BOURNEMOUTH UNIVERSITY

DOCTORAL THESIS

# From Brush Painting to High Relief


*Supervisor:*
Dr. Hongchuan YU

*Author:*
Yunfei FU

*Co-supervisors:*
Prof. Jian Jun ZHANG
Prof. Tong-Yee LEE
Prof. Zan GAO



*A thesis submitted in partial fulfilment of the requirements of Bournemouth University for the degree of Doctor of Philosophy*


National Centre for Computer Animation


Wednesday 14th August, 2019

# Copyright Statement

This copy of the thesis has been supplied on condition that anyone who consults it is understood to recognise that its copyright rests with its author and due acknowledgement must always be made of the use of any material contained in, or derived from, this thesis.

# Abstract

As an artistic form, relief is described as a hybrid between 2D painting and 3D sculpture. A novel approach for generating a texture-mapped high relief model from a single brush painting is presented in this work. The aim of this work is to extract the brush strokes from a painting and generate the individual corresponding relief proxies rather than recovering the exact depth map from the painting, which is a tricky computer vision problem, requiring assumptions that are rarely satisfied. The relief proxies of brush strokes are then combined together to form a 2.5D high-relief model. To extract brush strokes from 2D paintings, this work applies layer decomposition and stroke segmentation by imposing boundary constraints. The segmented brush strokes preserve the style of the input painting. By inflation and a displacement map of each brush stroke, the features of brush strokes are preserved by the resultant high relief model of the painting. As the adjacent brush strokes may share similar colours in some brush paintings, the layer decomposition method does not work well. To amend this issue, this work also proposed a deep learning based method for brush stroke extraction. This work demonstrates that it is able to produce convincing high-reliefs from a variety of paintings (with humans, animals, flowers, etc.). As a secondary application, this work shows how the proposed brush stroke extraction algorithm could be used for image editing. As a result, the proposed brush stroke extraction algorithm is specifically geared towards paintings with each brush stroke drawn very purposefully, such as Chinese paintings, Rosemailing paintings, etc.

# LIST OF BASIC TERMINOLOGY

**There are other terms to consider, but many of the important ones are explained in the report where relevant. These are terms not explicitly explained elsewhere.**

**Height/Depth map**: In this research, a depth map is a image that represents the height of a surface.

**Opaque**: Not able to be seen through; not transparent.

**Opacity**: The quality of lacking transparency or translucence.

**Alpha map**: The alpha channel of an RGBA image. Alpha indicates how opaque each pixel is(from fully transparent to fully opaque).

**Intensity**: Intensity represents brightness of the greyscale images. In most applications, measured between 0 (dark) and 255 (bright). For colour images, intensity is calculated after convert the colour images to greyscale.

**Stroke**: A mark made by drawing a pen, pencil, or paintbrush across paper or canvas.

**Brush stroke/Brushstroke**: A mark made by drawing paintbrush across paper or canvas.

# Table of contents

**List of figures** **vii**

**List of tables** **xi**

**1 Introduction** **1**
- 1.1 Relief Generation . . . 2
- 1.2 Motivation . . . 4
- 1.3 Aim and Objectives . . . 7
- 1.4 Contribution . . . 8
- 1.5 Thesis Outline . . . 10
- 1.6 Publication . . . 11

**2 Related Work** **12**
- 2.1 2D Image Based Relief Generation . . . 13
  - 2.1.1 Photograph Based Methods . . . 13
  - 2.1.2 2D Sketch Based Methods . . . 16
- 2.2 3D Model Based Relief Generation . . . 17
- 2.3 Brush Stroke Extraction . . . 18
  - 2.3.1 Painting Layer Decomposition . . . 19
  - 2.3.2 Soft Segmentation . . . 20
  - 2.3.3 Stroke Segmentation . . . 21
- 2.4 Deep Learning Based Segmentation . . . 22
- 2.5 Summary . . . 23

**3 Overview of Proposed Approach** **24**

**4 Layer Decomposition Based Brush Stroke Extraction—LStroke** **28**
- 4.1 Layer Decomposition . . . 29

4.1.1 Palette colours Extraction . . . . . 29
4.1.2 Determining Layer Opacity . . . . . 30
4.1.3 Modified Layer Decomposition . . . . . 31
4.1.4 Iterative Scheme . . . . . 35
4.1.5 Brush Stroke Completion . . . . . 38
4.2 Extraction of Brush Strokes . . . . . 39
4.2.1 MSERs Algorithm . . . . . 41
4.2.2 Modified MSERs Algorithm . . . . . 42
4.3 Result and Analysis . . . . . 44
4.3.1 Compare LStroke with [42]'s method . . . . . 46
4.4 Summary . . . . . 47

**5 Deep Network Based Brush Stroke Soft Segmentation—DStroke 49**
5.1 Training Dataset Generation . . . . . 52
5.2 Network Structure . . . . . 57
5.3 Soft Segmentation . . . . . 60
5.4 Result and Analysis . . . . . 62
5.4.1 Compare DStroke with LStroke . . . . . 63
5.4.2 Compare DStroke with [3]'s method . . . . . 65
5.5 Summary . . . . . 70

**6 High Relief Generation 72**
6.1 Displacement Map . . . . . 73
6.2 Model Generation . . . . . 75
6.3 Results and Analysis . . . . . 77
6.3.1 High relief generation by DStroke and LStroke . . . . . 78
6.3.2 Compare high relief generation with [81]'s method . . . 79
6.3.3 Limitations . . . . . 85
6.4 Summary . . . . . 85

**7 Other Applications 87**
7.1 Recoloring Paintings . . . . . 87
7.2 Inserting Objects . . . . . 88
7.3 Animating Strokes . . . . . 90

**8 Conclusions and Future Work 92**
8.1 Conclusion . . . . . 92

8.2 Future Work . . . . . 94

**References** **96**

# List of figures


1.1 High relief and Bas relief . . . . . 1
1.2 Features of brush strokes. . . . . 6

2.1 The structure of related works. . . . . 12
2.2 Sketch based modeling in [51]. . . . . 16
2.3 Bas relief generation from line drawing by Sykora et al. [68]. (a) Input drawing including user annotations. (b) Layering the segmented regions. (c) Resulting bas-relief-type mesh. . . . . 17
2.4 Relief generation from 3D model by Weyrich et al. [74]. . . . . 18
2.5 Image matting by Levin et al. [39]. (a) Input image. (b) Trimap. (c) Alpha map of foreground. . . . . 20
2.6 Soft segmented image by Aksoy et al.[3]. The soft segmented regions are represented in different colours. . . . . 22

3.1 Overview of proposed approach . . . . . 24

4.1 Layer decomposition in [70]. The geometry of pixels in original image(left) is analysed in RGB-space to extract its palette colours(middle), resulting in a translucent layer decomposition(right). 29
4.2 Simplification of convex hull in RGB space. . . . . 30
4.3 Edge Tangent Flow field and coherent lines of a Rosemaling painting. It contains lots of C and S strokes. . . . . 32
4.4 Comparison of layer decomposition by Eq. 4.3 and Eq. 4.9 at the 2nd layers. (a) and (b) show the results by using $E_{spatial}$ and $E_{flow}$ in Eq. 4.3; (c) and (d) show the results before and after using $E_{edge}$ in Eq. 4.9. . . . . 32

4.5 Comparison of before and after adding a new layer. The upper row shows the decomposed layers, while the below row shows those layers after adding a new layer. Each column shows the decomposed layers with a corresponding palette colour. The last column of the below row shows the new layer. . . . . 37
4.6 The masks of the shared and isolated regions when there are 4 and 5 layers respectively. (a) Shared regions (4 layers). (b) Shared regions (5 layers). (c)Isolated regions (4 layers). (d)Isolated regions (5 layers). . . . . 38
4.7 Illustration of in-painting on a layer. (a) User annotated mask (blue region) on the in-painting area. (b) In-painted layer. . . . 38
4.8 Comparison of the intensity (a) and opacity (b) of a layer with histograms (c)(d). . . . . 40
4.9 (a) Original image. (b) Intensity of image. (c) Original MSERs on the intensity of image. . . . . 41
4.10 Comparison of segmentation results by the original MSERs and modified version. (a) The decomposed layers. (b) The original MSERs on the intensity of layers. (c) Opacity of layers, and MSER regions on four layers by the modified MSERs. . . . . 42
4.11 (a) Input images (Painting ID from top to down: F374, F607, F411). (b) Brush strokes extracted by [42]. (c) Brush strokes extracted by **LStroke**. . . . . 44
4.12 Brush strokes extraction in [42] (Painting ID: F518) . . . . . 46
4.13 Brush strokes extraction by **LStroke** (Painting ID: F518); the upper row shows the opacity maps of each layer . . . . . 46
4.14 Manually marked brush strokes by [42]. . . . . 47

5.1 Paintings with highly mixed brush strokes in similar colours (upper row); Extracted brush strokes by **LStroke**(Chapter 4) (second row). Extracted brush strokes by **DStroke** (bottom row). . . . . 49
5.2 Overview of the Semantic Soft Segmentation[3]. . . . . 50
5.3 (a) Scene image. (b) Iterative stroke painting on canvas (upper row); height map of canvas (bottom row). (c) Illumination of canvas. . . . . 52
5.4 (a) Scene image. (b) Iterative stroke painting on canvas (upper row); edges of brush strokes (bottom row). . . . . 55

5.5 Input image and its augmentations. (a) Input image. (b) Adding noises. (c) Distortion. (d) Changing hue. (e) Flipping. . . . . 55
5.6 Different alpha maps of brush strokes from *kylebrush* [38]. . . . 56
5.7 The Pix2Pix network . . . . . 58
5.8 An example of blurred edges and gaps within stroke . . . . . 60
5.9 An example of soft segmentation. (a) Input image. (b) Hard segmentation. (c) A segmented brush stroke mask. (d) Opacity of the brush stroke. (e) Soft segmentation. . . . . 61
5.10 Highly mixed brush strokes in similar colours. (a) Input image. (b) Brush stroke extraction based on **LStroke** (Chapter 4). (c) Brush stroke extraction based on **DStroke**. . . . . 63
5.11 (a) Input image. (b) The result of [3]'s method. (c) The result of **DStroke**. . . . . 65
5.12 Compare the alpha map of a segmented stroke with groundtruth. (a) Input image. (b) Alpha map of a segmented stroke. (c) Alpha map of the groundtruth. (d) Difference map between prediction and the groundtruth. . . . . 66
5.13 Comparison of Running time in different image sizes . . . . . 67
5.14 Comparison of MSE in different image sizes . . . . . 67
5.14 (a) Input image. (b) Detected brush stroke edges. (c) Hard segmentation from **DStroke**. (d) The results of soft segmentation from **DStroke**. (e) Manual segmentation. . . . . 70

6.1 Displacement map generation from brush strokes. (a) Opacity of one layer. (b) Extracted three brush strokes. (c) Generated displacement map from three brush strokes. . . . . 73
6.2 (a) Inflation with markup cues (b) Smooth high relief surface. (c) High relief with the displacement map. (d) Texture-mapped high relief (rotated 30 degree from the original viewing direction). . 75
6.3 The effect of changing the layer number. (a) High relief generated from 4 layers. (b) High relief generated from 6 layers. (c) High relief generated from 9 layers. . . . . 78
6.4 (a) Input painting. (b) Segmentation by **DStroke**. (c) Segmentation by **LStroke**. (d) Relief generation by **DStroke**. (e) Relief generation by **LStroke**. . . . . 79

6.5 (a) Input images. (b) Brush strokes segmented by method in [81]. (c) The last three columns show brush strokes extracted by **LStroke** on each decomposed layers. . . . . 80
6.6 (a) Input images. (b) High reliefs from [81]'s method. (c) High reliefs generated by the this work. (d,e) Texture-mapped high reliefs generated by this work. . . . . 81
6.7 Extraction of overlapped brush strokes. (a) Input image. (b) The brush stroke segmentation by Yeh et al.[81]'s method ( blue rectangles indicate the overlapped regions of brush strokes). (c) The extracted brush strokes on one decomposed layer by **LStroke**, in which the overlapped regions are extracted. . . . . 82
6.8 (a) Input image. (b) Texture-mapped high relief by method in [81]. (c) Texture-mapped high relief by this work. . . . . 83
6.9 (a) Input paintings. (b) Textured mapped high relief results (rotated 45 degree to the left from the viewing direction). (c) Textured mapped high relief results (rotated 45 degree to the right from the viewing direction). . . . . 84
6.10 (a) A region from a van Gogh's painting (F779, Wheatfield with Crows). (b) High relief model. (c) Texture-mapped high relief. . 85

7.1 Column (a) shows 3 kinds of brush paintings, Rosemailing (Europe), van Gogh oil painting, and Chinese painting (East Asia). Column (b) shows recoloring one stroke and Column (c) shows recoloring multiple strokes. Column (d) shows inserting objects into these 3 paintings, in which the objects are opaque and are inserted in between brush strokes. . . . . 88
7.2 Insert object between brush strokes (Painting ID:F779) . . . . . 89
7.3 Animation by rotating strokes. . . . . 90

# List of tables

4.1 Opacity RMSE of Coherent Lines on Layers . . . . . 36

5.1 Evaluation and Comparison based on van Gogh's paintings . . . 64
5.2 Evaluation and Comparison of **DStroke** . . . . . 65

6.1 Running Time Performance . . . . . 83

# Acknowledgements

First of all, my sincere appreciation and thanks goes to my supervisors Dr. Hongchuan Yu and Prof. Jianjun Zhang for their priceless advices on my research. Their constantly pursue of the most advanced technology motivates me to work harder on my research at all stages.

Gratitude also goes to Sunny Choi, Jan Lewis and other colleagues at Bournemouth University and National Centre for Computer Animation for their help on all the works of my campus life.

I would like to express my special gratitude to Prof. Tony Lee and Chih-Kuo Yeh from the National Cheng Kung University for their invaluable guidance, help and our sleepless nights of work.

I would also like to thank all my labmates for the sharing of knowledge, collision of ideas, and our discussion on technologies, these makes the biggest support for my life here.

Last but not the least, I would like to thank my parents for their support, your encouragements has made me regain my confidence and power for so many times.

# Declaration

This thesis has been created by myself and has not been submitted in any previous application for any degree. The work in this thesis has been undertaken by myself except where otherwise stated.

Yunfei Fu
August 2019

# Chapter 1

# Introduction

(a) High relief (British Museum 2009a).

(b) Bas relief (British Museum 2009a).

Fig. 1.1 High relief and Bas relief

**As an artistic form, relief is described as a hybrid between 2D painting and 3D sculpture, as claimed by many previous works [7][74][33][34][83]. In essence, it creates a bridge between a full 3D sculpture and a 2D painting.** Relief is a kind of engraving art, carved on a plane or surface, unlike ordinary independent sculptures. In most cases, reliefs can be divided into two more specific categories: high relief and bas relief (Fig. 1.1). High relief shows much more depth information compared to bas relief. Relief refers to sculptural elements on flat surfaces, such as the trim on the Parthenon. Bas relief refers to the reliefs with shallow backgrounds that are a few inches deep at most. On the other hand, high relief forms more than half of the object, which can be up to a few meters from the background plane. For high relief, some parts of the relief could be completely separated from the background, while the bas relief is not separated

from the background. On this spectrum, high relief is closest to full 3D, whereas flatter artworks are described as bas-relief. Relief sculpting has been practiced for thousands of years. Since antiquity, reliefs have been created by artisans from many ancient cultures.

Nowadays, relief is useful for supporting an enormous number of applications such as making objects like commemorative medals, souvenirs, packaging and artistic sculptures. With the commonly and cheaply available 3D printing facilities, there is a growing trend for the need of relief art products. Traditionally, relief creation by hand is a time-consuming and laborious work, which also requires significant planning, professional skills, and a lot of effort in understanding human visual perception. More recently, attention has been given to computer-aided relief generation, which has been a significant topic in many research areas. Computer-aided relief generation also has a wide variety of purposes, including animation, stereoscopy, cartoon figures and rotating lenticular posters.

## 1.1 Relief Generation

Relief generation has been under research over the past two decades, it aims to generate relief model from a 3D or 2D input, either directly or with user indications. It is considered to be a more artist-friendly than natural approaches of relief sculpture.

As mentioned above, relief is described as an art form part way between 2D painting and full 3D sculpture [7][74][33][34][83]. Correspondingly, the existing relief generation researches can also be classified in two different categories with respect to their input [7]:

- 3D model based : using a 3D model (sculpture) as input
- 2D image based : using a 2D image as input

**How to generate a relief from a 3D sculpture?** The 3D model based relief generation methods focus on such a problem, in which a 3D sculpture is considered as a 3D digital model. These approaches almost follow the “bas-relief ambiguity”[6], that is, roughly speaking, the relief looks like a full 3D object from a frontal

view, and its disproportional depth would be revealed from a side view. The relief generation from 3D model was firstly studied in [14], then various existing 3D model based methods have demonstrated how to compress 3D model into relief [74][33][65][67]. However, these approaches require a 3D model as a starting point.

On the other hand, **how to generate a relief from a 2D painting?** Unfortunately, such a problem remains unsolved. There have been some relief generation methods available from a single image [83][76][5][77]. Reconstructing a surface from a single 2D image is an ill-posed problem in general. As described in most of the researches on 2D image based reconstruction algorithms, the problem that manifests itself immediately is that there is no complete knowledge of the depth within a single image. When watching a painting, although the audience simply knows that its content is not 3D, it can give some level of impression of the three-dimensional scene. The relief can go a step further by using the varying depth on the surface of the sculpture to convey the perception of the three-dimensional scene. A 2D painting can be considered as an image, however, the image based methods [83][76][5][77] are generally focusing on general photograph from real scene, which are based on the assumption that input image is formed from lighting and shading, which is especially unsuited for a brush painting, which does not obey the rules of lighting and shading, or emphasizes realism [41]. On the other hand, concerning the artistic intent of a painting, it is crucial that the style of the originals is preserved. In the case of reliefs, although there is no 3D model available, pseudo-3D effect reflecting the style and subtlety is crucial in preserving the artistic essence, which is not taken into account in image based methods.

Some research focus on relief generation from a line drawing [35][72][47][68]. However, maintaining the styles of paintings proves much trickier than simply manipulating the height of the contour lines, since line drawing based methods generate reliefs without consideration for surface details. They are limited to using the information contained in a line drawing, which is not effective for paintings containing information such as colour, texture and stroke shape.

## 1.2 Motivation

The research is motivated by both the theoretical studies and the practical applications:

- Theoretically, as mentioned above, relief is regarded as an art form part way between 2D painting and a full 3D sculpture, as claimed by many previous works [7][74][33][34][83]. Research so far has been primarily done based on a 3D model, and some other methods based on a single photograph[83][76][5][77] or a line drawing [35][72][47][68] as input. However, how to generate a relief from a 2D painting remains a problem.

  On the other hand, due to the various styles of painting, it is difficult to come up with a general relief generation method suitable for all 2D paintings. Some paintings may be too abstract for relief generation. Meanwhile, traditional classic western paintings developed in the Renaissance emphasize realism[13], so they could be handled by 2D image based methods. On the other hand, a brush painting usually does not emphasize realism, and the style and philosophies of brush painting have influenced other painting styles [13]. Study of relief generation from brush painting will naturally push forward the research frontier of relief generation from other paintings.

- In practice, relief generation also can be widely used in minting, decorations of furniture, walls, and artistic sculptures for blinds. With the commonly and cheaply available 3D printing facilities, there is a growing trend in the need of relief art products. On the other hand, as the input of the relief generation method in this work, the digital image of a brush painting is easily accessible on the Internet.

Therefore, designing an efficient method for relief generation from brush painting is important both theoretically and practically. In particular, this thesis focus on relief geometric reconstruction from a brush painting, which is useful for supporting applications such as 2D character design, animation, stereoscopy and rotating lenticular posters.

As mentioned above, surface reconstruction based on a single image is considered as an ill-posed problem since there is no complete knowledge of the depth within

a single image [83][76][5][77]. However, with the purpose of generating relief with acceptable quality, there are several factors to be considered specifically for the framing of the problem. Benzaid et al.[7] describe some core important factors in the relief generation process: depth information, silhouettes and edges, fine details.

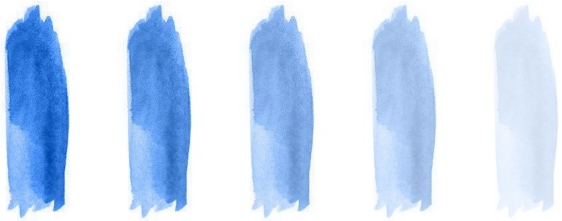
**Depth information:** There is no complete knowledge of the depth within a single image. A reasonable interpretation must be determined in some manners.

**Outlines:** Typically, outlines are used to separate objects from the background. The outlines produce a reasonable contrast between the foreground object and the background, and they are also established as the starting point for the interpretation of the height.

**Fine Details:** The image contains many small features and details. Small features and details are preferred for maintenance during the generation of relief from the 2D input.

On the other hand, a brush painting can be regarded as an union of brush strokes [79]. In a brush painting, all of the painter's aesthetic understandings are achieved through brush strokes. Without brush strokes, there would be no brush painting [41]. As shown in Fig. 1.2, a brush stroke has its own features(colour, shape and opacity).

(a) Brush strokes with the same opacity and shape but different colours.



(b) Brush strokes with the same colour and shape but different opacities.

(c) Brush strokes with different colours, shapes and opacities.

Fig. 1.2 Features of brush strokes.

**A brush stroke is a mark made by drawing paintbrush across paper or canvas.** A brush stroke serves the purpose of outlining contours of objects, separating space and preserving details [41] [18]. Generally, one brush stroke is employed to represent some specific real objects, such as a flower petal, a bud, or a feather [37][63][41]. In brush painting, it is observed that most brush strokes are rendered in a continuous, rhythmic motion, and the colour and opacity vary little along the direction of the brush strokes [37][79][41]. As mentioned above, it is impotent to maintain the **depth information, outlines, fine details** of the image in relief generation [7]. On the other hand, brush strokes serve the purpose of outlining objects, separating space and keeping the fine details in brush paintings[41][79][37]. Therefore, to produce reliefs with acceptable quality, it is essential to preserve these features (colour, shape and opacity) of brush strokes on relief.

In digital painting software, such as MyPaint and Photoshop, brush strokes are painted on one or more layers. In digital painting, it is often useful to paint different elements on separate layers, and then combine the resulting multiple layers into a single, final image called the composite. **A layer is an RGBA image, in which each pixel has an RGBA colour value (red, green, blue, alpha). Alpha indicates the degree of opacity of each pixel (from fully transparent to fully opaque) and allows the use of layer composition.**

The resulting questions are how to extract brush strokes from a painting and how to preserve the features of brush strokes on a relief. However, no consideration has been made for these problems in the previous 2D photograph based methods[83][76][5][77] or line drawing based methods [35][72][47][68]. Even though there are similarities between relief generation and brush painting, how to generate reliefs from brush paintings remains a problem.

## 1.3 Aim and Objectives

The aim of this research is to generate the geometry of relief from a brush painting. This work also argues that, because most brush paintings are produced with individual brush strokes, generating relief surface from each brush stroke would preserve the original features of the painting. Most brush paintings typically contain lots of brush strokes which exists as individual pieces of art [63][16]. Differing from the other relief generation methods, this work will preserve this very important feature by generating relief surfaces from the brush strokes individually. This approach nonetheless demands to conquer several challenges:

1. Unlike photographs, the opacity of brush strokes is an important feature of a brush painting. Artists are used to varying in colour, shape and opacity of the brush strokes to achieve certain painterly effects [46]. It is desired to preserve the features of a brush stroke on the relief.

2. Unlike the previous 2D image based methods focusing on photographs, painting does not obey the rules of lighting and shading exactly, which increases the level of difficulty to mimic the details on a relief surface.

3. Each brush stroke covers a region on the canvas and may overlap each other, some quite heavily in a painting. To make sure the information is retained, every brush stroke has to be faithfully extracted.

These questions served as an important rationale behind this research. To address these challenges, the following aims and objectives of this work are proposed.

The objectives of this thesis are to:

1. Design a technique to extract brush strokes regions accurately.

2. Design a method to detect the colour and opacity of brush strokes efficiently.

3. Develop an approach to preserve the features of brush stroke on relief surface.

4. Produce a method to generate a relief model based on a 2D brush painting.

## 1.4 Contribution

As mentioned above, the previous relief generation methods have demonstrated how to generate reliefs from 3D models, photographs and line drawings. In contrast, this research aims to generate a relief from a brush painting, aiming to preserve the original artistic features of the painting. To the best of the author's knowledge, this is the first work of relief model creation from a single brush painting input. The pop-up relief model is easily imported into the most popular industrial software including Houdini and Maya for realistic rendering.

This work particularly considers the overlapped brush strokes with complex shape profiles and their opacities. In contrast to the previous 2D image based methods, the proposed algorithm is also the first attempt to generate reliefs using opacity, and this research also demonstrates that opacity can preserve more details than intensity. Accompanying this method, a layer decomposition based brush stroke extraction algorithm—**LStroke** has been proposed. The major difference between **LStroke** and the previous works is that **LStroke** is based on a reformulated layer

decomposition algorithm, which makes it capable of extracting overlapped brush strokes without the prior knowledge of brush strokes.

The detailed contributions of this system are listed below:

1. Comparing with the previous brush stroke extraction method [42], this work is capable of extracting overlapped brush strokes and has higher accuracy.

2. Comparing with the previous 2D image based method [81], this work is more efficient for relief generation from brush painting. And this work uses opacity instead of intensity to generate reliefs, which can better preserve the feature of the input painting.

3. Comparing with the previous relief generation method using the raw image as texture [81], this work can preserve the opacities of brush strokes on relief surface, which better shows the feature and inter-relations of brush strokes on relief.

4. The proposed brush stroke extraction technique has other potential uses. This work demonstrates the utility of the decomposed brush strokes for image editing.

To better extract brush strokes with similar colours in a painting, this work also proposed **DStroke**, in which deep learning techniques are leveraged to facilitate the soft segmentation of brush strokes. Its contributions are:

1. To the best of the author's knowledge, this is the first work of brush stroke extraction based on deep learning. Comparing to previous semantic soft segmentation method [3] which can only segment semantic regions, the **DStroke** generates instance-level soft segmentation results, namely, brush strokes.

2. There is no available brush painting sample training dataset for the deep neural network at the moment. Manually marking brush strokes is a tedious task and time consuming, and it is unrealistic to manually create a large training dataset with segmented brush strokes on paintings. This work proposes an automatic algorithm to create brush paintings, and based on such an algorithm, this work creates a large training dataset containing

5000 samples. Each sample in the dataset contains a brush painting and the corresponding segmented brush strokes.

3. The higher efficiency. The proposed **DStroke** runs more than 100 times faster than the previous soft segmentation method [3], which is ideal for a large amount of brush strokes extraction.

4. The higher accuracy. In contrast to **LStroke** and soft segmentation method [3], the **DStroke** shows noticeable higher accuracy.

## 1.5 Thesis Outline

The organization of the document is as follows:

Chapter 1: **Introduction**. This chapter provides the background, the motivation, aim and challenges made in relief generation from painting, and lists the contributions of this work.

Chapter 2: **Related Work**. This chapter classifies and reviews related previous works in a systematic way. It reviews the development of relief generation over recent years. Relief generation methods are generally classified into types: 2D image based methods and 3D model based methods. The relative researches of brush stroke extraction are also discussed.

Chapter 3: **Overview of Proposed Approach**. This chapter explains the overview of this work and defines some basic concepts used in the research.

Chapter 4: **Layer Decomposition Based Brush Stroke Extraction—LStroke**. In this chapter, the layer decomposition based brush stroke extraction method — **LStroke** is introduced. The **LStroke** consists of two steps: layer decomposition and brush stroke extraction. This chapter introduces the concept "Layer Decomposition" in this research, which is the basis of the proposed **LStroke** method. This chapter also describes the proposed algorithm of brush stroke extraction. Finally, the results are evaluated and compared with related methods.

Chapter 5: **Deep Network Based Brush Stroke Soft Segmentation—DStroke**. This chapter presents a brush stroke soft segmentation method based on a deep neural network—DStroke. A data generation method is also proposed for training purpose. The results are evaluated and compared with related methods.

Chapter 6: **High Relief Generation**. This chapter shows how to generate high relief from extracted strokes. The results based on proposed method are discussed and compared related work.

Chapter 7: **Other applications**. Once the brush strokes are extracted, they can be used in a number of image editing operations. This chapter shows how to use the extracted brush strokes to enable interesting paint-aware applications.

Chapter 8: **Conclusions and Future Work**. This chapter concludes the progress up-to-date and discusses possible future work.

## 1.6 Publication

During the period of the research, a part of this work has been accepted or published at *IEEE Transactions on Visualization and Computer Graphics*:

**Fu, Y.**, Yu, H., Yeh, C. K., Zhang, J. J., Lee, T. Y. (2018). High Relief from Brush Painting. IEEE transactions on visualization and computer graphics.

# Chapter 2

# Related Work

This chapter reviews notable relief generation methods. They are classified into two categories: methods based on the 2D image and methods based on 3D model, which are reviewed in section 2.1 and section 2.2 respectively. Following with section 2.3 which reviews notable brush stroke extraction methods and relative works. The structure of the related works is shown in Fig. 2.1.

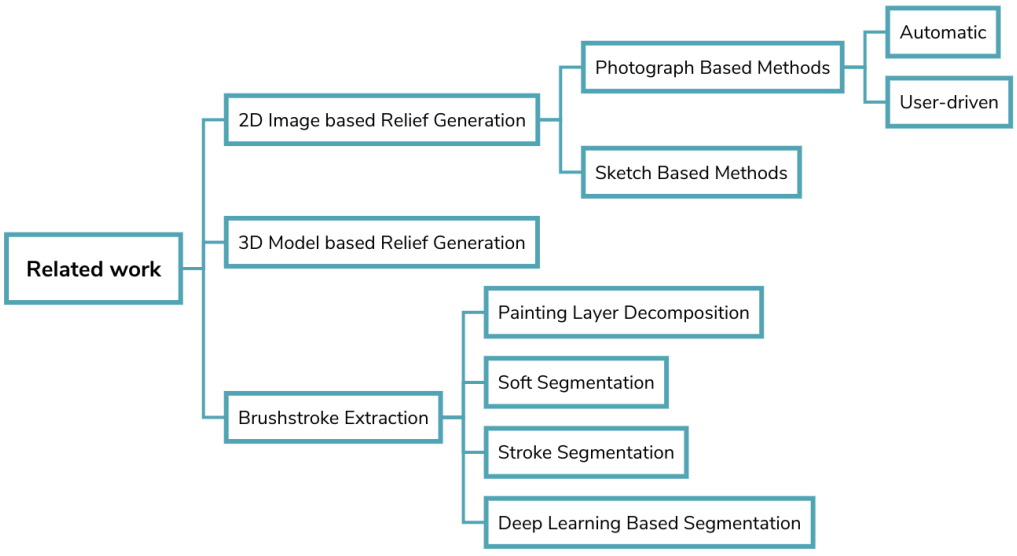



Fig. 2.1 The structure of related works.

## 2.1 2D Image Based Relief Generation

Inferring a 3D shape from an image is a highly under-constrained problem, which requires as much prior knowledge of the 3D shape of an object class as possible, so as to yield reasonable results. When the input is a man-made painting, it becomes more challenging. This is because (1) the man-made paintings may violate the rules of lighting and shading; (2) the input may consist of a lot of strokes that need to be specified by the users; and (3) these strokes are usually inter-related for artistic composition purposes. The closest approach to the topic of brush painting based relief production is the image based method. Since there is a huge volume of previous works in this area, this section reviews the important related works by classifying them according to their inputs.

### 2.1.1 Photograph Based Methods

***Automatic methods:***

A number of studies have been conducted on automatically reconstructing a 3D scene from a single image. [25] [26] [27] extracted the 3D structure of outdoor scenes by segmenting input image and classifying the image into different labels, such as ground, sky, and vertical objects. [60] created the a 3D scene using supervised machine learning to model both image depth cues as well as the relationships between different parts of the image. Similarly, [45] estimates the depth of each pixel by semantic segmentation.

Shape from shading (SFS) is another related area. It has been observed that it can be challenging to reconstruct original 3D shape from a single image using the Shape from Shading (SFS) method. Initially, the solution of a non-linear first order Partial Differential Equation (PDE) was used to formulate the SFS problem [28], and later a generalized equation was derived to reconstruct a surface based on a given non-integrable gradient field [1]. Many methods have been developed with different assumptions, such as orthographic SFS with a far light source [44], perspective SFS with a far light source [55], and perspective SFS with a point light source at the optical center [56]. These SFS methods assume the image is formed from lighting and shading, which may work well for realistic photographs with

relatively simple texture rather than paintings. Simply applying the SFS method is not enough to generate relief with acceptable quality for painting images.

[73] demonstrated a method by constructing reliefs from 2D images based on gradient operations. In their research, image gradients were calculated, then by smoothing gradients to smooth shape changes. Finally, they boost fine features with user input masks. The height of the image pixels is reconstructed based on the modified gradient field. Then, the pixel heights are compressed. [73] uses a triangle mesh to represent the relief, in which every vertex is assigned with a pixel position. Most features can be preserved by the proposed method. However, the depth relationship between segmented regions is not considered. [43] reconstructs depth map from the rubbing images of restoring brick and stone relief. The relief is separated into low and high frequency components. The low frequency component of relief is generated with mesh deformation, and the mesh deformation scheme is based partial differential equation. The high frequency can be generated from input images automatically or by user interaction. This method focuses on relief generation from stone rubbing images, which is unsuited for paintings.

In general, fully automatic methods are insufficient to support the reconstruction of complex objects due to insufficient 3D information available in a single image.

***User-driven methods:***

One cannot use the traditional SFS method for relief generation because of bas relief ambiguity unless there is human intervention. This is so because these ambiguities are not unique for a certain image, within the reconstructed surface shape when different lighting conditions are used [6]. As such, human intervention is needed to generate an appropriate surface. In order to by-pass this problem, additional information and some assumptions are needed to provide visual cues [82].

Zeng et al. [82] introduced a method that needs the users to first set a reasonable surface normal, and then the SFS method is applied locally to the reconstruction of each surface patch. These local solutions are combined to form a smooth global surface. Wu et al. [77] proposed another interactive system for the same purpose. Firstly, they improve existing SFS algorithms to reconstruct a correct normal for

local image regions. After which, they use a simple mark-up procedure to correct low frequency errors. They were able to increase the quality of surfaces, but their methods required considerable user interactions.

User interactions can be applied to relief generation. When the SFS method is used in high relief, even more user interactions are needed. Additionally, the method presents more problems when colour images inputs or those containing complex textures are used as input. Alexa et al. [5] came up with reliefs whose appearance is different when illumination from different directional light sources are used instead of ensuring that an image looks reliable under one constant lighting condition. They placed small pyramids at the centre of each image pixel and deformed them according to the desired reflectance properties. Their algorithm can produce reliefs that contain information about a pair of input images in one single piece of art. Moreover, colour information of a given image can be transferred to the relief representation if directional colour light sources are used. This is the first method to make full use of the nature of reliefs and their ambiguity and use them as a type of display. Nonetheless, it cannot prevent the large depth gradients on relief because of the considerate difference in regions when working with colour images. And the large depth gradients on the relief would make it hard to fabricate. However, SFS methods formed from lighting and shading, and even with user intervention, the details of painting such as the shape and transparency of brush strokes are not considered.

Oh et al.[53] represent a scene as a layered collection of depth images. Two user-assisted tools are employed, based on a painting metaphor, to assign depths and extract layers, and manually inpainted the hidden parts. [84] uses sparse user constraints on input images to generate a smooth shape. Yeh et al.[81] present an interactive user-driven method to reconstruct high-relief 2.5D geometry from a single photo by using user-specified depth cues. The method enables the creation of a double-sided organic shape. However, these user-driven methods cannot handle complex shapes and transparency of brush strokes, and the reconstructed mesh is smooth and so it cannot preserve the brush stroke details.

### 2.1.2 2D Sketch Based Methods

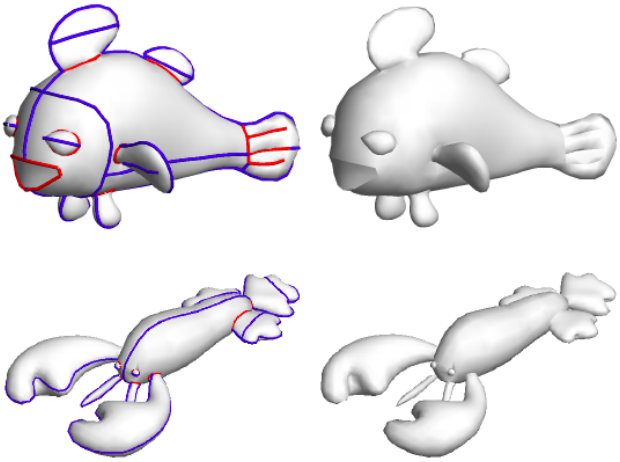
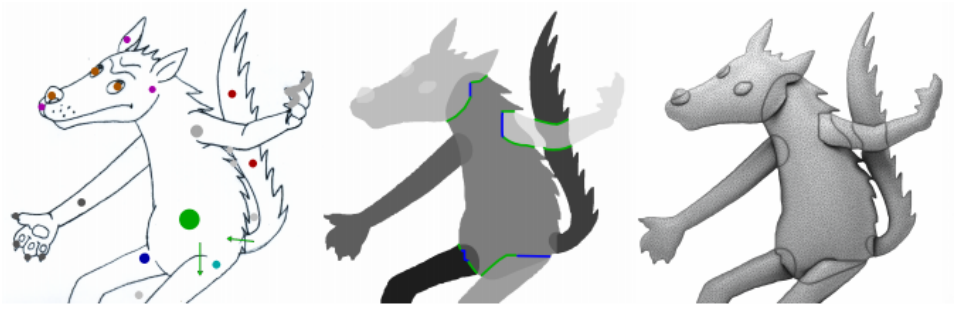


Fig. 2.2 Sketch based modeling in [51].

Sketch based modeling is another research topic relevant to this work, which provides a powerful paradigm for 3D modeling based on sketches. Sketches is a drawing made exclusively in lines. Rather than generating relief from a photograph of a real scene, some researches focus on relief generation methods from sketches. Igarashi et al. [29] firstly presents the Teddy system, a sketching interface for designing freeform models by interactive drawing 2D contours. Similarly, a range of methods use line contours from sketches to infer the shape of model. Karpenko et al. [32] generate smooth shapes by retrieving suggestive contours from visible-contour sketches. [51] enables users to add, remove, and deform control curves to construct a smooth surface, while the user-drawn strokes stay on the model surface and serve as handles for controlling the geometry. A common drawback of these approaches is that they require tedious specification of control curves to produce the desired shape, and since these methods are generally used for authoring 3D contents, they are not designed for 3D modeling with a fixed reference image.

Kolomenkin et al.[35] aims to reconstruct a model from a complex line drawing that consists of many inter-related lines. At first, they extract the curves from line drawing. Then, junctions between lines are detected and margins are generated. By analysing the connectivity between boundaries and curves, they reduce the problem to a constrained topological ordering of a graph. From these boundaries and curves with given depths, they use smooth interpolation across regions to



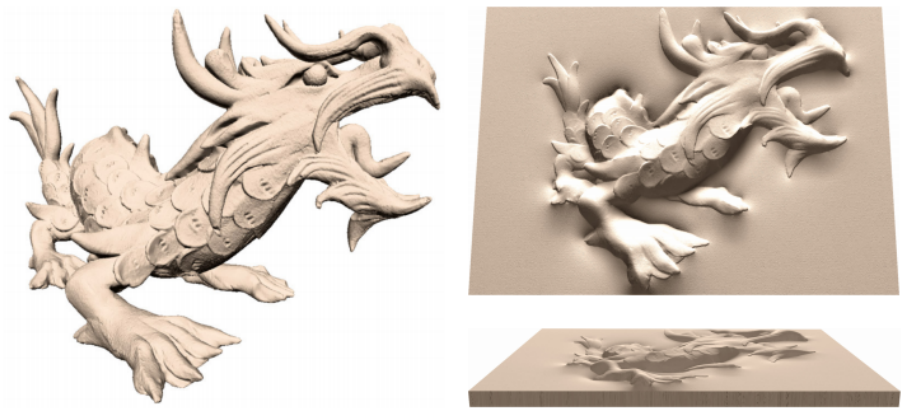
(a) (b) (c)

Fig. 2.3 Bas relief generation from line drawing by Sykora et al. [68]. (a) Input drawing including user annotations. (b) Layering the segmented regions. (c) Resulting bas-relief-type mesh.

generate the relief surface. Similarly, line labeling methods have been applied for shape construction from line drawings [72][68]. A labeling process would classify segmented lines into different labels, such as concave, convex and occluding, and these labels can give clues for the shape generation of relief. [68] proposed a relief generation method consisting of six main steps: segmentation, completion, layering, inflation, stitching, and grafting. The segmented regions are layered with each other (Fig. 2.3b) based on the user indications (Fig. 2.3a). This method combines user indications and shape inflation to model smooth relief shapes from line drawings, as shown in Fig. 2.3. However, this research aims at paintings with brush strokes, which are different from 2D line drawings. Line drawing based approaches are limited to using information contained in a line drawing, while a brush stroke does not only contain contour lines but also delineates a region, which contains information such as colour, texture, opacity. It is crucial to identify and extract brush strokes from a painting.

## 2.2 3D Model Based Relief Generation

A sizeable amount of methods has been considered relief generation from 3D model [14][74][33][65][67]. [14] firstly proposed a method creates relief models based on the input 3D models, in which the input model is compressed by a linear function. This method fails to handle the depth gap between in 3D models and the output reliefs can not successfully preserve the details on 3D models. [65] considered depth map of the 3D model as the high dynamic range (HDR) image.



(a) Original 3D model (b) Generated relief

Fig. 2.4 Relief generation from 3D model by Weyrich et al. [74].

The differential coordinates of the input models are calculated. Then, HDR image process method is applied to compress the 3D models. The output relief can preserve salient feature and de-emphasize the others, but sometimes the relief can be distorted and exaggerated. The methods proposed in [74][33] work on the gradient domain of depth map of 3D models, and apply the techniques of feature enhancement and saliency measure. However, these methods produce generally satisfactory relief output, and preserve the features of 3D models, although again some areas can be over-emphasized, as shown in Fig. 2.4. [67] generate the relief with an optimized contrast-limited adaptive histogram equalization (CLAHE) method, in which the depth map of the 3D model is compressed. Local contrast of depth map can be enhanced in this method,however, the detail preservation requires high resolution depth map from the 3D model.

These 3D model based methods can often generate relief with acceptable quality, but they require 3D models as input. 3D models are often unavailable or difficult to prepare, and obviously unsuited for relief generation from 2D paintings.

## 2.3 Brush Stroke Extraction

To the best of my knowledge, there is a lack of study on extracting brush strokes from paintings. Here, a brief overview of the relevant topics is given, i.e. decom-

posing images into layers and stroke segmentation, which are employed in this work.

### 2.3.1 Painting Layer Decomposition

In digital image editing, artists deposit colours throughout the image via a set of strokes, which can be regarded as individual layers with opacity values. However, scanned paintings and photographs have no such layers. Even for digital paintings, there is commonly no layer information in the output files. Without layers, simple edits may become very challenging. [57] presents an interactive approach for decomposing bitmap drawings and photographs into opaque and semi-transparent vector layers. [79] aims at decomposing Chinese paintings into a collection of layered brush strokes with an assumption that at most two strokes are overlapping and there is minimally varying transparency. However, it only focus on Chinese painting with limited brush strokes. [48], [49] present two generalized layer decomposition methods, in which pixels have individual layers and partially overlap to each other, and the layer orderings may be manipulated. [2] proposes a recolouring painting method for watercolour paintings based on the colour palette estimation and layer decomposition. [70] presents a layer decomposition method based on RGB-space geometry. They assume that all possible image colours are convex combinations of the palette colours. In [70], computing the convex hull of image colours and per-pixel layer opacities is converted into a convex optimization problem. Thus, their method can work well without prior knowledge of shape and overlap of strokes.

### 2.3.2 Soft Segmentation

(a) (b) (c)

Fig. 2.5 Image matting by Levin et al. [39]. (a) Input image. (b) Trimap. (c) Alpha map of foreground.

Segmentation is one of foci in image process all the time. The challenging issue is that the boundaries of some objects are usually vague. Manual segmentation is tedious and time consuming very much. For the objects with opacity, e.g. water, hair, brushstrokes in application, it is becoming infeasible. Soft segmentation of images is proposed to cope with such challenge since it can capture the soft transitions between image regions. **Soft segmentation** is decomposing an image into two or more segments where each pixel may belong partially to more than one segment [3]. On the other hand, for **hard segmentation**, each pixel can only belongs to at most one segment.

Soft segmentation is closely related to image matting, but there are distinct differences between them. Matting aims to estimate the per-pixel opacity of foreground region based on the foreground region indications that users provide [39][12][4]. The indications are typically represented as **Trimap**. A **Trimap** is a pre-segmented image consisting of foreground(white region), background(black region), and unknown opacity region(grey region), as shown in Fig. 2.5b. The extracted foreground is represented by a **alpha map**. An **alpha map** is a 2D image which indicates how opaque each pixel is (from fully transparent to fully opaque), as shown in Fig. 2.5c. The alpha value of each pixel on the alpha map is between 0(fully transparent) and 1(fully opaque).

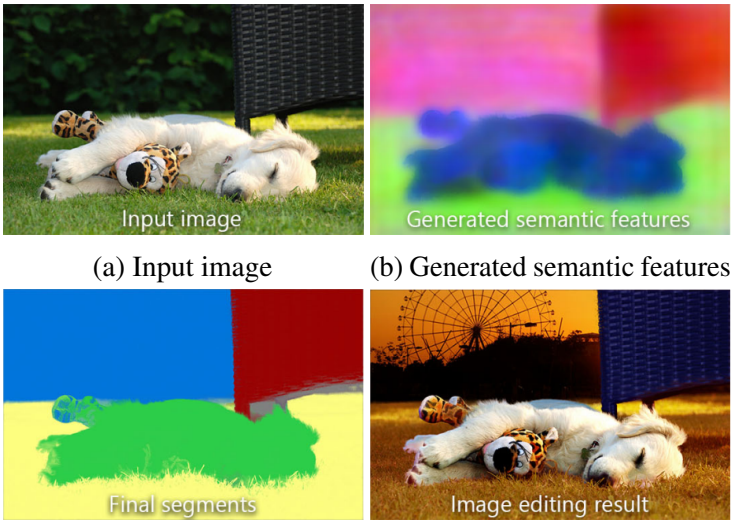
Instead of selecting the foreground objects from image, soft segmentation decomposes an image into multiple layers. As a result, each pixel is likely to be classified

into multiple layers [3][40]. [62] presented an approach of segmenting an image into multiple layers by the estimation of alpha maps, which needs to be optimized iteratively. Levin et al. [40] proposed spectral matting adopting matting Laplacian [39] and spectral decomposition to estimate a set of connected soft regions with fuzzy boundaries.

However, these soft segmentation works cannot generate semantically meaningful regions without user indications in the image. Most recently, Aksoy et al. [3] firstly leverage deep network for semantic soft segmentation focusing on this problem, see Fig. 2.6. i.e. using the high-level information (semantic information) from a deep network to define affinities between different regions in order to generate soft segmentation corresponding to semantic regions. On the other hand, low-level information such as colour affinity and matting affinity, are also calculated to construct the matting Laplacian matrix to handle fuzzy boundaries. Like spectral matting, regions with local soft transitions are generated using matting Laplacian and spectral decomposition. However, for brushstroke extraction, their methods suffer at least two limitations: (1) A brush painting normally contains hundreds or even thousands of brushstrokes. Unfortunately [3] and [40] only support a limited number of segmented regions. (2) Although using high-level information from a deep network, [3] cannot still separate different instance of the same class, e.g. two people are classified into the same layer due to the same semantic label. This will bring about serious errors for brush paintings since hundreds, or even thousands of instances (brushstrokes) in one painting may share the same semantic label.

### 2.3.3 Stroke Segmentation

Additionally, for oil paintings such as van Gogh's artefacts, [42] presented the seed growing based brushstroke extraction scheme for painting authentication and artist identification purposes. Firstly, some pixels are selected as seeds. Then, neighbouring pixels are exploited through region growing. Region growing can be controlled through a shape validation method. In their implementation, the metric of shape validation is defined based on 10 examples of brushstroke regions that are manually extracted from van Gogh's paintings. In section 4.3.1, this work is compared with their method. Usually, most of brushstrokes of oil paintings




(a) Input image (b) Generated semantic features

(c) Soft segmentation of input. (d) Image editing

Fig. 2.6 Soft segmented image by Aksoy et al.[3]. The soft segmented regions are represented in different colours.

tend to be opaque. Thus, the boundaries of brushstrokes can be detected. But the overlapped parts will be missed. [79] aims to decompose Chinese paintings into a collection of layered brush strokes with an assumption that at most two strokes overlap each other. However, their approach requires the order of strokes and a brushstroke library for support, which is built by professional artists.

On the other hand, some researches focus on the stroke segmentation of handwritten characters, such as pen strokes [22]. Pen stroke edges are distinct and can be extracted entirely in handwritten characters. As a result, a stroke is described by a set of geometric primitives. The hand-drawn primitives are further replaced by mathematically precise shapes to produce a neat final result. However, brushstrokes on a painting often contain individual colours and overlap each other. Moreover, the strokes may show a low contrast to the others or the background. The edges of strokes are blurred.

## 2.4 Deep Learning Based Segmentation

Krizhevsky et al. [36] introduced a deep convolution network called Alexnet which consisted of 8 layers and millions of parameters and trained on the ImageNet

dataset with 1 million images. Since then, even larger convolutional networks were designed and competed with the state of the art in image segmentation. A detailed review about deep learning based segmentation is beyond the scope of this research.

The state of art deep learning methods for segmentation includes [86], [11], [30], [19]. In [58], U-Net was proposed for biomedical image segmentation, which provides a pixel-wise accuracy for dense cells segmentation. Deep convolution neural networks are used to learn through minimizing a loss function. It is crucial to design an appropriate loss function. For instance, if simply using the Euclidean distance between the predicted and ground truth pixels as loss, the results may tend to be blurry. Unceasingly involving expert knowledge into loss functions can improve the performance of deep convolution neural networks. The discriminator network is combined with U-Net [58] in the Pix2Pix network [30], which can automatically learn a loss function appropriate for satisfying this goal. It improves the accuracy of segmentation. This work make use of this network [30].

However, the main challenge is that there is no available training dataset for brushstrokes extraction at the moment. The deep learning networks always prefer to a large dataset for training. Manually extracting brushstrokes is time consuming since a painting usually contains thousands of brushstrokes. To deal with this challenge, an automatic method is proposed to build up a large training dataset, in which brush paintings are created by a set of brushstrokes and the resulting edge maps of brushstrokes on paintings are available as well.

## 2.5 Summary

In this chapter, the related research works on current relief generation techniques has been discussed. To the best knowledge of the author, there is a lack of study on extracting brush strokes from paintings. A brief overview of the related topics is given. Inspired by these works mentioned above, a relief generating framework from brush paintings is proposed in this thesis with the combination of brush stroke extraction.

# Chapter 3

# Overview of Proposed Approach

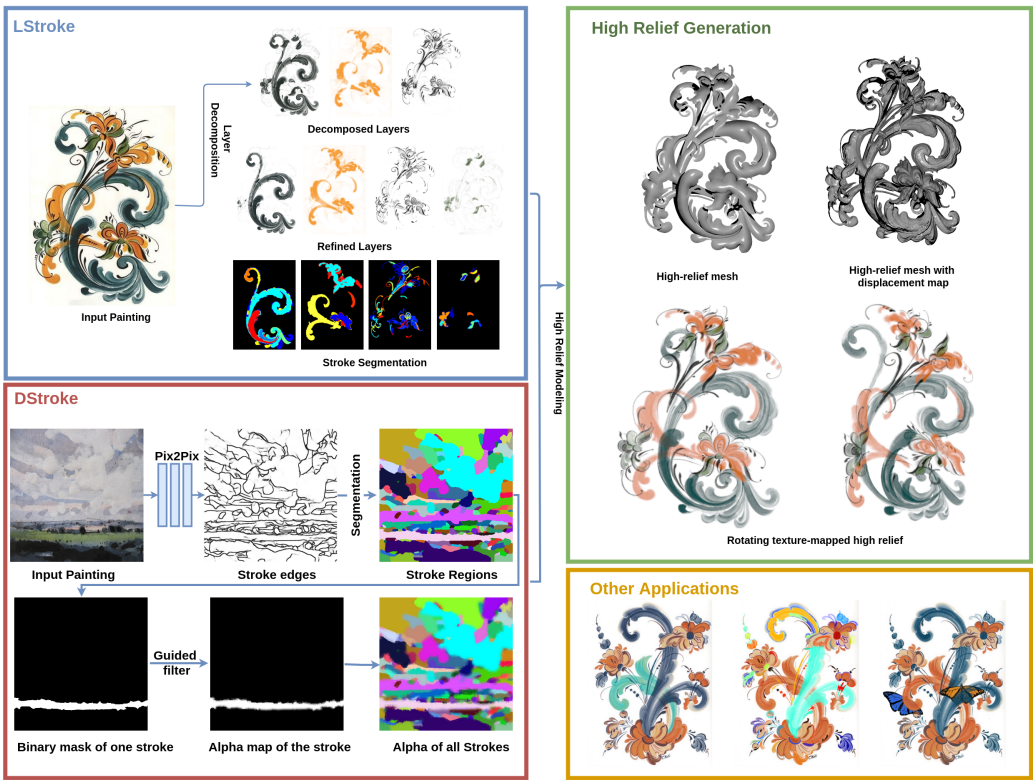



Fig. 3.1 Overview of proposed approach

As shown in Fig. 3.1, this work consists of two stages, the brush stroke extraction stage and a relief generation stage. For brush extraction, this work proposes two methods: a layer decomposition based brush stroke extraction method — **LStroke**

(Chapter 4); a deep learning based brush stroke extraction method — **DStroke** (Chapter 5). Then, the relief surfaces from each brush stroke are generated in the relief generation stage (Chapter 6).

**LStroke:**

In the **LStroke** (blue rectangle in Fig. 3.1), a given painting is firstly decomposed into a set of layers in terms of several specified palette colour values (see the "decomposed layers" in Fig. 3.1). Secondly, a new palette colour value is determined based on the unclassified regions, and the layers are recomputed accordingly in an iterative way (see the "refined layers" in Fig. 3.1). Thirdly, the brush strokes are extracted from each layer.

The key point is to extract the overlapped brush strokes. Overlapped strokes make colours blend. To tackle it, layer decomposition is employed here, which decomposes the painting into a set of translucent layers. In brush paintings, mostly a brush stroke only utilizes a single palette colour. Layer decomposition helps classify brush strokes separately into different layers based on the palette colour, so that every layer contains the strokes which are well separated.

However, wrong layer decomposition may cut one stroke into two or more layers. It is observed on multiple layers, brush strokes of such paintings typically follow same patterns. For instance, a scan of Rosemaling painting employs many C and S brush strokes, and the colour and transparency change very little in the direction of the stroke. However, if assigned the wrong palette colour, brush strokes may appear on multiple layers. The edge tangent flow (ETF) field and the coherent lines [31] are introduced to enhance such features in paintings, which are in favor of preserving the completeness of the strokes in every layer and effectively correct the errors due to wrong layer decomposition.

Moreover, for the paintings whose strokes cannot be clearly decomposed into a limited number of layers, an iterative scheme has been developed to refine the layers. The overlapping regions of multiple strokes with high opacity usually result in the gaps that break the strokes into one or more layers. To tackle this challenge, an inpainting technique is employed here. The coherent line is further involved in the MSERs algorithm [15] again for extracting strokes, which both preserves stroke continuity and removes spurious edges within one layer.

**DStroke:**

The layer decomposition benefits the brush strokes with different colours, however, the main challenge is that when the adjacent strokes share the similar colour in a brush painting, they would appear on the same layer and might be accidentally regarded as one brush stroke.

To tackle such a challenge, **DStroke** is proposed (red rectangle in Fig. 3.1), in which deep learning techniques are leveraged to facilitate extracting brush strokes from painting. To train the deep neural network, this work introduces how the generate the training data for brush stroke segmentation (Section 5.1). Firstly, the edge of brush strokes are detected by a trained network. The structure of the deep neural network and its implementation are explained (Section 5.2). Here, the deep neural network is trained and applied for brush stroke edge detection. Secondly, with the edges of brush strokes, each stroke region are easily extracted. Then, based on the extracted brush stroke regions, the opacity values of each brush stroke are further extracted (Section 5.3).

**High Relief Generation:**

With the brush strokes extracted by **LStroke** or **DStroke**, this work generates the texture-mapped high relief (green rectangle in Fig. 3.1). To generate the displacement maps of the strokes individually, this work performs shape from shading(SFS) on the opacity of the paintings instead of the intensity, since it better preserves the features of the strokes. It is desirable to transfer the features of the paintings, e.g. the transparency of brush strokes, to the surface of the brush stroke models. Using the inflation method proposed in [81], the surfaces for each brush strokes are inflated and combine it with the displacement maps. The shapes of all the strokes are then arranged to form the desired high-relief. In general, the background plane should be unchanged. Thus, generating strokes individually not only benefit the composition of reliefs, but also avoids this technique issue.

**Other Applications:**

Once the brush strokes are extracted, they can be used in a number of image editing operations to enable interesting paint-aware applications, as shown in Fig. 3.1. In section 7.1, the recolouring of specific brush strokes is demonstrated while keeping their opacity values. Similarly, another image synthesis application is

shown by inserting new objects into brush strokes, see section 7.2. By moving and rotating the texture mapped relief model, this work can also create simple 2.5D animations, see section 7.3.

# Chapter 4

# Layer Decomposition Based Brush Stroke Extraction—LStroke

Digital painting with different layers is an integral feature of digital image editing software, such as Photoshop and Sketchbook. Layers offer an intuitive way to edit the colour and geometry of components and localize changes to the desired portion of the final image. Without layers, brush stroke segmentation becomes extremely challenging, since they can overlap and blend with each other. To tackle these challenges, this work proposes a layer decomposition based brush stroke segmentation method—**LStroke**.

One layer can have different colours in image editing software such as Photoshop, but paintings are commonly generated from limited palette colours, so previous painting layer decomposition methods [70] [2][69] assumes single-colour layers, which means each layer has a single palette colour applied with varying opacity. This assumption also greatly reduces the amount of calculations. Since the brush stroke are normally contains only one palette colour, to separate the brush strokes in to different layers, the **LStroke** also assumes single-colour layers.

Wrong layer decomposition may cut one stroke into different layers. It is crucial to preserve the completeness and smoothness of the brush strokes in layer decomposition. To this end, the layer decomposition algorithm is modified in [70] by involving the coherent lines [31] in the implementation.

## 4.1 Layer Decomposition

The **LStroke** consists of two steps: Layer decomposition and brush stroke extraction. In the following sections, the layer decomposition algorithm in [70] is briefly addressed and then the modifications are discussed.

### 4.1.1 Palette colours Extraction

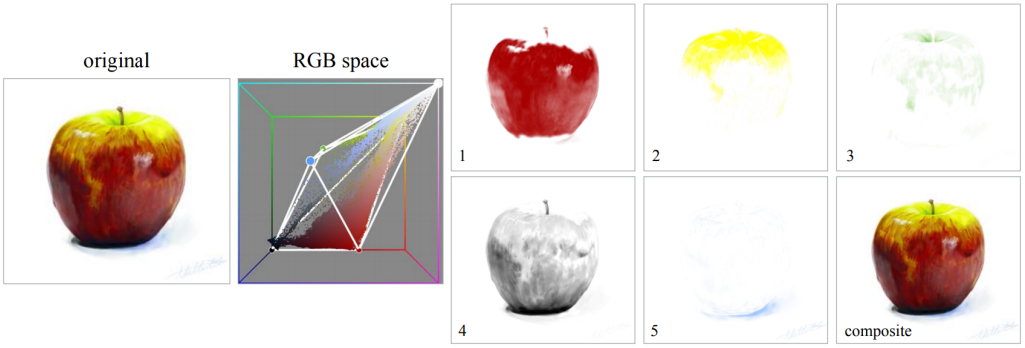



Fig. 4.1 Layer decomposition in [70]. The geometry of pixels in original image(left) is analysed in RGB-space to extract its palette colours(middle), resulting in a translucent layer decomposition(right).

As shown in Fig. 4.1, a input painting is firstly analysed in rgb-space to extract its palette colours , and then decomposed into translucent layers with different palette colours. Each layer represents a single-colored coat of input painting with varying opacity, and the Porter-Duff “A over B” operation is used for compositing these layers.

The “A over B” compositing and blend mode in [54] described that when the pixel A with colour $A_{RGB}$ and opacity $\alpha_A$ is placed over the pixel B with colour $B_{RGB}$ and opacity $\alpha_B$, the observed colour $(\frac{A}{B})_{RGB}$ is,

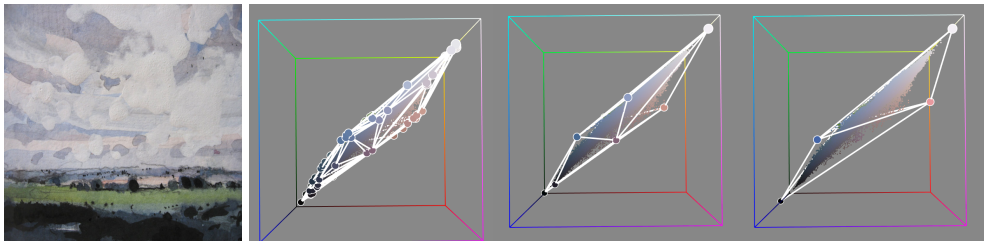
$$(\frac{A}{B})_{RGB} = \frac{\alpha_A A_{RGB} - (1-\alpha_A)\alpha_B B_{RGB}}{(\frac{A}{B})_\alpha} \tag{4.1}$$

where

$$(\frac{A}{B})_\alpha = \alpha_A + (1-\alpha_A)\alpha_B$$

As show in Fig. 4.1, the convex hull of image pixels in RGB-space are computed, and vertices of the convex hull are regarded as the palette(layer) colours of the input painting. And each pixel's colour is viewed as the convex combination of all layers' colours. The palette size and layer order are user-defined.

In practice, tightly wrapping pixels in rgb space by convex hull normally produces a complex shape(many vertices). Too many vertices would result in unmanageable number of layers, so the silhouette clipping method [59] is applied in [70] to progressively simplify the convex hull to a user-specified palette size $n$, as shown in Fig. 4.2.



(a) Input painting (b) Convex hull (c) 9 vertices (d) 5 vertices

Fig. 4.2 Simplification of convex hull in RGB space.

### 4.1.2 Determining Layer Opacity

For each pixel in the input image, the observed colour $p$ on input painting can be approximated by the recursive application of the Porter-Duff compositing operation (Eq. 4.1):

$$p = C_n + \sum_{i=1}^{n} \left( (C_{i-1} - C_i) \prod_{j=i}^{n} (1 - \alpha_j) \right) \tag{4.2}$$

where $C_i$ denotes the $i$-th layer's colour, $\alpha_i$ is the opacity of $C_i$ , the background colour $C_0$ is opaque. $n$ is the user-specified palette size.

Since the input painting should be recomposed by the layers (see Equation 4.2), the following 'polynomial' regularization term $E_{polynomial}$ penalizes the difference

between the observed colour $p$ and the polynomial approximation,

$$E_{polynomial} = \frac{1}{K}\left\| C_n + \sum_{i=1}^{n}\left((C_{i-1} - C_i)\prod_{j=i}^{n}(1-\alpha_j)\right) - p\right\|^2$$

and $K = 3$ or 4 depending on the number of channels (RGB or RGB-$\alpha$). Since the sparse solution for opacity is preferred in [70], namely, it is preferred to move $\alpha_i$ close to 0 or increase some other $\alpha_j$ to 1. The second opacity penalty term $E_{opaque}$ is expressed as,

$$E_{opaque} = \frac{1}{n}\sum_{i=1}^{n} -(1-\alpha_i)^2$$

The smoothness penalty term $E_{spatial}$ is expressed as,

$$E_{spatial} = \frac{1}{n}\sum_{i=1}^{n}(\bigtriangledown\alpha_i)^2$$

where $\bigtriangledown\alpha_i$ is the spatial gradient of opacity in the $i$-th layer. This term penalizes solutions which are not spatially smooth. However, the gradient of opacity is not always aligned with that of intensity, which may result in discontinuity at the edges.

The colour palette $C_n$ is calculated based on the simplified RGB-space convex hull of input image, which can be further selected to represent the colour of decomposed layers. The layer order as well as the number of layers $n$ are user-specified. The opacity for every layer may be solved by minimizing the following combined cost function,

$$E = \omega_{polynomial}E_{polynomial} + \omega_{opaque}E_{opaque} + \omega_{spatial}E_{spatial} \tag{4.3}$$

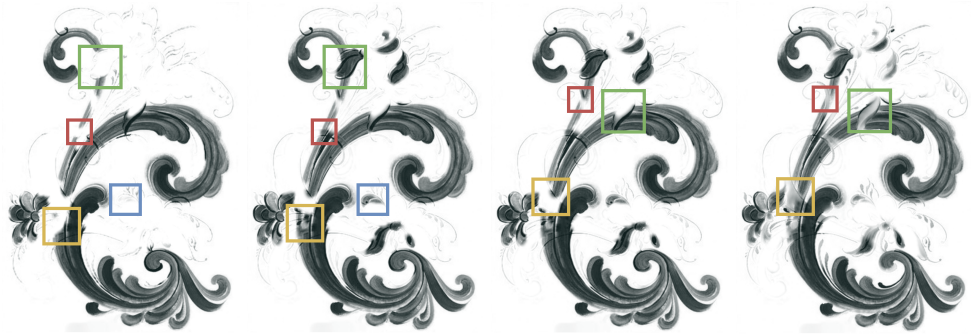
where $\omega_{polynomial} = 375, \omega_{opaque} = 1, \omega_{spatial} = 10$.

### 4.1.3 Modified Layer Decomposition

To enhance the smoothness and completeness of edges, the coherent line drawing technique in [31] is introduced to Eq. 4.3 , which is a flow-guided anisotropic filtering framework.

(a) Input painting (b) Edge Tangent Flow (c) Coherent lines

Fig. 4.3 Edge Tangent Flow field and coherent lines of a Rosemaling painting. It contains lots of C and S strokes.



(a) (b) (c) (d)

Fig. 4.4 Comparison of layer decomposition by Eq. 4.3 and Eq. 4.9 at the 2nd layers. (a) and (b) show the results by using $E_{spatial}$ and $E_{flow}$ in Eq. 4.3; (c) and (d) show the results before and after using $E_{edge}$ in Eq. 4.9.

Fig. 4.3b shows the edge tangent flow (ETF) field of a Rosemaling painting. First, the ETF field is involved into $E_{spatial}$. The ETF field $t$ is defined as,

$$t^{new}(x) = \frac{1}{k} \sum_{y \in \Omega(x)} \varphi(x,y) t^{current}(y) \omega_s(x,y) \omega_m(x,y) \omega_d(x,y) \tag{4.4}$$

where $t(x)$ denotes the normalized tangent vector at pixel $x$, and it is updated iteratively: $t^i(x) \rightarrow t^{i+1}(x)$. $t^{current}(x)$ donates the current tangent vector, and $t^{new}(x)$ represents the update one. The initial $t^0(x)$ is the normalized perpendicular vectors(counter-clockwise) of the image spatial gradient in greyscale, and by

default, $t(x)$ is updated 3 times. The gradient of an image is a vector of its partial derivatives, here the image gradient is calculated by sobel filter[64]. $\Omega(x)$ denotes the neighbourhood of the pixel $x$ with radius $r$, where $r = 5$. $k$ represents the vector normalization. The spatial weight function $\omega_s(x,y)$ employs the radially-symmetric box filter:

$$\omega_s(x,y) = \begin{cases} 1 & if \ \|x-y\| < r \\ 0 & otherwise. \end{cases}$$

The magnitude weight function $\omega_m$ indicates that the bigger weights are given to $y$ whose gradient magnitudes are higher than that of the central pixel $x$.

$$\omega_m(x,y) = \frac{1}{2}(1 + tanh(g(y) - g(x)))$$

$g(*)$ donates the normalized gradient magnitude. $\omega_m$ is a monotonically increasing function, and it ensures the preservation of the dominant edge directions. The direction weight function, $\omega_d$, increases when the two vectors are closely aligned.

$$\omega_d(x,y) = \left| t^{current}(x) \cdot t^{current}(y) \right|$$

It enhances the alignment of vectors, while suppressing swirling flows. In addition, the sign function $\varphi(x,y)$ is employed to enhance the alignment of vectors as well.

$$\varphi(x,y) = \begin{cases} 1 & if \ \ t^{current}(x) \cdot t(x)^{current}(y) > 0, \\ -1 & otherwise. \end{cases} \tag{4.5}$$

As shown in the Eq. 4.4, the ETF of each pixel is influenced by its neighbourhood and it is updated iteratively (3 times by default), so the ETF of each pixel are affected by a large area. As shown in the Fig. 4.3b, even the ETF on the background pixels are influenced by the brush strokes. It is also noteworthy that the ETF are updated from normalized image gradient, so even for the background in the input painting (Fig. 4.3a), some edge tangent flow are not zero.

Involving ETF filed of Eq. 4.3 in $E_{spatial}$, the smoothness penalty is rewritten as,

$$E_{flow} = \frac{1}{n} \sum_{i=1}^{n} \|t_i^{new}\| \left(\bigtriangledown_{\theta_i} \alpha_i\right)^2 \tag{4.6}$$

Here, $t_i^{new}$ represents the ETF on pixel $i$. $\theta_i$ denotes the direction of $t_i^{new}$, and $\bigtriangledown_{\theta_i}\alpha_i$ is the gradient of opacity in the direction of $t_i^{new}$. Moreover, this penalty is weighted by the norm of $t_i^{new}$. Applying the updated $E_{flow}$ to the layer decomposition of Eq. 4.3 instead of $E_{spatial}$, the brush strokes become complete and smooth, which can be noted in Fig. 4.4.

Moreover, the **coherent lines** are involved in layer decomposition of Eq. 4.3. The **coherent lines** are binary edges computed based on the ETF field [31]. Comparing with other techniques such as canny edge [10], the generated edges significantly enhance the coherence of the lines in the input image [31]. So, the generated edges are named **coherent lines**. Since the brush paintings are made by brush strokes, here, the **coherent lines** are used to detect the edge of brush strokes.

Herein, the **coherent lines** $l_c$ are defined as follows,

$$l_c(x) = FDoG(x,t,I) \tag{4.7}$$

Given a ETF field $t$ and input greyscale image $I$, the Flow-based Difference-of-Gaussians(FDoG) filter is employed here [31]. $l_c(x)$ represents the binary result at pixel $x$ as shown in Fig. 4.3c . Unlike Difference-of-Gaussians(DoG) filter [75], the kernel shape of FDoG is defined by the local flow encoded in ETF field $t$. In a image, it is most likely to make the highest contrast in the gradient direction. In FDoG filter, when moving along the edge tangent flow, the DoG filter is applied in the gradient direction. As a result, the FDoG filter significantly enhances the coherence of the generated edges, and suppresses noises. For details, please refer to [31].

Here, the coherent lines is regarded as the edges of brush strokes. To preserve the brush stroke edges, it is observed that the opacity along the coherent lines is consistent, i.e. $min \int_l \| \bigtriangledown \alpha \|^2 dx$, where $l$ denotes the pixel collection of coherent lines $l_c$. Hence, the constraint term is defined by applying Laplacian operator to the opacity along the coherent lines,

$$E_{edge} = \|LY\|^2 \tag{4.8}$$

where all the opacity $\alpha_i$ are stacked in the vector $Y$, and $L$ denotes the Laplacian connection matrix. The eight-connected neighbouring rule is utilized to construct

the connection matrix $L$, that is, if two adjacent pixels, i and j, stay on the same coherent line, the item of $L(i,j)$ is set to -1 ; otherwise 0. Fig. 4.4d shows that the brush strokes become visible and complete after involving $E_{edge}$ into Eq. 4.3. Accordingly, the layer decomposition of Eq. 4.3 is rewritten as,

$$E = \omega_{polynomial}E_{polynomial} + \omega_{opaque}E_{opaque} + \omega_{flow}E_{flow} + \omega_{edge}E_{edge} \quad (4.9)$$

where $\omega_{flow} = 10, \omega_{edge} = 1$ for all the examples. For comparison, the schemes of Eq. 4.3 and Eq. 4.9 are performed separately on the same set of brush paintings and compare the root-mean-square-error (RMSE) of the opacity of the coherent lines on each layer shown in Table 4.1. It has to be noted that there is no brush stroke on the background layer, so the results are not compared on the background layer. In other words, the evaluation starts from layer 1 (the index of the background layer is 0), as shown Table 4.1. The RMSE is defined as, $RMSE = \sum_{i=1}^{N} \sqrt{\sum_{j=1}^{n_i}(\alpha_{i,j} - \bar{\alpha}_i)^2/n_i}$. $N$ donates the number of coherent lines, where the $\bar{\alpha}_i$ is the average opacity of $n_i$ pixels on the $i^{th}$ coherent line, and $\alpha_{i,j}$ is the opacity value of $j^{th}$ pixel on the $i^{th}$ coherent line. The RMSE by Eq. 4.9 is noticeably less than that by Eq. 4.3. This means that the coherent lines have been embedded into the opacity of each layer. The weights are empirically determined in terms of the opacity RMSE of coherent lines. Moreover, the resulting layer by Eq. 4.9 is shown in the upper row of Fig. 4.5.

### 4.1.4 Iterative Scheme

The decomposed layers can be separated into the background and foreground (brush stroke regions) by threshold opacity. It can be noted in the first row of Fig. 4.5, there are some regions shared in multiple layers since such shared regions have visible opacity at the multiple layers. As there is a lack of layers, the colours of the shared regions have to be yielded by blending the colours of the current multiple layers. Moreover, it can be noted that the shared regions may be categorized into two kinds, one is the region overlapped by multiple brush strokes with palette colour, and the others are the isolated ones, as shown in Fig. 4.6. The former are always merged into the other regions within some layers, while the latter are always isolated in all the layers. It is natural to view the isolated regions as potential strokes. Therefore, average colour of the largest isolated region is set

Table 4.1 Opacity RMSE of Coherent Lines on Layers

| Painting ID | Number of Layers | Layer | Opacity RMSE of coherent lines | |
|---|---|---|---|---|
| | | | By Eq. 4.3 | By Eq. 4.9 |
| Rooster (Fig. 6.9, row 1) | 4 | 1 | 25.12 | 15.39 |
| | | 2 | 8.89 | 5.92 |
| | | 3 | 15.74 | 9.63 |
| Man (Fig. 6.9, row 2) | 4 | 1 | 38.41 | 26.33 |
| | | 2 | 21.28 | 16.37 |
| | | 3 | 5.26 | 4.19 |
| Bird (Fig. 6.9, row 3) | 4 | 1 | 8.24 | 6.12 |
| | | 2 | 7.15 | 4.25 |
| | | 3 | 9.24 | 6.71 |
| Lotus (Fig. 6.9, row 4) | 4 | 1 | 16.44 | 10.25 |
| | | 2 | 20.54 | 15.04 |
| | | 3 | 10.56 | 5.87 |
| Lotus2 (Fig. 7.1, row 3) | 4 | 1 | 17.58 | 13.87 |
| | | 2 | 19.78 | 8.72 |
| | | 3 | 22.47 | 17.24 |
| Rosemaling1 (Fig. 6.6, row 1) | 5 | 1 | 27.11 | 21.42 |
| | | 2 | 17.52 | 14.58 |
| | | 3 | 16.2 | 17.99 |
| | | 4 | 19.24 | 11.51 |
| Rosemaling2 (Fig. 6.6, row 2) | 5 | 1 | 15.44 | 12.84 |
| | | 2 | 21.22 | 24.71 |
| | | 3 | 25.72 | 19.95 |
| | | 4 | 24.98 | 21.36 |
| Rosemaling3 (Fig. 6.6, row 3) | 7 | 1 | 16.53 | 11.94 |
| | | 2 | 15.71 | 12.15 |
| | | 3 | 21.21 | 16.19 |
| | | 4 | 19.48 | 15.61 |
| | | 5 | 15.21 | 9.21 |
| | | 6 | 11.01 | 4.85 |
| Van gogh1 (Fig. 4.11 , row 1) | 5 | 1 | 10.18 | 28.11 |
| | | 2 | 12.27 | 14.11 |
| | | 3 | 22.34 | 13.41 |
| | | 4 | 11.85 | 8.65 |
| Van gogh2 (Fig. 7.1, row 2) | 4 | 1 | 15.89 | 14.25 |
| | | 2 | 19.54 | 15.49 |
| | | 3 | 17.21 | 12.98 |

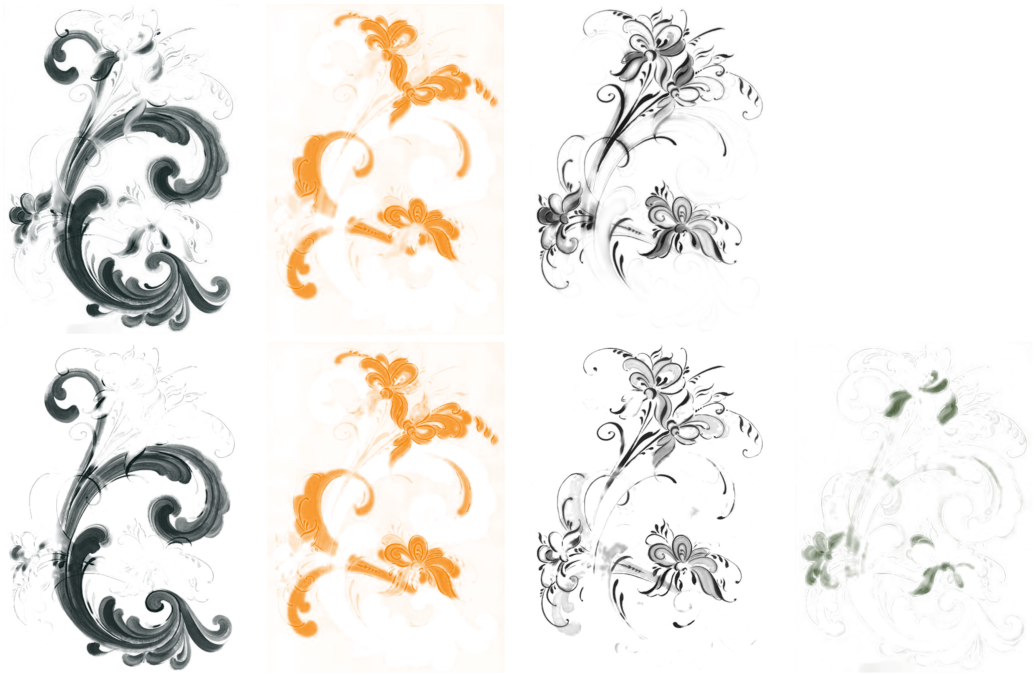


Fig. 4.5 Comparison of before and after adding a new layer. The upper row shows the decomposed layers, while the below row shows those layers after adding a new layer. Each column shows the decomposed layers with a corresponding palette colour. The last column of the below row shows the new layer.

as a new palette colour, and then recompute the opacity of each layer by solving Eq. 4.9. Fig. 4.5 shows the comparison of before and after adding a new layer. Furthermore, to remove the isolated regions, it can be achieved by adding the new layers in an iterative way. When no isolation region is detected, namely, no potential stroke is found, the iteration is stopped. Usually, after 1 or 2 iterations, there is no isolated region to appear in the new layer. This can be noted in Fig. 4.6, that is, after 1 iteration the isolated regions have a distinct change.

**Algorithm 1:** Iterative Scheme

**Input:** $N$ decomposed layers

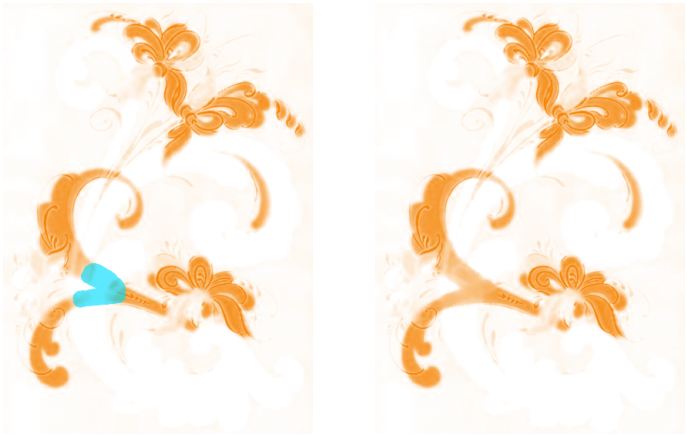
**Output:** $N+1$ decomposed layers

1 Binarize each decomposed layers;
2 Detect the isolated regions from the binarized layers;
3 Select average colour of the largest isolated region as new palette color $C_{n+1}$;
4 Re-compute the decomposition by solving Eq. 4.9 with $C_{n+1}$;
5 Stop the iteration when no isolation region is detected;

(a) (b) (c) (d)

Fig. 4.6 The masks of the shared and isolated regions when there are 4 and 5 layers respectively. (a) Shared regions (4 layers). (b) Shared regions (5 layers). (c)Isolated regions (4 layers). (d)Isolated regions (5 layers).

### 4.1.5 Brush Stroke Completion



(a) (b)

Fig. 4.7 Illustration of in-painting on a layer. (a) User annotated mask (blue region) on the in-painting area. (b) In-painted layer.

As shown in Fig. 4.5, some brush strokes may have other opaque brush strokes overlapped above, which bring about the gaps to break the brush strokes within a layer. Obviously, to make brush strokes complete and smooth, these gaps need to be filled. It is natural to involve user interventions, such as manual masks or sketches, which specify the gaps to be filled.

Once the overlapped regions are determined, the patch-based inpainting techniques [80] is employed here, that is, within one layer, the gap is specified by a mask, while the patches are extracted from the outside of the gap in all layers and are

utilized to create the patch dictionary. Then, the gap is filled through iteratively projecting each patch of the layer to its nearest neighbour in the dictionary. Fig. 4.7 shows that such patch-based inpainting methods can effectively deal with the scenario of a big gap. Since the inpainting method [80] requires user annotations, while the brush strokes extraction in [42] is automatic, for fair demonstration and comparison, other than the inpainting shown in Fig. 4.7, the inpainting technique is not involved for the other examples.

## 4.2 Extraction of Brush Strokes

Brush strokes normally follow certain rules in a brush painting and they vary depending on the style of the painting. Here, how to make use of such rules is discussed. Brush paintings, such as Rosmailing paintings, usually use subtle and vibrant colours to enhance colour contrast between overlapped strokes. As a result, the overlapped strokes tend to be classified into different layers. Extracting brush strokes within one layer is easier than directly from the input painting. The MSERs is employed which is proposed in [15] and [52] to extract brush strokes since it is invariant to affine intensity changes. MSERs algorithm requires a distinct difference between background and foreground while allowing a small variation of intensity within the selected stroke region. Usually, the strokes on the decomposed layers satisfy this requirement.

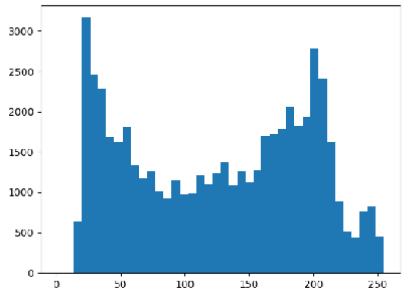
However, it is likely that MSERs may fail in segmentation with the following scenarios,

1. the brush stroke with the intensity very close to the background;
2. two adjacent brush stroke painted by different colours with the similar intensity;
3. overlapped brush strokes;
4. moreover, like the other existing segmentation approaches, the MSERs algorithm encounters over-segmentation issues as well.

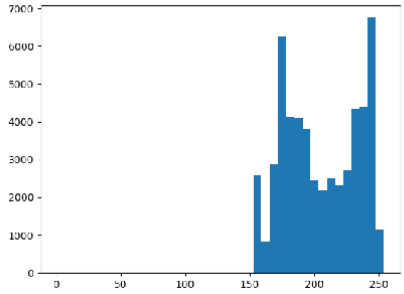
To tackle these challenges, the coherent lines [31] are introduced into MSERs, which both enhances the edges of strokes and preserves the completeness of strokes. For completeness sake, the MSERs algorithm is briefly addressed and then address the modifications.

(a) Opacity

(b) Intensity




(c) Histogram of alpha




(d) Histogram of intensity

Fig. 4.8 Comparison of the intensity (a) and opacity (b) of a layer with histograms (c)(d).

Intensity and opacity of one layer associated with the individual histograms are shown in Fig. 4.8. It can be noted that the opacity of the layer contains richer layered details than the intensity. Here, the layer intensity (represented as $L_I$) is based on the composition of its colour $L_{RGB}$, its opacity $\alpha_L$ , and the background colour $B_{RGB}$. Since the opacity of background is 1, based on Eq. 4.1, $L_I$ equals to $G(\alpha_L L_{RGB} - (1-\alpha_L)B_{RGB})$, in which $G(*)$ is the greyscale function.

(a) (b) (c)

Fig. 4.9 (a) Original image. (b) Intensity of image. (c) Original MSERs on the intensity of image.

### 4.2.1 MSERs Algorithm

MSERs can denote a set of distinguished regions that are detected in an intensity image. All of these regions are defined by an extremal property of the intensity function in the region and on its outer boundary, i.e. for a given extremal region S, the internal intensity is more than the intensity of boundary of S,

$$\forall p \in S, \forall q \in \partial S, \longrightarrow I(p) \geq I(q)$$

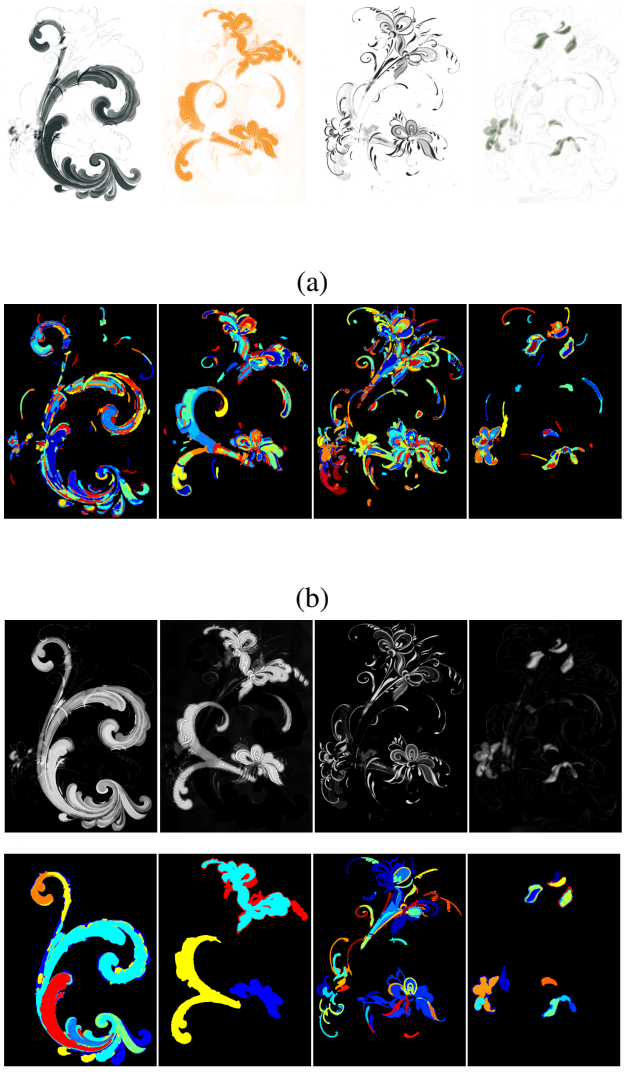
where $\partial S$ denotes the boundary of $S$. The extremal regions can be detected by changing threshold. With given intensity threshold $g$, all pixel with intensity larger than $g$ is black, otherwise is set to white. By changing threshold $g$, these black and white regions may further split or merge indicates the set of all extremal regions. The resulting extremal regions may be represented by the component tree. Accordingly, the change rate of the area of the extremal region is computed by

$$\gamma(S_i^g) = \frac{\left(\left|S_j^{g-\Delta}\right| - \left|S_k^{g+\Delta}\right|\right)}{|S_i^g|}$$

where $|.|$ denotes the cardinality, $S_i^g$ is the i-th region which is obtained by thresholding at an intensity value g and $\Delta$ is a stability range parameter. $g-\Delta$ and $g+\Delta$ are obtained by moving upward and downward respectively in the component tree from the region $S_i$ until a region with intensity value $g-\Delta$ or $g+\Delta$ is found.

$\{i, j, k\}$ are the indices of nodes of the component tree. MSERs correspond to those nodes of the component tree that have a stability value $\gamma$, which is a local minimum along the path to the root of the tree.

### 4.2.2 Modified MSERs Algorithm



(a)

(b)

(c)

Fig. 4.10 Comparison of segmentation results by the original MSERs and modified version. (a) The decomposed layers. (b) The original MSERs on the intensity of layers. (c) Opacity of layers, and MSER regions on four layers by the modified MSERs.

As shown in Fig. 4.9, in terms of the definition of the area change rate $\gamma$, MSERs may fail in segmentation with the following scenarios, (1) the region with the intensity most close to the background; (2) two adjacent regions in different colour but similar intensity; (3) the overlapped brush strokes.

Comparing with default MSERs algorithm use intensity of input, extraction in this work is performed on the opacity of generated layers. With successful layer decomposition, in each layer, the opacity of the background is close zero, which can be easily filtered out, see Fig. 4.10 . The opacity of the brush strokes is always independent of the colour, with successful layer decomposition, two adjacent or overlapped brush stroke painted by different colours would be separated into different layers, so, naturally, the proposed extraction is suitable for the second and third scenario, see Fig. 4.10. Moreover, the layer decomposition of Eq. 4.9 on a brush painting is performed, and the intensity and opacity of one layer associated with the individual histograms are shown in Fig. 4.8. It can be noted that the opacity of the layer contains richer layered details than the intensity. So, the first modification is to perform MSERs on the opacity of every layer.

Secondly, this work aim at the scenarios of two adjacent regions with the similar opacity. When the extremal region is growing up through changing threshold, it is feasible to restrict the region by introducing the coherent lines. According to the definition of the extremal region, the boundary of region S should satisfy,

$$\forall p \in S, \forall z \in \bar{S}, \forall q \in \partial S \longrightarrow I(p) \geq I(q) \text{ and } I(q) \leq I(z) \tag{4.10}$$

where $\bar{S}$ denotes the complement of $S$. The second modification is to simply modify the opacity of layers, that is, overlapping the coherent lines with the layer and then changing the opacity of coherent lines to the smallest value in the layer. To deal with the over-segmentation issue, the coherent lines play an important role. Given a region S, the area change rate $\gamma$ is modified as,

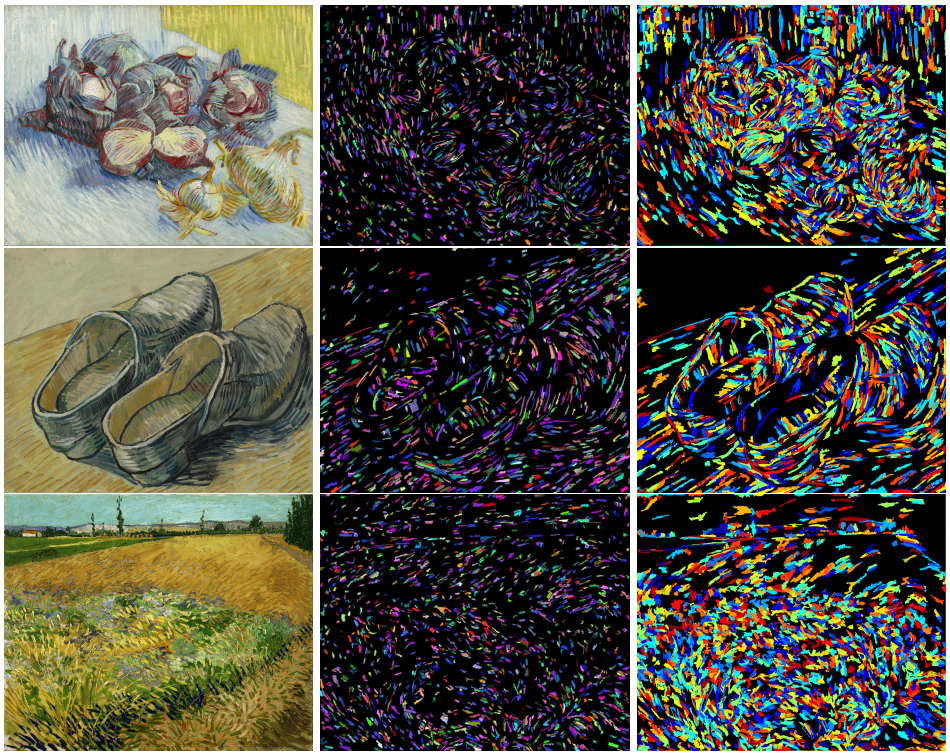
$$\gamma(S_i^g) = \left| \frac{S_j^{g-\Delta} - S_k^{g+\Delta}}{S_i^g} \right| + \left| \frac{Q_j^{g-\Delta} - Q_k^{g+\Delta}}{Q_i^g} \right| - \left| 1 - \frac{Q_i^g}{S_i^g} \right| \tag{4.11}$$

where $Q$ denotes the set of pixels which stay on the coherent lines and $Q \subset \partial S$. The third modification is to take into account the change of coherent lines to the

boundary of $S$, i.e. the third term penalizes that a small portion of the boundary $\partial S$ is occupied by coherent lines.

Fig. 4.10 shows the segmentation results by the modified MSERs, which correspond to brush strokes. It can be noted that performing MSERs on the intensity inevitable yields over-segmentation, as shown in Fig. 4.9 . Performing the modified MSERs on the opacity of layers, the strokes tend to be complete and smooth within one layer. Moreover, some small regions with the distinct opacity values against neighboring areas have been filtered out, and no region is selected from the background.

## 4.3 Result and Analysis



(a) (b) (c)

Fig. 4.11 (a) Input images (Painting ID from top to down: F374, F607, F411). (b) Brush strokes extracted by [42]. (c) Brush strokes extracted by **LStroke**.

To the best of the author's knowledge, there is lack of study on automatic brush strokes extraction. [42] presented a automatic brush stroke extraction method for van Gogh's paintings. The brush strokes extracted for some paintings are shown in Fig. 4.11. In this thesis, the van Gogh's painting ID (F-numbers) are given by the catalogue numbers in [71]. As shown in Fig. 4.11, the extracted brush strokes from the input paintings are indicated by different colours, and each brush stroke region is a set of pixel coordinates. To numerically evaluate their brush stroke extraction algorithm, they manually marked brush strokes on 10 selected regions of van Gogh's paintings, see Fig. 4.12.

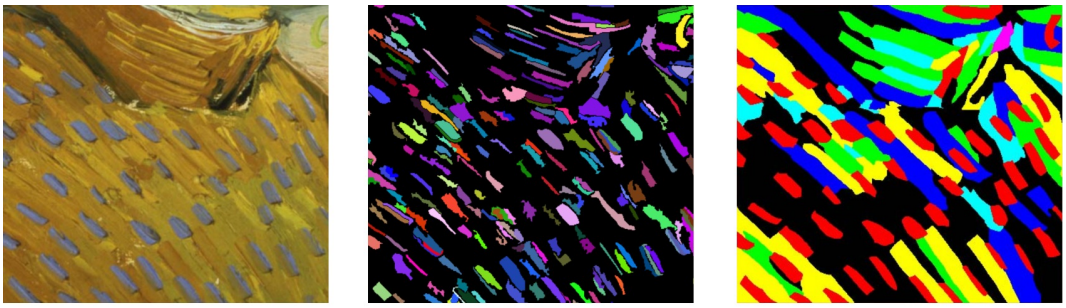
For the each input example region (Fig. 4.12a), they manually marked the brush strokes (Fig. 4.12b). Based on the manually marked examples, they define two parameters in order to evaluate the accuracy of extracted brush strokes and, therefore, can be used to compare between their method and this work: valid rate and detection rate.

For input painting regions, suppose there are $m$ manually marked brush strokes defined as $B_j^*$, $i = 1,...,m$, and $n$ brush strokes are detected by extraction algorithm represented by $B_i$, $i = 1,...,n$. $B_i$,$B_j^*$ are the set of pixel coordinates of brush strokes, and $B_i \cap B_j^*$ donates the overlapped pixels between the $j-th$ manually marked brush stroke and the $i-th$ detected brush stroke. $B_i$ is defined as valid covered once $\left|B_i \cap B_j^*\right| / |B_i| > 80\%$, which means more than 80% pixels of $B_i$ are in the overlapped region, and $C_{i,j}$ is set to 1 to indicate $B_i$ is valid covered by $B_j^*$, otherwise $C_{i,j} = 0$. $C_{i,\cdot} = \sum_{j=1}^{m} C_{i,j}$ which shows if the $B_i$ is valid covered by any manual brush stroke. If $C_{i,\cdot} = 1$, $B_i$ is defined as valid. On the other hand, $C_{\cdot,j} = \sum_{i=1}^{n} C_{i,j}$ which indicates how many extracted brush strokes are valid covered by manual brush stroke $B_j^*$. $B_j^*$ is defined as detected when $C_{\cdot,j} \geq 1$. If $C_{\cdot,j} \geq 1$, $D_{\cdot,j} = 1$, otherwise, $D_{\cdot,j} = 0$. Based on $C_{i,\cdot}$ and $C_{\cdot,j}$, valid rate and detection rate are used to assess the accuracy of extracted brush strokes.

Valid rate: $\sum_{i=1}^{n} C_{i,\cdot}/n$, the valid ratio of extracted brush strokes.

Detection rate: $\sum_{j=1}^{m} D_{\cdot,j}/m$, the ratio of detected manual brush strokes.

### 4.3.1 Compare LStroke with [42]'s method



(a) Input painting region (b) Brush stroke extracted by Li et al's method (c) Manually marked brush strokes

Fig. 4.12 Brush strokes extraction in [42] (Painting ID: F518)

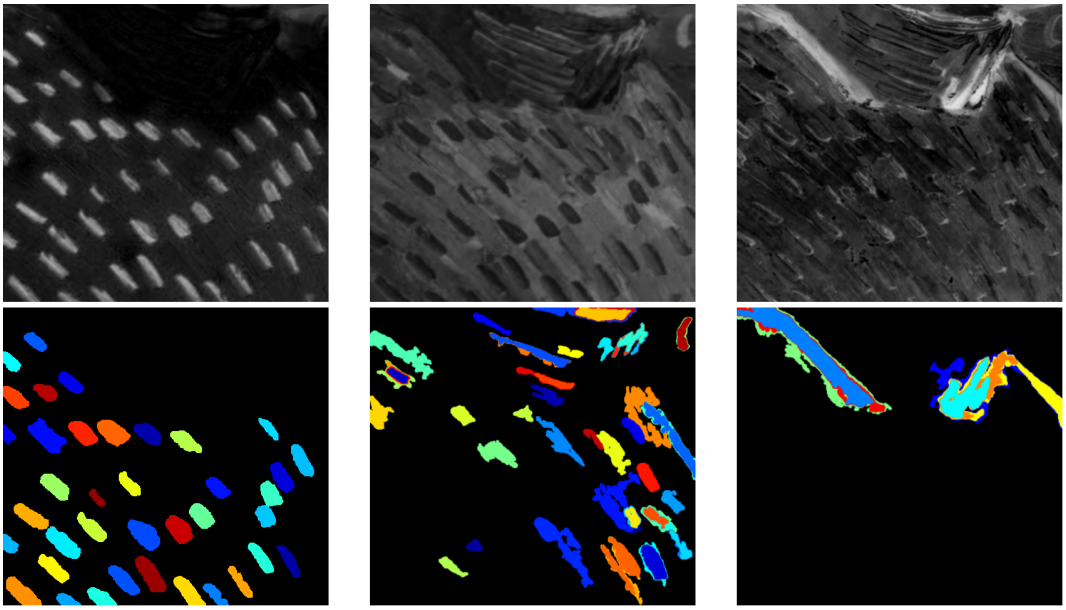


Fig. 4.13 Brush strokes extraction by **LStroke** (Painting ID: F518); the upper row shows the opacity maps of each layer

Even though **LStroke** is not specifically designed for van Gogh's paintings, for a fair comparison, brush strokes in 10 sample regions from the same collection of [42] are also manually marked by a professional artist from China Academy of Art. The manual marked brush strokes regarded as ground truth for evaluation, and they are marked individually on Mypaint[50]. Same as extracted brush strokes in [42], it is noteworthy that the extracted brush strokes by **LStroke** can also be represented as a set of pixel coordinates (see Fig. 4.13). And based on the manually marked brush, the valid rate and detection rate of brush strokes extracted

by **LStroke** are calculated. As shown in Fig. 4.14, in [42] the brush strokes are manually marked as ground truth for valid rate and detection rate.

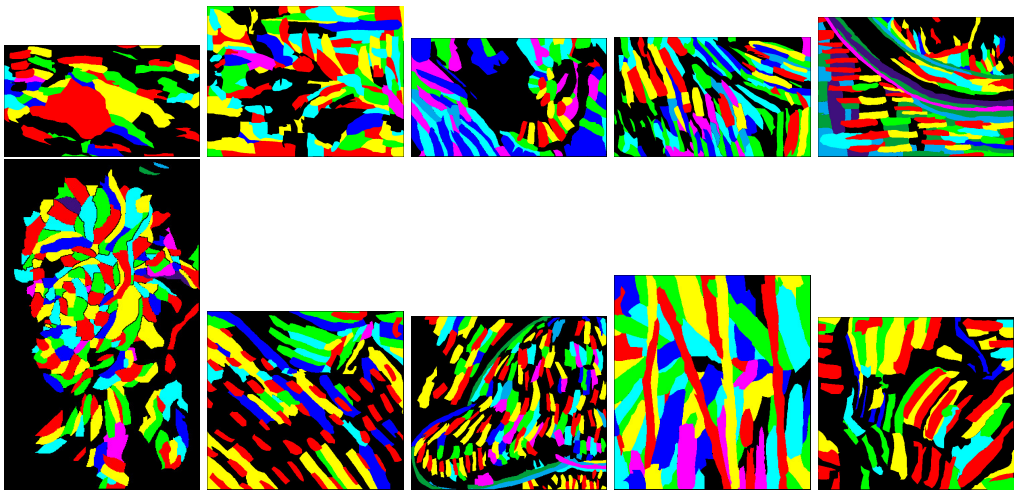


Fig. 4.14 Manually marked brush strokes by [42].

The average valid rate and average detection rate of **LStroke** are higher than the method in [42] and default MSERs algorithm, see Table 5.1 (in this thesis, the van Gogh's painting ID are given by the catalogue numbers in [71]).

As shown in Table 5.1, on average, **LStroke** extracts brush strokes with higher valid rate and detection rate. Additionally, compared to Li et al.'s method [42], **LStroke** is based on the alpha values from layer decomposition, it has better performance for the segmentation of overlapped brush strokes.

## 4.4 Summary

To generated relief model from brush strokes, brush strokes need to be firstly extracted. In this chapter, the layer decomposition based brush stroke extraction method — **LStroke** is introduced. The **LStroke** consists of two steps: layer decomposition and brush stroke extraction. In this chapter, how to extract brush strokes from decomposed layers by using modified Maximally Stable Extremal Regions (MSERs) algorithm is demonstrated. The default MSERs algorithm [15] and the modified algorithm are explained and their main characteristics compared.

With numerically evaluation, **LStroke** is compared with the most closely-related work [42]. As a result, the **LStroke** outperforms [42]'s in terms of valid rate and detection rate of brush stroke extraction. By successfully extract brush strokes, the proposed algorithm can also be used in image editing, see chapter 7.

# Chapter 5

# Deep Network Based Brush Stroke Soft Segmentation—DStroke

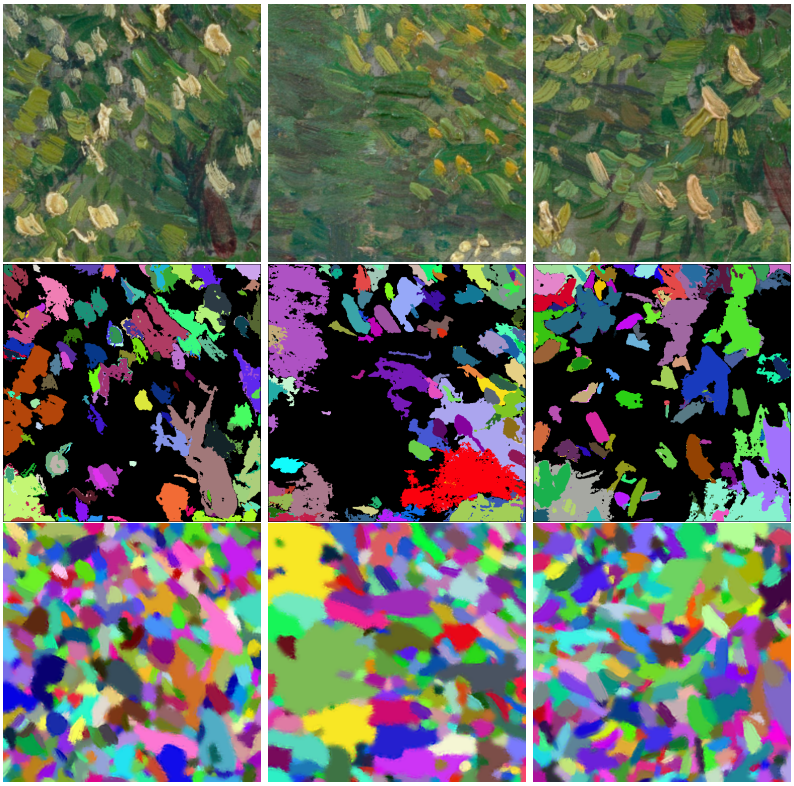


Fig. 5.1 Paintings with highly mixed brush strokes in similar colours (upper row); Extracted brush strokes by **LStroke**(Chapter 4) (second row). Extracted brush strokes by **DStroke** (bottom row).

In the previous method **LStroke** (Chapter 4), layer decomposition is used to split the brush strokes into different layers, so that each stroke can be easily segmented within one layer. However, the main challenge is that when the overlapped strokes share similar colours in some brush paintings, they would appear on the same layer and might be accidentally regarded as one brush stroke. Some brush paintings use subtle and vibrant colours to enhance the colour contrast between adjacent brush strokes. Applying palette colours to layer decomposition benefits the overlapped brush stroke segmentation in this scenario. However, some brush paintings do not emphasize the use of vibrant colours, or even the adjacent strokes may share the similar (or even the same) colours, the boundary of the brush strokes may be blurred, which results in the failure of stroke segmentation as shown in Fig. 5.1.

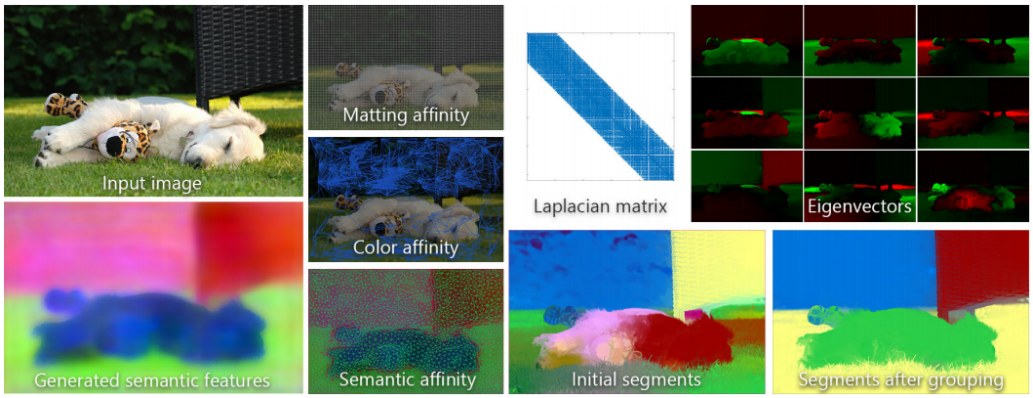



Fig. 5.2 Overview of the Semantic Soft Segmentation[3].

There have been many successful applications of deep neural networks in semantic segmentation. The recent work [3] firstly proposed the Semantic Soft Segmentation, which claims to successfully extract opacity of different semantic regions and determine the overlapped regions, as shown in Fig. 5.2.

The Semantic Soft Segmentation [3] was designed for accurate region selection (soft segmentation), in which a deep neural network is used for supplying semantic information. Moreover, it builds upon the matting Laplacian proposed in [39]. The matting Laplacian $L$ is matrix aiming to capture the affinity between each pair of neighbouring pixels in a small patch, normally, a 5 by 5 window. $\mathbf{a}$ is the vector of the layer alphas representing a soft segment, and calculated by solving the quadratic function $\mathbf{a}^T L\mathbf{a}$ with user constrains. As shown in the formulation, $L$

represents the Laplacian matrix, and how to construct the matting Laplacian plays an important role here. In [3], semantic information of image is generated by a deep neural network, and the matting Laplacian matrix is constructed according to the colour information and the semantic information, namely, the affinity of semantic information between pixels is also considered. After this, the spectral decomposition method [40] is applied for generating soft regions. The eigenvectors of matting Laplacian reveal semantic regions and soft transitions between them. The eigenvectors are regarded as a set of preliminary soft segments, and then grouped into 5 layers (a number set via empirical observations) get semantically meaningful soft segments.

Nevertheless, as mentioned in [3], the method suffers from least three limitations which fails in soft segmentation for strokes:

1. It only generates limited regions, and the number of resulting segmentation is fixed(5 by default), while a brush painting normally contains hundreds even thousands brush stroke, as shown in 5.11.

2. Even with high-level information from the deep network, [3] cannot separate different instance of the same class, since it cannot provide instance-aware semantic information. It is impossible to separate brush strokes since they belong to the same class.

3. The algorithm is not optimized for speed, the spectral decomposition of matting Laplacian is time-consuming. As mention in [3], their method takes 3 to 4 minutes for a 640x480 image.

Although deep learning techniques are leveraged to facilitate extracting soft regions from a image in [3], extracting brush stroke with deep neural networks is not straightforward and progression is limited due to the following issues:

1. There is no available brush stroke sample training dataset for the deep neural network at the moment. Labeling strokes by manual is a tedious task and time consuming very much. Usually, it cannot satisfy the big training dataset need for deep learning purpose.

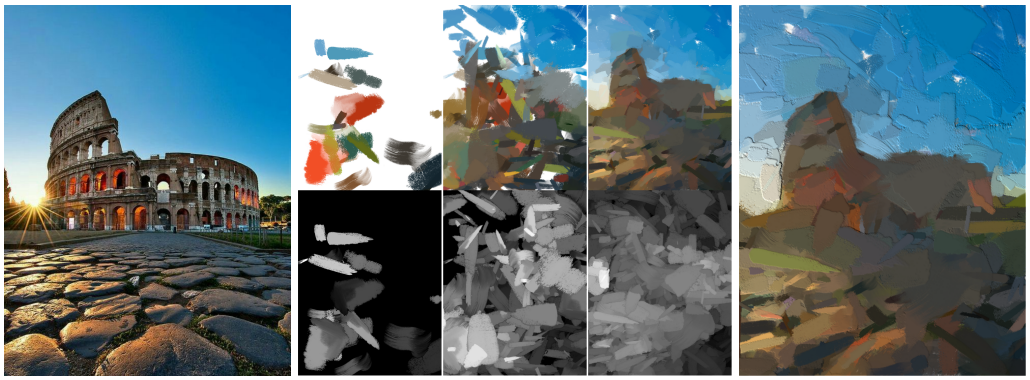
2. Similar to [3], in the scenario of brush stroke segmentation, how to deal with soft edges is a problem, which usually overlap with other regions in

terms of transparency. It is desired for segmentation to identify different brush stroke regions of the painting while also accurately representing the soft transition between them.

To tackle these challenges, a deep neural network based soft segmentation method—**DStroke** is proposed. The contributions include:

1. A big brush stroke training dataset can be automatically produced.
2. Instead of setting segmentation number via empirical observations, **DStroke** soft segments a painting into unlimited number of soft segments.
3. The **DStroke** generates instance-level soft segmentation results, namely, brush strokes.

## 5.1 Training Dataset Generation



(a) (b) (c)

Fig. 5.3 (a) Scene image. (b) Iterative stroke painting on canvas (upper row); height map of canvas (bottom row). (c) Illumination of canvas.

**Algorithm 2:** Dataset generation

**Input:** S: Sample image, SA: Stroke alpha map, ST: Stroke thickness

**Output:** C: Canvas, E: Edge map, H: Height map

$dis1 = \infty, H = 0$

$R = \frac{coveredregion}{canvas}$ // The ratio of painted region on canvas;

$threshold = rand(0.7, 1)$ // Random number between 0.7 and 1;

**while** $R <$ *threshold* **do**

  $dis1 = (S - C)^2$ ;

  $ROI = Position_{max}(dis1)$ // Choose the region of interest;

  $Color_{stroke} = avg(S[neighbor(ROI)])$ // Choose the average colour of the *ROI* neighbourhood on $S$ as stroke colour, which is a 3x3 window contains the *ROI* and its surrounding pixels;

  $C' = Compose(C, Stroke)$ // Paint the stroke on Canvas. The stroke image is firstly rotated to the vertical angle of the ROI gradient and then centre the stroke image is aligned to the ROI;

  $dis2 = (S - C')^2$ ;

  **if** $dis2 < dis1$ **then**

    keep the stroke ;

    save the stroke edge on E ;

    remove the overlapped edge on E ;

    $ST = SA$ // Thickness of stroke $ST$ is set equal to $SA$ ;

    $H[S] = Avg(H[S]), H = H + ST$ // update of Height map. $S$ is the stroke region. $H[S]$ represents the heights in stroke region, and $Avg(H[S])$ is the average height of region $S$. ;

    R= Size of painted region/ Size of canvas;

  **else**

    remove the stroke ;

  **end**

**end**

Rendering and Data augmentation.

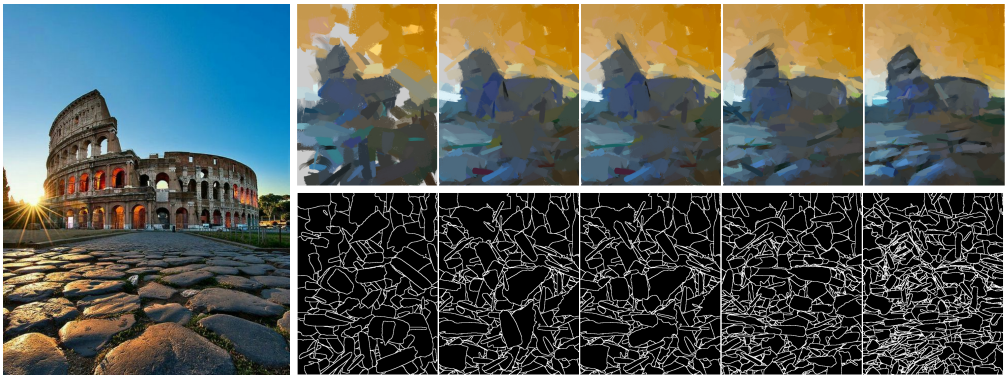
To produce a brush stroke sample training dataset for deep learning purpose, every input painting must contains the information of brush strokes. Moreover, the real painting usually have the individual height maps since pigment associated with each brush stroke on a canvas may has its own thickness. For reality purpose, the produced brush stroke samples should maintain illumination effect of the painting.

To this end, the following algorithm is proposed to generate paintings from images through mimicking brush strokes, see Fig. 5.3.

In the implementation of **DStroke**, firstly, the canvas $C$ is initialized as a blank white image, then the region of interest (ROI) represents the single pixel with the largest difference between scene image and the canvas ( as shown in Algorithm 2, $ROI = Position_{max}(dis1)$, in which $Position_{max}(*)$ returns the position with maximum value). The gradient of a pixel is a vector of its partials. Here, the Sobel operator [64] performs the gradient measurement on the greyscaled input image, and the Sobel operator consists of a pair of 3×3 convolution kernels. Secondly, based on the stroke library in *kylebrush* [38], a stroke is randomly picked up to put on canvas. It is worth noting that each brush stroke is represented as a rectangle image (Fig. 5.6) in the stroke library. The stroke image is firstly rotated to the vertical angle of the ROI gradient and then center the stroke image is aligned to the ROI. After that, the new ROI is detected over the updated image and is overlapped by a new stroke from the stroke library until the original image is mostly covered by strokes. This method does not aim to minimum the difference between scene image and canvas, which is time-consuming. Instead, paintings are generated once a certain percentage of the canvas is covered with strokes. For details, refer to **Algorithm 2**.

To simulate the physical appearance of a brush painting under different lighting circumstances, the thickness of the generated paintings are calculated. Here, the height map $H$ represents the per-pixel thickness of the painting, as shown in Fig. 5.3. Initially, the height map $H$ is set as zero. The thickness of a stroke is set equal to its alpha map ( $ST = SA$ ), and the height map is cumulative and updated each time a stroke is placed on the canvas. Firstly, the height of new stroke region is set to the its average height ( $H[S] = Avg(H[S])$, and $Avg(*)$ returns the average value) and then the height map is added with the new stroke thickness ( $H = H + ST$ ). When the painting and its thickness are completed, this method renders the final image. The normal of pixels on the canvas is calculated by the directional derivative of $H$. The illumination of each pixel is then calculated under the different illumination models.

The edge map $E$ is also updated as shown in Fig. 5.4. The edge map shows the stroke edges. Once a stroke is painted on the canvas $C$, the edge of the stroke is kept on $E$. Since the new strokes will cover the old strokes which makes the

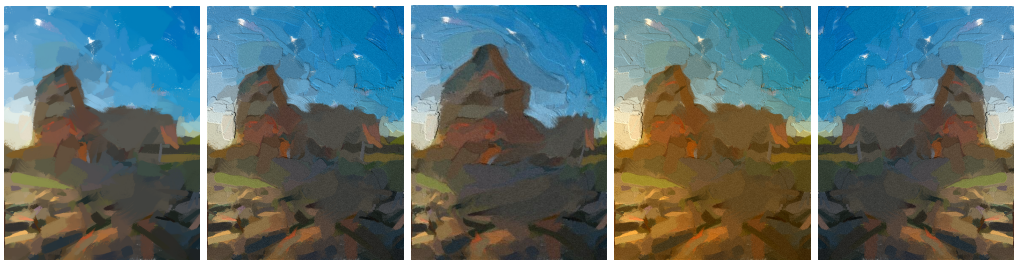


(a) (b)

Fig. 5.4 (a) Scene image. (b) Iterative stroke painting on canvas (upper row); edges of brush strokes (bottom row).

edges of the old strokes very difficult to see, the overlapped edges are removed on $E$. Eventually, each painting corresponds to an edge map that together forms the training data.

The *kylebrush* [38] contains brush strokes with various size and they are made by professional artists. On the other hand, based on the experiments, over-adjusting the stroke size will lead to a reduction in stroke extraction accuracy, so the strokes are not resized in the Algorithm 2.



(a) (b) (c) (d) (e)

Fig. 5.5 Input image and its augmentations. (a) Input image. (b) Adding noises. (c) Distortion. (d) Changing hue. (e) Flipping.

In some cases, the target application for a deep neural network exists in various conditions. However, the training data usually cannot sufficiently represent these various conditions. Data augmentation is a technique commonly used for such a scenario. It usually involves creating additional synthetically modified images by transforming (rotating, panning, scaling, adding some noise).

To further increase the variety of training dataset, as shown Fig. 5.5, data augmentation is applied: the paintings are randomly flipped, rotated, and slightly distorted; noises (Gaussian noise, salt and pepper noise) are added to the paintings; the hue is also changed randomly to shift the colour of paintings.

The input sample image for Algorithm 2 contains 1000 images downloaded from open source image library *stocksnap* [66] which contains various scene images created by artists and photographer (e.g. still life, animals, architecture, natural scenery). 1000 corresponding paintings are created based on the downloaded images, and then 4000 more augmented images are generated from the 1000 paintings.

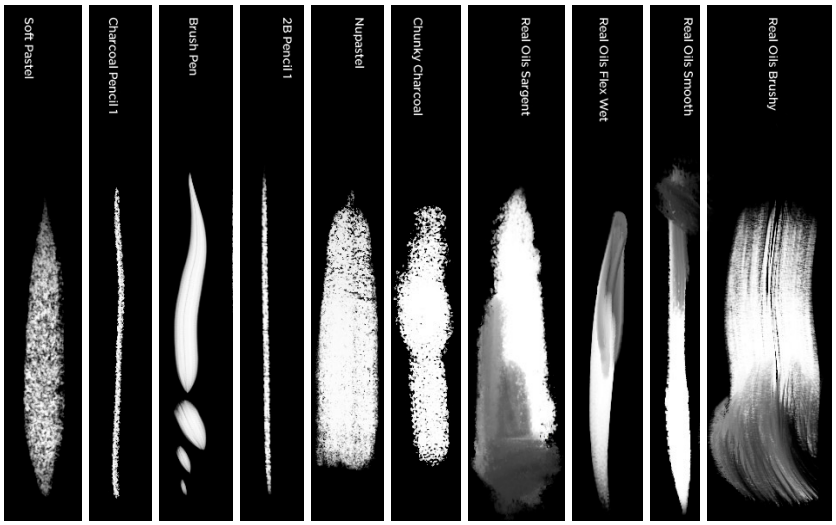



Fig. 5.6 Different alpha maps of brush strokes from *kylebrush* [38].

Compared to painterly rendering methods such as [78] [17] [24] [21] [23], there are three distinct differences in **DStroke**:

1. **DStroke** can better avoid over-fitting issue in training. This is because many stroke types are utilized for training instead of only one or a few of types, e.g. 520 (number of brush strokes types in the *kylebrush* library [38]) stroke types in the implementation. And data augmentation is also applied to increase the variety of data.

2. A weak similarity measurement is applied to simplify the computation, which drastically decreases computational complexity.

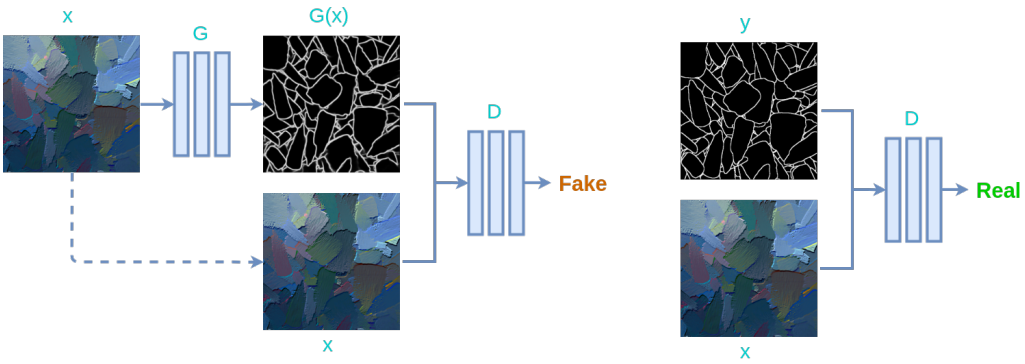
3. Based on the height maps, the output paintings are rendered under the different light circumstance to further avoid overfitting.

## 5.2 Network Structure

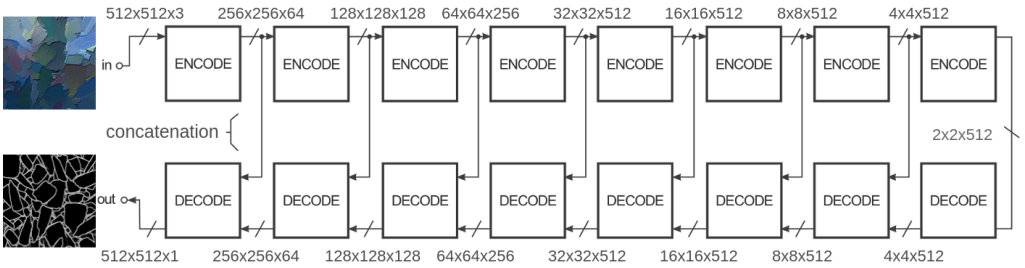
The **DStroke** is based on deep neural network. This section begins with a brief introduction to the network structure and then discusses its implementation.

The network of brush stroke extraction is based on the work of Pix2Pix [30]. The network consists of two parts: the generator $G$ that generates edges of brush strokes from input painting, and discriminator $D$ that classifies the ground truth with the generator's output, see Fig. 5.7.

The Generator is a U-Net architecture derived from the Fully Convolutional Network but is modified to ensure that it produces faster and more accurate segmentation. In general, the concatenation part of the U-Net encodes the input image into feature representations at multiple different levels while the expansion part propagates the contextual information to higher resolution layer which is necessary to predict a good segmentation map.




(a) The generator and discriminator




(b) Generator structure

Fig. 5.7 The Pix2Pix network

As shown in Fig. 5.7, the input of the generator $G$ are the painting images, output are the brush strokes edges. Let and $CD_k$ denote a Convolution-BatchNormDropout-ReLU layers with $k$ filters, $C_k$ represent a Convolution-BatchNorm-ReLU layer. The generator $G$ adopts 8 Convolution-BatchNorm-ReLU layers in the encoder, and all convolutions are 4x4 Convolution Layers with stride 2. The feature map numbers double at first 4 layers, starting with 64 feature maps for the first layer, 128 for the second, and so on. The architecture of encoder of $G$ can be represented as $C64 - C128 - C256 - C512 - C512 - C512 - C512 - C512$. Similarly, the decoder of $G$ is composed of 8 Convolution-BatchNorm-Dropout-ReLU layers. Each layer is concatenated with the corresponding downsampling feature map. Convolutions in the encoder of $G$ downsample by a factor of 2, whereas they upsample by 2 in the decoder. The decoder of $G$ consists of : $CD512 - CD1024 - CD1024 - C1024 - C1024 - C512 - C256 - C128$.

The discriminator $D$ adopts 4 Convolution-BatchNorm-ReLU layers, and the feature map number doubles starting with 64 maps for the first layer, 128 for the second, and so on. To map the last layer to a 1-dimensional output, a convolution

is applied here with a Sigmoid function. The architecture of discriminator $D$ is $C64-C128-C256-C512$.

The generator $G$ is trained to produce edges of strokes, while the discriminator, $D$, is trained to do distinguish the generator's outputs and the groundtruth [30]. In other words, objective of Pix2Pix network can be expressed as follows:

$$\mathscr{L}_{cGAN}(G,D) = E_{x,y}[logD(x,y)] + E_{x,z}[(1 - logD(x,G(x,z)))] \tag{5.1}$$

As shown in Fig. 5.7, the network should be trained to map painting $\rightarrow$ edge. Therefore, dataset $x,y$ consists of painting image $x$ and edges $y$ which is considered as ground truth, and as an another input to the generator, $z$ is Gaussian noise. Given the image $x$, the generator, $G$ tries to output the edge of painting $x$, $G(x)$ in order to fool the discriminator. The L1 loss function of Generator is defined as:

$$\mathscr{L}L_1(G) = E_{x,y,z} \|y - G(x,z)\|_1 \tag{5.2}$$

On the other hand, the discriminator, $D$, learns to distinguish between fake edge image, $G(x)$ , generated by the generator and the real image $x$:

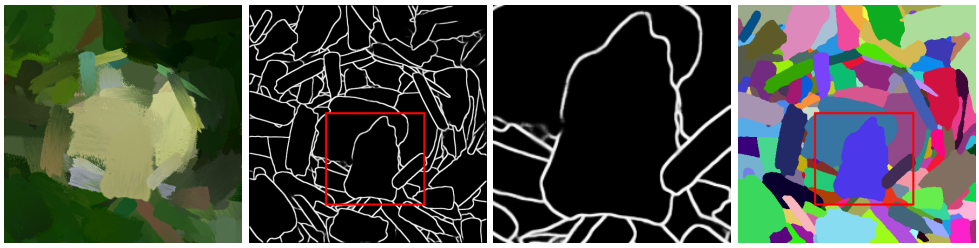
$$D(x,G(x)) = 0(fake), \quad D(x,y) = 1(real) \tag{5.3}$$

The idea here is that the generator needs to fool the discriminator in classifying fake edges as real but also it has to ensure that the output produced is near the ground truth output. The job of the discriminator does not change, that is, it just has to distinguish between real and fake edges. Where the generator $G$ tries to minimize this objective while the discriminator $D$ that tries to maximize it. As such, the final objective of the network is:

$$G^* = argmin_G max_D \mathscr{L}_{cGAN}(G,D) + \lambda \mathscr{L}_{L_1}(G) \tag{5.4}$$

In the implementation of **DStroke**, the 5000 training images generated from Section 5.1 are trained for 200 epochs, batch size 1, with random mirroring.

## 5.3 Soft Segmentation



(a) Input image (b) Edge detected (c) A small gap (d) Segmentation

Fig. 5.8 An example of blurred edges and gaps within stroke

In some cases, the edges obtained may have few small gaps due to complexity of brush stroke, as shown in Fig. 5.8. Therefore, trapped-ball segmentation [85] is applied here for stable extraction of brush strokes, as shown in Fig. 5.8d.

The guided filter has proved to be a good approximation of solution for matting Laplacian matrix [20]. To extracted the fuzzy transition between strokes and their transparency, the guided filter is employed here. The guided filter is very suitable for this task, i.e. brush stroke extraction, since the trimap is available. The hard segmentation regions can be regarded as the trimaps $p$ and assigned to every stroke accordingly, as shown in Fig 5.9. The other advantage is that guided filter is implemented in linear time.

More specifically, the cost function to be minimized to find the unknown alpha map of each stroke as follows:

$$E(q) = (q-p)^T \Lambda (q-p) + q^T L p$$

where, $N$ is the pixel number of input image $I$ (see 5.9a), $q$ is the $N \times 1$ unknown alpha map vector, $p$ is the constraint (e.g., a binary mask) as shown in 5.9c, $L$ is an $N \times N$ matting Laplacian matrix, and $\Lambda$ is a diagonal matrix encoded with the weights of the constraints.

To optimize the above-mentioned cost function, the linear system has to be solved:

$$(L+\Lambda)q = \Lambda p$$

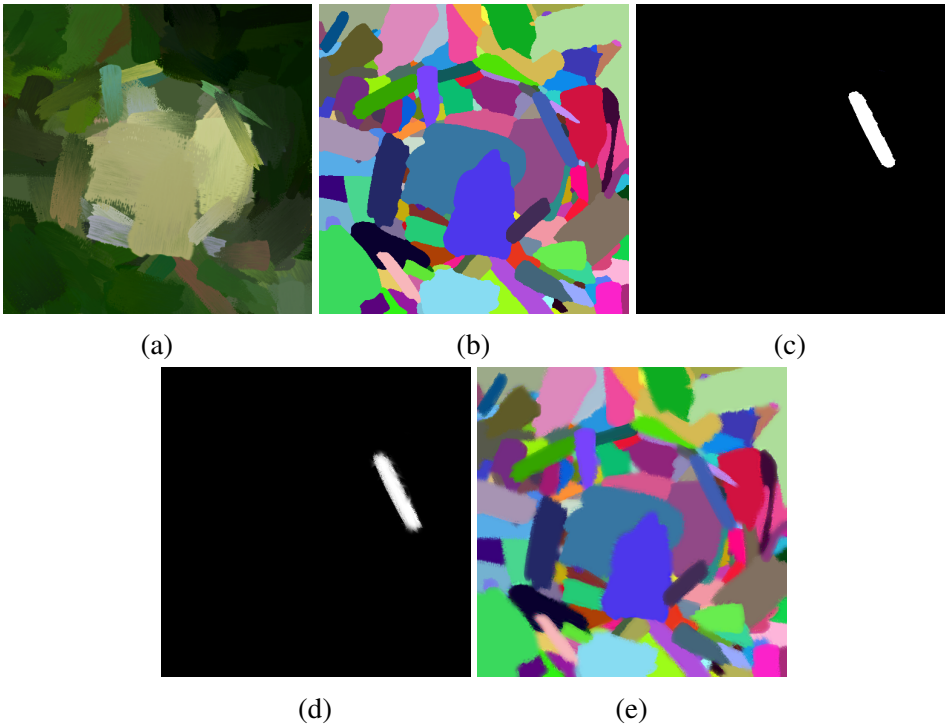


Fig. 5.9 An example of soft segmentation. (a) Input image. (b) Hard segmentation. (c) A segmented brush stroke mask. (d) Opacity of the brush stroke. (e) Soft segmentation.

The elements of the matting Laplacian matrix are given in [39]:

$$L_{i,j} = \sum_{k:(i,j)\in\omega_k} (\delta_{ij} - \frac{1}{\omega}(1 + \frac{(I_i - \mu_k)(I_j - \mu_k)}{\sigma_k^2 + \varepsilon}))$$

where the mean and variance of all pixel $i$ in a local window $w_k$ of image $I$ are represents as $\mu_k$ and $\sigma$, and $\varepsilon$ is a regularization parameter, which can be directly obtained by the guided filter kernel:

$$L_{ij} = |\omega| (\delta_{ij} - W_{ij})$$

where

$$W_{ij}(I) = \frac{1}{|\omega|^2} \sum_{k:(i,j)\in\omega_k} (1 + \frac{(I_i - \mu_k)(I_j - \mu_k)}{\sigma_k^2 + \varepsilon})$$

It has been proven [20] that $q_i$ can be re-written as a weighted sum as shown below:

$$q_i \approx W_{ij}(I)p_j \tag{5.5}$$

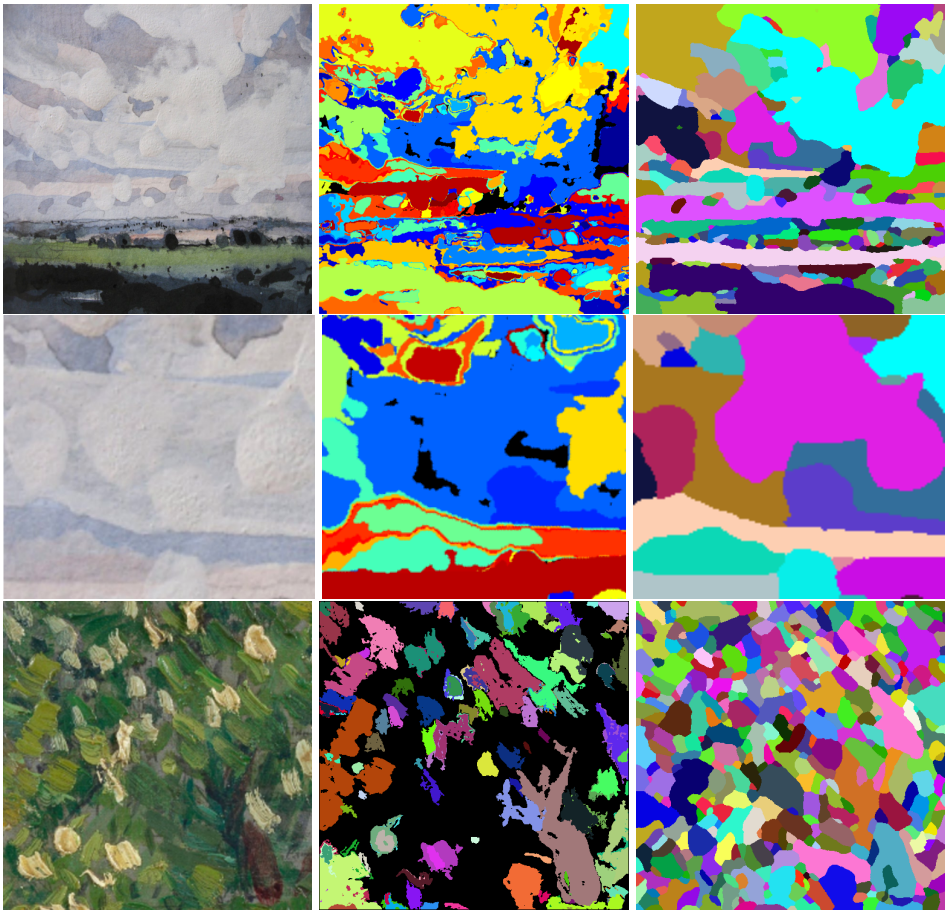
As such, the guided filter can be used to perform soft segmentation which is beneficial for this study as the guided filter is very suitable for this task, i.e. brush stroke extraction, since the trimap is available. The hard segmentation from deep network is regarded as the trimap $p$ and assigned to every stroke accordingly. The other advantage is that guided filter is implemented in linear time. This approach is more appropriate for this work because it would have been complicated to assign a trimap for each brush stroke to be extracted if the traditional matting method is used. Instead, when using the guided filter, binary masks of each stroke are used as trimaps $p$ (Fig. 5.9b) for guided filter to solve the fuzzy boundaries (Fig. 5.9d), namely, $q$ in Eq. 5.5. Here, to visualize the soft segmentation result, just like the [3], different colours are assigned to the alpha map of different strokes.

## 5.4 Result and Analysis

In this work, 5000 paintings are created associated with the individual set of whole strokes as the training dataset. To avoid over-fitting, several training strategies are applied here. First, the painting and edge are randomly flipped. Second, the pairs(painting, edge) are randomly cropped into different size and resized into 512*512. It takes less than two hours of training on a single GTX 1080 GPU.

In this section, **DStroke** is compared against several existing alternatives, and their implementations are obtained or reproduced. Specially, the stroke automatic extraction of stroke extraction of **LStroke** in Section 5.4.1 , and the Semantic Soft Segmentation[3] in Section 5.4.2 are experimented.

### 5.4.1 Compare DStroke with LStroke



(a) (b) (c)

Fig. 5.10 Highly mixed brush strokes in similar colours. (a) Input image. (b) Brush stroke extraction based on **LStroke** (Chapter 4). (c) Brush stroke extraction based on **DStroke**.

At first, to compare the **DStroke** method with the **LStroke**, the same 10 regions from van Gogh's paintings (see section 4.3) are also evaluated by **DStroke**, as shown in Table 5.1. As shown in Table 5.1, on average, **DStroke** extracts brush strokes with higher valid rate and detection rate than **LStroke**.

However,the **DStroke** is not specifically design for van Gogh's paintings. To better evaluate the proposed **DStroke** method, 11 more brush paintings are selected, including acrylic paintings, watercolor paintings, oil paintings. More specifically, to highlight the robustness of **Dstroke**, paintings are carefully picked, in which

Table 5.1 Evaluation and Comparison based on van Gogh's paintings

| Painting ID | Valid Rate (%) | | | | Detection Rate (%) | | | |
|---|---|---|---|---|---|---|---|---|
| | Li et al.'s method | Default MSERs | LStroke | DStroke | Li et al.'s method | Default MSERs | LStroke | DStroke |
| F218 (Glass with yellow roses) | 42.7 | 79.8 | 88.1 | 90.5 | 21.6 | 33.7 | 48.5 | 60.2 |
| F248a (Vase with Gladioli) | 75.4 | 79.4 | 90.2 | 93.4 | 78.8 | 82.1 | 88.1 | 92.2 |
| F297 (Skull) | 57.9 | 67.2 | 75.3 | 78.6 | 52.0 | 61.0 | 75.1 | 84.5 |
| F374 (Red Cabbages and Onions) | 58.2 | 67.9 | 75.9 | 81.2 | 63.2 | 60.8 | 82.4 | 84.8 |
| F386 (Still live with potatoes in yellow bowl) | 73.7 | 64.1 | 82.4 | 85.7 | 68.4 | 59.2 | 90.2 | 92.4 |
| F415 (Seascape near Les Saintes-Maries-de-la-Mer) | 46.9 | 50.3 | 58.1 | 65.2 | 60.0 | 80.1 | 92.1 | 92.8 |
| F518 (The little Arlesienne) | 60.7 | 55.7 | 71.2 | 78.2 | 75.2 | 86.1 | 78.9 | 85.2 |
| F538 (Portrait of Camile Roulin) | 49.0 | 58.4 | 84.2 | 88.7 | 44.9 | 61.4 | 81.3 | 83.6 |
| F572 (Willous at Sunset) | 83.9 | 68.9 | 82.4 | 89.5 | 65.6 | 71.2 | 87.0 | 89.7 |
| F652 (Pine with female figure in Sunset) | 50.0 | 60.1 | 75.8 | 76.5 | 72.5 | 60.1 | 67.3 | 81.1 |
| Average | 59.8 | 65.2 | 78.4 | 82.8 | 60.2 | 65.6 | 77.1 | 84.7 |

brush strokes in similar colours are heavily employed, as shown in Fig. 5.10. The proposed method is preformed on these complex paintings: due to the brush strokes in similar colours and the complexity of the paintings, some brush strokes are undetected in **LStroke** (the undetected regions are labelled in black). On the other hand, the segmentation of **DStroke** can extract the adjacent brush strokes in similar colours.

To numerically evaluate **DStroke**, same as **LStroke** and [42], the ground truth are also created through manually marking brush strokes by the professional artist. As shown in Table 5.2, the valid rate and detection rate of **DStroke** is noticeably higher than **LStroke**.

To better exam **DStroke**, the Intersection over Union(IoU) and pixel accuracy metrics are also used for evaluation. The IOU indicates the percent overlap ratio between the target mask(manual masked stroke) and the prediction mask(detected strokes). The mean of Intersection over Unions(meanIoU) are evaluated and compared here.

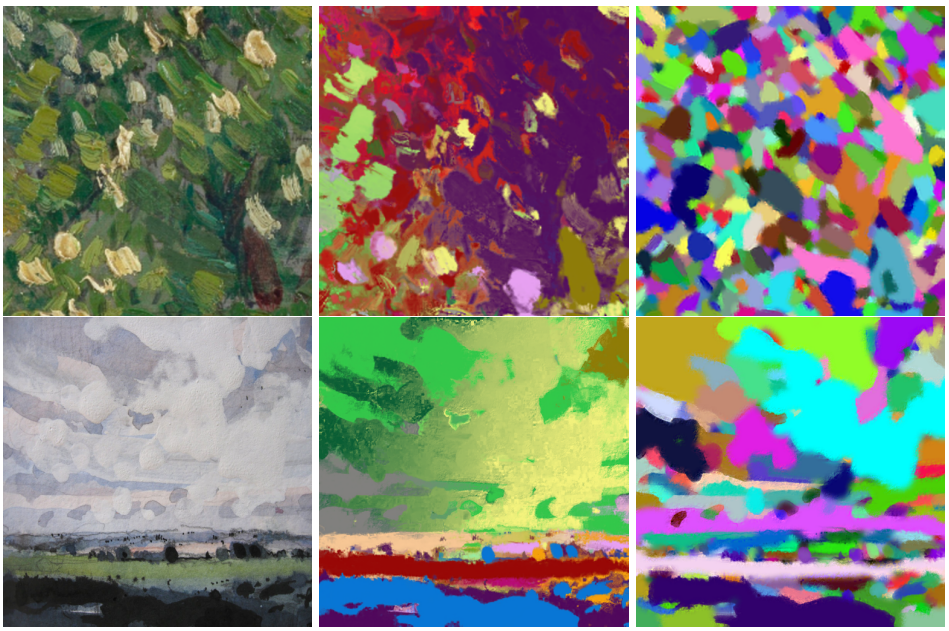
$$IoU = \frac{Target \bigcap Prediction}{Target \bigcup Prediction} \tag{5.6}$$

As shown in Table 5.2, the **DStroke** also outperforms in meanIoU.

Table 5.2 Evaluation and Comparison of **DStroke**

| Painting ID | LStroke | | | DStroke Method | | |
|---|---|---|---|---|---|---|
| | meanIoU(%) | Valid Rate(%) | Detection Rate(%) | meanIoU(%) | Valid Rate(%) | Detection Rate(%) |
| Mixed strokes1 (Fig. 5.14, row 1, oil painting) | 35.01 | 36.15 | 11.11 | 80.42 | 77.00 | 82.17 |
| Beach (Fig. 5.14, row 2, acrylic painting) | 33.25 | 40.91 | 8.37 | 82.11 | 80.92 | 87.07 |
| Mixed strokes2 (Fig. 5.14, row 3, oil painting) | 28.30 | 46.88 | 17.36 | 78.16 | 74.52 | 80.99 |
| Mixed strokes3 (Fig. 5.14, row 4, oil painting) | 32.88 | 39.06 | 14.08 | 81.99 | 79.27 | 82.99 |
| Cloud (Fig. 5.14, row 5, acrylic painting) | 31.86 | 38.64 | 12.86 | 86.66 | 88.43 | 90.95 |
| Dusk1 (Fig. 5.14, row 6, watercolor painting) | 35.39 | 40.54 | 12.84 | 76.13 | 74.81 | 89.91 |
| Dusk2 (Fig. 5.14, row 7, watercolor painting) | 22.56 | 38.71 | 12.12 | 65.68 | 58.97 | 69.70 |
| Trees (Fig. 5.14, row 8, acrylic painting) | 33.19 | 48.72 | 21.09 | 76.72 | 74.47 | 82.03 |
| Mixed strokes4 (Fig. 5.14, row 9, oil painting) | 35.23 | 34.13 | 11.37 | 76.04 | 70.26 | 77.84 |
| Road (Fig. 5.14, row 10, oil painting) | 26.79 | 46.15 | 19.59 | 76.69 | 74.44 | 85.57 |
| Landscape (Fig. 5.14, row 11, watercolor painting) | 27.53 | 35.00 | 11.79 | 79.22 | 78.00 | 88.97 |

### 5.4.2 Compare DStroke with [3]'s method

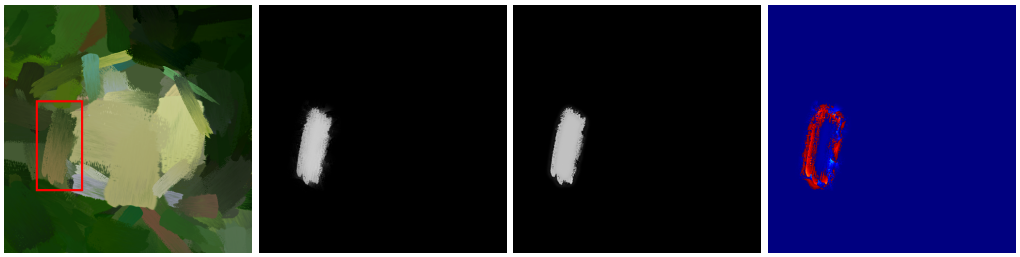


(a) (b) (c)

Fig. 5.11 (a) Input image. (b) The result of [3]'s method. (c) The result of **DStroke**.

The **DStroke** is also compared with the most closely-related work: Semantic Soft Matting by Aksoy et al. [3].

**DStroke** also apply deep learning for pre-segmentation, but **DStroke** does not require the pre-set segment number. Moreover, **DStroke** focus on instance-level segmentation instead of semantic segmentation. On the other hand, the linear complexity filter is applied, i.e. guided filter[20], to deal with the transparency of segments.



(a) (b) (c) (d)

Fig. 5.12 Compare the alpha map of a segmented stroke with groundtruth. (a) Input image. (b) Alpha map of a segmented stroke. (c) Alpha map of the groundtruth. (d) Difference map between prediction and the groundtruth.

To numerically evaluate **DStroke**, the generated brush paintings in dataset are used as groundtruth information. Here, the accuracy and running time of each method are tested. More specifically, 500 images with different image size (from 100 pixels to 1 million pixels) are examined. For accuracy, each soft segmentation are examined with the dataset by MSE(mean square loss), which is given as follows:

$$MSE = \frac{1}{M}\sum_{i=j}^{M}(\frac{1}{N}\sum_{i=1}^{N}(\alpha_{ij} - \hat{\alpha}_{ij})^2) \tag{5.7}$$

where, $\alpha_{ij}$ is the opacity value of $i$-th pixel on $j$-th segmented stroke, $\hat{\alpha}_{ij}$ is the corresponding groundtruth of $\alpha_{ij}$, $N$ is the stroke size, and $M$ is the number of all segmented strokes. Similarly, running time are also evaluated on the same 500 images. It can be seen that the proposed **DStroke** outperforms [3]'s method in both running time and accuracy, as shown in Fig. 5.14 and Fig. 5.13.

The main reasons why **DStroke** outperforms [3]'s method on accuracy is that [3]'s method is based on spectral matting [40] which only supports a limited number of segmented regions. By default, [3]'s method only output 5 regions, while a

brush painting normally contains hundreds of brushstrokes. On the other hand, the **DStroke** enables segmentation of thousands of strokes which makes it more accurate.

**DStroke** also outperforms [3]'s method on running time. [3]'s method uses the sparse eigendecomposition of the constructed Laplacian matrix to generate the soft segmentations. The sparse eigendecomposition is time-consuming [40], and it grows linearly with the number of pixels. On the other hand, **DStroke** is based on guided filter [20], which is much faster and implemented in linear time.

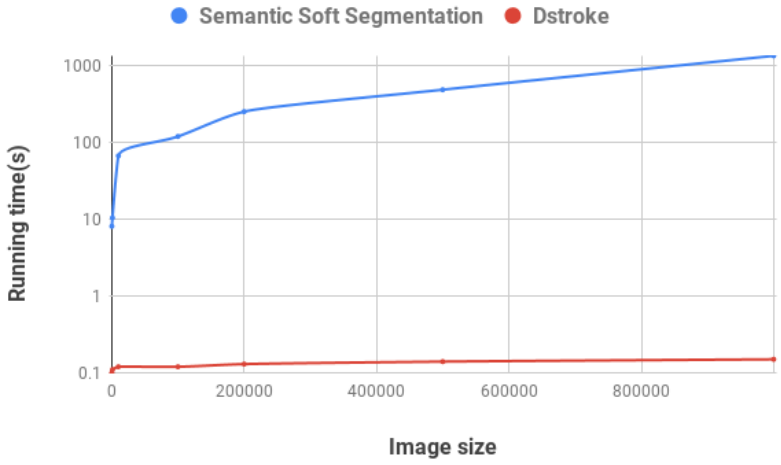



Fig. 5.13 Comparison of Running time in different image sizes


Fig. 5.14 Comparison of MSE in different image sizes

In summary, comparing to [3]'s method, the **DStroke** method has better performance in the following scenarios:

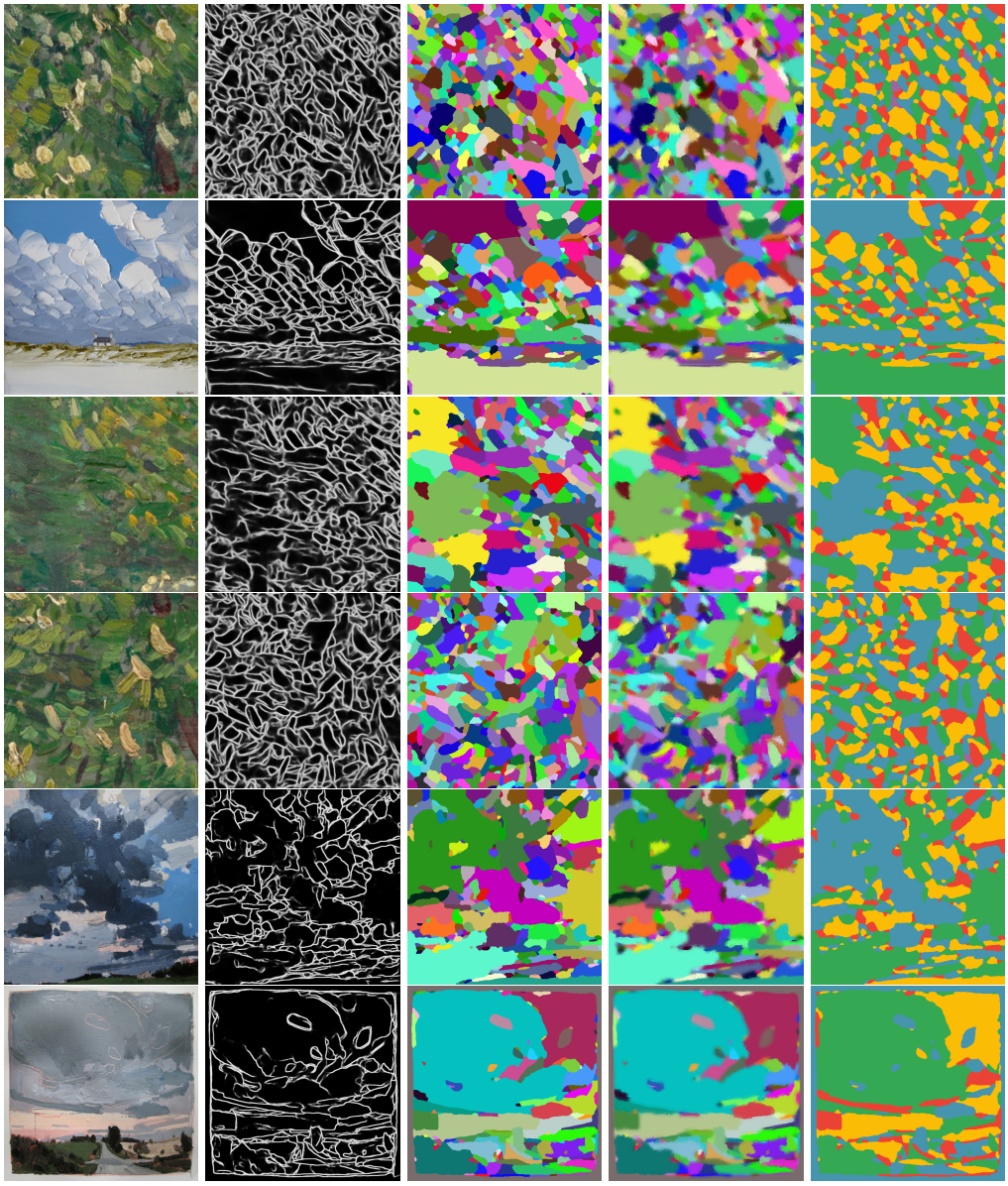
1. Comparing to [3]'s method, **DStroke** is able to generate instance-level soft segmentation results, as shown Fig. 5.11.

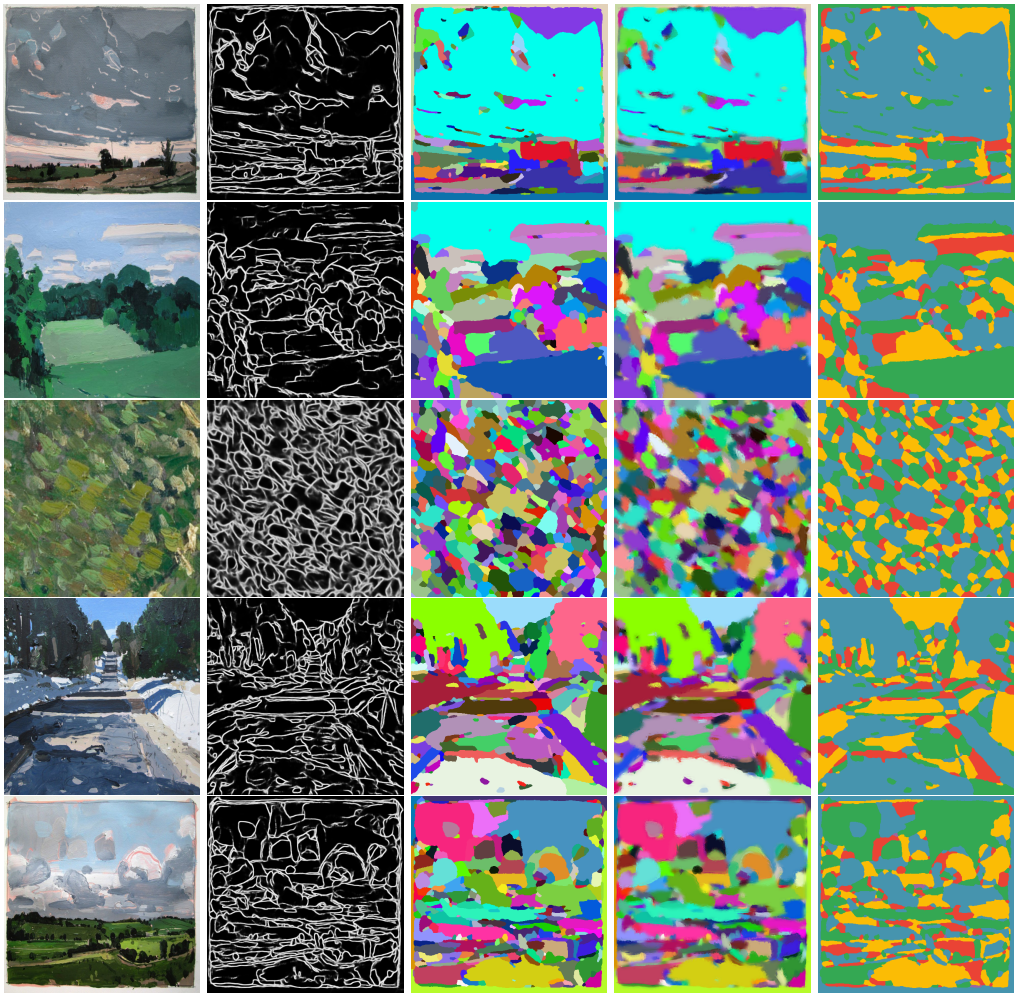
2. [3]'s method requires a pre-set fixed number of regions, while **DStroke** does not require that.

3. [3]'s method can only segment small number of regions, while **DStroke** enables soft segmentation for thousands of strokes, as shown Fig. 5.11.

4. As shown in Fig. 5.14, the accuracy of **DStroke** is noticeably higher than [3]'s method.

5. The run-time of **DStroke** method is much less time-consuming comparing to [3]'s method. This step takes around 3 minutes for a $640 \times 480$ image while **DStroke** takes less than 1 second, as shown in Fig. 5.13.





(a) (b) (c) (d) (e)

Fig. 5.14 (a) Input image. (b) Detected brush stroke edges. (c) Hard segmentation from **DStroke**. (d) The results of soft segmentation from **DStroke**. (e) Manual segmentation.

## 5.5 Summary

In this chapter, the deep learning based brush stroke extraction method — **DStroke** is addressed. The **DStroke** consists of two steps: edge detection (hard segmentation in Section 5.2 ) and brush stroke extraction (soft segmentation in Section 5.3). In this chapter, how to detect the edge of brush strokes with a deep neural network and extract the soft boundary of brush strokes are demonstrated.

With numerically evaluation, **DStroke** is compared with the most closely-related work **LStroke** and [3]. As a result, the **DStroke** outperforms **LStroke** in terms of valid rate, detection rate and MeanIOU of brush stroke extraction. And comparing with the state-of-art soft segmentation method [3], the **DStroke** also shows higher efficiency and accuracy.

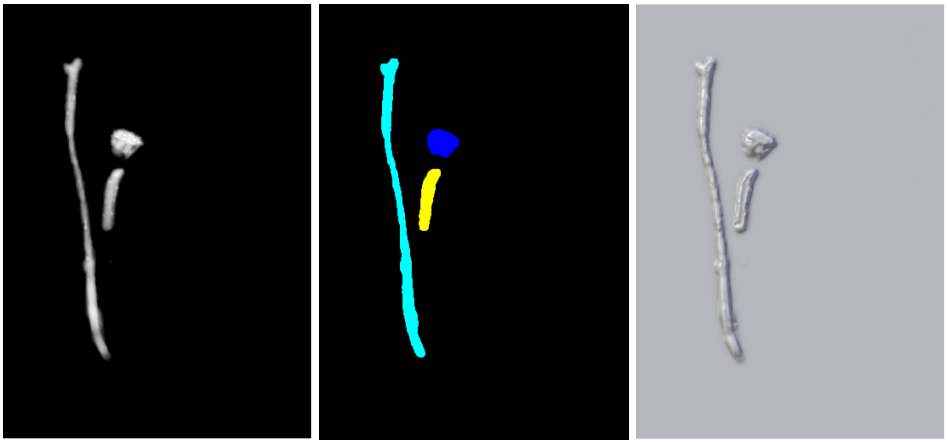
# Chapter 6

# High Relief Generation

In brush paintings, each brush stroke is often introduced to depict something specific in the real world [79].

Thus, the output of brush stroke extraction is a set of graphical objects that are meaningful with regard to the set of real objects the paintings depict. It is natural to generate depth information from brush strokes, here, how to generate relief from brush strokes is demonstrated. It is noteworthy that the formats of brush strokes extracted by **LStroke** and **DStroke** are the same. They are all brush stroke regions with colour and opacity values, and they all support the high relief generation method described in this chapter.

## 6.1 Displacement Map



(a) (b) (c)

Fig. 6.1 Displacement map generation from brush strokes. (a) Opacity of one layer. (b) Extracted three brush strokes. (c) Generated displacement map from three brush strokes.

To preserve the feature of brush stroke on the relief surface, the displacement maps for each brush stroke are generated . In this implementation, the orthogonal SFS [55] algorithm is employed on the segmented brush strokes. The brightness equation used in the SFS algorithm is expressed as,

$$I(x) = \frac{1}{\sqrt{1 + |\bigtriangledown h(x)|^2}} \tag{6.1}$$

$I(x)$ is the intensity at pixel $x$, $h(x)$ is the depth at pixel $x$. It can be noted that for higher intensity $I$, change of depth $h$ is smaller. Some brush strokes are painted by colours with high intensity or painted on a background with high intensity. As a result, if the shape from shading algorithm is performed on intensity, the resulting stroke models will become flat and lack of hierarchy. The opacity of brush strokes is independent of the colour (see Fig. 4.8). Each stroke has an appropriate distribution of opacity, which is in favor of a layered look. On the other hand, the intensity of stroke is determined by several facts: the colour and opacity of brush stroke, background colour and the overlapped strokes if there is any. Since colours of each layer are determined, based on Eq. 4.1, the intensity of brush stroke is a linear function of opacity. Displacement map generated from

intensity would be unavoidably influenced by the background colour, and the feature of the overlapped strokes, which is unwanted in each brush stroke mesh. Generating displacement map from opacity would better preserve the feature of brush strokes, so the equation is reformulated:

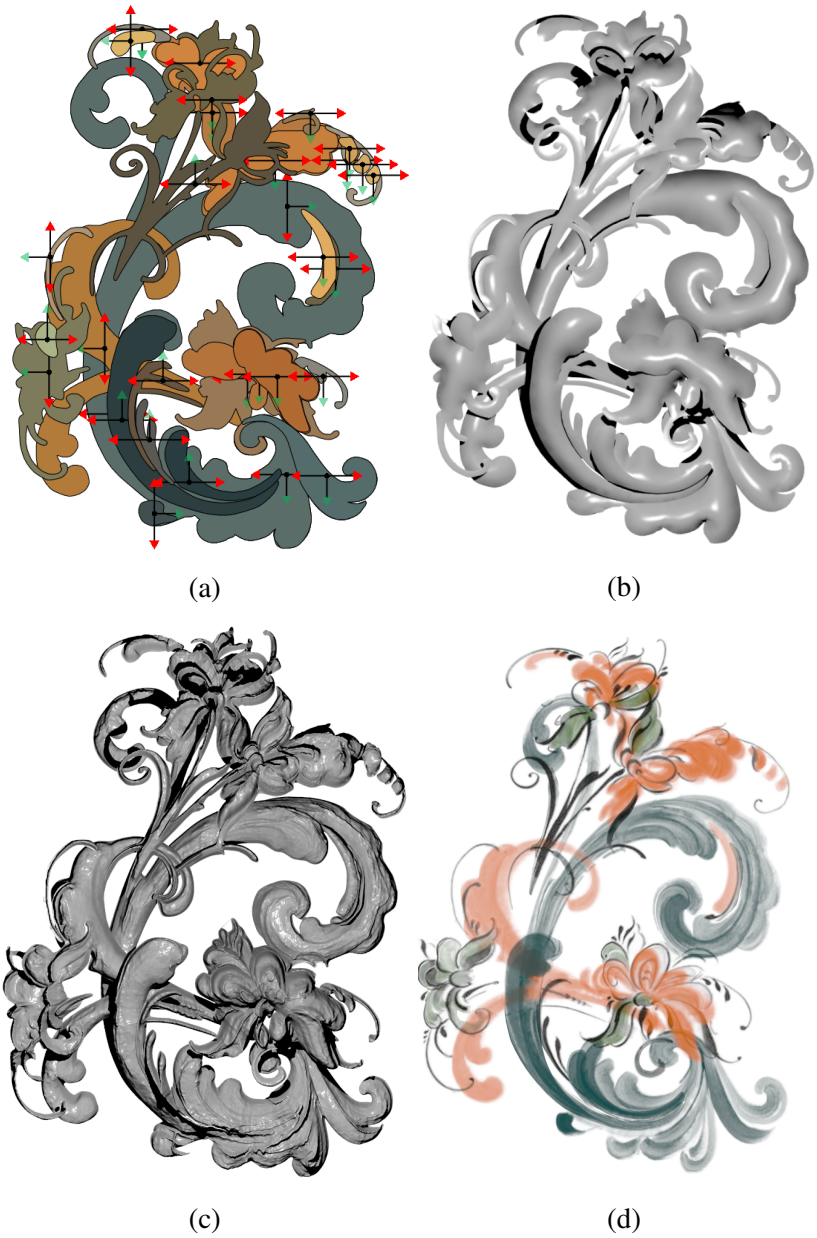
$$\alpha(x) = \frac{1}{\sqrt{1+|\bigtriangledown h|^2}} \tag{6.2}$$

$\alpha(x)$ is the opacity value of pixel $x$ on a brush stroke. To make the relief more inflated, it is rewritten as,

$$|\bigtriangledown h| = \sqrt{\frac{1}{|\alpha(x)|^2} - 1 + \Delta} \tag{6.3}$$

where $\Delta$ is a positive displacement which set to 0.2 by default. This modification may make the surface inflated. By using fast marching algorithm [61] to solve this equation, the displacement maps for each brush stroke are generated. Fig. 6.1 shows the opacity map of a decomposed layer and the extracted three brush strokes from it, which are depicted in different colour. Then, applying Eq. 6.3 to these three brush strokes generates the individual displacement maps. By merging them together, the displacement map of the entire input painting is generated accordingly.

## 6.2 Model Generation



(a) (b)

(c) (d)

Fig. 6.2 (a) Inflation with markup cues (b) Smooth high relief surface. (c) High relief with the displacement map. (d) Texture-mapped high relief (rotated 30 degree from the original viewing direction).

To further arrange and inflate the brush strokes, the user indication method proposed in [81] is applied, i.e. controlling the markup cues on extracted brush stroke regions to generate the inflated smooth surface for each brush stroke. As shown

in Fig. 6.2a, two very simple user-markup cues are used to guide the inflation of brush stroke regions.

The slope cues (the green icon) enable brush stroke regions to pop up from the image plane, and by dragging the arrow of the cues, users can control the slope $\vec{s}$ of the regions. Given brush stroke region $\phi(x,y)$ and slope cue $(x_i, y_i, \vec{s_i})$, the z-coordinate boundary are firstly computed. In [81]'s method, the gradient of region $\Omega$ is assumed to be smooth and curl-free, so the gradient $\vec{\Phi}$, $\vec{\Phi} = \nabla\phi(x,y)$, is reconstructed by minimizing:

$$\min_{\vec{\Phi}} \int\int_{\Omega} \left|\nabla\vec{\Phi}\right| + \left|\nabla \times \vec{\Phi}\right| dxdy$$

subjected to $\vec{\Phi}(x_i, y_i) = \vec{s_i}$. The first is smoothness term, and the second term minimize the curl. By setting one of the boundary vertices to 0, $\phi(x,y)$ can be simply computed through integration of $\vec{\Phi}$.

On the other hand, the curvature cue (red icon) constrains the local mean curvature of an extracted brush stroke, which allows users to further manipulate the inflation of the local shape. Similarly, the curvature $\widehat{k}$ amount is corresponding to the length of the curvature cues. The target curvature filed can be solve by following equation:

$$\Delta^2 f(x,y) = 0$$

where $\Delta$ is the Laplace-Beltrami operator, and $f$ represents the z-coordinates of the interior vertices, and it subjects to user-specified curvature cues $(x_i, y_i, \widehat{k_i})$, and Dirichlet boundary condition:

$$\Delta f(x_i, y_i) = \widehat{k_i} \;\; \forall (x_i, y_i) \in C$$
$$f(x_i, y_i) = \phi(x_i, y_i) \;\; \forall (x_i, y_i) \in B_D$$

Where $C$ represents the vertices on curvature cues, and $B_D$ represents the region boundary. For details, please refer to [81].

Then, as shown in Fig. 6.2c, by assigning the displacement map on each inflated surface, the features of the brush stroke are highlighted.

And instead of simply using the raw painting as the texture for the high relief model. To better show the features and inter-relations of brush strokes on high

relief, especially for the overlapped brush stroke regions, in this work, the extracted brush strokes are mapped on the corresponding reliefs. Fig. 6.2d shows that the extracted brush strokes with colour and opacity values are mapped to relief surfaces respectively. Up to now, the desired texture-mapped high relief model is generated.

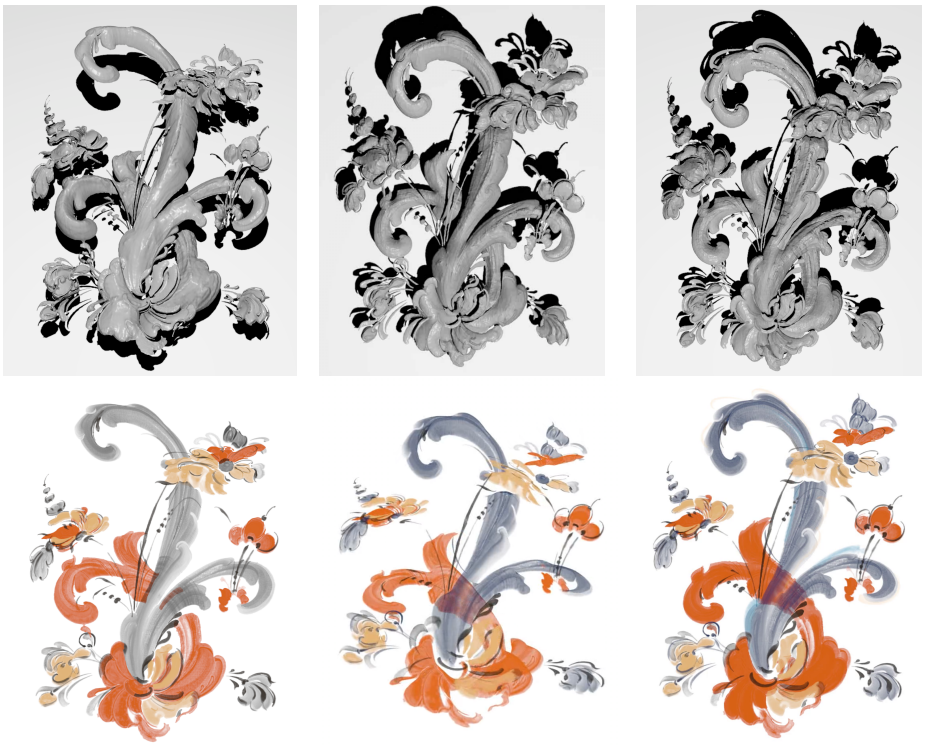
## 6.3 Results and Analysis

The proposed approach is performed on various brush paintings, including Rose-maling, van Gogh oil painting and Chinese brush paintings. All the tests were performed on a 6-core of 3.33 GHz Intel Core Xeon CPU with memory of 32 GB(RAM) and a single GTX 1080 GPU. Additionally, the current code of **LStroke** is developed with python and MATLAB, and **DStroke** is based on Keras and Tensorflow, the original code of Layer Decomposition, Pix2Pix and guided filter can be found on Github (at: https://github.com/CraGL/DecomposeSingle-Image-Into-Layers; https://github.com/affinelayer/pix2pix-tensorflow, https://github.com/clarkzjw/GuidedFilter). In the implementation, the parameters in Layer decomposition, ETF field, and Pix2Pix are set the default values as in the original codes.

Table 6.1 further shows the running time of the proposed approach. For fair demonstration and comparison, here **LStroke** is used for stroke segmentation here. To demonstrate the high relief effect, a further application is created: animating strokes in section 7.3. The implementation in this work is not multithreaded.

In LStroke, the number of layers (colour palette size) is chosen based on users' observation. Each layer aims to represent one coat of a painting with a single palette colour and it is desired to have a manageable number of layers. Too small palette size cannot generate all colour in the input image, and could result in wrong brush stroke segmentation. Too large palette size would result in an unwieldy number of layers. To make the proposed method more accessible, same as [70], the proposed decomposition (Eq. 4.9) is computed in a multi-resolution manner: the initial solution is computed based on recursively downsampled input images and the solution is upsampled as initial guess for larger image (finally, the input image size). By such manner, user can quickly preview the low resolution decomposed results (seconds for a 100x100 pixels image), which helps users to experiment

different layer numbers more efficiently. In other words, users can quickly choose a layer number based on the preview. From the same input image (2nd row of Fig. 6.6a), the effect of changing the layer number is shown in Fig. 6.3.

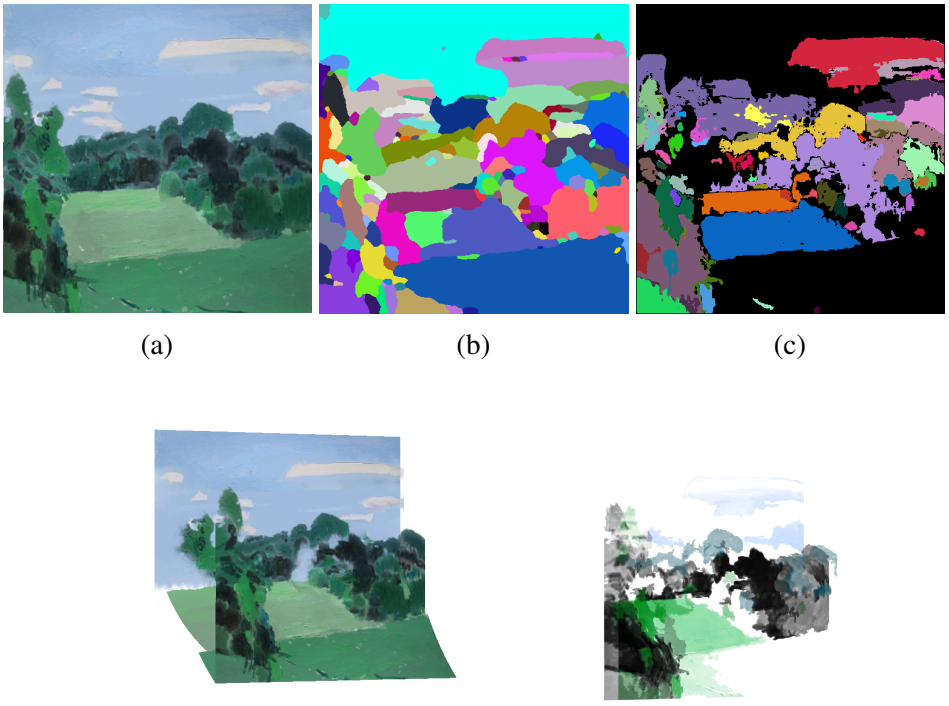


(a) (b) (c)

Fig. 6.3 The effect of changing the layer number. (a) High relief generated from 4 layers. (b) High relief generated from 6 layers. (c) High relief generated from 9 layers.

### 6.3.1 High relief generation by DStroke and LStroke

As mention in section 1.2, brush strokes are employed to represent the objects on brush paintings, and they serve the purpose of outlining the contours of objects. On the other hand, it is important to maintain the outlines of objects on the image in relief generation. In this work, reliefs are generated by inflating brush strokes individually, so undetected strokes would result in missing parts on the relief. It is preferred to extract brush stroke as accurate as possible.



(a) (b) (c)

(d) (e)

Fig. 6.4 (a) Input painting. (b) Segmentation by **DStroke**. (c) Segmentation by **LStroke**. (d) Relief generation by **DStroke**. (e) Relief generation by **LStroke**.

Based on the evaluation, **Dstroke** shows higher accuracy in brush stroke extraction (see section 5.4.1). So, relief generation by **DStroke** can better preserve the objects in the painting and shows less empty regions than **LStroke**.

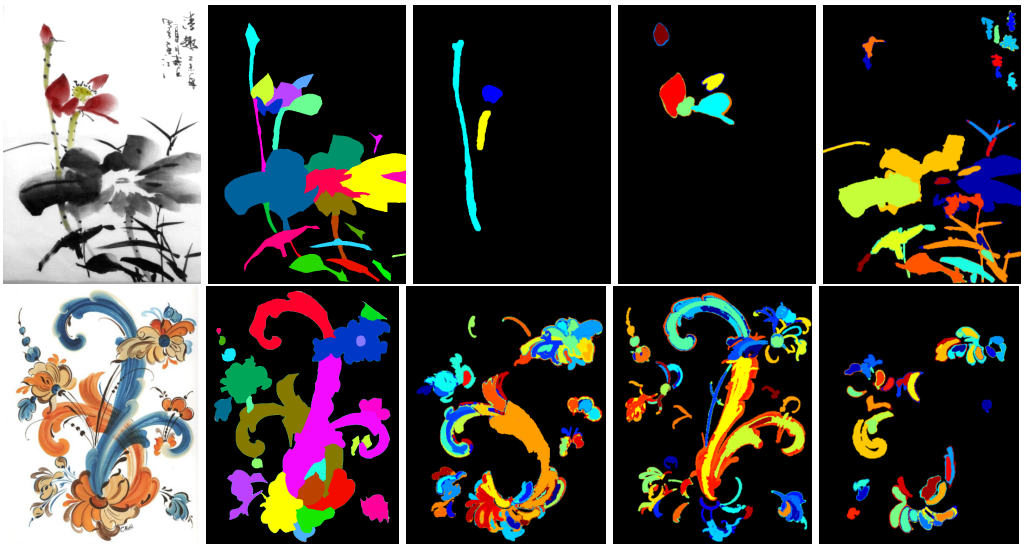
As shown in Fig. 6.4a and 6.4c, there are undetected regions in **LStroke**'s segmentation, which results in corresponding empty regions on the relief (e.g. undetected regions on sky in the painting, see Fig. 6.4e). On the other hand, relief generation by **DStroke** shows less empty regions and preserve more objects (Fig. 6.4d), due to the fact that **DStroke** extract brush strokes more accurately.

### 6.3.2 Compare high relief generation with [81]'s method

The proposed method is compared with the most closely-related work by Yeh et al. [81], mainly on four aspects: segmentation, local layering, inflation, and

texture-map. The method proposed in [81] was designed for high-relief 3D models from a single input image of organic objects with nontrivial shape profile. The proposed high relief generation process is similar to [81]'s method. The distinct difference is that the interactive user-driven region segmentation is replaced with the brush stroke extraction of each layer. After that, combining the SFS with the inflation, the high-relief is re-rendered based on the individual extracted brush strokes instead of the raw image.

**Segmentation:**

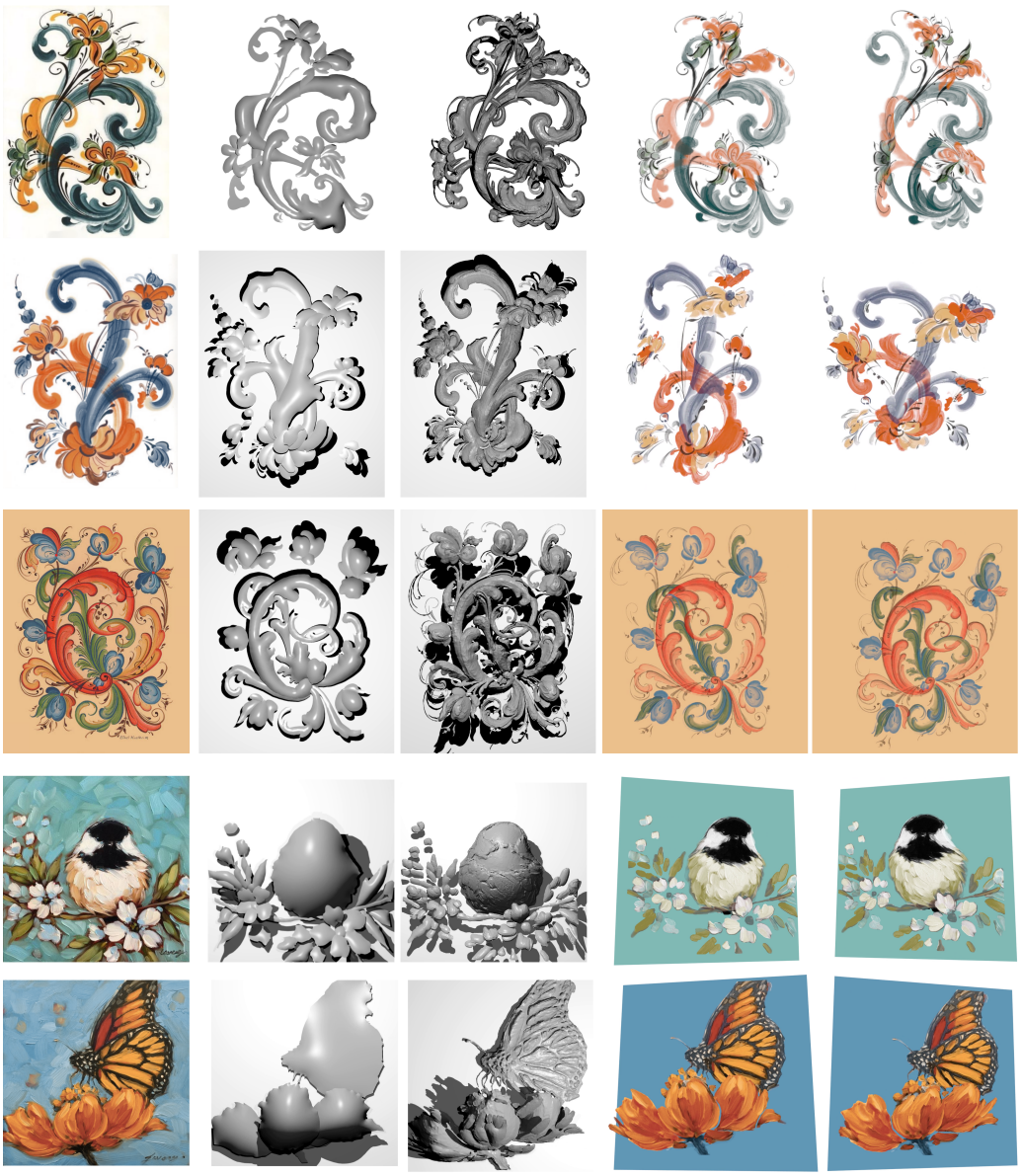


(a) (b) (c)

Fig. 6.5 (a) Input images. (b) Brush strokes segmented by method in [81]. (c) The last three columns show brush strokes extracted by **LStroke** on each decomposed layers.

Yeh et al.[81] extended the standard pixel-based graph-cut method [9] to be (2D) mesh-based to produce sharper segmentation boundary, in which the user scribbles foreground and background labels on the input image to subdivide it into different regions. However one painting may contains hundreds of brush stroke. To successfully segment the brush strokes, the users must specify a tedious amount of scribbles. It is likely to fail to segment thin and fine structures commonly seen in brush strokes. As a comparison, this work is automatic and able to preserve the thin and complex shapes of brush strokes, as shown in Fig. 6.5.

**Inflation:**



(a) (b) (c) (d) (e)

Fig. 6.6 (a) Input images. (b) High reliefs from [81]'s method. (c) High reliefs generated by the this work. (d,e) Texture-mapped high reliefs generated by this work.

[81]'s method is designed for reconstructing smooth and organic shapes, assuming the inflation model to be smooth. To preserve the fine details of the brush strokes, opacity is applied to generate the surface, and the displacement map to the high relief, which better maintains the details of brush strokes. Fig. 6.6 demonstrates the difference of the relief surfaces by this work and [81]'s.

The overall impression of [81]'s results is fine, but as a comparison, this work can better preserve the thin strokes and the fine details of brush strokes.

**Local layering:**

(a) (b) (c)

Fig. 6.7 Extraction of overlapped brush strokes. (a) Input image. (b) The brush stroke segmentation by Yeh et al.[81]'s method ( blue rectangles indicate the overlapped regions of brush strokes). (c) The extracted brush strokes on one decomposed layer by **LStroke**, in which the overlapped regions are extracted.

Each brush stroke covers a region on the canvas and they may overlap with each other, some quite heavily in a painting. In order to achieve faithfully complete brush strokes, [81] requires clear edge of region to form the T-junctions. For brush strokes with a complex shape, users have to manually divide the edge and label the layers. In this work, the overlapped brush strokes with different colours can be extracted by layer decomposition, in which the information of each brush strokes is retained. Fig. 6.7 clearly shows that the overlapped brush strokes can be extracted by this work.

**Texture-mapping:**

Compared to [81], this work exploits the opacity of brush strokes to transfer the details to high relief, which effectively preserves the artistic feelings of the paintings. Fig. 6.8 shows the texture-mapped high-relief results that are generated from a painting. Herein, Fig. 6.8a shows the input (single) image, while Fig. 6.8b and 6.8c show views of the models with large rotation from the original view. As shown in Fig. 6.8b, [81] simply uses the raw image as texture for each inflated brush stroke region, which is inappropriate, especially for the overlapped brush

(a) (b) (c)

Fig. 6.8 (a) Input image. (b) Texture-mapped high relief by method in [81]. (c) Texture-mapped high relief by this work.

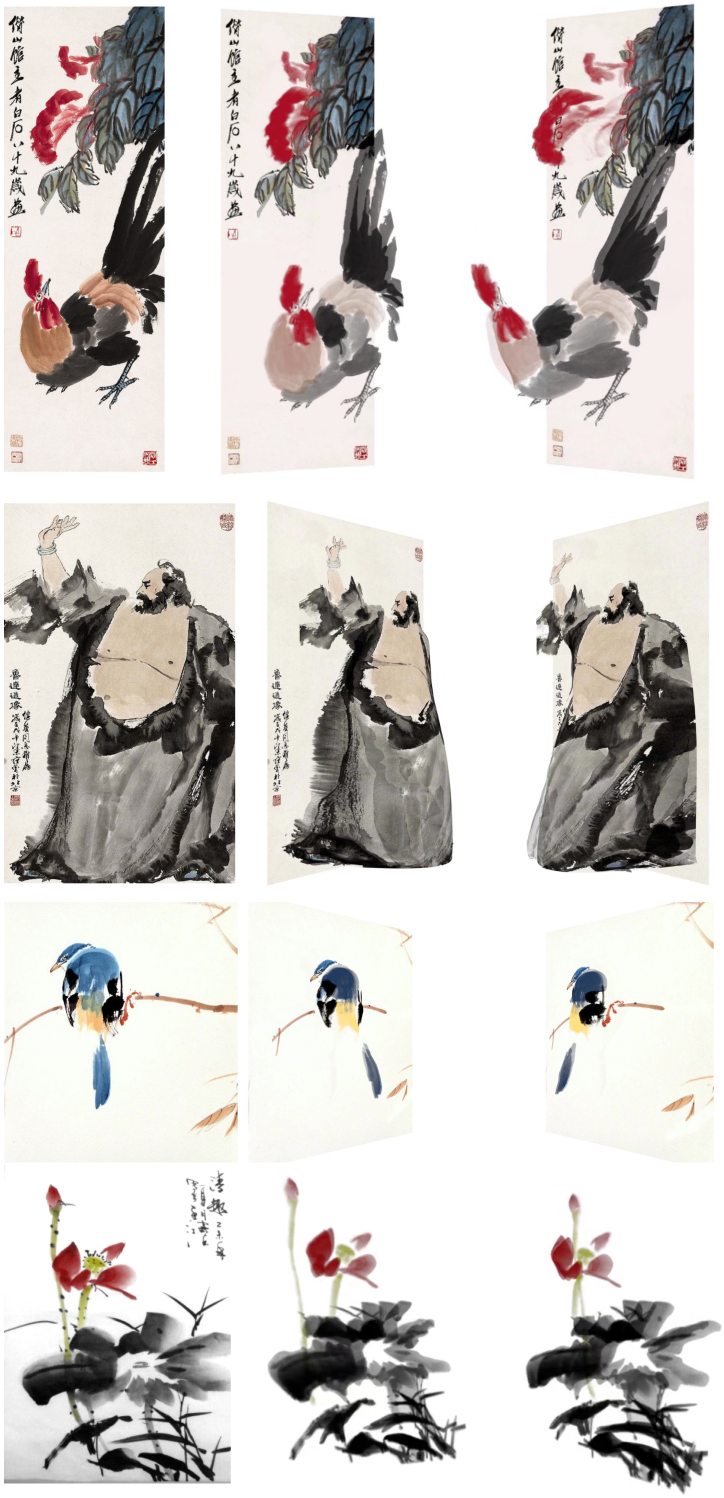
stroke region. Instead of using the raw image as texture, the extracted brush strokes with opacity values are mapped to the relief surface in the proposed method, which can better show the features and inter-relations of brush strokes on high relief. More results are shown in Fig. 6.9.

**Running time:**

Table 6.1 Running Time Performance

| Painting ID | [81]'s method | | | | This work | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Segmentation | | Relief generation | Total time | Segmentation | | | Relief Generation | | Total time |
| | Stroke number | Time | | | Stroke number | Runtime of Layer | Runtime of MSERs | Runtime of SFS | User annotation& Inflation | |
| Rooster (Fig. 6.9, row 1) | 112 | 35m31s | 9m44s | 45m15s | 292 | 311.25s | 0.64s | 8.68s | 11m37s | 16m57s |
| Man (Fig. 6.9, row 2) | 53 | 18m12s | 8m12s | 26m24s | 110 | 162.21s | 0.85s | 7.58s | 10m43s | 13m34s |
| Bird (Fig. 6.9, row 3) | 28 | 08m21s | 4m25s | 12m46s | 38 | 70.54s | 0.32s | 6.54s | 5m12s | 6m39s |
| Lotus (Fig. 6.9, row 4) | 34 | 15m01s | 7m21s | 22m22s | 45 | 112.21s | 0.34s | 8.25s | 8m31s | 10m32s |
| Lotus2 (Fig. 7.1, row 3) | 120 | 39m08s | 10m36s | 49m44s | 258 | 387.11s | 0.85s | 17.21s | 8m21s | 15m1s |
| Rosemaling1 (Fig. 6.6, row 1) | 42 | 20m22s | 8m09s | 28m31s | 114 | 164.87s | 0.75s | 7.26s | 9m19s | 12m12s |
| Rosemaling2 (Fig. 6.6, row 2) | 62 | 25m27s | 17m54s | 43m21s | 220 | 157.32s | 0.45s | 5.85s | 15m5s | 17m49s |
| Rosemaling3 (Fig. 6.6, row 3) | 35 | 19m45s | 6m15s | 26m0s | 105 | 60.23s | 0.55s | 5.92s | 7m04s | 8m11s |
| Bird2 (Fig. 6.6, row 4) | 42 | 18m35s | 5m31s | 24m06s | 85 | 110.48s | 0.82s | 7.11s | 12m13s | 14m11s |
| Butterfly (Fig. 6.6, row 5) | 6 | 5m54s | 3m15s | 9m09s | 30 | 50.10s | 0.44s | 5.86s | 6m24s | 7m21s |

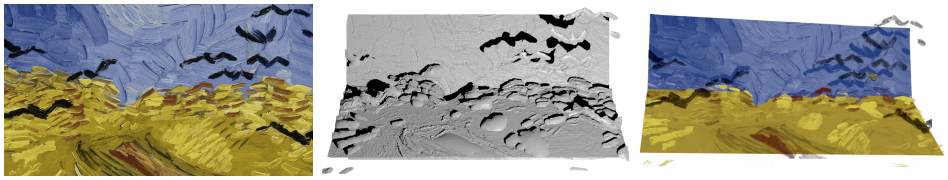
More specifically, the running time in segmentation and relief generation are compared between this work and [81]'s method. For segmentation, the proposed method is less time-consuming, especially for a input painting containing many brush strokes. Relief generation in [81]'s method includes several steps: layering, completion, user annotation and stitching, which requires more time for inflation on a single region than the proposed method. As shown in Table 6.1, the proposed method is less time-consuming, even though more regions are inflated than [81].



(a) (b) (c)

Fig. 6.9 (a) Input paintings. (b) Textured mapped high relief results (rotated 45 degree to the left from the viewing direction). (c) Textured mapped high relief results (rotated 45 degree to the right from the viewing direction).

### 6.3.3 Limitations



(a) (b) (c)

Fig. 6.10 (a) A region from a van Gogh's painting (F779, Wheatfield with Crows). (b) High relief model. (c) Texture-mapped high relief.

The proposed segmentation does not involve semantic information, while a van Gogh's painting may employ hundreds of strokes to represent a semantic region, such as a wheat field (Fig. 7.1b), a onion (first row in Fig. 4.11a). In both **LStroke** and **DStroke**, these semantic regions would be segmented into many brush strokes, and it's hard to maintain the original artistic feeling by inflating those brush strokes (see Fig. 6.10b). Meanwhile, it would be very time-consuming to adjust the makeup cues for every brush stroke in such semantic regions. Here a small region is selected from a van Gogh's painting (F779, Wheatfield with Crows) for high relief generation (see Fig. 6.10).

## 6.4 Summary

To generated relief model from brush strokes, these brush strokes need to be inflated and arranged. In this chapter, the high relief generation method is introduced. The high relief generation consists of three steps: displacement map generation, inflation and texture mapping. At first, displacement maps are generated based on opacity of each extracted stroke. By user indications, the extracted brush stroke regions are inflated to smooth surface. Then, the displacement maps are assigned to each surface. Moreover, the extracted brush strokes with colour and opacity values are mapped on the relief model. By such method, how to preserve the feature of brush strokes on the model surface is demonstrated.

The generated high reliefs are compared with the most closely-related work [81]. As a result, the proposed method outperforms [81]'s method in terms of inflation, local layering, segmentation, texture-map and running time.

By successfully extract brush strokes, this work can also be used in 2.5D animation, see chapter 7.

# Chapter 7

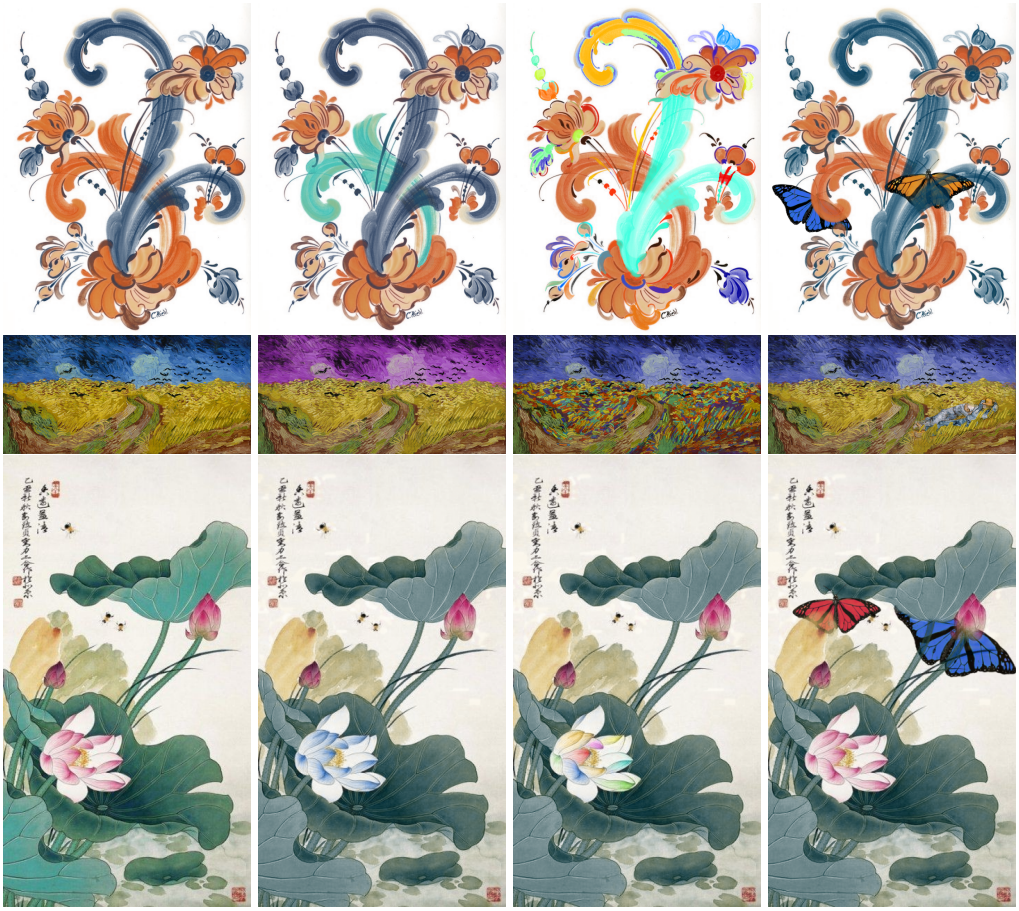
# Other Applications

Once the brush strokes are extracted, they can be used in a number of image editing operations to enable interesting paint-aware applications.

## 7.1 Recoloring Paintings

It is natural to recolor specified strokes for recomposition. Once the brush strokes are available, recoloring strokes with a new palette colour is becoming as simple as combining RGBA images. Fig. 7.1 shows recoloring strokes on three paintings respectively, Rosemaling, van Gogh oil painting, and Chinese brush painting. As the strokes are extracted, it is easy to separately recolor one or more strokes with different colours. Once the brush strokes are extracted, this work can simply recolor strokes by assigning RGB value to the brush stroke while keeping its opacity value. For example, suppose there are $n$ decomposed layers in **LStroke** and $m$ brush strokes extracted on the decomposed $i$-th layer $L_i$, and it is desired to change colour of $j$-th brush stroke on $L_i$. $L_i$ can be considered as a RGBA image with colour and opacity value at each pixel. Since the pixel coordinates in $j$-th brush stroke is known, for each pixel coordinates in the selected brush stroke, the *RGB* value of $L_i$ is changed while keeping the opacity value unchanged. By changing the *RGB* value to the desired colour, the $j$-th brush stroke is recolored on $L_i$. Then, the recolored $i$-th layer is composed with the other layers to generate the new image based on Eq. 4.1.

As shown in Fig. 7.1c, the selected brush strokes are assigned with some random colours. It is noteworthy that the pixels positions of extracted brush strokes are known, so they can be easily selected by users.



(a) (b) (c) (d)

Fig. 7.1 Column (a) shows 3 kinds of brush paintings, Rosemailing (Europe), van Gogh oil painting, and Chinese painting (East Asia). Column (b) shows recoloring one stroke and Column (c) shows recoloring multiple strokes. Column (d) shows inserting objects into these 3 paintings, in which the objects are opaque and are inserted in between brush strokes.

## 7.2 Inserting Objects

Fig. 7.2 and Fig. 7.1d show stroke manipulation by inserting objects. One task of image editing is to change a specified region with a new object in a seamless

and effortless manner. Here this work is interested in inserting new objects into painting while keeping the transparency of the painting. The inserted object is contained in an RGBA image with foreground opaque and background completely transparent. Note that the objects are opaque and are inserted between brush strokes. The occluded regions of the objects are visible due to the transparency of the brush strokes. Unlike simply pasting the object on the image (Fig 7.2c), this implementation works on the strokes, which can insert objects while keeping the opacity of brush strokes on the painting.

(a) Image of insert object

(b) Image of painting

(c) Paste object on painting

(d) Insert object between strokes

Fig. 7.2 Insert object between brush strokes (Painting ID:F779)

## 7.3 Animating Strokes

(a) Select the strokes to be rotated (the red circles show the selected strokes).

(b) First frame of the animation (before rotation of the strokes).

(c) 30th frame of the animation (after rotation of the strokes).

Fig. 7.3 Animation by rotating strokes.

Since the relief model of each stroke is generated separately in this work (Fig. 6.2), they can be used to generate some simple animations.

Here, a simple animation is created by moving and rotating the high relief. As shown in Fig. 7.3, the brush stroke on the head and tail of the rooster are firstly selected manually, and then their rotation angles and directions are manually set. Here, the head and tail are rotated by 30 degrees in 2 seconds.

Two frames of a created 2.5D animation are showed in Fig. 7.3. The direction and speed of stroke rotation are manually set in Blender [8]. The complete animation could be found in the following link: https://youtu.be/UOHXUwGbzgo.

# Chapter 8

# Conclusions and Future Work

## 8.1 Conclusion

Relief is an art form partway between a 3D sculpture and 2D painting. However, how to generate a relief from a 2D painting remains unsolved.

The traditional relief sculpturing techniques cost lots of time to achieve the desired visual effect. It impairs devising the personalized relief model for a wider range of uses. The image based relief modeling techniques have attracted increasing attention from the computer graphics community. It constructs relief models without traditional demands of tedious and time-consuming artistic work. However, the existing image based relief generation researches are based on real photos or sketches, which is not suitable for brush paintings.

The relief generation from brush painting is a challenging task. The diversity of the brush strokes and their artistic features make the relief generation even more difficult to accomplish.

This thesis addresses this challenge by introducing a brush strokes based relief modelling framework. This work presents a new approach for generating a relief from a single brush painting, aiming to preserve the original artistic features of the painting. The basic concept of this framework is to extract the strokes from a single painting and generate the high relief model. The proposed brush stroke extraction to encode the features of brush paintings is the foundation of this framework.

This work particularly consider the overlapped brush strokes with complex shape profiles and their opacities.

To extract the brush strokes from a painting, this work introduces **LStroke** which applies layer decomposition and stroke segmentation by imposing boundary constraints. By layer decomposition, brush strokes with different palette colours are separated into different layers. To further extract the brush strokes, the revised MSERs algorithm is applied here, which both preserves stroke continuity and removes spurious edges within one layer.

To better deal with the brush strokes mixed with similar colours, this thesis introduces **DStroke**, in which deep learning techniques are firstly leveraged to facilitate extracting brush strokes from paintings. Meanwhile, for training purpose, this work also proposes an automatic algorithm to create paintings and the associated brush stroke edge maps as training dataset. The **DStroke** enables soft segmentation for thousands of strokes. Comparing with the alternative works, **DStroke** shows higher accuracy and efficiency.

Based on the extracted brush strokes, the displacement maps of the strokes are generated individually, this work performs shape from shading(SFS) on the opacity of the paintings instead of the intensity, which transfers the features of brush paintings to the surface of the brush stroke models. With user indications, this work inflates the surface for each brush stroke and combine it with the displacement maps. The shapes of all the strokes are then arranged to form the desired high-relief. After that, instead of using the raw image as texture, the extracted brush strokes with opacity values are mapped to the relief surface in this work, which can better show the features and inter-relations of brush strokes on high relief. As a result, this work bridges the gap between painting and relief.

The findings of this work also support several promising applications. One is recoloring of the specified strokes for recomposition. Once the brush strokes are extracted, this work can simply recolour strokes by assign RGB values to the brush strokes while keep their opacity value. Similarly, this work showed another image synthesis application by inserting new objects into painting while keeping the transparency of the brush strokes. Another application is animating stroke. By moving and rotating the texture mapped relief of certain brush stroke, some simple animations can be created.

## 8.2 Future Work

The work in this thesis leads to the following directions for future work:

**Wider Range of Paintings:**

In the future, I will attempt a wider range of paintings and investigate the issues of their individual layer decomposition and stroke segmentation. Different paintings have different stroke patterns. Understanding the individual rules will improve the success rate for a wider range of paintings. Rational and automatic colour separation in the overlap regions is not trivial and will also be studied in the future.

**Automatic Palette Selection:**

I plan to investigate methods for better palette colour selection from the image contents, such as local brush stroke analysis. Secondly, I would study how colour composition equation would influence the brush stroke extraction, and refine this work based on it.

**Smart Cues:**

The brush stroke extraction methods proposed in this thesis does not involve semantic information, while some paintings may employ hundreds of strokes to represent a single semantic region. It is inefficient to adjust the make-up cues for such regions. It can be dramatically improved by region clustering or semantic labelling, once the brush strokes in one semantic region are clustered, the user mark-up cues can be assigned uniformly.

**End-to-End Training:**

In the proposed **DStroke** method, the deep neural network is trained to extract the contour of brush strokes, then the opacity of brush strokes are extracted based on the hard segmentation. In **DStroke**, the opacitiy of brush strokes in dataset is used for evaluation.

Thus, it is expected that designing an end-to-end deep neural network to utilize the opacity values of brush strokes and improve the accuracy of brush stroke extraction.

# References


[1] Agrawal, A., Raskar, R., and Chellappa, R. (2006). What is the range of surface reconstructions from a gradient field? In *European Conference on Computer Vision*, pages 578–591. Springer.

[2] Aharoni-Mack, E., Shambik, Y., and Lischinski, D. (2017). Pigment-based recoloring of watercolor paintings. In *Proceedings of the Symposium on Non-Photorealistic Animation and Rendering*, page 1. ACM.

[3] Aksoy, Y., Oh, T.-H., Paris, S., Pollefeys, M., and Matusik, W. (2018). Semantic soft segmentation. *ACM Transactions on Graphics (TOG)*, 37(4):72.

[4] Aksoy, Y., Ozan Aydin, T., and Pollefeys, M. (2017). Designing effective inter-pixel information flow for natural image matting. *arXiv preprint arXiv:1707.05055*.

[5] Alexa, M. and Matusik, W. (2010). Reliefs as images. *ACM Trans. Graph.*, 29(4):60–1.

[6] Belhumeur, P. N., Kriegman, D. J., and Yuille, A. L. (1999). The bas-relief ambiguity. *International journal of computer vision*, 35(1):33–44.

[7] Benzaid, Z. (2017). *Analysis of Bas-Relief Generation Techniques*. PhD thesis, The University of Wisconsin-Milwaukee.

[8] Blender Online Community (2019). *Blender - a 3D modelling and rendering package*. Blender Foundation, Blender Institute, Amsterdam.

[9] Boykov, Y. and Kolmogorov, V. (2004). An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE transactions on pattern analysis and machine intelligence*, 26(9):1124–1137.

[10] Canny, J. (1987). A computational approach to edge detection. In *Readings in computer vision*, pages 184–203. Elsevier.

[11] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2017). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848.

[12] Chen, Q., Li, D., and Tang, C.-K. (2013). Knn matting. *IEEE transactions on pattern analysis and machine intelligence*, 35(9):2175–2188.

[13] Chu, N. S. and Tai, C.-L. (2004). Real-time painting with an expressive virtual chinese brush. *IEEE Computer Graphics and applications*, 24(5):76–85.

[14] Cignoni, P., Montani, C., and Scopigno, R. (1997). Computer-assisted generation of bas-and high-reliefs. *Journal of graphics tools*, 2(3):15–28.

[15] Donoser, M. and Bischof, H. (2006). Efficient maximally stable extremal region (mser) tracking. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 553–560. IEEE.

[16] Girshick, R. B. (2004). *Simulating Chinese brush painting: the parametric hairy brush*. ACM.

[17] Gooch, B., Coombe, G., and Shirley, P. (2002). Artistic vision: painterly rendering using computer vision techniques. In *Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*, pages 83–ff. ACM.

[18] Goodman, N. (1976). Languages of art (rev. ed.). *Indianapolis, IN: Hackett*.

[19] He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE.

[20] He, K., Sun, J., and Tang, X. (2013). Guided image filtering. *IEEE transactions on pattern analysis & machine intelligence*, (6):1397–1409.

[21] Hegde, S., Gatzidis, C., and Tian, F. (2013). Painterly rendering techniques: a state-of-the-art review of current approaches. *Computer Animation and Virtual Worlds*, 24(1):43–64.

[22] Herold, J. and Stahovich, T. F. (2014). A machine learning approach to automatic stroke segmentation. *Computers & Graphics*, 38:357–364.

[23] Hertzmann, A. (2002). Fast paint texture. In *Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*, pages 91–ff. ACM.

[24] Hertzmann, A. (2003). A survey of stroke-based rendering. Institute of Electrical and Electronics Engineers.

[25] Hoiem, D., Efros, A. A., and Hebert, M. (2005a). Automatic photo pop-up. *ACM transactions on graphics (TOG)*, 24(3):577–584.

[26] Hoiem, D., Efros, A. A., and Hebert, M. (2005b). Geometric context from a single image. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 654–661. IEEE.

[27] Hoiem, D., Efros, A. A., and Hebert, M. (2007). Recovering surface layout from an image. *International Journal of Computer Vision*, 75(1):151.

[28] Horn, B. K. and Brooks, M. J. (1986). The variational approach to shape from shading. *Computer Vision, Graphics, and Image Processing*, 33(2):174–208.

[29] Igarashi, T., Matsuoka, S., and Tanaka, H. (2007). Teddy: a sketching interface for 3d freeform design. In *Acm siggraph 2007 courses*, page 21. ACM.

[30] Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. *arXiv preprint*.

[31] Kang, H., Lee, S., and Chui, C. K. (2007). Coherent line drawing. In *Proceedings of the 5th international symposium on Non-photorealistic animation and rendering*, pages 43–50. ACM.

[32] Karpenko, O. A. and Hughes, J. F. (2006). Smoothsketch: 3d free-form shapes from complex sketches. In *ACM Transactions on Graphics (TOG)*, volume 25, pages 589–598. ACM.

[33] Kerber, J., Tevs, A., Belyaev, A., Zayer, R., and Seidel, H.-P. (2009). Feature sensitive bas relief generation. In *Shape Modeling and Applications, 2009. SMI 2009. IEEE International Conference on*, pages 148–154. IEEE.

[34] Kerber, J., Wang, M., Chang, J., Zhang, J. J., Belyaev, A., and Seidel, H.-P. (2012). Computer assisted relief generation—a survey. In *Computer Graphics Forum*, volume 31, pages 2363–2377. Wiley Online Library.

[35] Kolomenkin, M., Leifman, G., Shimshoni, I., and Tal, A. (2011). Reconstruction of relief objects from line drawings. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 993–1000. IEEE.

[36] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[37] KWO, D. (1978). Chinese brushwork: Its history, aesthetics, and techniques.

[38] KYLEBRUSH.COM, K. T. W. . (2018). kylebrush. https://www.kylebrush.com/.

[39] Levin, A., Lischinski, D., and Weiss, Y. (2008a). A closed-form solution to natural image matting. *IEEE transactions on pattern analysis and machine intelligence*, 30(2):228–242.

[40] Levin, A., Rav-Acha, A., and Lischinski, D. (2008b). Spectral matting. *IEEE transactions on pattern analysis and machine intelligence*, 30(10):1699–1712.

[41] Li, J. and Gardner, H. (1993). How domains constrain creativity: The case of traditional chinese and western painting. *American Behavioral Scientist*, 37(1):94–101.

[42] Li, J., Yao, L., Hendriks, E., and Wang, J. Z. (2012a). Rhythmic brushstrokes distinguish van gogh from his contemporaries: findings via automated brushstroke extraction. *IEEE transactions on pattern analysis and machine intelligence*, 34(6):1159–1176.

[43] Li, Z., Wang, S., Yu, J., and Ma, K.-L. (2012b). Restoration of brick and stone relief from single rubbing images. *IEEE Transactions on Visualization and Computer Graphics*, 18(2):177–187.

[44] Lions, P.-L., Rouy, E., and Tourin, A. (1993). Shape-from-shading, viscosity solutions and edges. *Numerische Mathematik*, 64(1):323–353.

[45] Liu, B., Gould, S., and Koller, D. (2010). Single image depth estimation from predicted semantic labels. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1253–1260. IEEE.

[46] Loon, A. v. et al. (2008). *Color changes and chemical reactivity in seventeenth-century oil paintings*.

[47] Malik, J. (1987). Interpreting line drawings of curved objects. *International Journal of Computer Vision*, 1(1):73–103.

[48] McCann, J. and Pollard, N. (2009). Local layering. *ACM Transactions on Graphics (TOG)*, 28(3):84.

[49] McCann, J. and Pollard, N. S. (2012). Soft stacking. In *Computer Graphics Forum*, volume 31, pages 469–478. Wiley Online Library.

[50] mypaint contributors (2018). mypaint. https://github.com/mypaint.

[51] Nealen, A., Igarashi, T., Sorkine, O., and Alexa, M. (2007). Fibermesh: designing freeform surfaces with 3d curves. *ACM transactions on graphics (TOG)*, 26(3):41.

[52] Nistér, D. and Stewénius, H. (2008). Linear time maximally stable extremal regions. *Computer Vision–ECCV 2008*, pages 183–196.

[53] Oh, B. M., Durand, F., and Chen, M. (2007). Image-based modeling and photo editing. US Patent 7,199,793.

[54] Porter, T. and Duff, T. (1984). Compositing digital images. In *ACM Siggraph Computer Graphics*, volume 18, pages 253–259. ACM.

[55] Prados, E. and Faugeras, O. (2004). Unifying approaches and removing unrealistic assumptions in shape from shading: Mathematics can help. *Computer Vision-ECCV 2004*, pages 141–154.

[56] Prados, E. and Faugeras, O. D. (2003). " perspective shape from shading" and viscosity solutions. In *ICCV*, volume 3, page 826.

[57] Richardt, C., Lopez-Moreno, J., Bousseau, A., Agrawala, M., and Drettakis, G. (2014). Vectorising bitmaps into semi-transparent gradient layers. In *Computer Graphics Forum*, volume 33, pages 11–19. Wiley Online Library.

[58] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

[59] Sander, P. V., Gu, X., Gortler, S. J., Hoppe, H., and Snyder, J. (2000). Silhouette clipping. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 327–334. ACM Press/Addison-Wesley Publishing Co.

[60] Saxena, A., Sun, M., and Ng, A. Y. (2009). Make3d: Learning 3d scene structure from a single still image. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):824–840.

[61] Sethian, J. A. (1999). *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*, volume 3. Cambridge university press.

[62] Singaraju, D. and Vidal, R. (2011). Estimation of alpha mattes for multiple image layers. *IEEE transactions on pattern analysis and machine intelligence*, 33(7):1295–1309.

[63] Smith, R. and Lloyd, E. (1997). Art school.

[64] Sobel, I. (2014). History and definition of the sobel operator. *Retrieved from the World Wide Web*, 1505.

[65] Song, W., Belyaev, A., and Seidel, H.-P. (2007). Automatic generation of bas-reliefs from 3d shapes. In *Shape Modeling and Applications, 2007. SMI'07. IEEE International Conference on*, pages 211–214. IEEE.

[66] stocksnap.io, S. W. . (2018). stocksnap. https://www.stocksnap.io/.

[67] Sun, X., Rosin, P. L., Martin, R. R., and Langbein, F. C. (2009). Bas-relief generation using adaptive histogram equalization. *IEEE transactions on visualization and computer graphics*, 15(4):642–653.

[68] Sỳkora, D., Kavan, L., Čadík, M., Jamriška, O., Jacobson, A., Whited, B., Simmons, M., and Sorkine-Hornung, O. (2014). Ink-and-ray: Bas-relief meshes for adding global illumination effects to hand-drawn characters. *ACM Transactions on Graphics (TOG)*, 33(2):16.

[69] Tan, J., DiVerdi, S., Lu, J., and Gingold, Y. (2018). Pigmento: Pigment-based image analysis and editing. *IEEE transactions on visualization and computer graphics*.

[70] Tan, J., Lien, J.-M., and Gingold, Y. (2016). Decomposing images into layers via rgb-space geometry. *ACM Transactions on Graphics (TOG)*, 36(1):7.

[71] Van Gogh, V. and Faille, J.-B. (1970). *The works of Vincent van Gogh: his paintings and drawings*. Meulenhoff International.

[72] Varley, P. A. and Martin, R. R. (2002). Estimating depth from line drawing. In *Proceedings of the seventh ACM symposium on Solid modeling and applications*, pages 180–191. ACM.

[73] Wang, M., Chang, J., Pan, J., and Zhang, J. (2010). Image-based bas-relief generation with gradient operation. In *Proceedings of the 11th IASTED International Conference Computer Graphics and Imaging, Innsbruck*, volume 679, page 33.

[74] Weyrich, T., Deng, J., Barnes, C., Rusinkiewicz, S., and Finkelstein, A. (2007). Digital bas-relief from 3d scenes. In *ACM Transactions on Graphics (TOG)*, volume 26, page 32. ACM.

[75] Winnemöller, H., Olsen, S. C., and Gooch, B. (2006). Real-time video abstraction. In *ACM Transactions On Graphics (TOG)*, volume 25, pages 1221–1226. ACM.

[76] Wu, J., Martin, R. R., Rosin, P. L., Sun, X.-F., Langbein, F. C., Lai, Y.-K., Marshall, A. D., and Liu, Y.-H. (2013). Making bas-reliefs from photographs of human faces. *Computer-Aided Design*, 45(3):671–682.

[77] Wu, T.-P., Sun, J., Tang, C.-K., and Shum, H.-Y. (2008). Interactive normal reconstruction from a single image. In *ACM Transactions on Graphics (TOG)*, volume 27, page 119. ACM.

[78] Xie, N., Zhao, T., Tian, F., Zhang, X. H., and Sugiyam, M. (2015). Stroke-based stylization learning and rendering with inverse reinforcement learning.

[79] Xu, S., Xu, Y., Kang, S. B., Salesin, D. H., Pan, Y., and Shum, H.-Y. (2006). Animating chinese paintings through stroke-based decomposition. *ACM Transactions on Graphics (TOG)*, 25(2):239–267.

[80] Xu, Z. and Sun, J. (2010). Image inpainting by patch propagation using patch sparsity. *IEEE transactions on image processing*, 19(5):1153–1165.

[81] Yeh, C.-K., Huang, S.-Y., Jayaraman, P. K., Fu, C.-W., and Lee, T.-Y. (2017). Interactive high-relief reconstruction for organic and double-sided objects from a photo. *IEEE transactions on visualization and computer graphics*, 23(7):1796–1808.

[82] Zeng, G., Matsushita, Y., Quan, L., and Shum, H.-Y. (2005). Interactive shape from shading. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 343–350. IEEE.

[83] Zeng, Q., Martin, R. R., Wang, L., Quinn, J. A., Sun, Y., and Tu, C. (2014). Region-based bas-relief generation from a single image. *Graphical Models*, 76(3):140–151.

[84] Zhang, L., Dugas-Phocion, G., Samson, J.-S., and Seitz, S. M. (2002). Single-view modelling of free-form scenes. *Computer Animation and Virtual Worlds*, 13(4):225–235.

[85] Zhang, S.-H., Chen, T., Zhang, Y.-F., Hu, S.-M., and Martin, R. R. (2009). Vectorizing cartoon animations. *IEEE Transactions on Visualization and Computer Graphics*, 15(4):618–629.

[86] Zhao, H., Shi, J., Qi, X., Wang, X., and Jia, J. (2017). Pyramid scene parsing network. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 6230–6239. IEEE.