國立交通大學

多媒體工程研究所

碩 士 論 文

利 用 線 條 平 移 與 合 併 簡 化 線 條 繪 畫

Line Drawing Simplification by Stroke Translation and Combination

研 究 生：簡　韻

指導教授：莊榮宏　教授

　　　　　林文杰　教授

中 華 民 國　一百零一　年 九 月

利用線條平移與合併簡化線條繪畫

學生：簡　韻　　　　　　　指導教授：莊榮宏　博士

林文杰　博士

國立交通大學

多媒體工程研究所

摘　要

我們提出了一個新的簡化線條繪畫草稿的演算法。首先，我們會根據使用者需求決定是否要將輸入的線條分段。我們會觀察曲率 (curvature)，並在曲率過大的位置切開線條。接著，面對畫家隨意繪出的鬆散線條，我們使用低通濾波 (low-pass filter) 來聚集這些線條。我們會對繪畫做低通濾波，然後再利用濾波後的結果作為權重 (weight)，算出每條線的權重，並將線條平移至畫面上權重較高的地方。走向相近、距離接近的鬆散線條將會被聚集在一起，使我們能夠在之後的步驟更輕易地合併線條。之後，我們會依據線條屬性，找出相近的線條配對並合併，減少線條總數來達到較乾淨的線條繪畫。這個系統會減少雜亂的短線條，將它們合併成較長的線條。使雜亂的草稿變成較乾淨的繪畫。

Line Drawing Simplification by Stroke Translation and Combination

Student: Yun Chien

Advisor: Dr. Jung-Hong Chuang

Dr. Wen-Chieh Lin

Institute of Multimedia Engineering

College of Computer Science

National Chiao Tung University

ABSTRACT

In this thesis, we propose a new algorithm for simplifying line drawing sketches. First, if user want to segment, we segment the strokes at the points of large curvature. Then, we perform a low-pass filter and use the result to assign a weight to every stroke. The strokes are moved to the position of the higher weight. After that, we find the stroke pairs and combine them to reduce the total number of the strokes, resulting in a cleaner line drawing art. This system also cuts down the disordered and confusing small strokes and combines them to form long strokes.

Acknowledgments

I would like to thank my advisors, Professors Jung-Hong Chuang and Wen-Chieh Lin, for their guidance, inspirations and encouragement. I would also like to thank my thesis committee members, Professors Hung-Kuo Chu and Yu-Ting Tsai for giving me excellent advice and viewpoints. I am also grateful to Pascal Barla and Amit Shesh for providing me the testdata in their papers. I appreciate my senior colleagues' help, Tan-Chi Ho, who taught me lots of knowledge of computer graphics and programming skills, and Tsung-Shian Huang, who gave me kind assistance and advice. I want to express my gratitude to all my colleagues and junior colleagues in CGGM lab. Besides, I would like to thank Ceqblack, who helped me to understand and use some mathematic formulas. Finally, I would like to thank my parents for their love and support.

Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

The process of making the digital art usually contains three parts, the sketch, line-art and coloring; as shown in Figure 1.1. The sketches are usually used to demonstrate the idea of the whole picture. It can be drawn freely and quickly. Next, the artists must make the picture "cleared". They will draw the contour of the art carefully using the beautiful lines. Finally, they color the line-art by using the base color to fill the line-art, and then adding the shadows and details to finish the art.

The process of making the sketch to the line-art, however, could be very time consuming because it is nearly impossible to draw a long beautiful stroke directly. To prevent the shiver

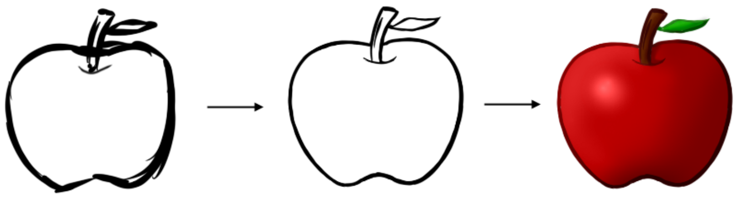


Figure 1.1: The process of making the digital art.

of hand, the stroke must be drawn in one breath. Artists usually draw many short beautiful strokes and link them carefully, resulting in a complete and pretty single stroke. Drawing the line according to sketches seems easy but it is very time-consuming. This motivates us to study how to translate the sketch into line-art automatically. Although it is intuitive for human to understand the shape from lots of discontinuous and disordered lines, it is very hard for the computer. The goal of our study is to find out a method to simplify the sketch to a fewer strokes, making the contours to be as long as possible and close to the artists' instinct of linking the short strokes and reducing the density of the hatchings.

Barla et al. proposed the concept of ε-line and ε group [BTS05]. The combined strokes are restricted to the cases that they can not fold onto themselves. Shesh and Chen presented an efficient and dynamic stroke simplification [SC08] based on the overlap of the stroke pairs. However, most of the contour strokes of a sketch do not overlap others that much. Orbay and Kara proposed a new method not only simplify the sketches but also beautify them [OK11]. Their focus is the process of contour strokes, not the hatching strokes. In previous work, too many rules are usually given to restrict the strokes dealt with or the strokes are combined locally regardless their global feature.

In this thesis, we provide a simple algorithm to simplify the strokes, especially for the sparse strokes. We compute the features of strokes, such as tangent and curvature, then use a low-pass filter to gather the sparse strokes together. Finally, we will find out the stroke pairs by the feature of the strokes and combine the stroke pairs to obtain a simplified line-art.

1.1 Contribution

The contributions of our line simplification can be summarized as follows:

- Present a unified simplification framework for contours and hatching.
- Present a stroke simplification based on a low-pass filtering

1.2 Organization of the Thesis

The following chapters are organized as follows. Chapter 2 gives a literature review and the background knowledge. Chapter 3 introduces the details of preprocessing, segmentation, stroke translation, stroke pairing and combination. Chapter 4 shows the experimental results of our line simplification algorithm and comparisons to previous work [BTS05, SC08]. Finally, we summarize our study and discuss the future work in Chapter 5.

CHAPTER 2

Related Work

Several novel techniques have been proposed for the simplification of line-drawing. We classify them into three groups: progressively improving method, density-based simplification, and curve extracting method.

2.1 Progressively Improving Method

These systems simplify the strokes while users are drawing. New curves that the users just add are used to modify the old curves. Baudel' s design is based on patterns extracted from the artists' drawing and editing [Bau94]. It allows input with many strokes to modify a single stroke. Igarashi et al. proposed a method to beautify the strokes one after another by considering the geometric constraints among the curves [IMKT97]. They also provided multiple candidates to avoid the ambiguity. Kara et al. used conceptual sketches to update the existing 3D template models [KDS06]. They modified the points on the original models by using the attraction of the points from the input curves.

A common problem of these methods is that the artists must give up their natural drawing style, which could disrupt their idea of drawing.

2.2 Density-based Simplification

There are several studies that are based on density control of the line rendering. Preim and Strothotte sorted the lines with respect to the screen position, local density and line length [PS95]. They cut down the strokes according to this order. Deussen and Strothotte proposed a method for drawing the trees [DS00]. They used the depth information of a tree model to simplify the foliage. Praun et al. created Level-of-Detail (LOD) for the hatching [PHWF01]. They constructed a sequence of mipmapped hatch images corresponding to different tones, collectively called a “tonal art map”. They used the textures of hatching which fit the user defined tones. Wilson and Ma created the complexity map of the screen image. They could render the 3D model in different style such as silhouette or hatching [WM04]. Grabli et al. also created a density map [GDS04]. They used the density map and other information selected from the 3D scene (silhouettes, depth, etc.) to evaluate the significance of the lines and delete the least significant lines. Cole et al. used a “priority buffer” to determine the significant of lines [CDF$^+$06].

These methods only reduce the existing lines and do not add the new lines, which are not suitable for the sketch drawn by the artists. Because the artists usually use many short lines to form a long line.

2.3 Curve Extracting Method

Many techniques for drawing simplification have been proposed to extract the geometric curves from completed sketches. Saund extracted the arcs from the curve using “curve element tokens” [Sau92] and then used the least-squares arc to fit the new curve. Their results, however, are too limited due to the presentation of arcs. Rosin proposed three aspects to group the strokes: continuation, parallelism and proximity, but they did not provide good experiments [Ros94]. Barla et al. used the concept of “ε-lines” and “ε-group” to simplify the line-drawings [BTS05]. However, the ε-line can not fold onto itself, and hence is not applicable to the folding or self-intersecting curves. They used the proximity of the curves as the constraint to combine the curves. Pusch et al. divided the curve into small rectangular regions until the region can only be presented by a single curve [PSNW07]. Later, they linked the small boxes to get new curves. They could not process the self-intersecting curves. Shesh and Chen used the information of the overlap, parallelism, and proximity of the strokes to combine the strokes [SC08]. Their concept of overlaping can be weak because the contour curves may not overlap that much. Orbay and Kara proposed a new method not only simplify the sketches but also beautify them [OK11]. They used trainable stroke clustering which use the neural network takes the input of the feature extracted from the stroke then decided whether the two strokes should group together. Their feature contains three parts: continuation, parallelism and proximity. They focus on the contour of sketches and did not mention the hatching.

CHAPTER 3

Research Method

In this chapter, we will introduce the concepts of input and rendering, preprocessing, segmentation, stroke translation and stroke simplification. The stroke simplification includes point pairs finding and stroke pairs finding and the combination of stroke pairs. We discuss and compare the different approaches at last.

3.1 Definition

We define the line drawing art $\mathbf{L}$ as a set of strokes in two-dimensional space. A stroke l is a continuous path which represented by a sequence of points $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, ..., \mathbf{p}_n$, where n is the number of the points and $n \geq 2$. $\mathbf{p} = (\mathbf{p}_x, \mathbf{p}_y)$ is the point on the stroke, containing three attributes: radius r_p, tangent $\mathbf{t} = (\mathbf{t}_x, \mathbf{t}_y)$ and curvature κ_p. $\mathbf{p}_x$, $\mathbf{p}_y$, r_p, $\mathbf{t}_x$, $\mathbf{t}_y$, κ_p are all real numbers. In addition, there are no neighbors with same position ($\mathbf{p}_i \neq \mathbf{p}_{i+1}$) and radius. The position and radius vary smoothly between the points. The line segment $\overline{\mathbf{p}_i, \mathbf{p}_{i+1}}$ is the line between two neighboring points of the same stroke, and the $\|\overline{\mathbf{p}_i\mathbf{p}_{i+1}}\|$ is the length of all its line segment. The length of a stroke is the sum of the length of the line segments, denoted by l_{length}.

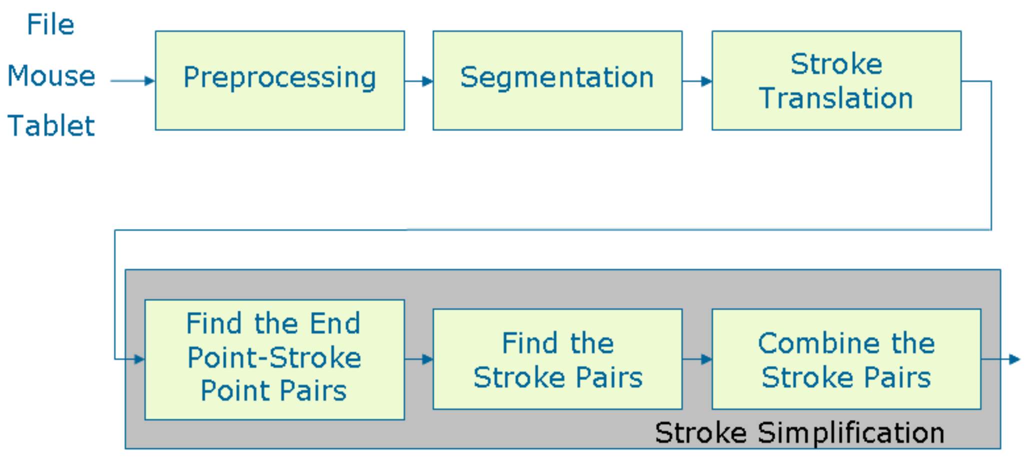


Figure 3.1: Overview.

3.2 Overview

Our thesis provides a simple algorithm to simplify the strokes. First, we calculate the attributes of the stroke's points, such as tangent and curvature. Then we segment the strokes that are too zigzag or winding, hence we truncate the points with large curvature. In the next step, we use a low-pass filter, such as Gaussian kernel to move the sparse or overwriting strokes more closer such that we can simplify them with a shorter distance tolerance. Then we find out the end point-stroke point pairs and stroke pairs by using the tangent and position of the strokes. Finally, we combine the stroke pairs and obtain a simplified line drawing art.

3.3 Input and Rendering

Our system accepts three types of the input strokes. First, read predefined input file. The file defines a stroke starting with a symbol "*", followed by a line containing the attributes of the stroke. Then, the points are defined in the following lines, each starts with a character "P", and positions $(\mathbf{p}_x, \mathbf{p}_y)$ and the radius r. For example, the file

```
*
2
P 177 366 1
P 172 369 1
P 172 374 1
*
2
P 175 352 1
P 172 354 1
```

defines two strokes. The first stroke has three points (177, 366), (172, 369), (172, 374) and the second stroke has two points (175, 352), (172, 354). The radiuses of all points are 1.

The second type of input is the strokes given by the user using the mouse. The user can input the stroke radius r_{user} and draw on the screen. When user clicks the left button of the mouse, the system starts to record points of the mouse's path. When the user releases the left button, the stroke will be created if the point set contains at least two points.

Last, the user can use the tablet to draw the strokes. The user draws on the screen with a pressure-sensitive pen. When the user put the pen-point on the tablet, the system will start to record the points on the path. After each little time period, our system will record the point according to the pen-point's position and pressure. The radius $r = \frac{f_{pen}}{f_{full}} r_{user}$, where f_{pen} is the pressure of the pen-point and f_{full} is max pressure allowed for the tablet. For examples, f_{full} for WACOM Intuos series®is 1024. When the user pulls up the pen, the stroke will be created if the point set contains at least two points.

The system can also delete the last stroke or save all strokes to the file.

We render the stroke by using Hertzmann's method [Her02] as follows:

1. Compute the tangent $\mathbf{u}$ at each point by using $\mathbf{u}_i = (\mathbf{u}_{xi}, \mathbf{u}_{yi}) = \mathbf{p}_{i+1} - \mathbf{p}_{i-1}$ and $\mathbf{u}_1 = \mathbf{p}_2 - \mathbf{p}_1$ and $\mathbf{u}_n = \mathbf{p}_n - \mathbf{p}_{n-1}$. Notice that $\mathbf{u}$ is used only for the rendering.

2. Compute stroke normal directions as $\mathbf{n}_i = (\mathbf{n}_{xi}, \mathbf{n}_{yi}) = (\mathbf{u}_{yi}, -\mathbf{u}_{xi})/\|\mathbf{u}_i\|$.

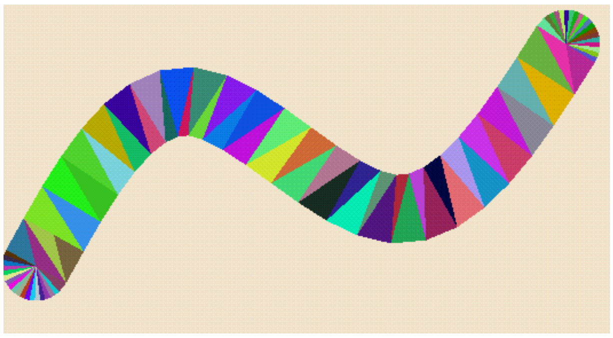


Figure 3.2: Stroke rendering [Her02].

3. Compute the boundary points of the stroke by using the radius r along the normal direction of the stroke. The points are $\mathbf{a}_i = \mathbf{p}_i + r\mathbf{n}_i$ and $\mathbf{a}_i = \mathbf{p}_i - r\mathbf{n}_i$
4. Tesselate the stroke as shown in Figure 3.2.
5. Add circular caps for end points as triangle fans.
6. Render the triangles.

3.4 Preprocessing

Before stroke simplification, we need to check the drawing range. If the drawing was drawn out of the viewport, we need to normalize the line drawing into the viewport because we will render the drawing to the texture and do the stroke translation later. Suppose the viewport size is $b_{windowsize}$. We will translate the strokes later so we make it blank at the rim of the viewport. Let rim size be $\delta_{translation}$ which is the limited offset of stroke translation distance for the next step. To transform the point $(\mathbf{p}_{ix}, \mathbf{p}_{iy})$ on the line drawing to a relative position $(\mathbf{p}_{newx}, \mathbf{p}_{newy})$ within the viewport, we do the following

$$\mathbf{p}_{newx} = \mathbf{p}_{ix} \cdot XY_{scaling} + X_{translation}, \tag{3.1}$$

$$\mathbf{p}_{newy} = \mathbf{p}_{iy} \cdot XY_{scaling} + Y_{translation}, \tag{3.2}$$

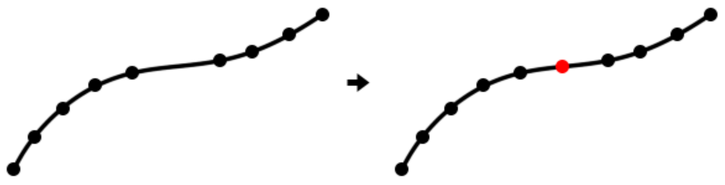


Figure 3.3: Subdivision.

since

$$\frac{\mathbf{p}_{newx} - \delta_{translation}}{\mathbf{p}_{ix} - X_{min}} = \frac{\mathbf{p}_{newy} - \delta_{translation}}{\mathbf{p}_{iy} - Y_{min}} = XY_{scaling} \tag{3.3}$$

where

$$X_{min} = min\{\mathbf{p}_{ix} | \mathbf{p}_{ix} \in l, l \in \mathbf{L}\} \tag{3.4}$$

$$Y_{min} = min\{\mathbf{p}_{iy} | \mathbf{p}_{iy} \in l, l \in \mathbf{L}\} \tag{3.5}$$

$$X_{max} = max\{\mathbf{p}_{ix} | \mathbf{p}_{ix} \in l, l \in \mathbf{L}\} \tag{3.6}$$

$$Y_{max} = max\{\mathbf{p}_{iy} | \mathbf{p}_{iy} \in l, l \in \mathbf{L}\} \tag{3.7}$$

$$X_{scaling} = \frac{b_{windowsize} - 2 \cdot \delta_{translation}}{X_{max} - X_{min}} \tag{3.8}$$

$$Y_{scaling} = \frac{b_{windowsize} - 2 \cdot \delta_{translation}}{Y_{max} - Y_{min}} \tag{3.9}$$

$$XY_{scaling} = \min(X_{scaling}, Y_{scaling}, 1) \tag{3.10}$$

$$X_{translation} = -XY_{scaling} \cdot X_{min} + \delta_{translation} \tag{3.11}$$

$$Y_{translation} = -XY_{scaling} \cdot Y_{min} + \delta_{translation}. \tag{3.12}$$

If the input points are too few or too sparse, we need to do the subdivision. It allows us to compute the distance between the strokes by calculating the distance between points of the strokes. We did not use the curve fitting or regular sampling because we do not have the parametric form of the strokes and we want to make it simple. To do the subdivision, we first calculate the distance between the neighboring points $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ and insert a new point between $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ if the distance is larger then a threshold $\delta_{subdivision}$. The newly added point $\mathbf{p}_{new}$ is the midpoint of $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$.

The subdivision is recursively applied to $\mathbf{p}_i$ and $\mathbf{p}_{new}$, and $\mathbf{p}_{new}$ and $\mathbf{p}_{i+1}$. The subdivision stop when the errors of all line segments are smaller than the $\delta_{subdivision}$. We set $\delta_{subdivision} = 5$ for all of the experiments in Chapter 4.

In the next step, we compute the tangent $\mathbf{t}$ and curvature κ for each point. Because the input may be influenced by the jiggling of the hand, we use the offset to get the reference point before and behind for the point we want to compute. The tangent and curvature of point $\mathbf{p}_i$ are defined as

$$\mathbf{t}_i = (\mathbf{p}_i^+ - \mathbf{p}_i^-)/\|\mathbf{p}_i^+ - \mathbf{p}_i^-\| \tag{3.13}$$

where

$$\mathbf{p}_i^- = \begin{cases} \mathbf{p}_1, & \text{for } i \leqslant Offset1 \quad (3.14) \\ \mathbf{p}_{i-Offset1}, & \text{for } i > Offset1 \quad (3.15) \end{cases}$$

$$\mathbf{p}_i^+ = \begin{cases} \mathbf{p}_{i+Offset1}, & \text{for } i < n - Offset1 \quad (3.16) \\ \mathbf{p}_n, & \text{for } i \geq n - Offset1 \quad (3.17) \end{cases}$$

$$\kappa_i = \frac{2 \cdot \mathbf{t}_i^- \times \mathbf{t}_i^+}{\|\mathbf{t}_i^-\| + \|\mathbf{t}_i^+\| + \|\mathbf{t}_i^- + \mathbf{t}_i^+\|} \tag{3.18}$$

where

$$\mathbf{t}_i^- = \begin{cases} \mathbf{p}_i - \mathbf{p}_1, & \text{for } i \leqslant Offset2 \quad (3.19) \\ \mathbf{p}_i - \mathbf{p}_{i-Offset2}, & \text{for } i > Offset2 \quad (3.20) \end{cases}$$

$$\mathbf{t}_i^+ = \begin{cases} \mathbf{p}_{i+Offset2} - \mathbf{p}_i, & \text{for } i < n - Offset2 \quad (3.21) \\ \mathbf{p}_n - \mathbf{p}_i, & \text{for } i \geq n - Offset2. \quad (3.22) \end{cases}$$

We set $Offset1 = 5$ and $Offset2 = 10$ in practice.

3.5 Segmentation

The winding and zigzag stroke could make it hard to be combined. So this kind of stroke need to be segmented into several new strokes. We have found out that the curvature will be large at the bent point of a stroke. Apparently, we need to cut the stroke at the points of high

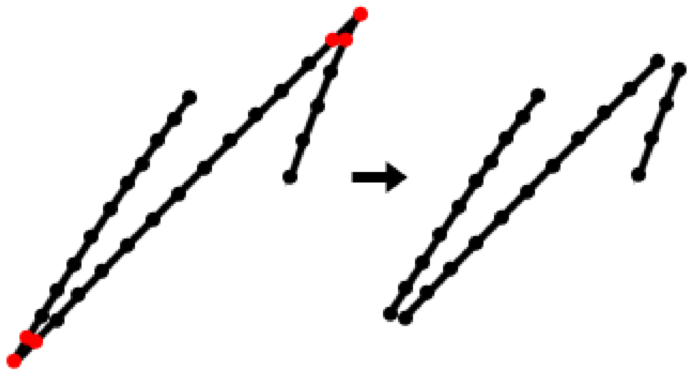


Figure 3.4: Segmentation.

curvature. However, if we just separate a stroke at the point of the largest curvature, the end points' tangents are still can not present the alignment of the stroke well. It will still cause the wrong combination of the strokes. So we cut out all of the parts with large curvatures, i.e., we remove the points if $\kappa > \delta_{curvature}$ for a user-specified $\delta_{curvature}$.

We did not consider the continuity of the strokes after they are segmented because we think it is not important for the contours or hatchings. However, it is a very interesting question of whether we need to link them after segmentation or not.

3.6 Stroke Translation

When artists want to express a thick stroke, they will draw many strokes at the same place to present the thickness. However, it will cause us hard to combine the stroke because the distance between stroke is one of the important properties for stroke combination. Sometimes, the strokes are just too sparse or overwriting. No matter what reason is, we need a method to move the strokes to the center of the artists' expectation. At least, we need to move these strokes as close as possible.

We have tried the Moving Least Squares (MLS) method but failed. MLS is a point based translation method. However, if we do not make sure each group of points can only be presented by a stroke, the points from other strokes will influence the results. If we just do the MLS after we get two strokes to combine, the results are still not good enough since the two strokes are not enough to gather the strokes to the center as we wish. Because we may not find out the center of final long stroke from these two strokes.

At last, we use a very simple method, low-pass filter to locate where the center of the strokes is. First, we draw all the strokes to a texture **T** and apply the low-pass filter to this texture. We use Gaussian kernel and standard deviation σ is calculated by the filter size δ_f. For $e^{-\frac{(\delta_f/2)^2 \cdot 2}{2\sigma^2}} < 0.05$, we need to set $\sigma^2 > -\frac{1}{4}\delta_f^2 \cdot ln(0.05)$.

The filter result of pixel **p** is

$$w(\mathbf{p},\sigma) = \int_{\mathbf{q}\in T} T(\mathbf{q})G(\mathbf{p},\mathbf{q},\sigma)d\mathbf{q} \tag{3.23}$$

where $T(\mathbf{q})$ is the value of texture **T** at pixel **q** and $G(\mathbf{p},\mathbf{q},\sigma)$ is the Gaussian kernel

$$G(\mathbf{p},\mathbf{q},\sigma) = \frac{1}{2\pi\sigma^2}e^{-\frac{\|\overrightarrow{\mathbf{pq}}\|^2}{2\sigma^2}}, \tag{3.24}$$

where $\frac{\delta_f}{2} \geq \mathbf{p}_x - \mathbf{q}_x \geq -\frac{\delta_f}{2}$ and $\frac{\delta_f}{2} \geq \mathbf{p}_y - \mathbf{q}_y \geq -\frac{\delta_f}{2}$.

We use the filter result as the weight and calculate the sum of the weight the points for each stroke l,

$$w_l = \sum_{\mathbf{p}\in l} w(\mathbf{p},\sigma). \tag{3.25}$$

Then, we need to find out the offset $\mathbf{d} = (\mathbf{d}_x, \mathbf{d}_y)$ to make the stroke with the highest weight so we give the limited offset to the x and y to all points of the stroke and calculate all cases in this small square area, where $\delta_{translation} \geq \mathbf{d}_x \geq -\delta_{translation}$ and $\delta_{translation} \geq \mathbf{d}_y \geq -\delta_{translation}$. $\delta_{translation}$ is the limited offset of stroke moving distance. The modified weight is

$$w_{l+\mathbf{d}} = \sum_{i=1}^{n} w(\mathbf{p}+\mathbf{d},\sigma). \tag{3.26}$$

We will translate the stroke to the position with the highest weight by the offset $\mathbf{d_{max}}$ where

$$\mathbf{d_{max}} = \max(w_{l+\mathbf{d}}). \tag{3.27}$$

We translate points of the stroke with the same offset instead of move the points with different offsets since we want to preserve the tangent of the line segments of the strokes. The stroke's

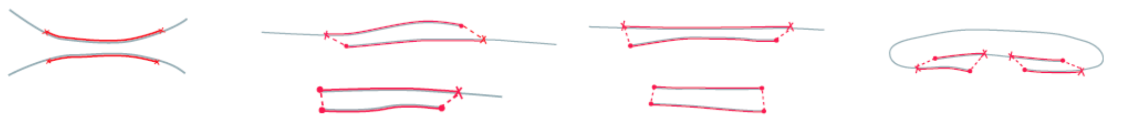

(a) Fork
(b) Four possible configurations with one path
(c) Two paths: closed curve

Figure 3.5: Conditions of stroke combination [BTS05].

new point positions are calculated as followed:

$$l_{new} = l + \mathbf{d}_{max}. \quad (3.28)$$

3.7 Stroke Simplification

There are many methods to pair and combine the strokes. The main differences are in the processing units and the conditions to pair the strokes. The processing units can be points or line segments. In this section, we will discuss how to pair the strokes in 3.7.1 and the method of stroke combination in 3.7.2.

3.7.1 Stroke Pairing

The three papers discussed in the related work section use different methods. Barla et al. [BTS05] used the definition of the ε-line to pair the strokes. They conclude that there are only 5 cases to pair the ε-line as shown in Figure 3.5(b) and Figure 3.5(c). They calculate the distances between strokes and the end of the strokes but did not mention that the unit used is point or line segment. Shesh and Chen [SC08] used points to pair the strokes. They find out every point pairs of two strokes. They pair two strokes if the percentage of the point pairs between two strokes is high. Orbay and Kara [OK11] found out the stroke pair by finding the closest points to end points then use the attributes of these points to calculate the feature of stroke pair. However, they use neural network to determine the stroke pairs at last.

We do not use the stroke overlap percentage as Shesh and Chen [SC08] did because that in most of the cases, especially for the contours, the overlap percentage does not mean a lot. In the

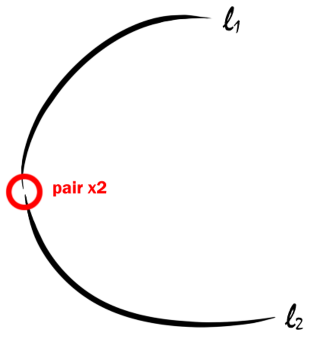


Figure 3.6: Combination.

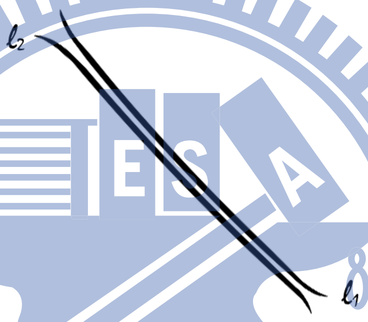


Figure 3.7: Combination of fork.

extreme case, they will overlap only at the end point of the strokes as shown in Figure 3.6. We think that the continuity [OK11] is more important for the contour case.

We believe that the first method - "the combination only occurs at the end points" is true, but we ignore the combination case as in Figure 3.5(c) because the real cases are more complicated. In most cases, the strokes will combine to strange closed strokes instead of a beautiful circle. Moreover, once the stroke is closed, it can only wait other stroke to fully overlap with it and it can not link out again. In other word, it will be harder to find the stroke pair.

We choose the first method to pair the strokes, but there is a problem - "the fork cases are very complicated too." Figure 3.5(a) is the fork case that we must avoid. But we may want to combine the strokes in Figure 3.7.

We have tried two methods but failed. First, we defined the line segments as "head", "mid-

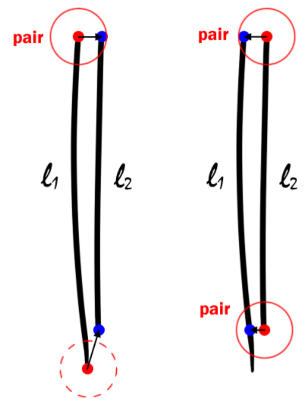


Figure 3.8: End point-stroke point pairs.

dle" or "end" by the length of the stroke and reject the line segment pairs with "middle" to "middle." We also tried another method to check only a limited percentage of the points near the end point with other strokes' line segments. The second method may be better but it is time-consuming with no outstanding result.

So we only check the error between the end points of a stroke and the points of the other stroke. For example, Figure 3.8 shows that l_1's end points can find a point pair from l_2 and l_2's end points can find two point pairs from l_1.

Although the method of P. L. Rosin [Ros94] did not present good results, they brought out the concept of the rules for simplifying the strokes. They have mentioned about the Gestalt school of psychology which studied the grouping of sensory. The following papers use this concept a lot. We also use the proximity and parallelism for our simplification.

The proximity is the spatial error between the strokes. We can also call it the distance between two strokes. Several measures have been proposed, the maximum distance [BTS05], the minimum distance [OK11], or point distances [SC08]. We calculate the distance between end points of a stroke and the points of the other strokes. There are four end points need to be checked between two strokes. So there are at most four pairs between two strokes. The error E_s is calculated by an end point $\mathbf{p}_e$ and a point from other strokes $\mathbf{p}_o$

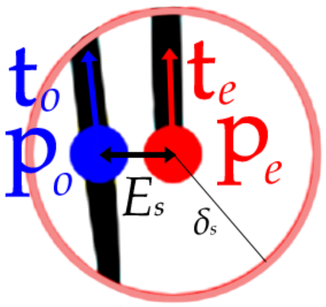


Figure 3.9: Error calculation between $\mathbf{p}_e$ and $\mathbf{p}_o$.

$$E_s = \|\overrightarrow{\mathbf{p}_e\mathbf{p}_o}\|. \tag{3.29}$$

The parallelism is the angular error between the strokes. The error E_a is calculated by the tangent of an end point $\mathbf{t}_e$ and the tangent of another stroke's point $\mathbf{t}_o$

$$E_a = \mathbf{t}_e \cdot \mathbf{t}_o, \tag{3.30}$$

where $\mathbf{t}_e$ and $\mathbf{t}_o$ are all unit vectors.

The user can set the threshold of the spatial and angular error δ_s and δ_a. Then we will record the stroke pair if there are more than two end point-stroke point pairs ray traced two strokes. We can not count the pair if there is only one pair between two strokes. It means that the strokes are arranged as a T junction as in Figure 3.10. They can not be combined to a single stroke. The end point-stroke point pair should record the stroke indexes and the overlap stroke point's index for the combination step.

3.7.2 Stroke Combination

Now, we know all the candidates for stroke pairs. We need to check if the stroke pair can be combined together or not. The additional conditions about the order of combination and the grouping of the strokes are important.

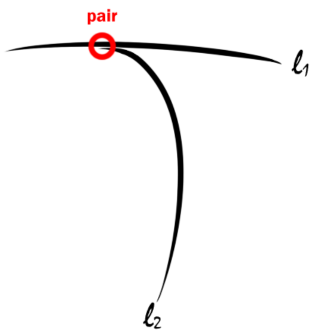


Figure 3.10: T junction.

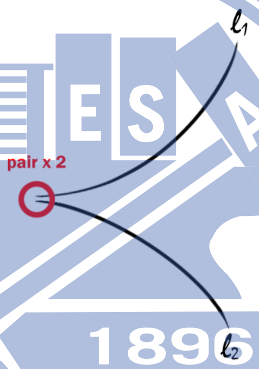


Figure 3.11: V junction.

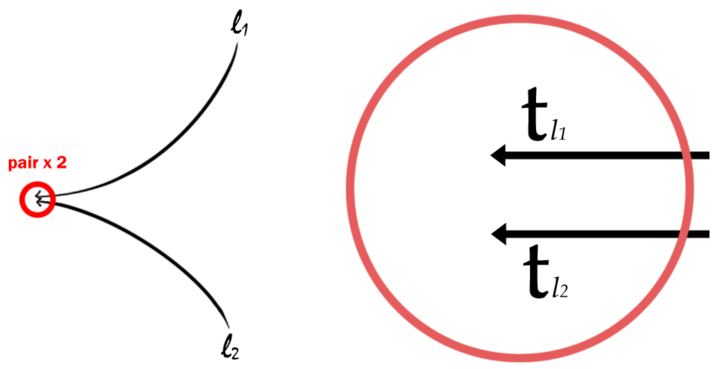


Figure 3.12: Example 1 of V junction.

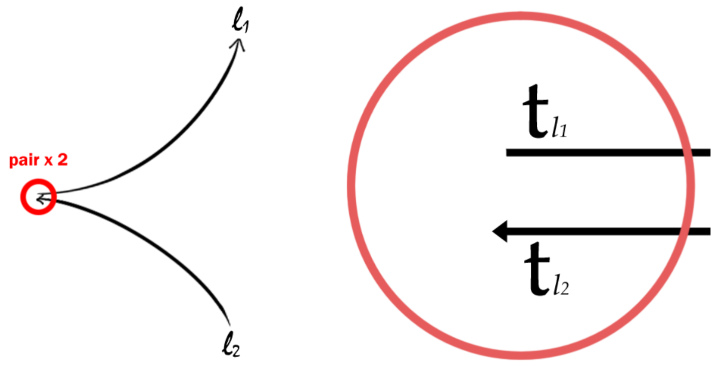


Figure 3.13: Example 2 of V junction.

The first additional condition is to prevent the case as shown in Figure 3.11. If the strokes only have two end point-stroke point pairs, we need to check their orientation. We notify the stroke 1's tangent of overlapped point $\mathbf{t}_{l1}$ and $\mathbf{t}_{l2}$ for stroke 2. We can have $\mathbf{t}_{l1} \cdot \mathbf{t}_{l2} < 0$ if stroke 1's head is paired with stroke 2's head or stroke 1's tail is paired with stroke 2's tail as shown in Figure 3.12. So as $\mathbf{t}_{l1} \cdot \mathbf{t}_{l2} > 0$ for stroke 1's head pair with stroke 2's tail or stroke 1's tail pair with stroke 2's head as shown in Figure 3.13. In the Figures, the head means the start point of the stroke, denoted as $\mathbf{p}_1$. The tail means the last point of the stroke, denoted as $\mathbf{p}_n$.

The second additional condition is to prevent the case as shown in Figure 3.14. We need to check if the strokes are fully overlap to each other at the overlap area. The point pairs define two sub-strokes of overlap area as the red part of the Figure 3.14. We need to make sure the distance between sub-strokes is smaller than the spatial error. Actually, we need to find a sequence of corresponding relationship between two strokes' points in the overlap area. However, we just simply check if all of the points can find a corresponding point on another stroke within a user-specified distance δ_s.

The order of the combination should be important too. Because after the combination, the stroke may change and can not be combined with another stroke pair candidate. One case is that it should be related to the error E_s and E_a. Another case is that it should be related to the weight texture $\mathbf{T}$. Due to the difficulty of the error update during the combination of the first case, we choose the second case.

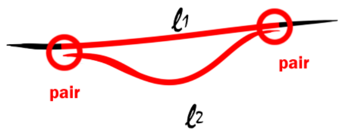


Figure 3.14: Stroke pair candidate without full overlap.

According to the drawing principles summarized by Fu et al. [FZLM11], artists mostly draw from rough to fine. They start with a rough sketch and then gradually refine the drawing strokes. So we prefer to delete the short and minor strokes first. That is, the strokes with small weights.

We calculate the weight sums of the points on a stroke l and average it with the length of the stroke l_{length} as followed:

$$\bar{w}_l = \frac{1}{l_{length}} \sum_{\forall \mathbf{p} \in l} w(\mathbf{p}, \sigma). \tag{3.31}$$

For a stroke pair, we will choose the smaller value of the strokes to represent the weight of the stroke pair. And we sort the strokes in increasing order of the weight. Then we can combine the strokes with small weight first.

The grouping of the strokes can be separated into two problems. First, we need to decide whether we need the classification of the contour and hatching and how to combine the contour and hatching. Second, the strokes should be combined by “two to one” or “many to one” or “many to many”.

At first, we want to use percentage of overlapping to classify the contour and hatching but failed. We observed that the hatchings overlap a lot so we classify the strokes with high percentage of overlapping as hatching and that with low percentage of overlapping as contour. A stroke in the gray area between the two gaps will be classified by the distance and add it into the closest group. There is another condition for the hatching group. The hatching group must have three or more strokes, otherwise it is a contour group.

In this failed method, we combine the hatching group by deleting some of the strokes in the group. Our goal is to decrease the density of the hatching group. But we use a simple method. For each pair of hatching strokes, we choose one of the strokes to be the new stroke. The parent

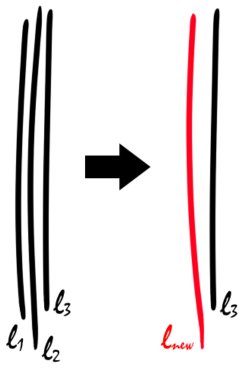


Figure 3.15: Problem of the order of combination.

pointer will point to the new stroke too. If one of the stroke l_1 has been simplified and another stroke l_2 is not simplified, l_2's parent pointer will point to the l_1's parent. Therefore, there will be only under half of the strokes left.

Although we believe that the contour and hatching need to be processed differently, we find out that our classification of the contour and hatching is still very weak. Especially for the short hatchings, we can not calculate the percentage of overlapping precisely. Finally, we give up this classification method.

The next problem is combining the strokes by “two to one” or “many to one” or “many to many”. “Two to one” is the easiest way to combine the stroke. But the order of the combination will influence the result. After some combinations, some of the strokes may not be combined as in Figure 3.15. At first, l_1 and l_2 can be combined together and l_2 and l_3 can be combined together too. After the combination of l_1 and l_2, l_3 can not be combined with the new stroke. “Many to one” can be considered the best path for all the strokes in the group. But it is hard to guarantee that there should be only one stroke after combination. Orbay and Kara [OK11] used lots of steps to eliminate this problem. Although they give many successive cases, we are still not sure if they will fail in other cases. “Many to many” also can be considered the path for all the strokes in the group. The problem is to decide the start and the end of the stroke. In the worst case, the number of the strokes may increase.

We choose the “two to one” method to combine the strokes because it is the easy and steady.

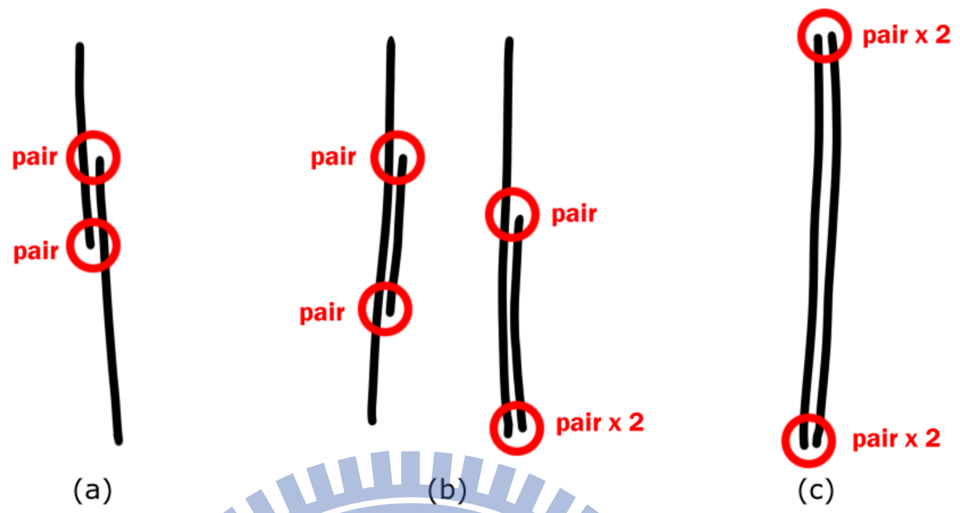


Figure 3.16: Cases of combination.

According to the order of stroke pair as we mentioned above, we will combine the stroke pairs one by one. For each stroke pair, we need to check their end point-stroke point pairs again. By the point pairs, we will classify them into three cases as in Figure 3.16. In addition, we still need to check the orientation and distance of the overlapping area as we mentioned above.

Case A is that the two strokes only have two point pairs and the end points belong to different stroke as shown in Figure 3.16(a). We will get two sequences of the points in the overlap area. The sequence of points are the end point to the point of recorded index. Then we will interpolate the points of two sequences. For two sequences $P = \{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, ..., \mathbf{p}_n\}$ and $Q = \{\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3, ..., \mathbf{q}_m\}$, we will choose the sequences with more points. For example, if $n > m$, we choose P as a base and for each $\mathbf{p}_i \in P$ we find the closest point $\mathbf{q}_{\mathbf{p}_i} = \mathbf{q}_j$ on Q and interpolate them as shown in Figure 3.17. The point positions and radiuses of the new stroke in the overlapping area are calculated by

$$\mathbf{p}_{newi} = \frac{n-i}{n}\mathbf{p}_i + \frac{i}{n}\mathbf{q}_{p_i}, \tag{3.32}$$

where $0 \leq i \leq n-1$. It is also a simplified method because we did not derive the sequence of point pair for the P and Q. The points outside of overlapping area are still at the position as

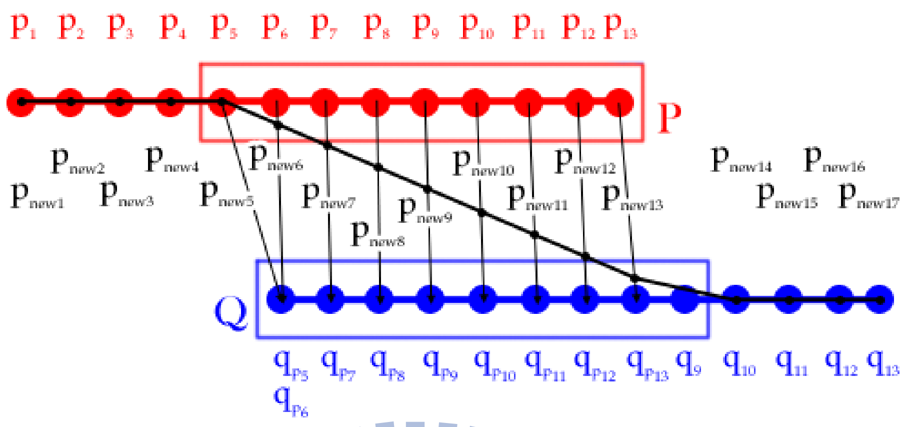

p_1 p_2 p_3 p_4 p_5 p_6 p_7 p_8 p_9 p_{10} p_{11} p_{12} p_{13}
P
p_{new1} p_{new2} p_{new3} p_{new4} p_{new5} p_{new6} p_{new7} p_{new8} p_{new9} p_{new10} p_{new11} p_{new12} p_{new13} p_{new14} p_{new15} p_{new16} p_{new17}
Q
q_{P5} q_{P7} q_{P8} q_{P9} q_{P10} q_{P11} q_{P12} q_{P13} q_9 q_{10} q_{11} q_{12} q_{13}
q_{P6}

Figure 3.17: Interpolation of Case A.

they are on the old strokes. If the radius $r < 1$, we set $r = 1$ to prevent the stroke from too thin for drawing. We need to subdivide the stroke and calculate the tangent for the new stroke for the next stroke combination.

The radius should be related to the old positions before stroke translation and every point should record the farthest combined point during the combination. And the radius should vary smoothly between points. However, we did not finish this part. We just interpolate the radius.

In Case B (Figure 3.16(b)), we only choose the stroke which is not fully overlapped to be the new stroke. In Case C (Figure 3.16(c)), we are free to choose any one of the strokes as the new stroke. After the combination of each stroke pair, we will change the parent pointer to point the new stroke. We will update the average weight $\bar{w}_l$ for the stroke pairs after each combination and then sort them in increasing order and choose the first pair in the list for pair combination. We repeat this process until all of the stroke pairs are processed. When all stroke pair candidates are processed, we complete the simplification.

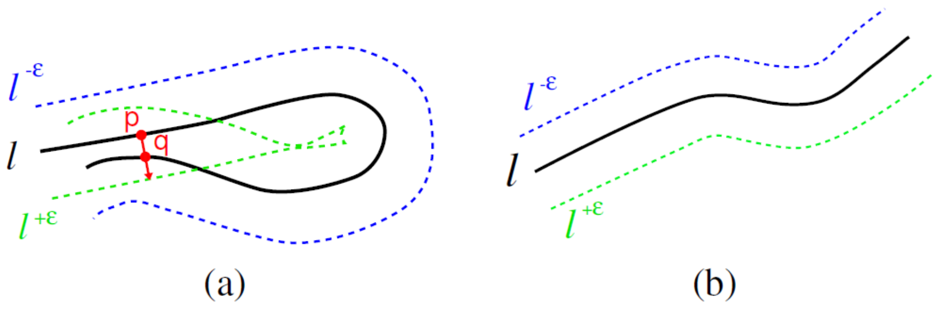


Figure 3.18: (a) l is not an ε-line. (b) l is an ε-line [BTS05].

3.8 Discussion

There are several previous work discussing line drawing simplification in the context of Non-Photorealistic Rendering (NPR). We will discuss the differences with their work in this section.

Barla et al. proposed the concept of ε-line [BTS05]. They cluster the original curves into consistent groups. Each group can be replaced by an ε-line. An ε-line is a line that does not fold onto itself at the scale ε (see Figure 3.18). Each point of the ε-line has no point along the normal at a distance less than ε that also belongs to the same line.

After they split the initial lines into ε-lines, they will find the pair of ε-lines. They have observed that only five kinds of combination for the paths can be combined as a new line (see Figure 3.5). The pair of ε-lines will be clustered as ε-groups.

An ε-group is a group of ε-lines that can be converted to an ε-line at the scale of ε, as shown in Figure 3.19. They will calculate the error for each pair of ε-lines. It is defined by the spatial error and other attributes of the lines such as color. The constraint will be set in order to forbid clustering if the line attributes are too different. They build a graph, each node represents an ε-line and each edge represents the error between two ε-lines. Then, they use a greedy algorithm to choose the edge of the smallest error at each clustering step. The algorithm stops when no more clusters can be created. In practice, they use two standards, namely pre-defined strategies to choose the new line. The average line interpolates all the original line in the cluster for the

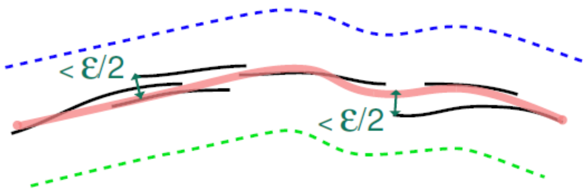


Figure 3.19: ε-group [BTS05].

contour. The most significant line is one of the original lines chosen according to an application-defined priority measure (base on length, nature...) for the hatching.

Shesh and Chen presented an efficient and dynamic line simplification [SC08]. They define a parameter δ as the maximum distance between two strokes and a parameter ρ as the percentage of overlapping between two strokes. They also define a value to measure how "near", how "local parallel" and how similar in the color of stroke pairs. They use an s-spanner [GGN06] to find the point pairing. They build the spanner by 2D points on the screen. Given the distance δ, querying the s-spanner will return all pairs of points that are within a distance δ. Then they will built two tables $C(\mathbf{p}, S)$ and $Q(S, T)$ from all the pairs returned by the spanner. $C(\mathbf{p}, S)$ maintains the "closest" points in every stroke S to a given point $\mathbf{p}$. $Q(S, T)$ maintains the geometric likelihood that stroke S and T will be paired together. They pair stroke S with a stroke T if ρ percent of the points of S are paired to some points in T with comparable corresponding local gradient. However, if multiple strokes have sufficient overlapping with a single stroke S, only the first stroke (in the order of the time they were created) will be paired with the stroke S during that frame. After they pair the strokes, they will parameterize [Gos00] and sort [KDMF03] the points in S and T into a common domain. And use Rational Gaussian Curve [Gos95] for interpolation. However, they encountered a problem when some of the lines are drawn too long and winding. So they segment such strokes by monitoring abrupt curvature changes if need.

Orbay and Kara proposed a new method not only simplifies the sketches but also beautifies them [OK11]. They used a trainable stroke clustering which uses the neural network that first takes the input of the feature extracted from the stroke and then decide whether the two strokes should be grouped together. Their feature contains three parts: remoteness (shortest distance

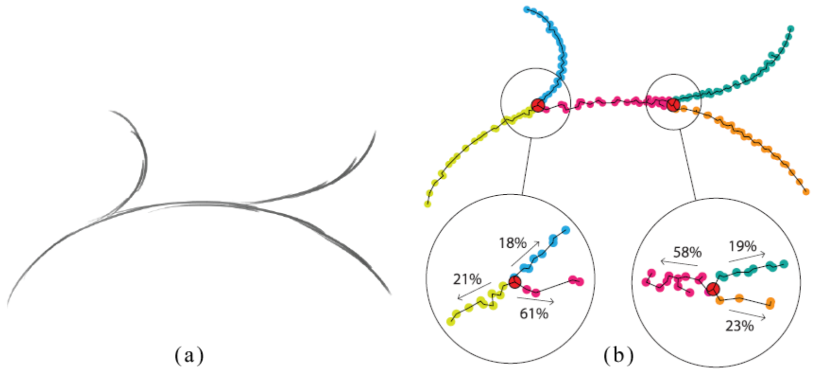


Figure 3.20: (a) original strokes (b) bifurcation points [OK11].

between two strokes), misalignment (tangent vectors at the points of minimum distance) and discontinuity (the likelihood of one stroke naturally continuing another one). After the neural network finished the bottom-up stroke fusion, they will do the top-down stroke group fission. They split the curves by calculating the minimum spanning tree (MST) [Kru56, Pri57] of the point set. Only three significant branches will be considered if the point has more than three branches. The bifurcation points (see Figure 3.20) are identified as those having a $\frac{BL_{min}}{BL_{max}} > \varepsilon$, where BL_{min} and BL_{max} are the minimum and maximum cumulative branch lengths emanating from the point, and ε is the threshold 0.05 in their implementation. For n bifurcation points, this results in $1 + 2n$ distinct point groups. The next step is merging the selective branches. For each point group, they identify the stroke that contributes the most number of points to that group (or multiple if tie) as illustrated in Figure 3.21(a). They use these strokes as the natural "skeleton" and use the tangents of these strokes' end point to decide the natural combination among the branches (see Figure 3.21(b)). In the end, for n number of bifurcation points, $1 + n$ final stroke clusters are generated (see Figure 3.21(c)). At this point, each cluster is just a set of unorganized coordinate points. They reorder the point by Laplacian Eigenmaps [BN03] as shown in Figure 3.22. Last, they beautify the reordered points by curve fitting and smoothing. They use a technique similar to moving least squares [Lee00]. Finally, the parameterization will be calculated (see Figure 3.23). The user may select to fit a cubic B-spline to the entire point set when desired. They employ a modified error function that incorporates the stylus pressure.

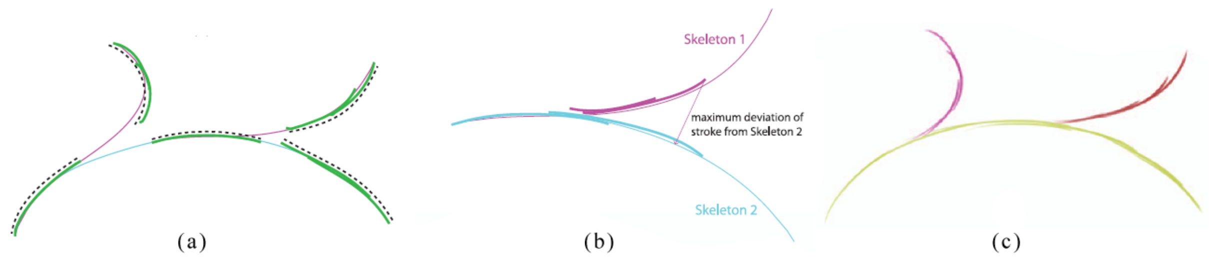


Figure 3.21: (a) natural skeleton (b) natural combination (c) final stroke clusters [OK11].

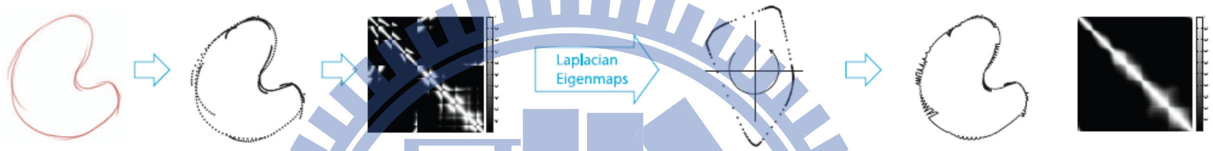


Figure 3.22: Laplacian eigenmaps [OK11].

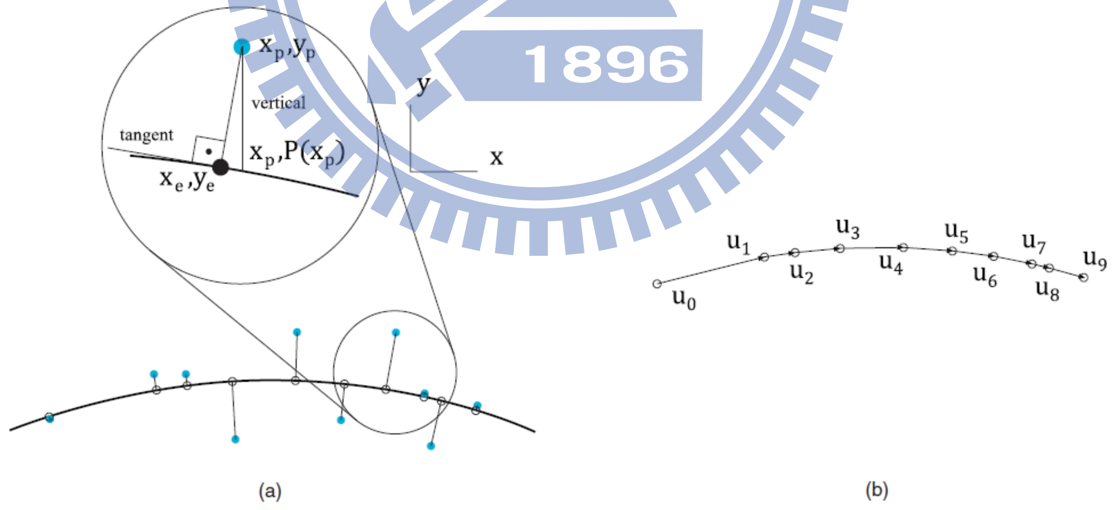


Figure 3.23: Locally quadratic curve is fit to a fixed number of points. (a) The original point $\mathbf{x}_p$, $\mathbf{y}_p$ is projected parallel to the local coordinate frame. (b) Curve parameterization is computed after the point placements [OK11].

The only thing in common is that every method uses the feature of the strokes to find the stroke pairs. The spatial error, angular error and the continuity are mostly used. Some of the methods use other attributes such as color. The detail calculations are different. For stroke combination, every method uses different ways of combining the strokes.

However, the ε-line can not fold onto itself. The method of Barla et al.[BTS05] can not deal with the folding or self-intersecting curves. Shesh and Chen's concept of overlapping [SC08] can be weak because the contour curves may not overlap that much. For the method of Orbay and Kara [OK11], they focus on the contour of sketches and did not mention the hatching. They use the neutral network, which need the training and their "many to many" combination method relies on the stability of the guarantee that there should be only one stroke after the combination.

CHAPTER 4

Results

ES A 1896

In this chapter, we present the experiment results of the stroke simplification. We present some experimental results with different parameters. Then we show the final results and compare the results of our algorithm to the results of the algorithm proposed by Barla et al. [BTS05] and Shesh and Chen [SC08]. We also list the computation time of the stroke translation, the stroke pairing and combination, and the total time.

All experiments are performed on Intel Core CPU i5-3450 3.1GHz $\times$ 4 cores with 8G ram, using NVIDIA Geforce GTX 560 graphics hardware. The operation system is Windows 7 x64. All results are rendered in 512 $\times$ 512 resolution.

4.1 Stroke Translation

There are two steps for the stroke translation. First, draw all of the strokes to the texture and do the low-pass filter using Gaussian kernel. Users can define the size of the filter. The size should cover the thickest part of the overwriting strokes. The results of using the filter size $\delta_f = 5, 10$, and 15 are shown in Figure 4.1. In Figure 4.1(a), the center of the feature strokes is not lighter than the other pixels. Use $\delta_f = 10$ or 15 will be better.

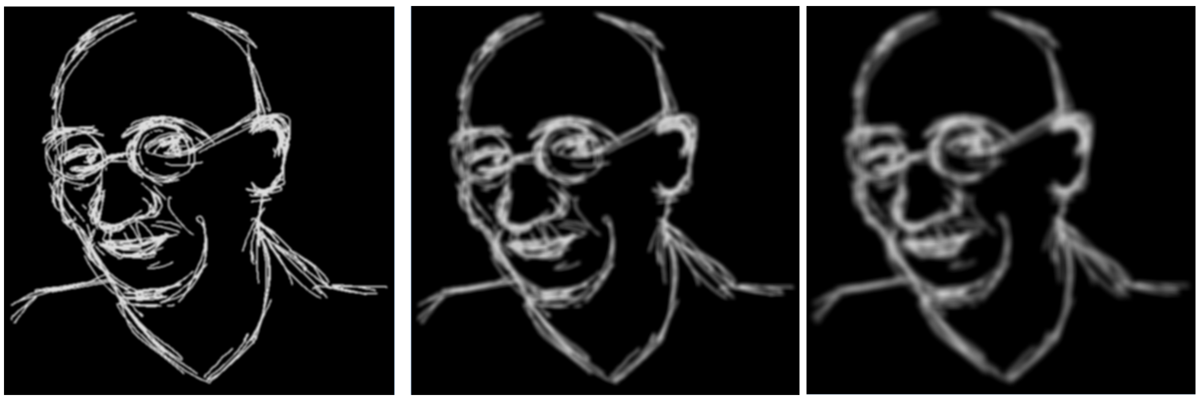

(a) $\delta_f = 5$ (b) $\delta_f = 10$ (c) $\delta_f = 15$

Figure 4.1: Results of using Gaussian low-pass filter with different filter size.

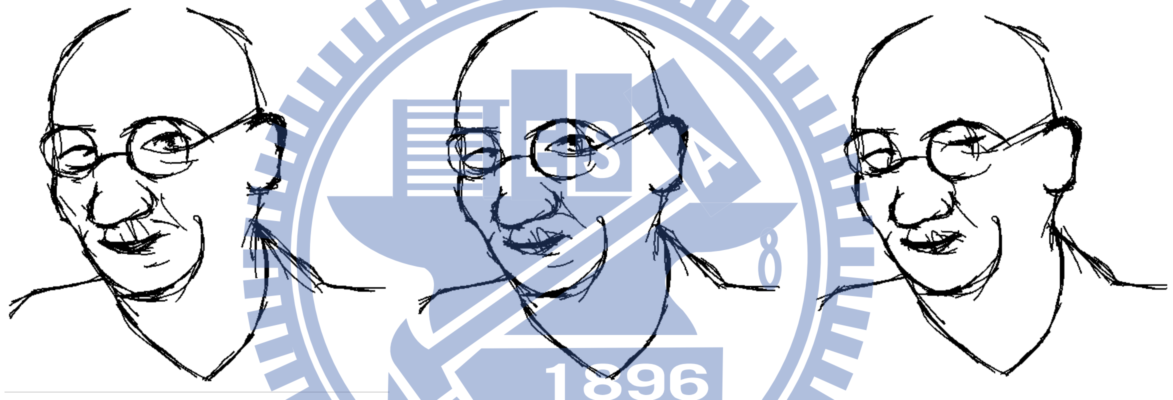
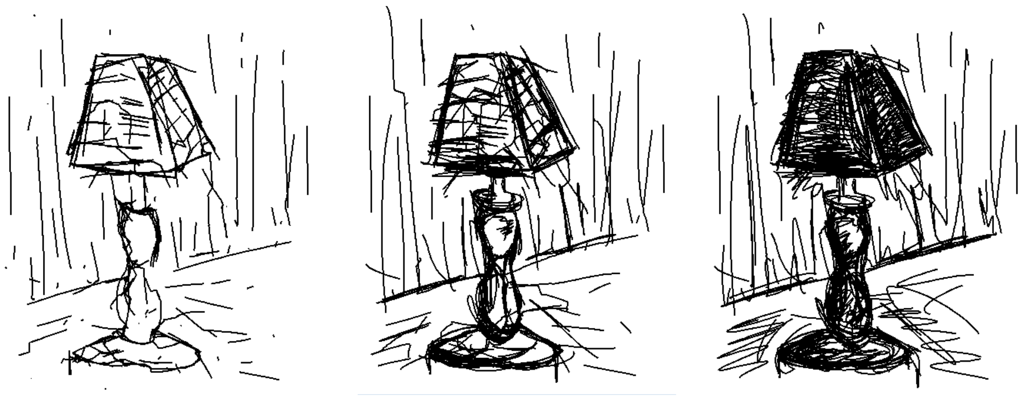

(a) $\delta_{translation} = 5$ (b) $\delta_{translation} = 10$ (c) $\delta_{translation} = 15$

Figure 4.2: Stroke translation results with different moving distance.

We choose $\delta_f = 10$ to continue the next step. The second step is to translate the strokes. We move the strokes to the position of the highest weight in the limited range. The limited range should cover the thickest part of the overwriting strokes too. The limited offset of stroke moving distance with $\delta_{translation} = 5, 10$, and 15 will produce the results shown in Figure 4.2. In Figure 4.2(c), strokes are moved too far. Use $\delta_{translation} = 5$ or 10 will be better.

With stroke translation, we can centralize the strokes and we define smaller and more appropriate spatial error δ_s for our final results.



(a) $\delta_{curvature} = 1$ (b) $\delta_{curvature} = 3$ (c) $\delta_{curvature} = 10$

Figure 4.3: Results with different curvature limit.

4.2 Experiment with Different Parameters

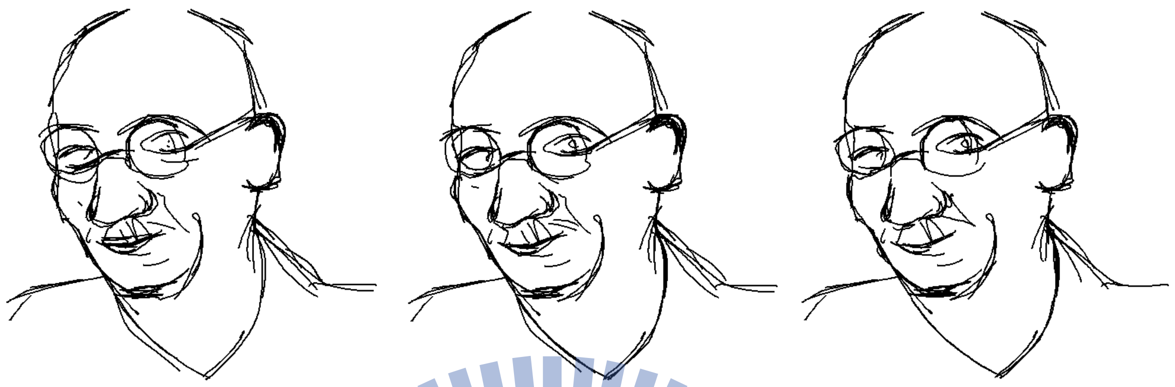
The user-defined parameters are curvature limit $\delta_{curvature}$, filter size δ_f, stroke moving distance $\delta_{translation}$, spatial error δ_s, and angular error δ_a. Each of them has influence on the results.

The curvature limit $\delta_{curvature}$ influences on how many strokes we will have after segmentation. We want to segment the strokes if they are too zigzag or winding, but the best $\delta_{curvature}$ differs from model to model.

As shown in Figure 4.3, a too small $\delta_{curvature}$ will result in too many shattered strokes, on the other hand, a too big $\delta_{curvature}$ can not segment the zigzag strokes as desired.

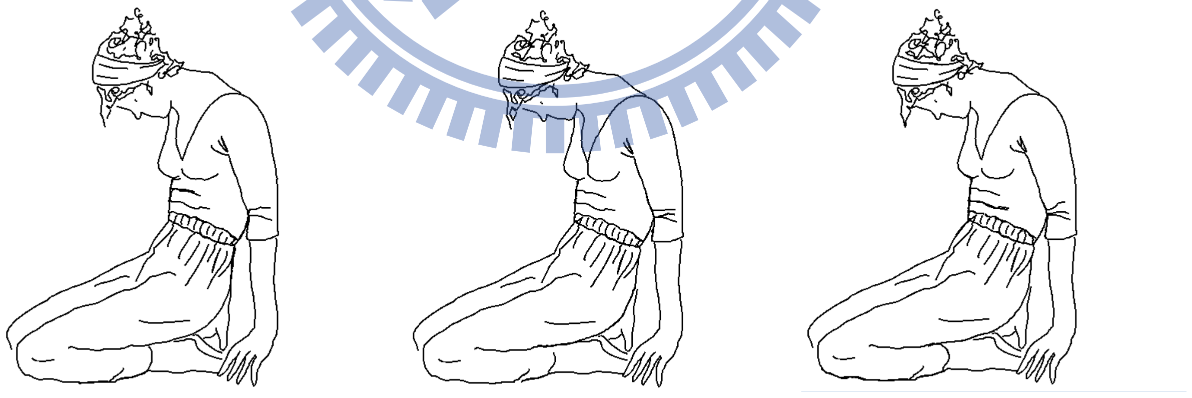
The filter size δ_f determines the weight map of the result. Not only the stroke translation but also the combination order will be based on the weight map. We use two models to demonstrate the results. One model has stroke translation and one has not. The results of using $\delta_f = 5, 7,$ and 10 are shown in Figures 4.4 and 4.5. The results are very similar but still have some slight differences.

An appropriate filter size should allow the center of the strokes obtain the highest weight. It should blur the strokes to the thick stroke that the artists want to present. We can also see that even without the stroke translation, the weight map for combination order still influence the results.



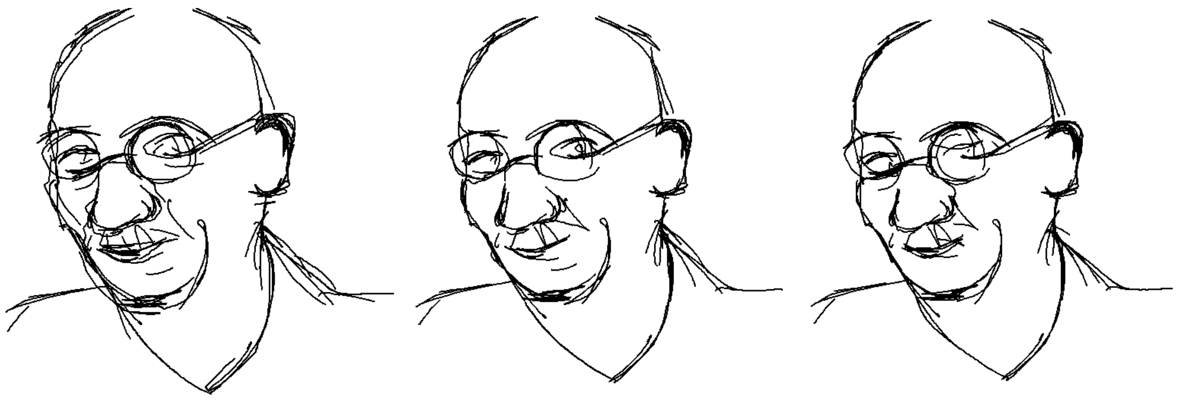
(a) $\delta_f = 5$ (b) $\delta_f = 7$ (c) $\delta_f = 10$

Figure 4.4: Results with different filter size with stroke translation.



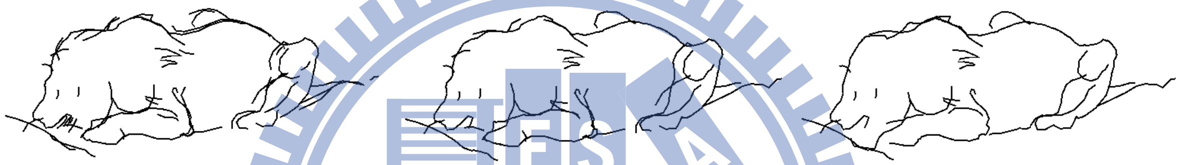
(a) $\delta_f = 5$ (b) $\delta_f = 7$ (c) $\delta_f = 10$

Figure 4.5: Results with different filter size without stroke translation.



(a) $\delta_{translation} = 0$ (b) $\delta_{translation} = 10$ (c) $\delta_{translation} = 15$

Figure 4.6: Results with different stroke moving distance.



(a) $\delta_s = 5$ (b) $\delta_s = 10$ (c) $\delta_s = 15$

Figure 4.7: Results with different spatial error.

The stroke moving distance $\delta_{translation}$ determines the effect of stroke translation. As we mentioned before, a good stroke translation can allow us define a smaller and more appropriate spatial error δ_s for the final results. We choose $\delta_{translation} = 0, 10,$ and 15 to demonstrate the results as shown in Figure 4.6.

If the $\delta_{translation}$ is too small (see Figure 4.6(a)), we can not combine the strokes with small δ_s. If the $\delta_{translation}$ is too big (see Figure 4.6(c)), the original line drawing will be destroyed as Gandhi's eyes. The stroke translation also avoid the new stroke to be attracted apart from the center of thick stroke that the artists desire. We also can avoid the problem of combination order as shown in Figure 3.15. We use this method to get the advantage of the combination method of "many to one" or "many to many". It is the most important part of our algorithm.

The spatial error δ_s is the maximum distance of the stroke combination after the stroke translation. The results using $\delta_s = 5, 10,$ and 15 pixels are shown in Figure 4.7.

A too small δ_s fails to combine strokes that are expected to be combined (see Figure 4.7(a)).

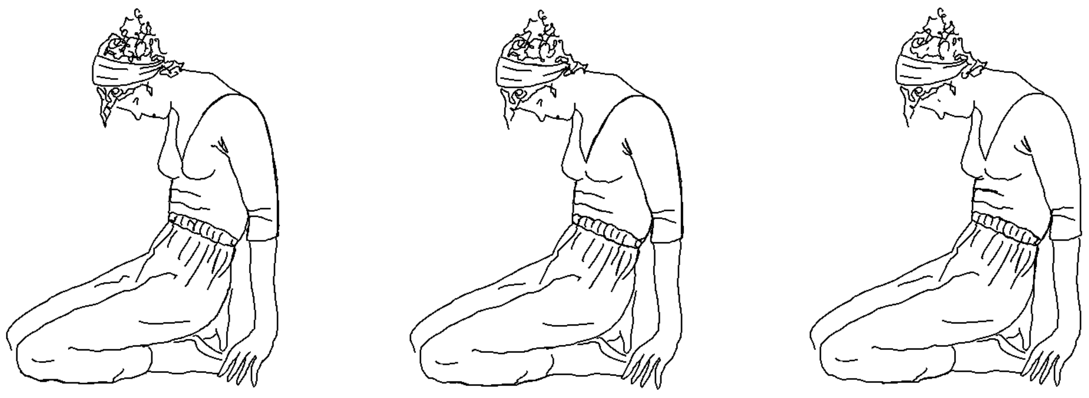

(a) $\delta_a = 30$ (b) $\delta_a = 60$ (c) $\delta_a = 90$

Figure 4.8: Results with different angular error.

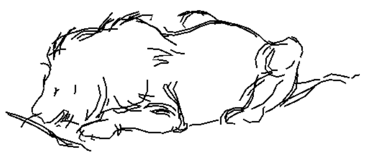
A too big δ_s can combine too many strokes than necessary and destroys the drawing such as the lion's jaw (see Figure 4.7(c)).

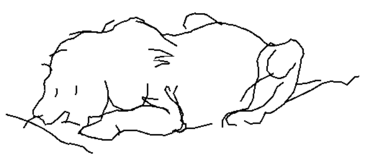
The angular error δ_a is the maximum difference of the angle for the stroke combination. The results using $\delta_a = 30, 60$, and 90 degrees are shown in Figure 4.8. The δ_a should be around 30 degrees. Most of the artists will not draw two strokes having more than 30 degrees difference and expect to link them together. However, with higher degree difference, we can combine more strokes.

4.3 Simplification Results and Comparison

In this section, we compare our simplification results to the ones proposed by Barla et al. [BTS05] and Shesh and Chen [SC08]. Barla et al. used the definition of ε-line to combine the strokes. Shesh and Chen [SC08] found the point pairs and used the percentage of overlapping to combine the strokes. We do not use the special definition nor the overlapping percentage to justify if the strokes should be combined or not. Instead, we find the end point-stroke point pairs to combine the strokes. Moreover, we use the filter result to centralize the strokes in some overwriting cases, making us easier to combine the strokes. We do not use the attribute of the color, neither.

In Figure 4.9, we do not compute the new radius by the points' distance. So the stroke's


(a) Original


(b) Our result

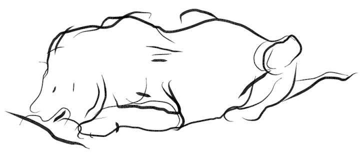
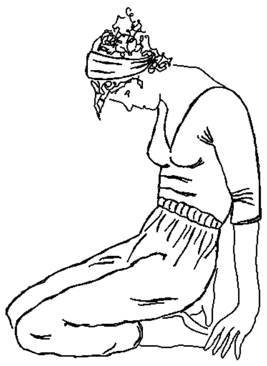
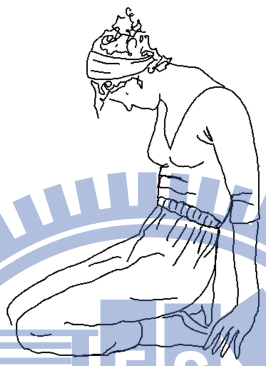

(c) Result of Barla et al. [BTS05]

Figure 4.9: Simplification results of lion.


(a) Original


(b) Our result

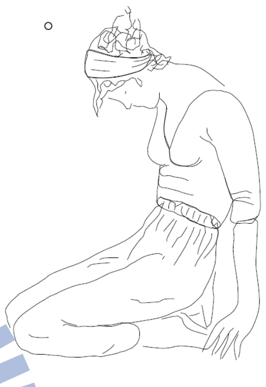
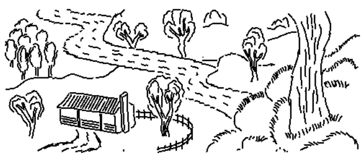

(c) Result of Barla et al. [BTS05]

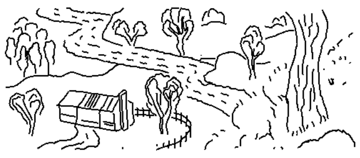
Figure 4.10: Simplification results of woman.

points' radiuses remain the same as original. We obtain a better combination at the lion's jaw and front leg. With stroke translation, we can combine the fur and the stroke of the jaw better. The strokes of the lion' s back are not combined well. We think it is due to the order of combination.

In Figure 4.10, we do not use the stroke translation. It is because this drawing does not have overwriting strokes. We get better combinations at left breast, left sleeve, the creases of the pants and waist. However, the crease of the cloth near the stomach and armpit are a little strange. The strokes are very close to each other but they are not combined. We think it is due to problems on the full overlap check.

In Figure 4.11, we do not use the stroke translation, neither. The picture has color but Barla et al. did not use the color attribute in their result. We obtain a better result for the house and the grass. But some of the long lines break at some places on the river or the road in front of the house. We think it is because the original strokes are too short and are combined based on case


(a) Original


(b) Our result

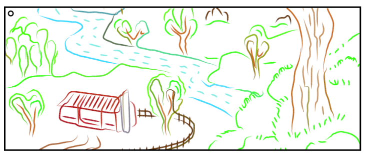
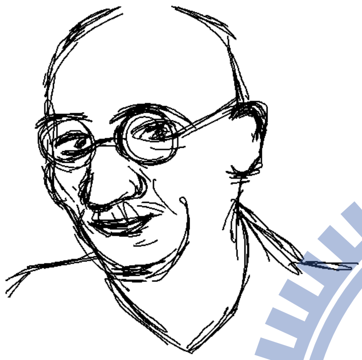
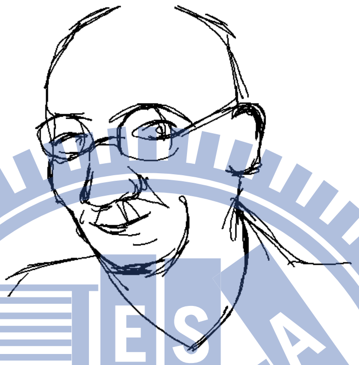

(c) Part of the Result of Barla et al. [BTS05]

Figure 4.11: Simplification results of passage.


(a) Original


(b) Our result

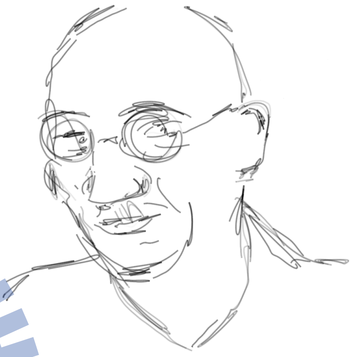
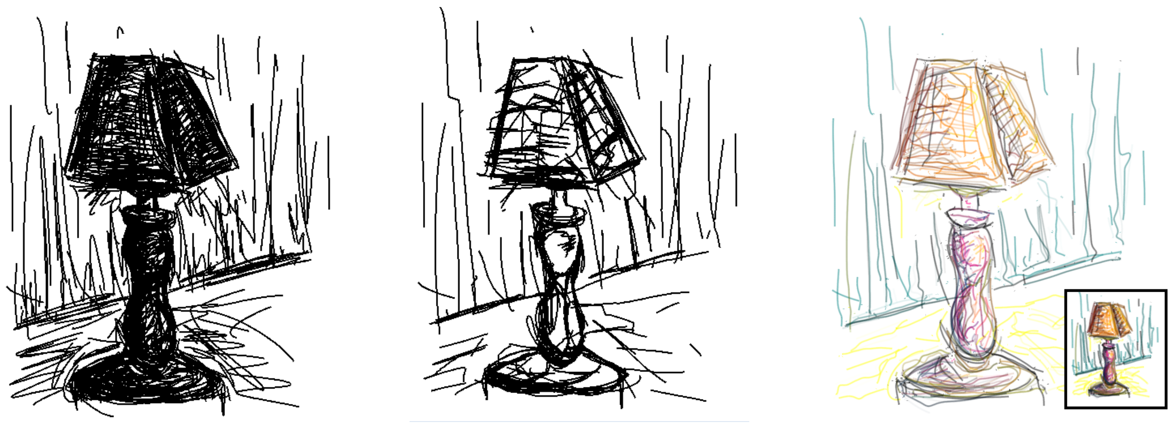

(c) Result of Shesh and Chen [SC08]

Figure 4.12: Simplification results of Gandhi.

B or case C. So we delete some of the short strokes and make the space bigger than the spatial error. The roof of the house still has some strokes. They should be combined together but they are not.

In Figure 4.12, although we use stroke translation to gather the strokes, we encounter a strange problem as we mentioned above. Even the strokes are very close to each other but they are not combined together. We still obtain better results at the shoulder, the neck and the glasses. But the contour of the head is not combined as expected.

In Figure 4.13, we use segmentation to segment the long zigzag strokes as Shesh and Chen did. But we do not use the color attribute in the combination error. We use stroke translation to move the hatching together. So we can get a clear result that does not have hatching everywhere. However, the contour are still remained with too many strokes. It should be the problem that we mentioned above and we are still trying to find why. Other test results are shown in Figures



(a) Original (b) Our result (c) Result of Shesh and Chen [SC08]

Figure 4.13: Simplification results of lamp.

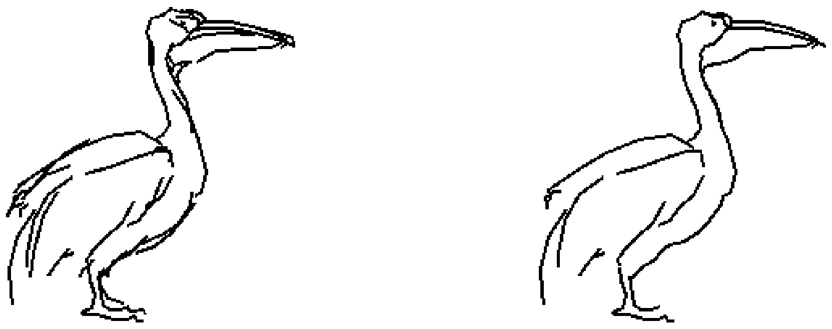
4.14 to 4.17.

The original strokes number $N_{original}$ and the number after segmentation $N_{segmentation}$ and the final number N_{final} of the strokes are shown in Table 4.1.

The stroke translation time T_t, the stroke pairing and combination time T_c and the total time T_{total} are shown in Table 4.2. Even if we do not do the stroke translation, we still need to draw the strokes to the texture for computing the weight order of the stroke combination. With bigger filter size, we also take more time to compute. And more strokes need more time to combine.

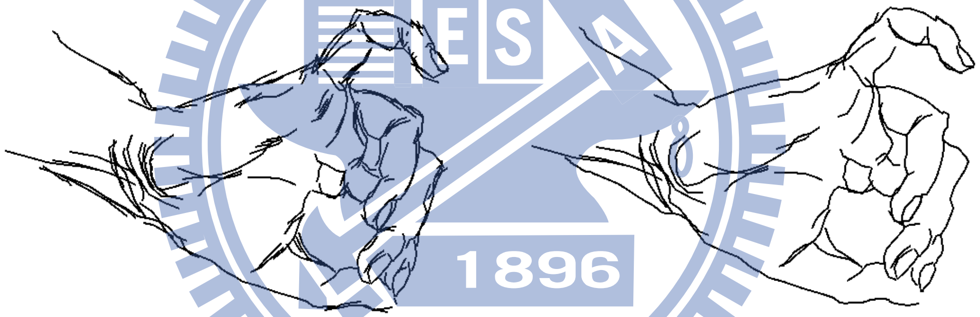
Finally, Table 4.3 shows the parameters we use for all of the testing results.

The experiments with different parameters use the same parameters as the final comparison results except the given different parameters. For example, the experiments of $\delta_{curvature}$ are the lamps shown in Figure 4.3, and their parameters are the same as the final comparison results shown in Figure 4.13. Only the parameters of $\delta_{curvature}$ are different from Figures 4.3(a) to (c).



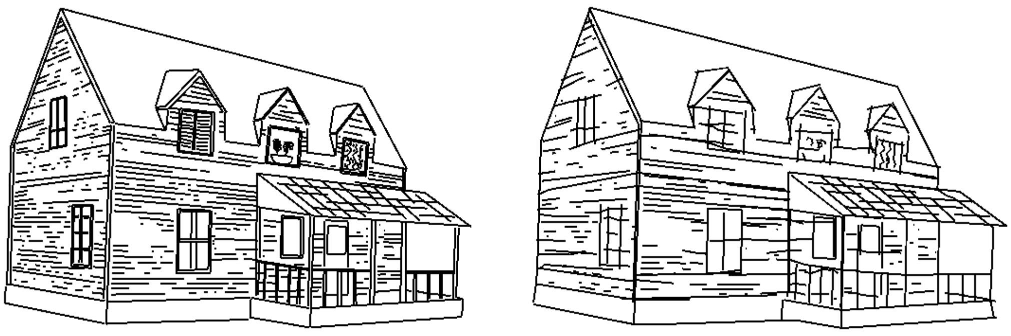
(a) Original (b) Our result

Figure 4.14: Simplification result of bird.



(a) Original (b) Our result

Figure 4.15: Simplification result of hand.



(a) Original (b) Our result

Figure 4.16: Simplification result of house.

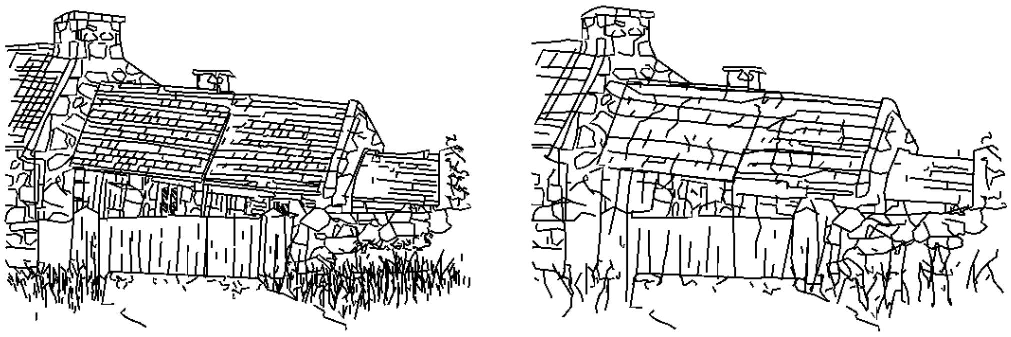

(a) Original (b) Our result

Figure 4.17: Simplification result of houses.

Drawing	$N_{original}$	$N_{segmentation}$	N_{final}
Lion	103	105	25
Woman	321	321	65
Passage	532	532	183
Gandhi	205	285	79
Lamp	406	983	179
Bird	48	48	18
Hand	171	172	45
House	648	648	290
Houses	867	867	259

Table 4.1: Stroke numbers of different drawings.

Drawing	T_t (sec)	T_c (sec)	T_{total} (sec)
Lion	0.10	0.14	0.25
Woman	0.06	1.23	1.31
Passage	0.11	5.83	5.98
Gandhi	0.19	1.09	1.31
Lamp	0.53	30.31	30.90
Bird	0.08	0.00	0.08
Hand	0.02	0.04	0.07
House	0.08	0.17	0.26
Houses	0.03	0.87	0.90

Table 4.2: Computation time of different drawings.

Drawing	$\delta_{curvature}$	δ_f	$\delta_{translation}$	δ_a	δ_s
Lion	20.0	10	10.0	70.0	10.0
Woman	2.0	5	10.0	90.0	8.0
Passage	2.0	10	10.0	60.0	6.0
Gandhi	2.5	10	10.0	30.0	10.0
Lamp	3.0	10	13.0	30.0	10.0
Bird	32.0	10	5.0	30.0	8.0
Hand	22.0	5	5.0	90.0	8.0
House	12.0	10	5.0	60.0	5.0
Houses	32.0	5	5.0	30.0	11.0

Table 4.3: Parameters of the results.

CHAPTER 5

Conclusions

We give a brief summary and conclusion of our stroke simplification algorithm in this chapter. We also propose several directions of the future work.

5.1 Summary

We have proposed a novel algorithm to simplify the line drawing. We calculate the tangent and curvature of the input strokes as stroke features. Then we segment the strokes based on the curvature and produce the strokes that are easier to be combined. After segmentation, the end points can have more appropriate tangent to present the orientation of the strokes. We apply the stroke translation to centralize the strokes. We then draw the strokes to the texture and apply a low-pass Gaussian filter. The strokes will be translated to the position of highest weight based on a weight map of the filtering results.

Then we pair the strokes based on the end points' position and tangent. We calculate the end points' spatial error and angular error with other stroke's points. After checking the conditions of combination, we pair and combine the strokes. The combination is ordered according to the weight of the strokes. At each combination step, we only deal with two strokes. We combine the

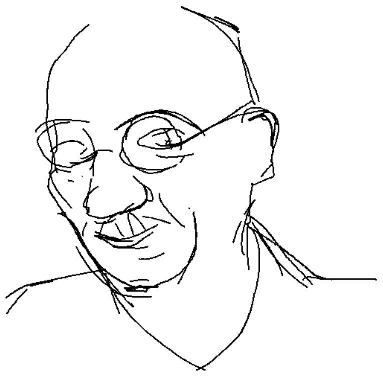


Figure 5.1: Problem of full overlap check.

pairs in an increasing order of the weight. By this simple algorithm, we can obtain a simplified line drawing.

5.2 Future Works

The combination process seems lack of some conditions, since we can not combine the strokes even if the strokes are very close to each other. Obviously, these phenomena are due to the check of full overlapping between two strokes. If we neglect this limit, some of the results will lose some strokes at strange place. But in the model "Gandhi", combination can be done better as shown in Figure 5.1. We are still finding the lacking conditions.

Although we can simplify the strokes with the proposed simple algorithm, we have to use many user-defined parameters. The users need to try and error to get the best result. They need to know the meaning of each parameter. We also have not found the best simplification order. Furthermore, we still can not simplify the strokes locally. Last, too many strokes will result in a very long computation time.

In the future, we need to solve the above three problems. We need to build a level-of-detail (LOD) structure and reduce the parameters. It should be a graph in which the points are the strokes and the edges are the error. We need to update the error after each stroke combination. We need to find an easy and fast method for the error updating. We need to find the better order of simplification. If the graph is built, this will be easy and just simplify from the smallest error edge. With the density measure, we may control the local simplification. We can also find some methods to reduce the computation time. We need to find the correct and faster method to find the series of point pairs to combine the stroke pairs. The calculation of the radius of new stroke is not finished. We have found the central of the thick stroke where the artists want but we did not calculate the radius of the thick stroke. If we have a new method to classify the contour and hatching, we may get the better results. We use the isotropic filter kernel for our simplification. There are some papers discussed the anisotropic filter [KKD09], and it may produce the better result if we use it.

There are many applications of the stroke simplification algorithm for line drawing. We can apply them to the animation or storyboard. With a series of line drawings, we may find a better way to simplify the strokes. There is another application. We can use the artist drew line-art to guide the simplification of the sketch. We can use the technique such as stroke corresponding [LCY$^+$11]. We may use a database of line-art to simplify the sketch. We can learn more about the drawing style of artists and the style of the strokes. Moreover, the beautification and smoothing of the stroke simplification can let the result be closer to the line-art.

Bibliography

[Bau94] T. Baudel. A mark-based interaction paradigm for free-hand drawing. In *Proceedings of the 7th Annual ACM Symposium on User Interface Software and Technology*, pages 185–192, 1994.

[BN03] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[BTS05] P. Barla, J. Thollot, and F. X. Sillion. Geometric clustering for line drawing simplification. In *Proc. Eurographics Symp. Rendering*, pages 183–192, 2005.

[CDF$^+$06] F. Cole, D. DeCarlo, A. Finkelstein, K. Kin, K. Morley, and A. Santella. Directing gaze in 3d models with stylized focus. In *Eurographics Symposium on Rendering*, volume 2, 2006.

[DS00] O. Deussen and T. Strothotte. Computer-generated pen-and-ink illustration of trees. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, pages 13–18, 2000.

[FZLM11] Hongbo Fu, Shizhe Zhou, Ligang Liu, and Niloy Mitra. Animated construction of line drawings. *ACM Trans. Graph. (Proceedings of ACM SIGGRAPH ASIA)*, 30(6): 133, 2011.

[GDS04] S. Grabli, F. Durand, and F.X. Sillion. Density measure for line-drawing simplification. In *Computer Graphics and Applications, 2004. PG 2004. Proceedings. 12th Pacific Conference on*, pages 309–318, 2004.

[GGN06] J. Gao, L.J. Guibas, and A. Nguyen. Deformable spanners and applications. *Computational Geometry*, 35(1):2–19, 2006.

[Gos95] A.A. Goshtasby. Geometric modelling using rational gaussian curves and surfaces. *Computer-Aided Design*, 27(5):363–375, 1995.

[Gos00] A.A. Goshtasby. Grouping and parameterizing irregularly spaced points for curve fitting. *ACM Transactions on Graphics*, 19(3):185–203, 2000.

[Her02] A. Hertzmann. Stroke-based rendering. SIGGRAPH, 2002.

[IMKT97] T. Igarashi, S. Matsuoka, S. Kawachiya, and H. Tanaka. Interactive beautification: A technique for rapid geometric design. In *Proceedings of the 10th Annual ACM Symposium on User Interface Software and Technology*, pages 105–114, 1997.

[KDMF03] R.D. Kalnins, P.L. Davidson, L. Markosian, and A. Finkelstein. Coherent stylized silhouettes. *ACM Transactions on Graphics*, 22(3):856–861, 2003.

[KDS06] L.B. Kara, C.M. D' Eramo, and K. Shimada. Pen-based styling design of 3d geometry using concept sketches and template models. In *Proceedings of the 2006 ACM Symposium Solid and Physical Modeling*, volume 6, pages 149–160, 2006.

[KKD09] J.E. Kyprianidis, H. Kang, and J. Döllner. Image and video abstraction by anisotropic kuwahara filtering. In *Computer Graphics Forum*, volume 28, pages 1955–1963, 2009.

[Kru56] J.B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956.

[LCY$^+$11] Dongquan Liu, Quan Chen, Jun Yu, Huiqin Gu, Dacheng Tao, and Hock Soon Seah. Stroke correspondence construction using manifold learning. *Comput. Graph. Forum*, 30(8):2194–2207, 2011.

[Lee00] I.K. Lee. Curve reconstruction from unorganized points. *Computer Aided Geometric Design*, 17(2):161–177, 2000.

[OK11] G. Orbay and L.B. Kara. Beautification of design sketches using trainable stroke clustering and curve fitting. *Visualization and Computer Graphics, IEEE Transactions on*, 17(5):694–708, 2011.

[PHWF01] E. Praun, H. Hoppe, M. Webb, and A. Finkelstein. Real-time hatching. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, page 581, 2001.

[Pri57] R.C. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36(6):1389–1401, 1957.

[PS95] B. Preim and T. Strothotte. Tuning rendered line-drawings. In *Proceedings of Winter School in Computer Graphics–The third Central European Conference on Computer Graphics' 95*, pages 227–237, 1995.

[PSNW07] R. Pusch, F. Samavati, A. Nasri, and B. Wyvill. Improving the sketch-based interface. *The Visual Computer*, 23(9):955–962, 2007.

[Ros94] P.L. Rosin. Grouping curved lines. In *5th British Machine Vision Conf*, pages 265–274, 1994.

[Sau92] E. Saund. Labeling of curvilinear structure across scales by token grouping. In *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR'92., 1992 IEEE Computer Society Conference on*, pages 257–263, 1992.

[SC08] A. Shesh and B. Chen. Efficient and dynamic simplification of line drawings. *Computer Graphics Forum*, 27(2):537–545, 2008.

[WM04] B. Wilson and K.L. Ma. Rendering complexity in computer-generated pen-and-ink illustrations. In *Proceedings of the 3rd International Symposium on Non-photorealistic Animation and Rendering*, pages 129–137, 2004.