after median. (f) Interpolation with (c) and (e).

5.3 Color Wash Effect by Mean Shift Filter

In this section, we discuss how to synthesize the color-wash effect from the interpolated median image by means of image processing. We choose the algorithm, the mean shift filter, described by [8] to achieve this goal. The mean shift filter is based on the mean shift procedure.

5.3.1 The Mean Shift Procedure

The mean shift procedure is a non-parametric technique for the analysis of a complex multimodal feature space. A feature space is a mapping of the input obtained through the processing of the data in small subsets at a time. A typical example is shown in Figure 5.6. The feature space can be regarded as the empirical probability density function (p.d.f.) of the represented parameter. Dense regions in the feature space thus correspond to local maxima of the p.d.f., that is, to the *modes* of the unknown density.

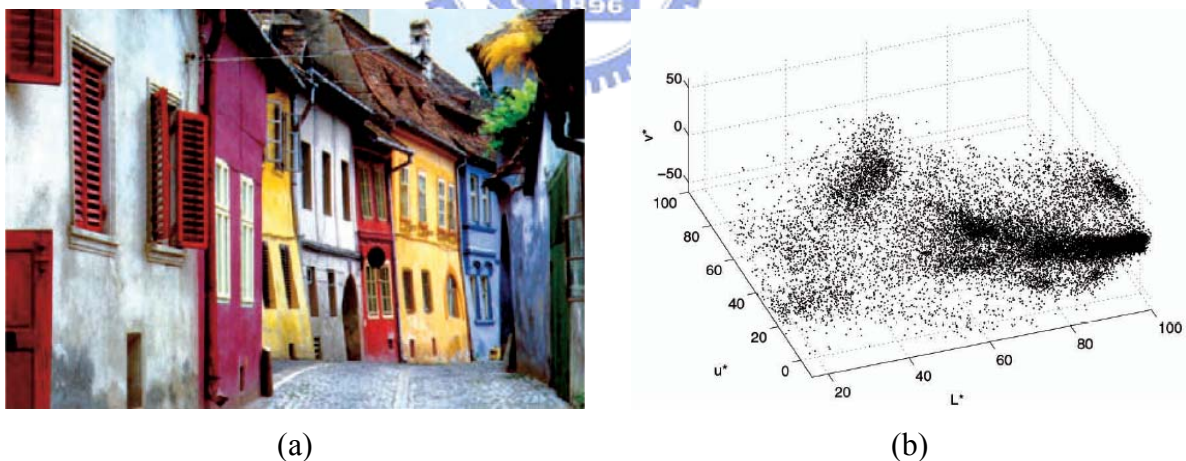


Figure 5.6 Example of a feature space. (a) A 400×276 color image. (b) Corresponding L^*u^*v color space with 110,400 data points.¹

Given n data points \mathbf{x}_i , $i = 1, \dots, n$ in the d -dimensional Euclidean space R^d , the *multivariate kernel density estimator* is

¹ Figure 5.6 to Figure 5.9 are excerpted from [8].

$$\hat{f}(x) = \frac{1}{nh^d} \sum_{i=1}^n K\left(\frac{x-x_i}{h}\right), \quad (1)$$

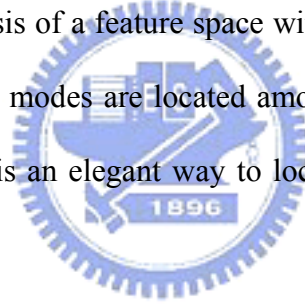
where K is the selected kernel, and h is the window radius or bandwidth.

The quality of a kernel density estimator is measured by the mean of the square error between the density and its estimate, integrated over the domain of definition (MISE). The measure is minimized by the Epanechnikov kernel [21], and its formula is

$$K_E(x) = \begin{cases} \frac{1}{2} c_d^{-1} (d+2) (1-\|x\|^2) & \|x\| \leq 1 \\ 0 & \text{otherwise,} \end{cases} \quad (2)$$

where c_d is the volume of the unit d -dimensional sphere.

The first step in the analysis of a feature space with the underlying density $f(\mathbf{x})$ is to find the modes of this density. The modes are located among the zeros of the gradient $\nabla f(\mathbf{x}) = 0$ and the mean shift procedure is an elegant way to locate these zeros without estimating the density.



The density gradient estimator is obtained as the gradient of the density estimator

$$\hat{\nabla}f(x) \equiv \nabla \hat{f}(x) = \frac{1}{nh^d} \sum_{i=1}^n \nabla K\left(\frac{x-x_i}{h}\right) \quad (3)$$

For the Epanechnikov kernel (2) the density gradient estimator (3) becomes

$$\begin{aligned} \hat{\nabla}f(x) &= \frac{1}{n(h^d c_d)} \frac{d+2}{h^2} \sum_{x_i \in S_h(x)} [x_i - x] \\ &= \frac{n_x}{n(h^d c_d)} \frac{d+2}{h^2} \left(\frac{1}{n_x} \sum_{x_i \in S_h(x)} [x_i - x] \right) \end{aligned} \quad (4)$$

where the region $S_h(\mathbf{x})$ is a hyperspace of radius h having the volume $h^d c_d$, centered on \mathbf{x} , and containing n_x data points. The last term in (4)

$$M_h(x) \equiv \frac{1}{n_x} \sum_{x_i \in S_h(x)} [x_i - x] = \frac{1}{n_x} \sum_{x_i \in S_h(x)} x_i - x \quad (5)$$

is called the sample mean shift.

The quantity $\frac{n_x}{n(h^d c_d)}$ is the kernel density estimator $\hat{f}(x)$ computed with the hyperspace $S_h(\mathbf{x})$ (the uniform kernel), and thus we can rewrite (4) as

$$\begin{aligned} \hat{V}(f) &= \frac{n_x}{n(h^d c_d)} \frac{d+2}{h^2} \left(\frac{1}{n_x} \sum_{x_i \in S_h(x)} [x_i - x] \right) \\ \hat{V}f(x) &= \hat{f}(x) \frac{d+2}{h^2} M_h(x), \end{aligned} \quad (6)$$

which yields

$$M_h(x) = \frac{h^2}{d+2} \frac{\hat{V}f(x)}{\hat{f}(x)} \quad (7)$$

The mean shift vector has the direction of the gradient of the density estimate at \mathbf{x} when this estimator is obtained with the Epanechnikov kernel. Since the mean shift vector always points towards the direction of maximum increase in the density, it can define a path leading to a local density maximum, i.e., to a mode of the density.

The mean shift procedure, obtained by successive

- computation of the mean shift vector $M_h(\mathbf{x})$,
- translation of the kernel (window) $S_h(\mathbf{x})$ by $M_h(\mathbf{x})$.

is guaranteed to converge. The behavior of this feature space analysis technique is illustrated in Figure 5.7. In Figure 5.7(a) the data set representing the first two components of the L^*u^*v space shown in Figure 5.6. In Figure 5.7(c) the peaks retained for the final classification are

marked with red dots.

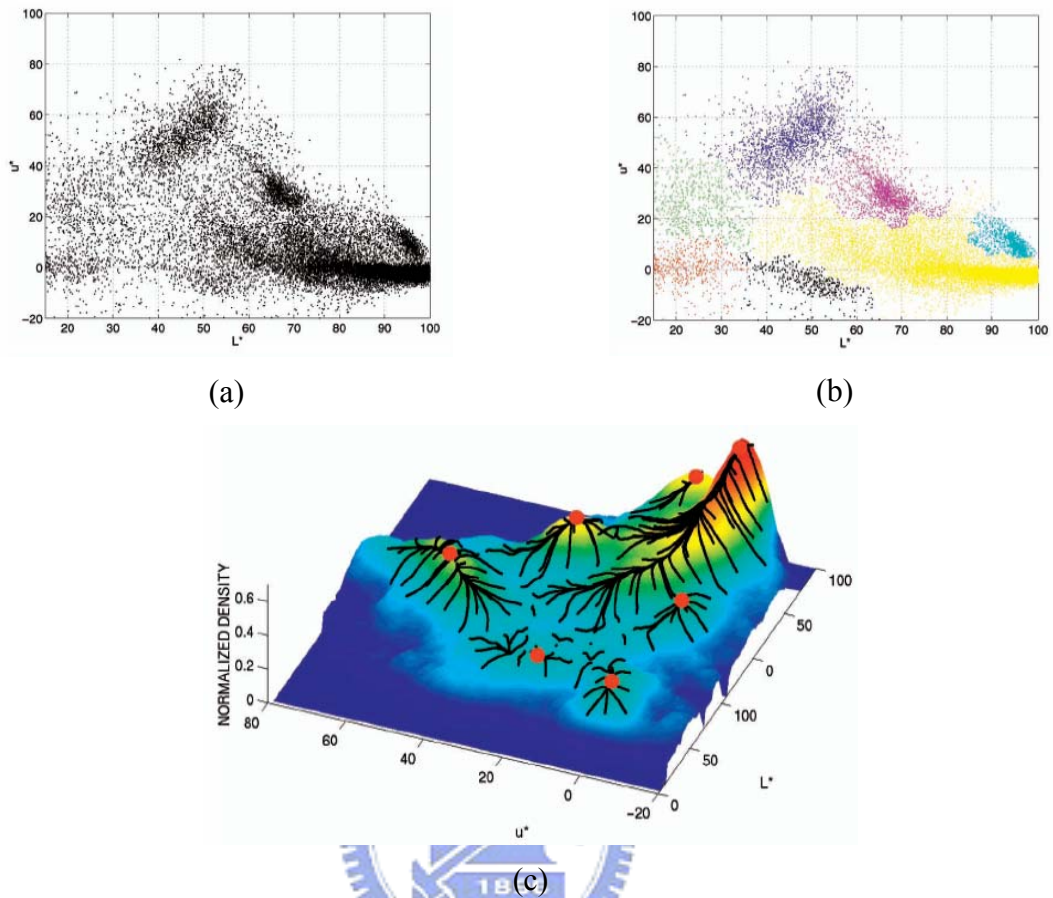


Figure 5.7 Example of a 2D feature space analysis. (a) Two-dimensional data set. (b) Seven clusters. (c) Trajectories of the mean shift procedures.

5.3.2 The Mean Shift Filter

Smoothing through replacing the pixel in the center of a window by the (weighted) average of the pixels in the window indiscriminately blurs the image, removing not only the noise but also salient information. Discontinuity preserving smoothing techniques, on the other hand, adaptively reduce the amount of smoothing near abrupt changes in the local structure, i.e., edges. Mean shift filtering is one of approaches to achieve this goal.

Let \mathbf{x}_j and $\mathbf{z}_j, j = 1, \dots, n$, be the d -dimensional input and filtered image pixels in the joint spatial-range domain.

Mean Shift Filtering

For each pixel,

1. Initialize $k = 1$ and $\mathbf{y}_k = \mathbf{x}_j$.
2. Compute $y_{k+1} = \frac{1}{n_k} \sum_{x_i \in S_1(y_k)} x_i$ until convergence.
3. Assign $\mathbf{z}_j = (x_j^s, y_{conv}^r)$

The superscripts s and r denote the spatial and range components of a vector, respectively. The last assignment specifies that the filtered data at the spatial location x_j^s will have the range component of the point of convergence y_{conv}^r .

Mean shift filtering with uniform kernel having $(h_s, h_r) = (8, 4)$ has been applied to the often used 256×256 gray-level cameraman image as shown in Figure 5.8.

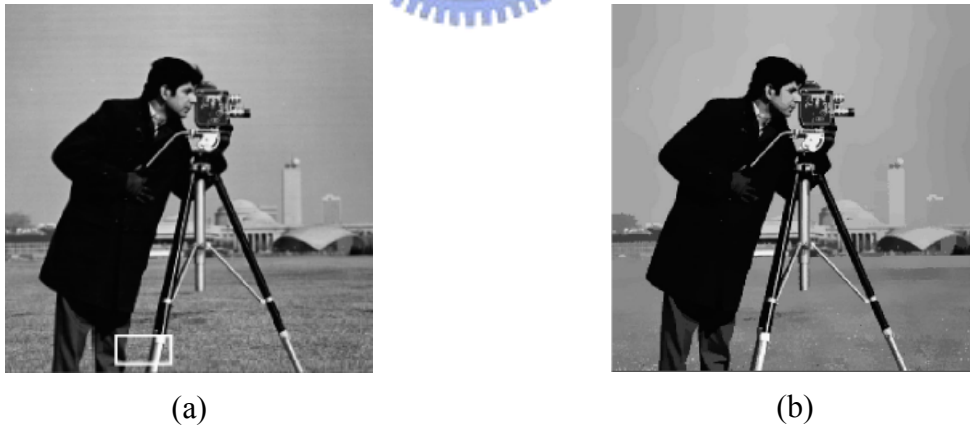


Figure 5.8 Cameraman image. (a) Original. (b) Mean shift filtered $(h_s, h_r) = (8, 4)$.

The regions containing the grass field have been almost completely smoothed while details such as the tripod and the buildings in the background were preserved. This is what we desire for the synthesized wash effect of animal paintings: not only smooth the regions that contain the animal hair, but also keep the features such as the eyes and the mouth in detail.

To illustrate the effectiveness of the filtering process, the 40×20 window marked in Figure 5.8 (a) is represented in three dimensions in Figure 5.9(a). In Figure 5.9(b) the mean shift paths associated with each pixel from the plateau and the line are shown and the black dots are the points of convergence.. Note that the convergence points (black dots) are situated in the center of the plateau, away from the discontinuities delineating it. Similarly, the mean shift trajectories on the line remain on it.

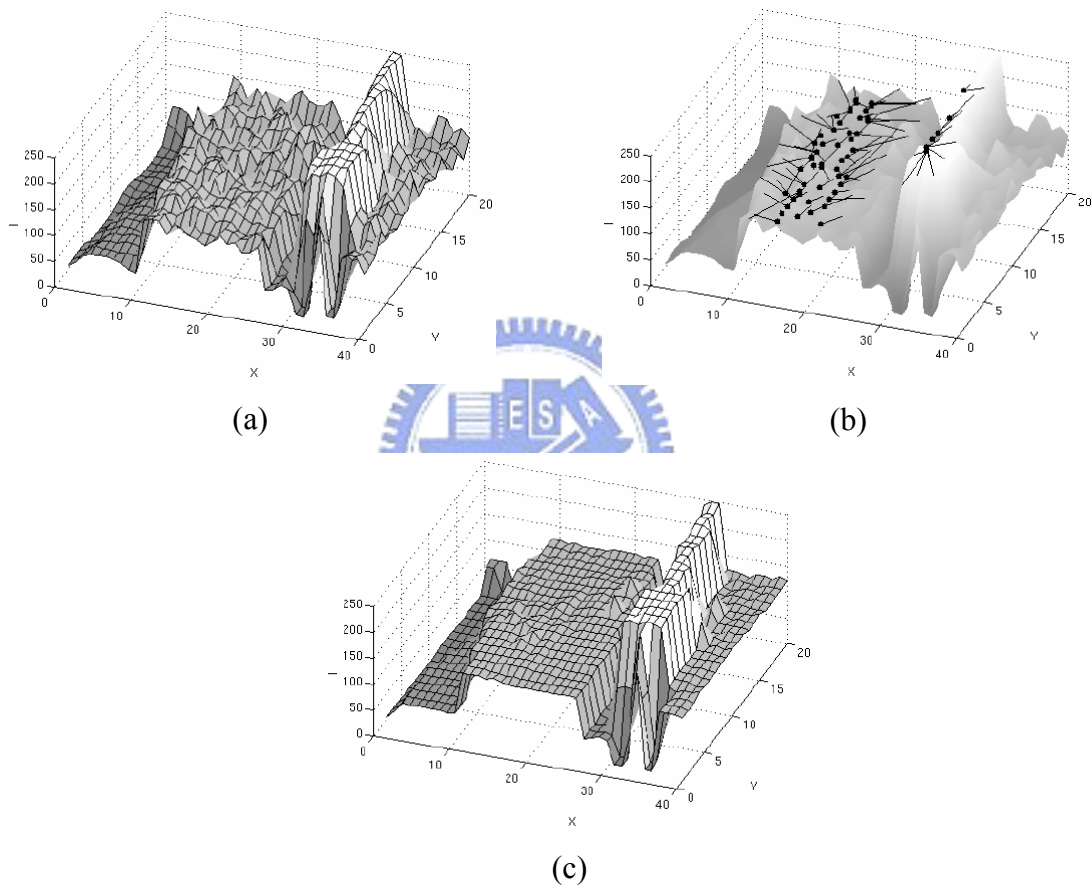


Figure 5.9 Visualization of mean shift-based filtering and segmentation for gray-level data. (a) Input. (b) Mean shift paths for the pixel on the plateau and on the line. (c) Filtering result $(h_s, h_r) = (8, 4)$.

After the Gaussian convolution, we apply mean shift filter to the blurred image. Figure 5.10 shows two examples.



Figure 5.10 Examples of mean shift filtering with $(h_s, h_r) = (10, 15)$.

5.4 Hair Brush Rendering

As wash brush rendering described in Section 5.1, the way to render the hair painting is similar. The difference between hair brush rendering and wash brush rendering is that the stroke textures may be different and the canvas used for hair brush rendering is scaled by the user-defined factor. In order to draw on a canvas of arbitrary size, the process for hair brush rendering takes the scale factor of canvas into account. The anchor position of each stroke and the lengths along both directions are all translated and scaled by the factor.

The artist does not draw the hair brushes over the featured regions so that the featured regions are kept clean and neat. To emulate this, we look over the precision map before drawing a hair brush texture. If this stroke anchor is within the region of interest, our system skips it and goes to the next stroke. If it is outside of the precision map, we render it on the hair brush canvas with a modified color.

In order to represent the gradation of the hair brush such that it won't look the same as the wash painting beneath it, its color that extracted from the reference image is "darkened". To achieve the heuristic that the greater the saturation is, the more the color gets darkened, we first transform the color space from RGB to HSB. Then we reduce the brightness of this color

according to its saturation value by the following equation:

$$\text{Brightness} = \text{Brightness} - \text{Saturation} * \text{factor}S$$

where *factor S* is a constant whose default value is 0.05. Examples of hair brush rendering are shown in Figure 5.11.

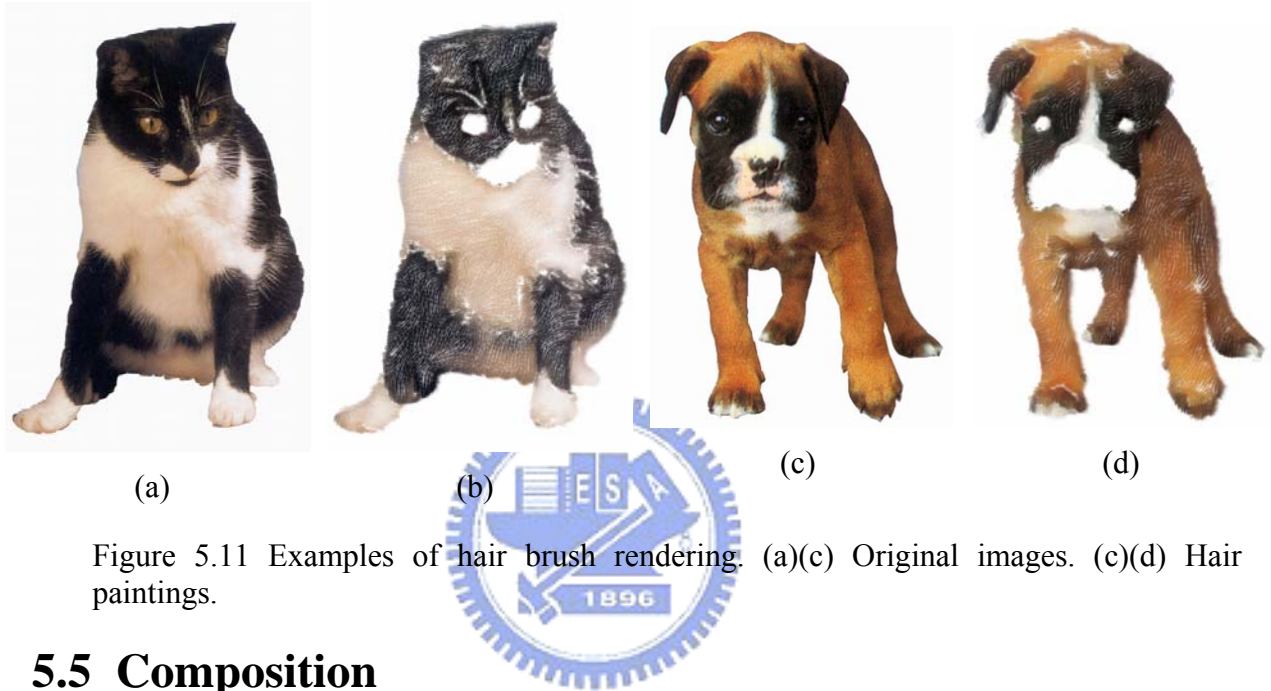


Figure 5.11 Examples of hair brush rendering. (a)(c) Original images. (c)(d) Hair paintings.

5.5 Composition

The last step of our rendering process is the composition of the wash painting and the hair painting. Note that the canvas of wash painting needs to be scaled by the user-defined factor such that it is compatible with the size of the hair painting. The reason not to scale the canvas of wash painting before rendering is to reduce the computation time of mean shift filtering.

Chapter 6

Implementation and Results

In this chapter, the implementation and results are presented. All the figures in this chapter are rendered on the PC platform with a P4 2.80 GHz CPU and 1.50GB RAM by which the presented algorithm is implemented in Java language with package of Java Advanced Imaging (*JAI*).

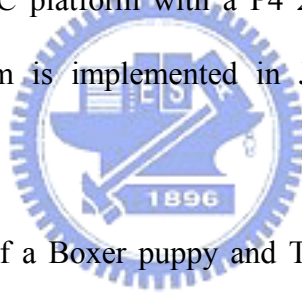


Figure 6.1 is the source of a Boxer puppy and Table 6.1 shows the parameters of this example. Figure 6.2 illustrates the resulting image and layers of strokes. Figure 6.3 is the source of a Dachshund puppy and Table 6.2 is its style parameters. In Figure 6.5, the source is a cat. Its parameters are listed in Table 6.3. The rendered results of example 2 and example 3 are shown in Figure 6.4 and Figure 6.6 respectively. The standard derivation of each layer is interpolated according to its w_b between the user-defined maximum and minimum values. As well the Gaussian width is interpolated by the same way.

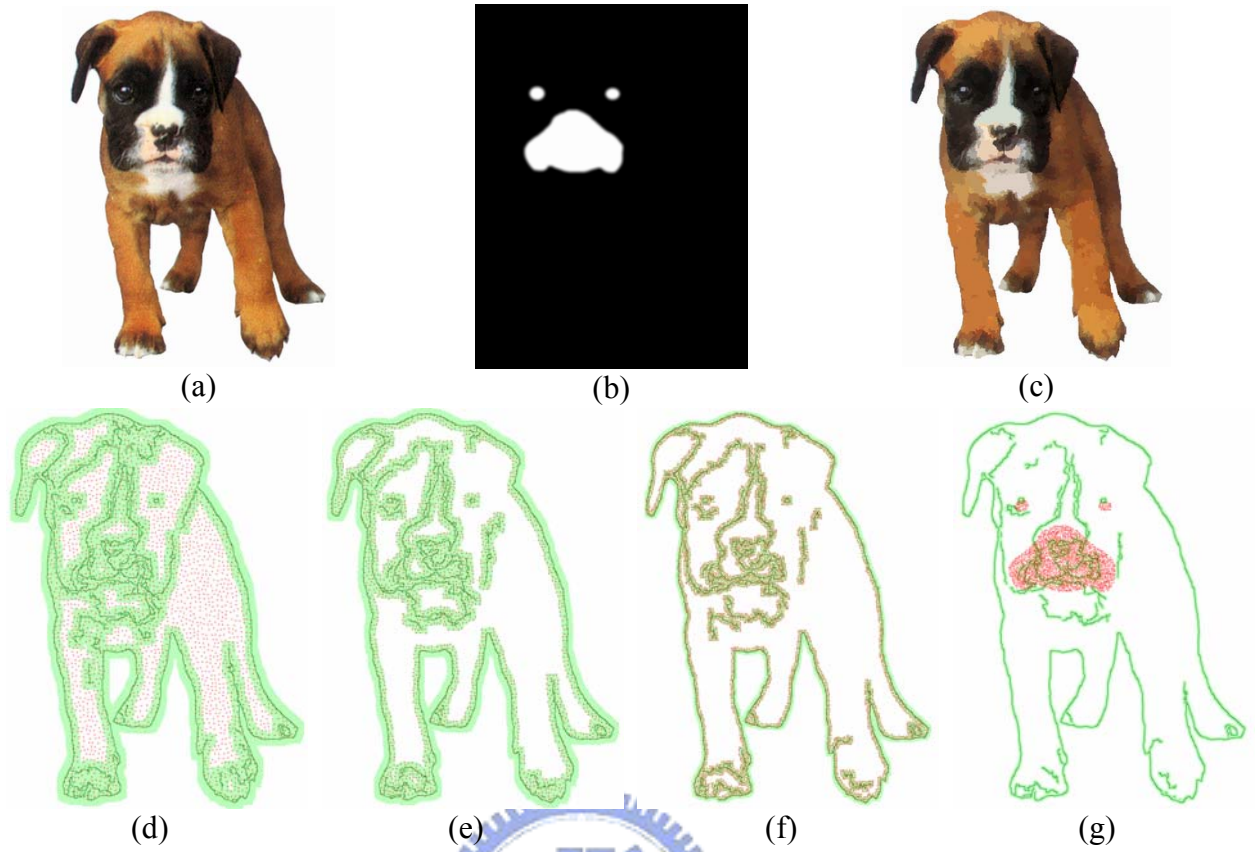


Figure 6.1 Example 1. (a) Reference image. (b) Precision map. (c) Color-fused image. (d)- (g) Edge map and stroke distribution for Layer 0 to Layer 3 respectively.

Original Image Size	600×800	Basis Point Radius	30.0	
Canvas Scale	1.0	Gradient Threshold	0.18	
Layer Number	4	Alpha	0.5	
m_1	2	m_2	5	
Canny T_{low}	1	Canny T_{high}	10	
σS	10	σR	15.0	
	Layer 0	Layer 1	Layer 2	Layer 3
Gaussian σ	1.5	1.6587301	1.7380953	1.8333334
Gaussian Width	11	7	5	3
w_r	12.0	10.0	5.0	3.0
w_b	25.0	15.0	10.0	4.0

Table 6.1 Style parameters of Example 1.



(a)



(b)



(c)



(f)



(d)



(e)

Figure 6.2 Resulting images of example 1. (a)- (d) Hair strokes for Layer 0 to Layer 3 respectively. (e) Wash painting. (f) Result.

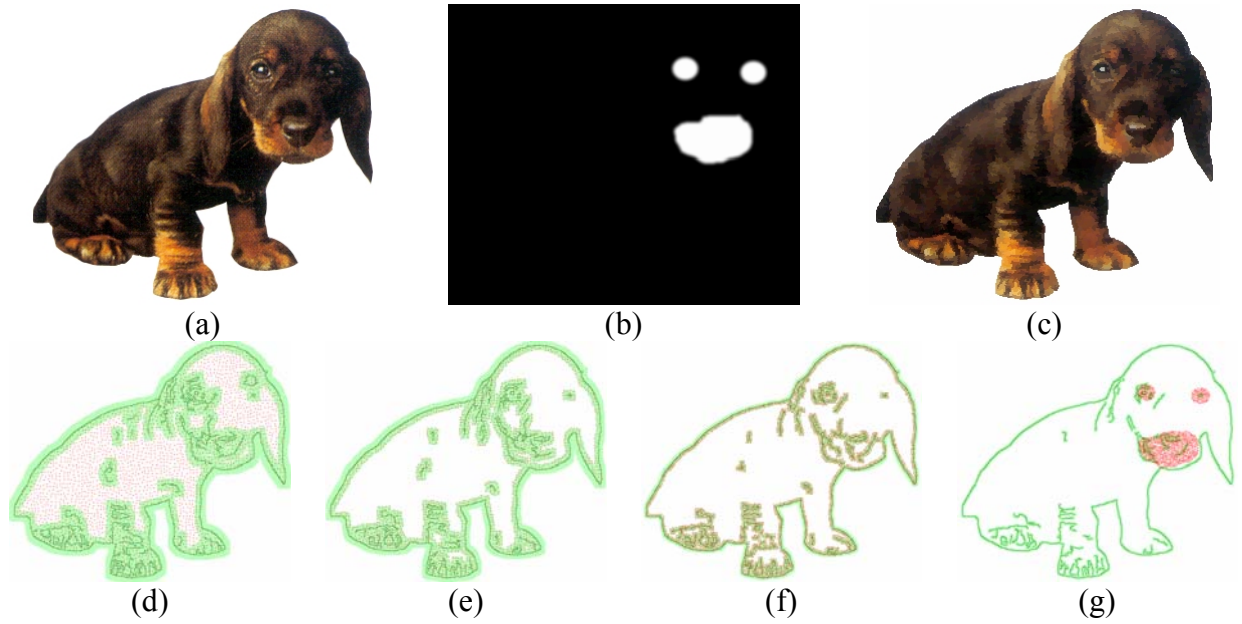


Figure 6.3 Example 2. (a) Reference image. (b) Precision map. (c) Color-fused image. (d)-(e) Edge map and stroke distribution for Layer 0 to Layer 3 respectively.

Original Image Size	700×600	Basis Point Radius	50	
Canvas Scale	1.0	Gradient Threshold	0.2	
Layer Number	4	Alpha	0.5	
m_1	2	m_2	5	
Canny T_{low}	1	Canny T_{high}	10	
σS	10	σR	15.0	
	Layer 0	Layer 1	Layer 2	Layer 3
Gaussian σ	1.5	1.6587301	1.7380953	1.8333334
Gaussian Width	11	7	5	3
w_r	12.0	10.0	5.0	3.0
w_b	25.0	15.0	10.0	4.0

Table 6.2 Style parameters of Example 2.



(f)



(a)



(b)



(c)



(d)



(e)

Figure 6.4 Resulting images of example 2. (a)- (d) Hair strokes for Layer 0 to Layer 3 respectively. (e) Wash painting. (f) Result.

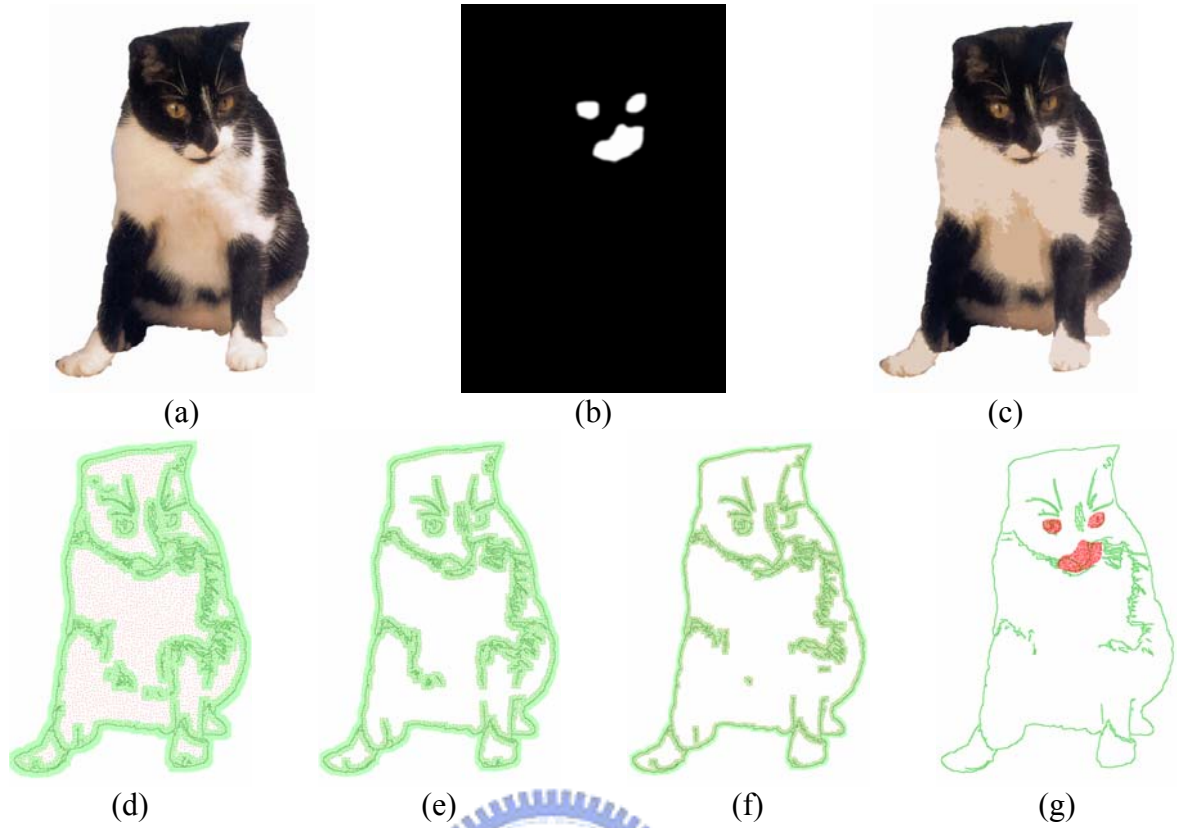


Figure 6.5 Example 3. (a) Reference image. (b) Precision map. (c) Color-fused image. (d)- (e) Edge map and stroke distribution for Layer 0 to Layer 3 respectively.

Original Image Size	760×1120	Basis Point Radius	30.0	
Canvas Scale	1.0	Gradient Threshold	0.15	
Layer Number	4	Alpha	0.8	
m_1	2	m_2	5	
Canny T_{low}	1	Canny T_{high}	10	
$sigmaS$	7	$sigmaR$	15.0	
	Layer 0	Layer 1	Layer 2	Layer 3
Gaussian σ	1.5	1.6515151	1.7272727	1.8333334
Gaussian Width	11	7	5	3
w_r	15.0	12.0	8.0	2.0
w_b	25.0	15.0	10.0	3.0

Table 6.3 Style parameters of example 3.

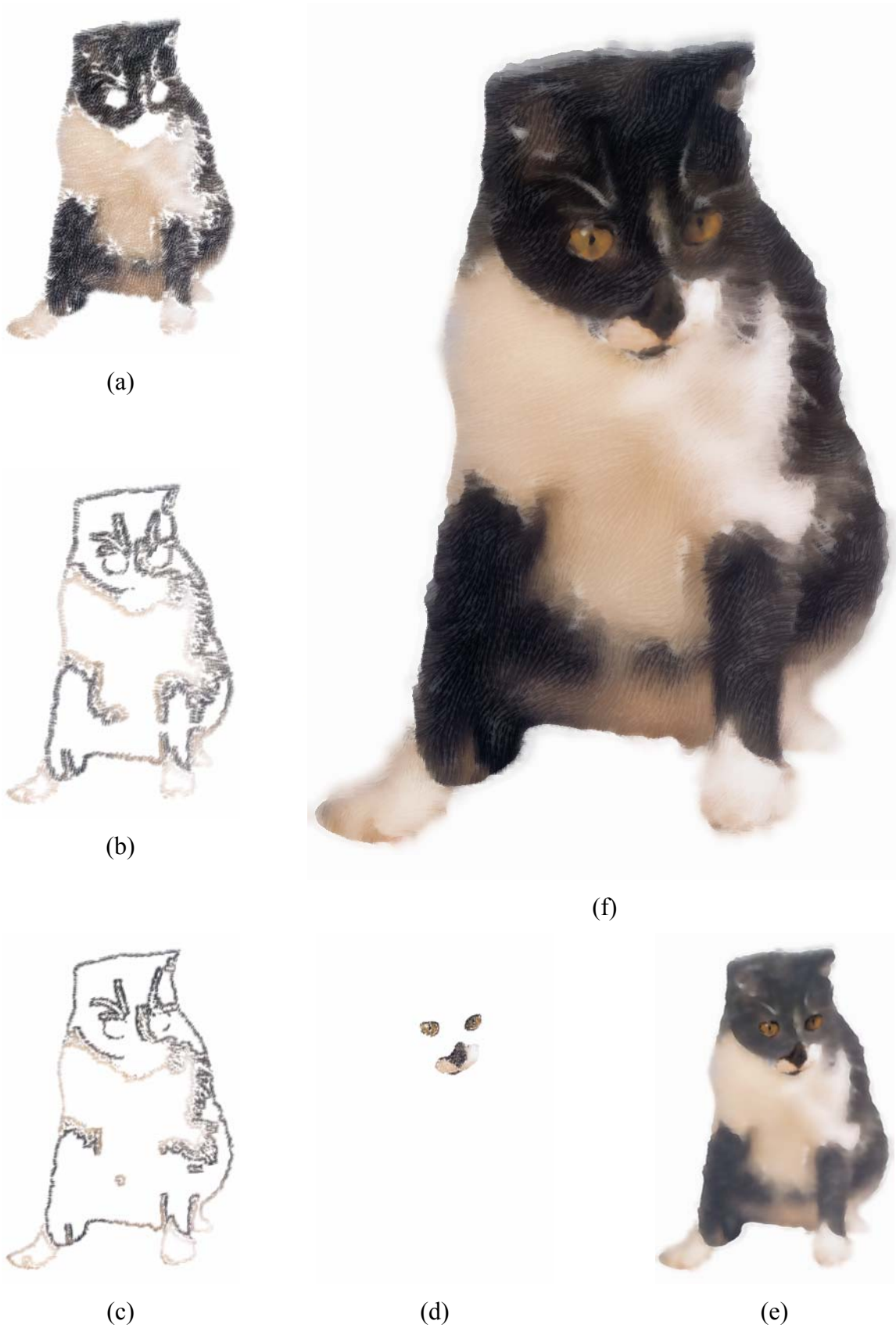


Figure 6.6 Resulting images of example 3. (a)- (d) Hair strokes for Layer 0 to Layer 3 respectively. (e) Wash painting. (f) Result.

Chapter 7

Conclusions and Future Works

In this thesis, we propose a method to synthesis Chinese fine-brushwork painting on animals. The rendered results could be in different styles by various types of strokes and parameters. The whole process is semi-automatic. Users just specify the style parameters and define the featured regions over a given photograph. The rest of work is done by the computer. Moreover, the algorithm is relied on imagery analysis. The central element in our work is a brush stroke with properties. While rendering, these strokes are used by putting textures layer-by-layer and based on image processing to generate wash and hair paintings. The major contributions of our approach are as follows:

1. Radial basis functions are used to interpolate globally brush stroke orientation. It makes coherence of outline of the subject.
2. Painterly refinement is guided by edges. Although colors in our renders may differ from the reference image, yet the images are still comprehensible because the paintings are structurally refined.
3. Use real brush stroke textures and thus the result is similar to the painting by Chinese brush.

4. Treat strokes in two different usages to synthesis wet wash and dry brush effects, the particular style of Chinese painting.
5. Our system is general and easy to use to a beginner. It takes a lot of time to accomplish a painting in reality and also it is not easy for everyone to become a talent painter. With our aid, one animal drawing in Chinese fine-brushwork can be generated with effortless.

However, there are still some issues left to enhance our system in the future.

1. In this thesis, we focus on “Hair Painted with a Flattened Brush” (破鋒絲毛法) . Although it works well on many kinds of animals such as dogs and cats, there are still other painting skills could be developed such as “Linear Style with Dots”(戳點畫毛法), “Hair Painted with Broken Color”(以墨破色法) and “Painting with Ink and Colors”(點染畫毛法) which are introduced in [1]. As well the painting skills on animal face features could be coupled with. If we make our way, the complexity of various paintings could be improved enormously.
2. Our system can be extended to generate animation by adding abilities to handle object extracting and tracking. We only need to distinguish the animal from the varied background. Furthermore the frame coherence is an important issue as well.
3. Our method handles textures by simple translation and rotation. It is more general to treat textures by texture warping with curved brush strokes.

Reference

- [1] 施伯雲繪著，《貓·虎畫法》藝術圖書公司，1990 初版。
- [2] 張克齊繪著，《鳥語花香—張克齊工筆畫集》，藝術圖書公司，1995 年初版。
- [3] 劉奎齡繪著，《飛禽走獸花鳥畫集》，藝術圖書公司，1988 年再版。
- [4] 勝大莊美術館，Fine Chinese Paintings and Calligraphy，1991 年出版。
- [5] Adrain G. Bors. 2001. “Introduction of the radial basis function (rbf) networks.” *Online Symposium for Electronics Engineers, vol. 1 of DSP Algorithms: Multimedia*.
- [6] John Canny. “A Computational Approach to Edge Detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 8, No. 6, Nov. 1986.
- [7] Cassidy J. Curtis, Sean E. Anderson, Joshua E. Seims, Kurt W. Fleischer, and David H. Salesin. 1997. “Computer-generated watercolor.” *In Proceedings of SIGGRAPH 97, Computer Graphics Proceedings*, pp.421-430.
- [8] Dorin Comaniciu and Peter Meer. 2002. “Mean shift: A robust approach toward feature space analysis,” *IEEE Trans. on Pattern Analysis and Machine Intelligence* 24, 5.
- [9] Frédo Durand, Victor Ostromoukhov, Mathieu Miller, François Duranleau, and Julie Dorsey. “Decoupling strokes and high-level attributes for interactive traditional drawing.” *Eurographics Workshop on Rendering 2001*, pages 71–82, June 2001.

- [10] Rafael C. Gonzalez and Richard E. Woods. 2002. "Digital Image Processing." *Prentice-Hall*.
- [11] Paul Haeberli. 1990. "Paint by Numbers: Abstract image representations." *Computer Graphics*, 24(4): 207-214, August 1990.
- [12] Aaron Hertzmann. 1998. "Painterly rendering with curved brush strokes of multiple sizes." *In Proceedings of SIGGRAPH 98*, Computer Graphics Proceedings, pp.453-460.
- [13] Aaron Hertzmann and Ken Perlin. 2000. "Painterly rendering for video and interaction," *NPAR 2000: First International Symposium on Non Photorealistic Animation and Rendering*, pp.7-12.
- [14] Peter Litwinowicz. 1997. "Processing images and video for an impressionist effect." *In Proceedings of SIGGRAPH 97*, Computer Graphics Proceedings, pp. 407-414.
- [15] Sheng-Wen Huang. 2002. "Physical-based model of ink diffusion in Chinese ink paintings." *Master thesis*, National Chiao Tung University.
- [16] Thomas Porter and Tom Duff. "Compositing Digital Images", *In Proceedings of SIGGRAPH 84*, pp.253-259.
- [17] M. J. D Powell. 1987. "Radial Basis Functions." *In J.C.Mason, M.G.Cox (eds.): Algorithms for Approximation*. Oxford University Press, New York.
- [18] Michael P. Salisbury, Sean E. Anderson, Ronen Barzel, and David H. Salesin. 1994. "Interactive pen-and-ink illustration." *In Proceedings of SIGGRAPH 94*, Computer Graphics Proceedings, pp.101-108.
- [19] Michael P. Salisbury, Michael T. Wang, John F. Hughes and David H. Salesin. 1997.

- “Orientable textures for image-based pen-and-ink illustration.” *In Proceedings of SIGGRAPH 97*, Computer Graphics Proceedings, pp.401-406.
- [20] Mike Salisbury, Corin Anderson, Dani Lischinski and David H. Salesin. 1996. “Scale-dependent reproduction of pen-and-ink illustrations.” *In Proceedings of SIGGRAPH 96*, Computer Graphics Proceedings, pp.461-468.
- [21] Robert Sedgewick. “Algorithms in C,” *Addison-Wesley*, 1990.
- [22] B. W. Silverman. “Density Estimation for Statistics and Data Analysis.” *New York: Chapman and Hall*, 1986
- [23] Der-Lor Way and Zen-Chung Shih. 2001. “The synthesis of rock textures in Chinese landscape painting.” *In Proceedings of Eurographics 01*, pp.C123-C131.
- [24] Der-Lor Way, Yu-Ru Lin and Zen-Chung Shih. “The synthesis of trees in Chinese landscape painting using silhouette and texture strokes.” *Journal of WSCG* Volume 10, Number 2, pp. 449-506, 2002.
- [25] Shan-Zan Weng, Zen-Chung Shih and Hsin-Yi Chiu. 1999. “The synthesis of Chinese ink painting.” *National Computing Symposium 99*, pp.461-468.
- [26] Georges Winkenbach and David H. Salesin. 1994. “Computer-generated pen- and-ink illustration.” *In Proceedings of SIGGRAPH 94*, Computer Graphics Proceedings pp.91-100.
- [27] Georges Winkenbach and David H. Salesin. 1996. “Rendering Parametric Surfaces in Pen and Ink.” *In Proceeding of SIGGRAPH 96*, Computer Graphics Proceedings pp.469-476.