PSAN2

FYP Final Report

Shape Processing and its Application to Stroke Rendering and Stylization

"An automated inking system"

by

Tsang Hao Fung

PSAN2

Advised by

Prof. Pedro SANDER

Submitted in partial fulfillment

of the requirements for COMP 4991

in the

Department of Computer Science

The Hong Kong University of Science and Technology

2013-2014

Date of submission: April 24, 2014

Abstract

This project aims to develop a stylization system that generates ink work from plain curves that can match the quality of those drawn by artists. We state the importance of stroke interactions and propose a novel data-driven model that can capture the art of inking. We address the existing limitations and present a stack of shape processing algorithms used to construct the automated inking system. Our system is able to generate plausible inking of Chinese calligraphy and cartoons at a quality level not reached before.

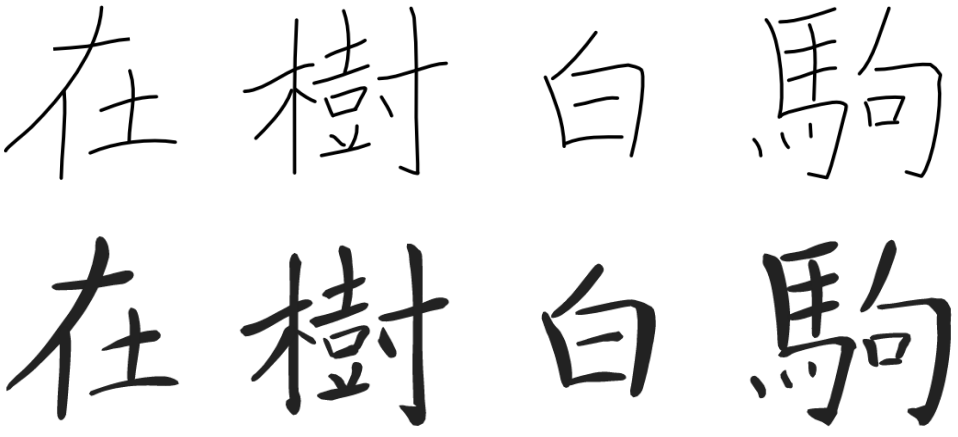

Table of Contents

1. Introduction

1.1 Overview

Motivation

Inking is one of the major steps of work in illustration. It is the process of tracing curves to form strokes. High quality strokes require careful control of a brush by an artist, and it is a time consuming work. Until now, there is no known automated inking tool that meets artists' quality requirements. A typical animation has 24 frames per second. Yet much manual work is required to ink and color each frame in the current state of the 2D animation industry. Because proper inking is very time consuming, only modest inking is utilized in most commercial animation production.

Calligraphy is a specialized form of illustration. In Chinese calligraphy, variations of strokes follow an implicit but strict set of rules. Comparing to other languages, there are not many Chinese fonts available. This is because a typical Chinese font has to support at least 10,000 distinct characters, while a major part of that is manual work. Components can be reused to generate font characters, but this cannot be applied to calligraphic fonts which have more natural variation.

The quality of strokes

Physically, to paint a stroke, an artist holds a brush or pen and drags it along the desired path, maintaining and varying the posture over the course. By carefully controlling the posture, speed and pressure, resulting strokes can exhibit a wide range of shapes. The distinctive shape of each stroke is called its quality.

Expressiveness

Expressiveness of a brush depends on the freedom of variation the brush shape exhibits along a trajectory. A flexible real brush has high expressiveness, and oriental brushes tend to produce footprints slightly more flexible than western brushes. Figure 1 shows an example of Chinese calligraphy. Observe the high variability of the strokes.

Figure 1. An example of Chinese calligraphy. Artwork courtesy 李克杰, reproduced without permission.

1.2 Objectives

The objectives of this project were

1) Investigate existing shape processing techniques and establish new ones

2) Design and develop an automated inking system

3) Determine what level of automated inking can be achieved, in theory and practice

The scope of this study was limited only to inking, so coloring and shading were not of concern. Strokes with higher expressiveness were the focus of this study.

1.3 Literature Survey

In order to achieve the objectives of this project, it was necessary to have a basic understanding of three important topics in computer graphics: stroke rendering, stylization and shape analysis and processing.

Stroke rendering

There is not a generally accepted representation of strokes. It depends highly on application requirements and the design of the graphics pipeline.

Stroke generation

Physically, a stroke is generated by dragging a brush or pen over a painting surface. The same concept can be applied to a digital stroke, where a shape representing the brush footprint moves along a trajectory. A stroke is represented by the area swept by the footprint.

The exact shape of the brush varies according to the application. In the case of fixed width strokes, the shape is simply a circle with a fixed radius. Most software designed for drawing makes use of pressure and orientation data provided by graphics tablets to change the shape dynamically to different extents.

Recently after the turn of the century, [1] and [2] presented a physically based simulation model to generate realistic brush footprints. This idea has been adopted in other research, like [3] and [4], although the simulation model varied significantly in these implementations.

Raster versus vector pipeline

In a raster pipeline, strokes are produced and stored as pixels. Raster strokes cannot be edited, so artists have to frequently undo and redo them in order to make perfect strokes. In a vector pipeline, produced strokes remain in vector representation and thus can be further manipulated. Raster graphics software typically produces more expressive strokes with a range of pixel effects that vector graphics software lacks. In 1995, TicTacToon was developed [5] to provide a complete vector workflow for professional use.

Vector representations

Output strokes have different vector representations. Systems that follow the stroke generation model typically evaluate B-spline into polyline and then perform stroke operations on the discrete point set, producing another discrete point set that describes a polygon. It can then be optionally converted back to B-splines by fitting methods like [6]. There are some exceptions, like [7], which can perform spline to spline stroke operation by direct computation following a closed form solution. However, in general, closed form solutions cannot be derived for deforming or complex brushes as stated in [8].

Skeletal strokes

Another class of stroke representation without the mental model of a *process* of moving a shape along a trajectory, is called skeletal strokes [9]. A figure is deformed by a skeleton to form a textured stroke in which the resulting stroke is naturally vivid. But this representation is less able to control the *shape* of a stroke, because the deformation is non-affine and distorts the figure to a high degree.

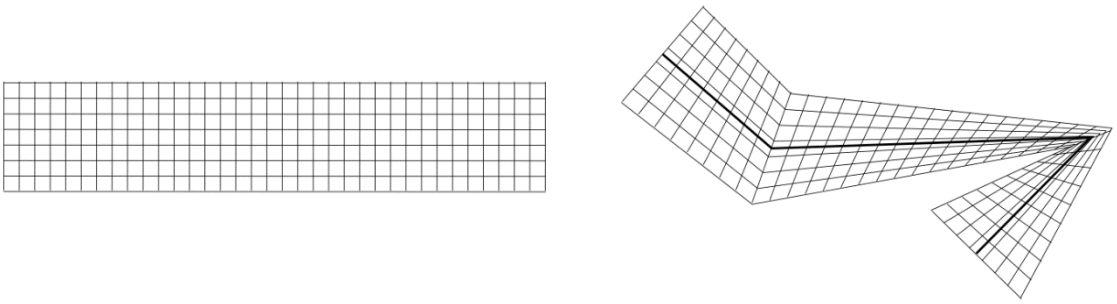

Figure 2. Original texture (left), deformed stroke (right).

Stylization

The typical application of stylization includes non-photorealistic rendering of 3D models, enhancement of 2D drawings and painterly rendering of photographs. Most stylization techniques vary only the thickness along a stroke. There are two stylization strategies, data driven and procedural.

Data driven

Data driven stylization systems have a database of example strokes. For each input curve, the system searches for the best matches in the database. Then, it transfers the style of the matched stroke to the input curve. If multiple best-matches are permitted, they will be interpolated following a certain scheme. Data driven approaches have the advantage of being representation neutral, as exemplified by [10] and [11], where the same data-driven technique is applied to two different stroke representations.

Procedural

A stylization procedure can be as simple as equations specifying the relationship between output thickness to curve curvature [12], or as complex as using machine learning techniques to determine the transformation of the brush shape [13], or, in most other literature, sine functions with random period and phase.

Literature	Stroke Representation	Stylization Strategy	Output Format
[3]	Non-physically-based brush simulation	-	Raster and vector
[7]	Circular brush along trajectory	-	Vector (B-spline)
[1],[2],[4]	Physically based brush simulation	-	Raster
[5]	Elliptic brush along trajectory	-	Vector
[9]	Skeletal stroke	-	Raster
[10]	Skeletal stroke	Data driven	Raster
[11]	Position and posture samples	Data driven	Raster and vector
[12]	Varying thickness along curve	Simple equation	Vector
[13]	Fixed brush shape, variable size and rotation along trajectory	Machine learning (Hidden Markov Models)	Raster and vector
Calligraphic brush tool in Adobe Illustrator [14]	Arbitrary brush along trajectory, curve fitting afterwards	-	Vector
Adobe Photoshop [15]	Arbitrary brush along trajectory	-	Raster

Table 1. Taxonomy of different stroke rendering and stylization implementations

Shape analysis and processing

External and internal classification

Shape analysis and processing techniques can be classified into external and internal [16]. Techniques that concern only the contour (or boundary) of shapes are external, while techniques that consider the entire volume are internal. For example, morphological image processing is a family of internal methods. Most vector graphics editors today, for example Illustrator and Inkscape, manipulate shape outlines in splines, and are thus external.

Minkowski sum-based analysis

[17] presented a set of theories and algorithms to manipulate shapes algebraically by using Minkowski addition as the basic operator and a notion of "negative shapes". The theoretic framework is powerful, but is not very suitable for stroke rendering. In particular, the theory cannot model stroke generation as a *dynamic process* of a brush that changes its shape or configuration (size and rotation) while moving along a trajectory. As a result, strokes represented by this method are restricted to fixed brushes, which do not provide enough expressiveness to strokes.

Morphological image processing

Morphological image processing (MLP) has its basic dilation operator closely related to Minkowski addition. While Minkowski sum-based analysis is generally algebraic and does not imply an image-based implementation, MLP only works with shapes represented as images. MLP algorithms have the advantage of being computationally efficient and in many cases straight forward to implement.

A well-known concept in MLP is thinning or skeletonization. It reduces any shape into a thin skeleton for further analysis. There are different thinning algorithms with different accuracy, noise sensitivity and computational complexity [18].

2. Design

2.1 Analysis

Representation

The problem of automated inking should not be viewed solely as a problem of stylization. Stylization merely generates a representation of a stylized stroke and relies on a rendering system to output the result. That means restrictions of a stroke representation are inherent to a stylization system.

A vector stroke representation is favored, because vector strokes can be interpolated among animation frames or tweaked by artists manually. As stated in the previous section, Minkowski sum-based representation is not suitable for representing expressive strokes. Hence, a contour vector representation is used throughout this system.

Stroke intersections and interactions

Stylization schemes that consider curves individually are artistically not sufficient. If a stylization system processes each curve independently without considering other curves, it is said to be a piecewise stylization system. A group consists of all curves that intersect each other. Figure 3 shows an example of inked illustration and some group patterns. Curves that intersect with each other also interact, and are rendered differently from what they would be alone. The rules observed from the interactions contribute to a large part of the illustration's style.

Figure 3. An example of inked illustration (first on left) and some group patterns (right)

Relational graph

A graph can be constructed from a group, using the curves as nodes and intersections as edges. Boolean relations (intersect or not?) among curves is not sufficient to describe a style. Attributes must be associated, for example the relative position of the intersection and the included angle. In Figure 3 & 4, it can be observed that the strokes' tapering depend on where and how the intersections occur.

Components

A Chinese character is made of several isolated groups of strokes and they are called components.

Figure 4. Some components in Chinese calligraphy

Stylization approach

As stated in the previous section, there are two approaches to stylization, namely procedural and data driven. After a comprehensive study, it is concluded that the rules are not definite and cannot be generalized without adding many special cases. In contrast, in a data driven approach, one does not need to specify the rules explicitly. Instead, the most-likely sample is selected when ambiguity occurs. In additional, a data driven system can accommodate different styles of illustration by using a different library. Thus, a data driven system is designed.

Then it raises the question of how to obtain data for stylization. The stylization system should be able to "learn" (extract data) from existing inked illustrations. High resolution scans can be readily obtained. But they must be converted to vector and processed before extracting data from them.

Stroke characteristics

The most important characteristics of a stroke are at the corners, which are associated with the local curvature characteristics of the central axis. Corners are highly specific, that means the stroke characteristics at one corner should not be applied to a non-corner curve section, and probably would not look good if applied to a different corner. Not all strokes have corners. If the central axis is largely smooth, then the characteristic feature will be the relation between thickness and curvature. Tapering occurs at both ends of a curve. The characteristic is in how fast the thickness wears off when approaching an end.

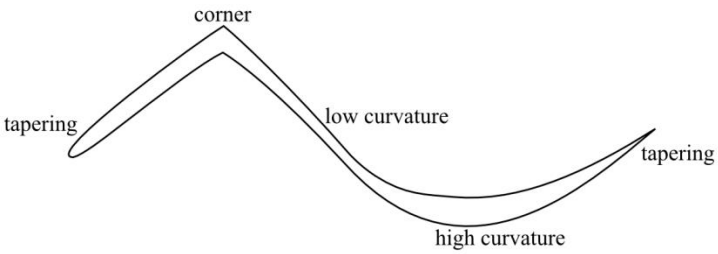

Figure 5. Stroke characteristics

Stroke model

Given a stroke and its central axis, the stroke can be regarded as a result of transformation using the central axis as skeleton. It means the stroke can be transformed inversely to get a straightened stroke. The straight stroke can then be applied to other curves. However, we argue, this model does not transfer the characteristics of a stroke well.

For simple strokes, that is, strokes with a smooth trajectory and circular brush, the skeletal model works very well (For example, strokes shown in figure 3). Simple strokes can be represented by the "varying thickness

along curve" model. Strokes shown in Figure 1 are out of this class, that they cannot be represented by the "thickness along curve" model. In skeletal stroke, there exists an inverse transform for them, but, they will be in a highly distorted form because the transformation is non-affine especially at joints. A slight perturbation to the straight stroke has the possibility to make a large visual artifact due to magnification.

A model suffers from loss of specificity when it converts information to a canonical representation. Thus, a stroke transfer model without intermediate representation is designed. The idea is to deform the sample stroke to fit the input curve directly.

2.2 The scenario

Suppose some scans of calligraphy or inked illustrations drawn by artists are obtained. Given some unstroked input curves, the system needs to apply strokes to the sketched curves in a style that mimics the reference style.

2.3 The automated inking system

The automated inking system is outlined as follows:

Stage A (offline). Library construction

1. Input images of stroke samples
2. Perform thresholding, edge extraction and vectorization
3. Decompose the shapes into individual strokes
4. Thinning: compute the central axis of each stroke
5. Compute the skeletal representation of each stroke
6. Perform component analysis
7. Build a library of component strokes

Stage B (online). Matching and stylization

1. Input digitized unstroked curves
2. Perform component analysis
3. Search for the best match in the library
4. Compute correspondence between input curves and sample curves
5. Compute the skeletal stroke deformation
6. Output the stroked set

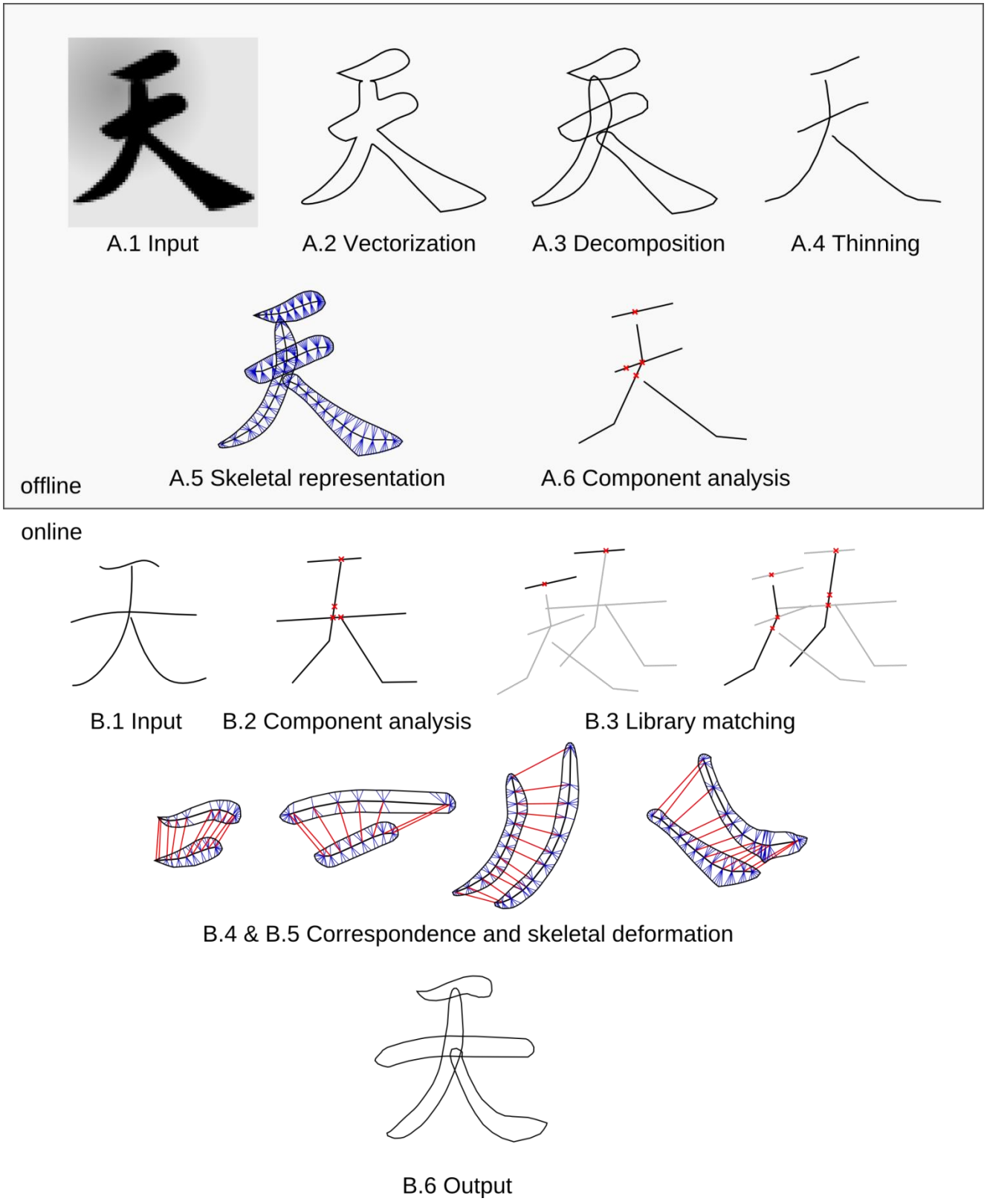

Figure 6. Visual summary of the automated inking system

3. Algorithms

3.1 Notations

The contour of a closed shape S is given as $\vec{d} = S[t]$, where *d* is a position vector and *t* is a discrete parameter. There are total *T* points, for *t* goes from 0 to *T-1*. The point at *t*=0 is called the origin and can be chosen arbitrarily along the contour. Since *S* is a closed loop, *S*[*T*] actually goes back to *S*[0].

The ∠ denotes the full-circle angle of a velocity vector.

shape contour

$$\vec{d} = S[t] \; where \; 0 \le t \le T-1 \quad (1)$$

closed loop constraint

$$S[0] = S[T] = \vec{0} \quad (2)$$

velocity

$$\vec{v}[t] = S[t+1] - S[t] \quad (3)$$

closed loop constraint

$$\sum_0^{T-1} \vec{v}[t] = \vec{0} \quad (4)$$

angle function

$$\theta = \angle\vec{v}[t] \; where \; 0 \le \theta \le 2\pi \quad (5)$$

discrete curvature

$$\nabla\theta = \frac{\angle\vec{v}[t] - \angle\vec{v}[t-1]}{|\vec{v}[t]|} \quad (6)$$

3.2 Shape decomposition

Introduction

Given a shape that is a composition of several strokes, a human observer is able to decompose the shape into individual strokes. In order to decompose a shape algorithmically, one has to define composition in a precise sense.

Basics

Composition

In general, composition is the union of several overlapping shapes. For strokes, composition is the union of two or more intersecting strokes. By intersection it means no parallel overlap or end to end connection is allowed. Intersection must yield a junction. To enable proper decomposition, composition must yield identifiable junctions. There may be some welding artifacts in junctions. There is information loss in the composition process, where sections of the outline in the junction are deleted.

Two assumptions are made about the composition operator. First, it is associative. A composition of several composites is also a valid composite, which means that it can be equivalently composed by individual strokes. Second, it is commutative. It implies if “drawing” is a process of adding one stroke at a time, then the order of adding strokes is not important.

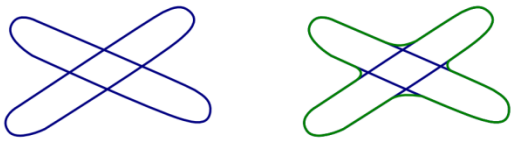

Figure 7. Composition of two sausages (blue), and the resulting contour with welding artifacts (green)

Decomposition

Decomposition is the reverse process of composition; it is an attempt to recover the individual strokes and guess the shape of the lost sections. Decomposition works by identifying and breaking the junctions on the contour. The commutative assumption allows breaking the junctions in arbitrary order, while the associative assumption allows decomposition in a recursive manner.

The algorithm

Summary

The shape decomposition algorithm is recursive on the grand scale. First, the contour is analyzed for inward corners. Then, several corners are selected to form junctions. If a junction is identified, the contour is segmented into several sections. Then, the sections are connected according to the junction type, and gaps are bridged by interpolation. The newly created shape patches are sent to the decomposition routine again. The process stops when all decomposition attempts fail.

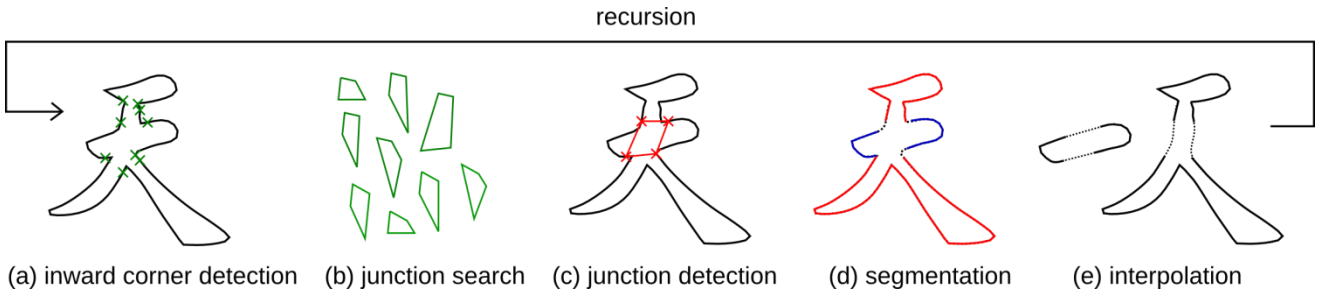

Figure 8. Visual summary of the decomposition algorithm

Inward corner detection

Assume the contour is coded anticlockwise. Then the inward corners are where the curvature shows negative impulses.

Junction search

For each inward corner, select the 7 closest corner points. In this set of 7 points, try all possible combination of 2 or 3 points (depending on the junction type). For example, to detect an X junction, 35 combinations are tried for each corner point.

Junction detection

There are 3 types of junctions presented. In theory there may be more, but in practice these are sufficient for decomposing the test shapes.

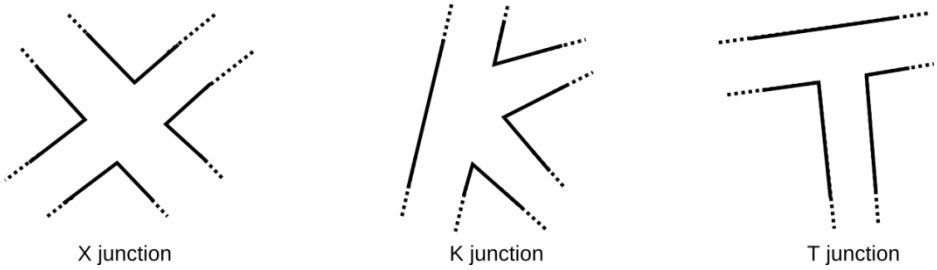

Figure 9. Junction type X, K and T

X junction

An X junction has 4 corners. There are several criteria to test for a good junction. The 4 points have to be "rectangular looking", with every internal angle within a certain range. It has to have 4 pairs of bridges.

K junction

A K junction has 3 corners. The 3 points have to be "short isosceles triangle looking", with 1 longest side and 2 shorter sides of similar length. It has 1 bridge. The bridgeable side is called the trunk. There are 2 branches.

T junction

A T junction has 2 corners. It has 1 bridge. There is 1 trunk and 1 branch.

Detection order

Because an X junction is also a valid T junction, X is always detected before trying T.

Bridge

A bridge is a potential smooth connection between two sections running in near parallel. It respects the sections before and after the bridge, making sure that the bridge *B* has small angle deviations with v_a and v_b. Several samples around the corner points are tried.

Branch

A branch is made by two sections running in the opposite sense (anti parallel).

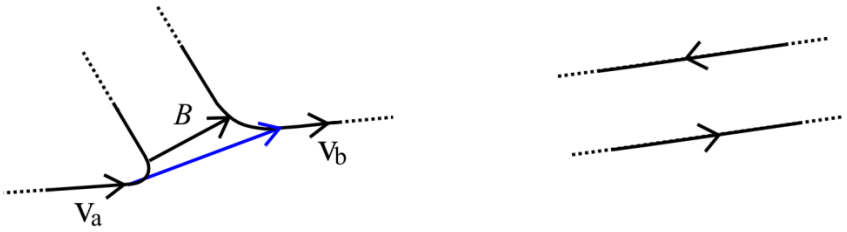

Figure 10. Bridge (left) and branch (right)

Segmentation

Segmentation has to respect the running sense of the contour. It cuts and divides the contour into multiple sections. It does not cut exactly at where the corners are, but advance or retreat a certain distance in order to remove welding artifacts.

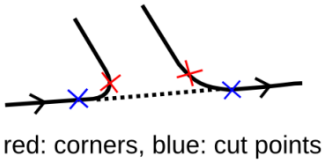

Figure 11. A smooth connection

Interpolation

When building a bridge, the velocity vectors before and after the bridge are scaled to half the distance of the gap and used to compute the control points of a cubic Bezier section. This ensures a smooth bridging.

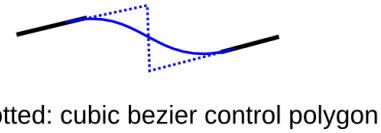

Figure 12. Interpolation

Hollow shapes

In reality, composition of several strokes can enclose a hollow space. The algorithm uses only local information around corners, without paying attention to the topology of the entire contour except at segmentation stage. If the outer contour is defined clockwise, then the inner contour must be anti-clockwise. Each topological case has to be handled specifically, and the result can be 1 to 2 contours. At the end of decomposition process, the resulting shape(s) must be non-hollow.

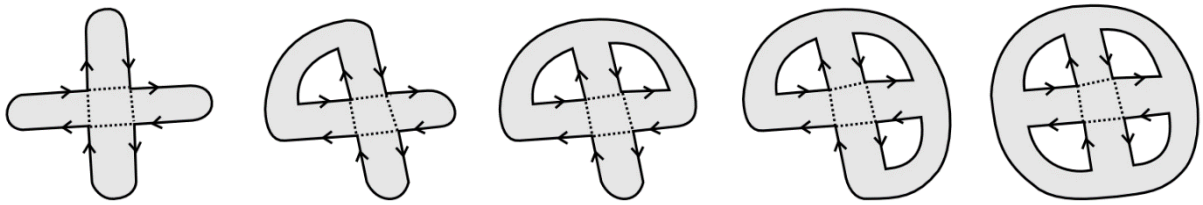

Figure 13. Topologies of X junction segmentation

Robustness

According to the composition assumption, composition can be done in arbitrary order and equivalent results are guaranteed. However, the reverse does not always hold in practice. Decomposition changes the shape and can potentially affect the decomposition result in the next iteration. If multiple junctions are detected in one iteration, choosing different ones to process first may lead to drastically different final results. The larger the welding artifact, the more likely that divergence will occur. Spikes and noise must also be removed from the contour before running the algorithm to reduce the number of false corner points.

3.3 Central axis

Introduction

As its name suggests, the central axis lies along the center of a stroke. In the generative model, a stroke is defined as “the shape produced by sweeping a convex and changing brush along a trajectory”. In reality, the

footprint of a physical brush is usually convex and longitudinally symmetric. If this is the case, then the central axis is the most reasonable guess of the original trajectory given a generated stroke.

The process to compute the central axis is known as thinning. It has its counterpart with the same name in morphological image processing. In fact, the results of morphological thinning and contour thinning are highly similar. In Figure 14, the left set of stokes is superimposed on the thinning result generated by our method, while the right shows the result generated by the Matlab command `bwmorph(Image,'thin',100);`

.

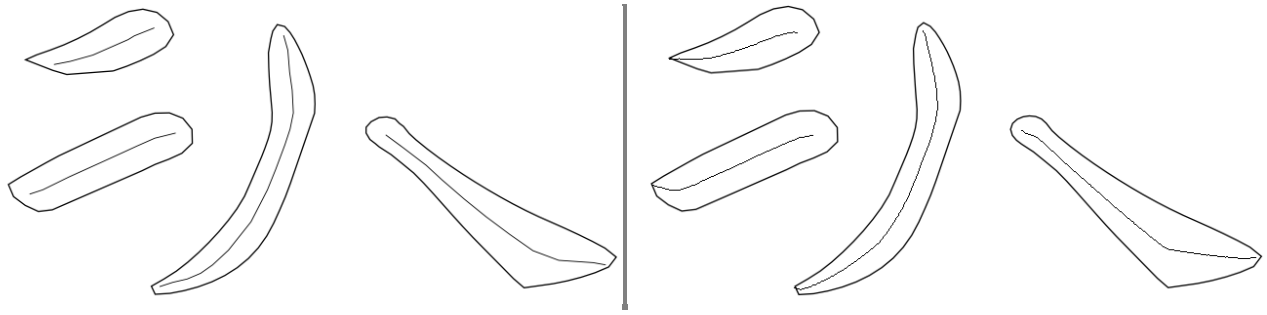

Figure 14. Our method (left); morphological thinning (right)

Advantages

It is completely feasible to rasterize the shape into a bitmap, perform morphological thinning, and then convert the resulting curve back to vector form. However there are two advantages in keeping a full-vector processing pipeline:

- Precision. There is huge amount of information loss in rasterization and backward conversion. In comparison, vector computation has a very high precision.
- Efficiency. The most accurate morphological thinning algorithms are iterative, and the number of iterations until convergence is proportional to the rendering resolution (for the same shape). The complexity is thus a cubic function of 1/precision.

The algorithm

The basic idea of the algorithm is to pair up points on the contour, and for every pair, compute the midpoint. The connected set of midpoints then form the central axis.

A pair must have velocities in opposite directions. Point **A** forms a pair by searching over a certain section on the contour and selecting the best point **B** that minimizes:

(1) The length of the section line: $\left|\vec{D}\right| = |\mathrm{A} - \mathrm{B}|$

(2) the angle difference: $|\angle\overrightarrow{v_b} + \angle\overrightarrow{v_a}|$

(3) $|\angle\overrightarrow{-D} - \angle\overrightarrow{v_a} - 90°| + |\angle\vec{D} - \angle\overrightarrow{v_b} - 90°|$

where expression (3) actually maximizes the perpendicularity between ($\overrightarrow{v_a}$ and $\vec{D}$) and ($\overrightarrow{v_b}$ and $\vec{D}$).

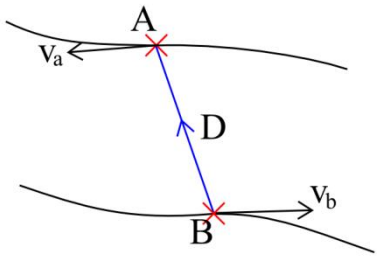

Figure 15. Pairing up points on the contour

Initially, to establish the first pair, the scope of the search is the whole contour. Starting from the first pair, the pairing process goes toward both ends of the stroke. The search for the next pair is done to a certain distance along the contour, ahead or behind the previous pair-points, which then advances A and B. The searching distance depends on the current thickness of the stroke. To illustrate, at the most abrupt corner of an ideal round jointed stroke, the contour distance to the next pair is at most πD, where D is the length of the section line. At least one of A and B must advances. The process stops when no valid next pair can be found. Because of the search-for-the-best strategy, not every point on the contour will contribute to the central axis. This is good for reducing the effect of noise and salient parts.

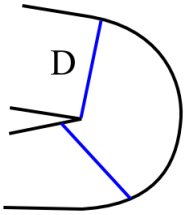

Figure 16. A sharp turn

Complexity

Given the same shape, if the number of sample points N increases, then the total amount of work of the thinning process can increase with N^2 in big O notation. However, if the concern is how the performance is affected by different shapes, then it has a different expression. It depends on the thickness of the stroke. Given the same thickness and sampling density, the amount of work actually grows linearly with the length of the stroke. In most practical cases, strokes are thin, then the complexity is close to linear. Thus it is extremely efficient to compute.

Adjusting the weights

The weight of the three minimization criteria can be adjusted to yield different results. If the weight of (1) length of the section line is increased, then the central axis will follow the tapering trend more closely.

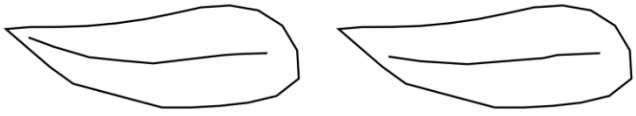

Figure 17. Pairing that favors shorter distance (left); pairing that favors smaller angle difference (right)

Example

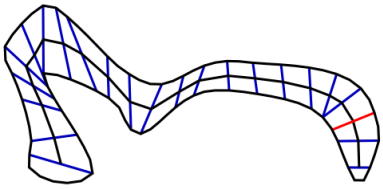

Figure 18. Thinning of a stroke. Section lines are rendered in blue. The initial pair is marked in red.

The problem of salient parts

A salient part on the contour can create a branch in the resulting skeleton. It is thus necessary to clean up the short branches (known as hairs) afterwards. Our algorithm avoids small salient parts by searching for the best pairs. A longer search distance gives a higher tolerance to salient parts.

Figure 19. The same stroke as Figure 18, thinned by morphological method

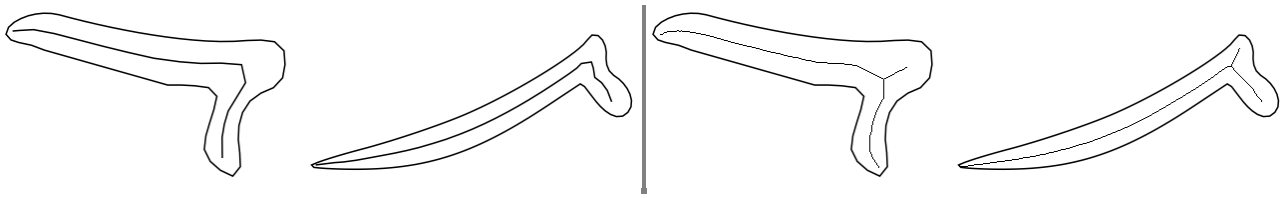

Figure 20. Our method (left); morphological thinning (right)

3.4 Skeletal stroke representation and deformation

Introduction

Not to confuse the name “skeletal stroke representation” with skeletal strokes. The algorithms given in this section coincidentally also follow a skeletal concept, but has no theoretic relation to skeletal strokes. As mentioned in the last section, stroke characteristics are strongly associated with the local characteristics of the central axis. This observation makes it reasonable to associate every point on the contour to a point on the central axis.

Skeletal stroke representation

The algorithm is quite straight forward. For every point on the contour, search for the closest visible point on the central axis and add the contour point to the list of that closest axis point. When finished, every point on the central axis has a list of one or more associated contour points. Visible means that the central axis point must connect to a contour point without occlusion.

Skeletal stroke deformation

Under the skeletal model, when the backbone (central axis) is deformed, the stroke should undergo a kind of soft body deformation.

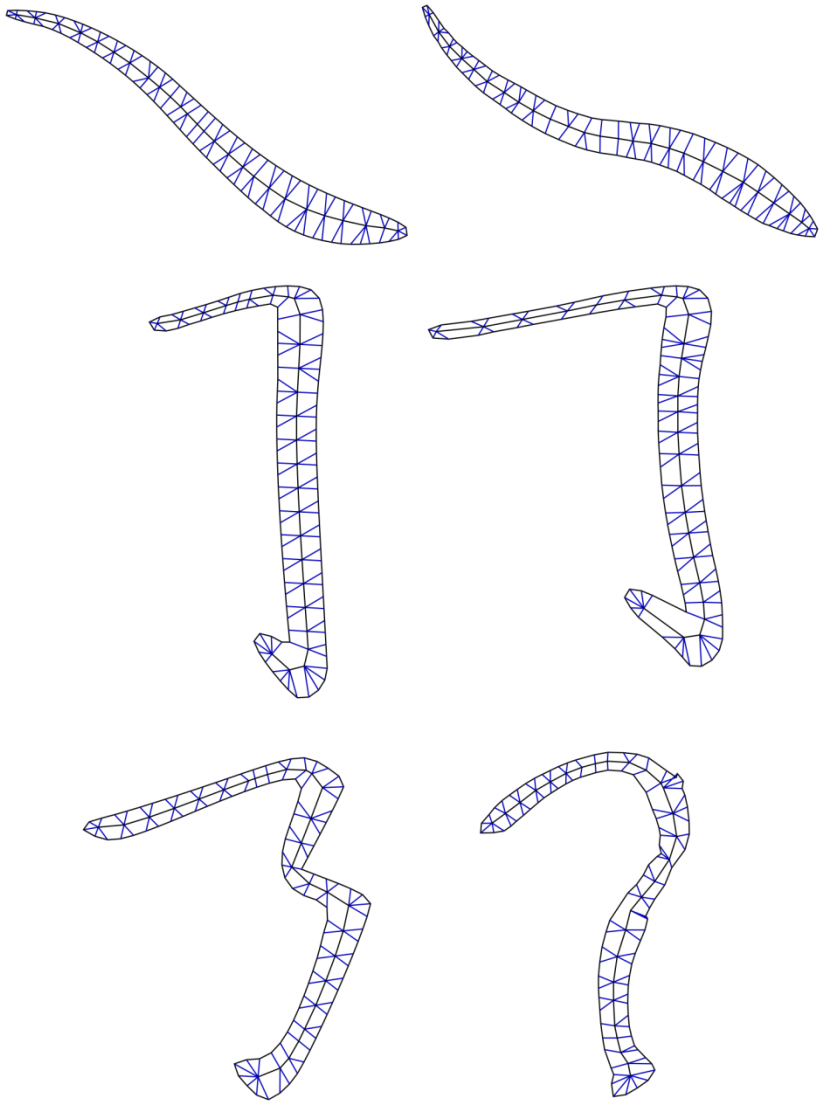

Figure 21. Skeletal stroke representations (left) and their various deformations (right). Deformation includes curvature reflex (top), contraction and elongation (middle), and freeform (bottom).

The algorithm has 4 stages: rigid deform, joint fanning, length stretch and rectify.

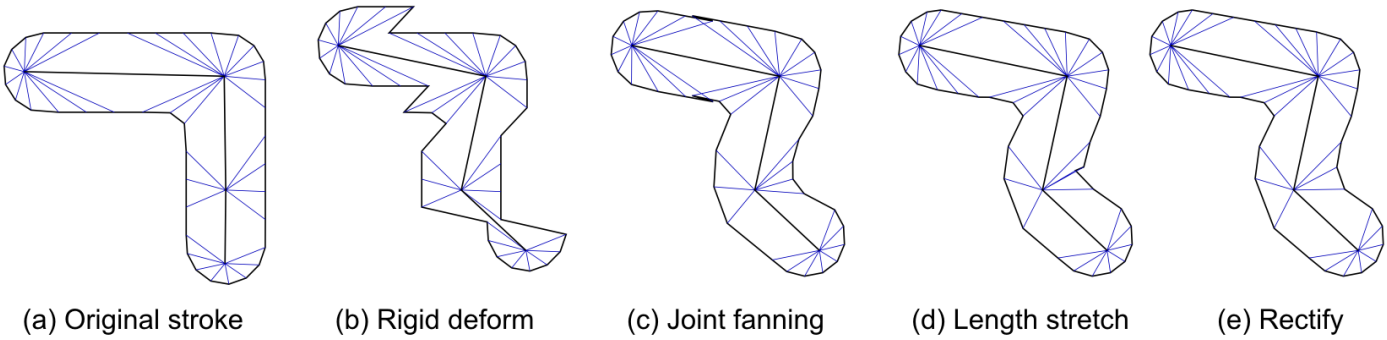

Figure 22. A visual summary of the deformation algorithm

Notation

The original stroke and principal axis is denoted as S and AX respectively. The new principal axis is denoted as NX, and the deformed stroke is denoted as N. All of them are array of points.

The skeleton, which is unchanged, is denoted as SK. SK is an array, and each element SK[i] is an array of indices x referring to some points S[x] associated with AX[i] and NX[i].

+ - * are the vector add, subtract and scale operator respectively, if the operands are vectors. `rotate` rotates a given vector for a specific angle. `rotate_to` rotates a given vector **to** a specific angle. Other vector operators do what their names suggest.

Initialize

```
N = S
```

Rigid deform

Rigid deform is applying the displacement to every stroke point, according to the displacement from AX to NX. Below is the pseudo code for rigid deform.

```
for each i in SK
      var D = NX[i]-AX[i]
      for each j in SK[i]
            N[SK[i][j]] += D
```

Joint fanning

For each joint, the “fan” is stretched linearly. Angular displacement is applied to the end points. Below is the pseudo code for joint fanning.

```
for i from 1 to SK.length-2
      var aa = angle AX[i-1]-AX[i]
      var ab = angle AX[i+1]-AX[i]
      var na = angle NX[i-1]-NX[i]
      var nb = angle NX[i+1]-NX[i]
      for each j in SK[i]
            var ap = angle S[SK[i][j]]-AX[i]
            var ar = ap/(ab-aa)
            var np = na+ar*(nb-na)
            var V = N[SK[i][j]]-AX[i]
            var NV = V rotate_to np
            N[SK[i][j]] = NX[i]+V

Var i=0
var aa = angle AX[i+1]-AX[i]
var na = angle NX[i+1]-NX[i]
for each j in SK[i]
      var V = N[SK[i][j]]-NX[i]
      var NV = V rotate na-aa
      N[SK[i][j]] = NX[i]+NV

Var i=SK.length-1
var aa = angle AX[i-1]-AX[i]
var na = angle NX[i-1]-NX[i]
for each j in SK[i]
```

```
var V = N[SK[i][j]]-NX[i]
var NV = V rotate na-aa
N[SK[i][j]] = NX[i]+NV
```

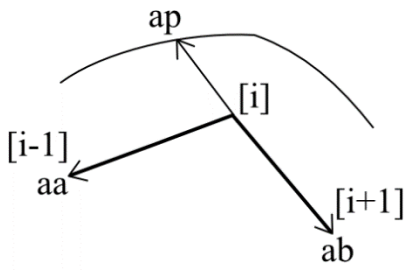

Figure 23. Vectors at a joint

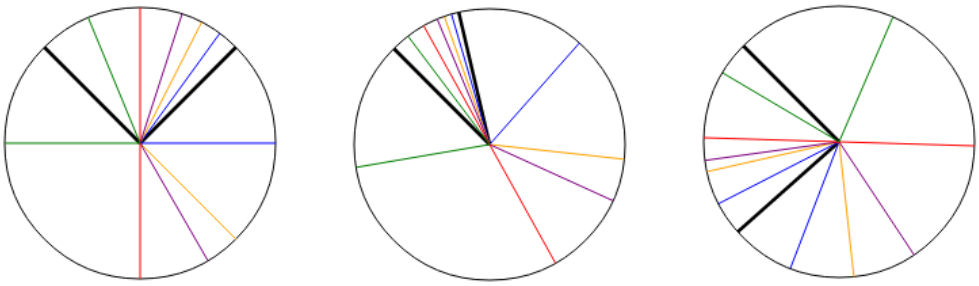

Figure 24. Joint fanning. Original fan (left), stretched fans (center and right).

Length stretch

For each point B on the contour, the component parallel to axv is scaled proportionally according to the ratio of nxv/axv, and the perpendicular component is kept as is. nxv and axv are the corresponding segment on NX and AX respectively.

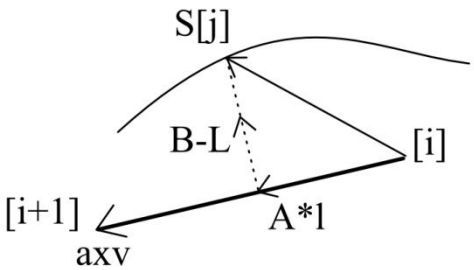

Figure 25. Projection diagram

Below is the pseudo code for length stretching.

```
for i from 0 to SK.length-2
      for each j in SK[i]
            stretch(i,SK[i][j]);
      for each j in SK[i+1]
            stretch(i,SK[i+1][j]);

function stretch(i,j)
{
      var axv = AX[i+1]-AX[i]
```

```
        var nxv = NX[i+1]-NX[i]
        var nxn = nxv rotate π/2
        var A = normalize axv
        var B = S[j]-AX[i]
        var l = A dot B
        if( 0<=l && l<=length axv)
        {
              var pl = nxv*(length A*l / length axv)
              var pp = nxn*(length B-L / length nxn)
              var NP = NX[i]+pl+pp
              N[j] = (N[j]+NP)*0.5
        }
}
```

Rectify

After length stretching, some interior fan leaves may overlap or cross each other. They must be rectified.

Conclusion

The stroke deformation algorithm is an efficient single pass computation. It may be “less correct” than an energy minimization based algorithm, but its advantages are being straight forward to implement and extremely efficient.

3.5 Curve correspondence

Introduction

The goal of correspondence is to pair up characteristically important points on the two curves. Naive correspondence is done by sampling the two curves evenly with the same number of samples, and associates the two arrays of points sequentially. It often produces bad results.

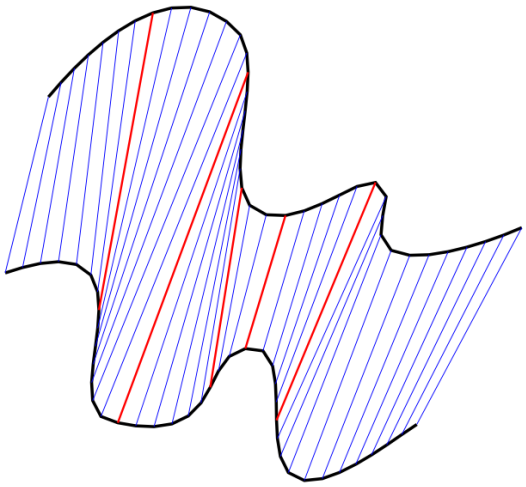

Figure 26. Naive correspondence. Both curves have 40 points. The peaks and troughs are wrongly corresponded.

The algorithm

The algorithm has 3 stages: curve simplification, correspondence and resampling.

Curve simplification

Curves are simplified to a characteristic polyline. There are two passes: lower sampling and corner merging.

Lower sampling

Starting from one end of the curve, points are dropped until the cumulative angular error reaches a certain threshold, in this case $\pi/4$.

Corner merging

Butt corner points are merged from 2 to 1.

Figure 27. Original curve (left), after lower sampling (middle), after corner merging (right).

Correspondence

The points of the two characteristic polylines are matched. For each point, the cost to every points of the other polyline is computed and the point with the lowest cost is chosen. The cost function is composed of 3 terms:

$$5 \cdot | l[i] - l[j] | + | \theta[i] - \theta[j] | + | \nabla\theta[i] - \nabla\theta[j] |$$

where i, j are indices of polyline A and B respectively. $l[x]$ denotes the normalized path length at x. θ denotes the angle of the velocity vector. $\nabla\theta$ denotes the discrete curvature.

The path length error has 5 times the weight than the other terms. Only the mutual pairs, i.e. A[i] is connected to B[j] **and** B[j] is connected to A[i], are kept. Some points will be left unmatched.

Resampling

The sections of curve between two corresponded pairs are resampled to a definite number of points.

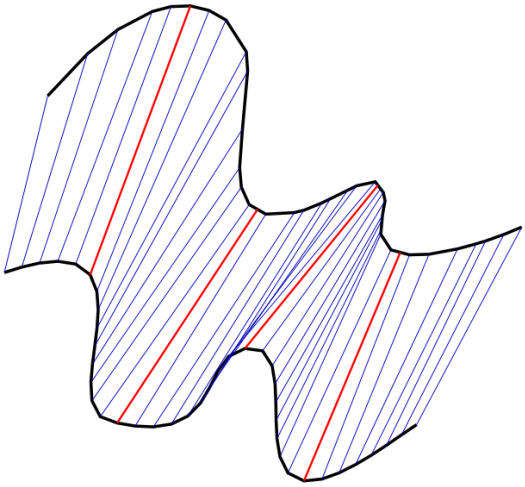

Figure 28. Correspondence result. Note the peaks and troughs are correctly matched.

3.6 Component representation and matching

Introduction

To achieve plausible stylization, the goal is to maximize matches in the attributed related graph of intersecting strokes. In unordered character recognition, there exists technique like [19] that performs maximal sub-graph matching. However the computational complexity is too high. In the context of stylization non-optimal matches impose less harm. It is decided to perform graph matching heuristically in order to reduce the runtime complexity.

Component representation

Each curve is processed and analyzed to create a component profile containing various properties of the curve.

Characteristic polyline

The algorithm stated in section 3.5 is used.

Joins

A join means that a curve's end is very close to the end of another curve. It holds 2 Boolean values.

Hooks

A hook means a short, sharp turning segment at the end of a polyline. It holds 2 Boolean values.

Dot

A Boolean value indicates whether the polyline is a short segment. A dot is likely to have a round looking stroke.

Intersects

The intersections of a polyline with all others in the same group are computed. It is an array of the following data structure:

- **a,b** is the normalized path length of where the intersection occurs on A and B respectively, where A is the characteristic polyline of the current curve, and B is the other one.
- **x,y** is the coordinate of intersection. This data is used for visualization only, not for matching.
- **ag** is the included angle at the intersection
- **i** is the id of B in the group

Other dots

A dot is usually isolated, i.e. without intersection with other components. It is often reasonable to relate them with other dots to capture the proximity. Other_dots is an array of the following data structure:

- **x,y** is the relative position of the other dot
- **i** is the id of the other dot in the group

Bound

It is a single data structure of the bounding box, containing xmin, xmax, ymin and ymax. centerx, centery, width and height can then be derived.

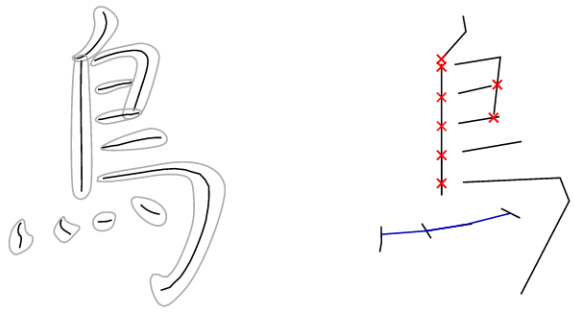

Figure 29. Visualization of a component profile. Stroke and path (left), characteristic polyline (right). Intersections are marked as red crosses and dot proximities in blue lines.

Component Analysis

Component analysis is done in the offline stage that builds a stroke library by analyzing sample inked illustrations. The image is thresholded, vectorized and decomposed into a vector graphics composed of many strokes beforehand. The strokes are grouped to reduce the computational burden of the $O(N^2)$ intersecting test. Grouping can be done manually. It can also be done automatically during earlier stages. For example, during decomposition, strokes must be in the same group if they are decomposed from the same shape.

The output stroke library is a collection of the following data structure:

- **stroke**: stroke contour
- **path**: principal axis
- **comp**: component profile

Component analysis is done online before component matching, which is the same as in offline but just without the stroke data included.

Component similarity

Component similarity is a numerical measurement of the difference between two component profiles. It constitutes of the following:

Correspondence cost

The correspondence cost from section 3.5 is used. Unmatched points add a high penalty.

Intersects

Elements from the two sets are optimally matched and the assignment cost is taken, using the following cost function:

$$\text{Cost}(A,B) = 5\cdot|\ A.a-B.a\ |\cdot w(A.a) + |\ A.b-B.b\ |\cdot w(A.b) + |\ A.ag-B.ag\ |$$

where

$$w(x) = 1.25 \cdot \left(\frac{1}{\frac{1}{2}\sqrt{2\pi}} \right) \cdot \exp\left(-\frac{(x-0.5)^2}{2(0.5)^2} \right)$$

which is a bell shaped function having its peak 1.0 at x=0.5 and falls off to 0 beyond ±1.5 range. It means that intersections located in the middle are more important than at ends. Intersections out of the polyline itself are also considered, although weighting is much lower. Unmatched points add extra penalty.

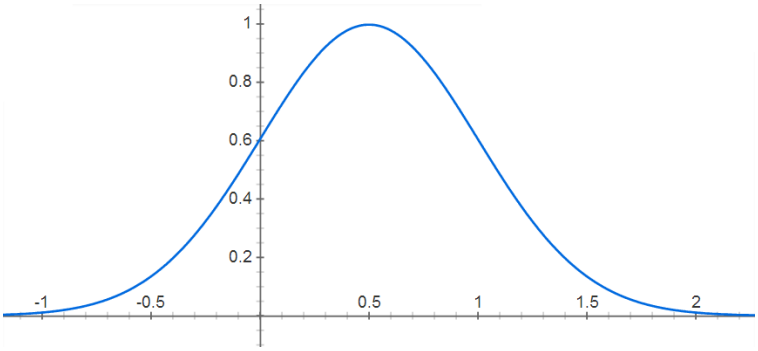

Figure 30. Graph of y=w(x)

Other dots

Elements of "other dots" are matched by the following cost function:

$$\text{Cost}(A,B) = 10 \cdot |\angle A - \angle B| + \big| |A| - |B| \big|$$

The assignment cost is taken. Unmatched points add extra penalty.

Bound error

The bound error is composed of 2 terms, aspect ratio and area error:

$$\big| \text{A.width/A.height} - \text{B.width/B.height} \big| + \big| \text{A.width} \cdot \text{A.height} - \text{B.width} \cdot \text{B.height} \big|$$

Other differences

Others differences like dot, joins and hooks add certain penalty.

Weighted sum

The above values are weight-summed to a real number. The exact weightings vary according to the style of illustration and unit of measurement.

Component matching

Given a query component profile, it is searched linearly in the library for the 5 candidates with the highest similarity.

Component promotion

To attain a maximal sub-graph match heuristically, candidates which are neighbors in both the query graph and library graph space are promoted. After that, the best match is selected for stroke transfer.

3.7 Stroke transfer

Given a sample stroke and an input curve, the goal is to transfer the characteristics of the stroke to the curve. Stroke transfer has 4 steps. First, the sample stroke is thinned to get its central axis. Second, the correspondence between the central axis and the input curve is computed. Then, the skeletal representation of the sample stroke is computed. Lastly, the input curve is used as the new backbone to deform the stroke. The quality of stroke transfer depends on the quality of correspondence and deformation.

Skeletal deformation is translation and rotation invariant, but not scale invariant. By scaling the sample stroke and its central axis before deformation, the thickness of the output stroke can be effectively controlled.

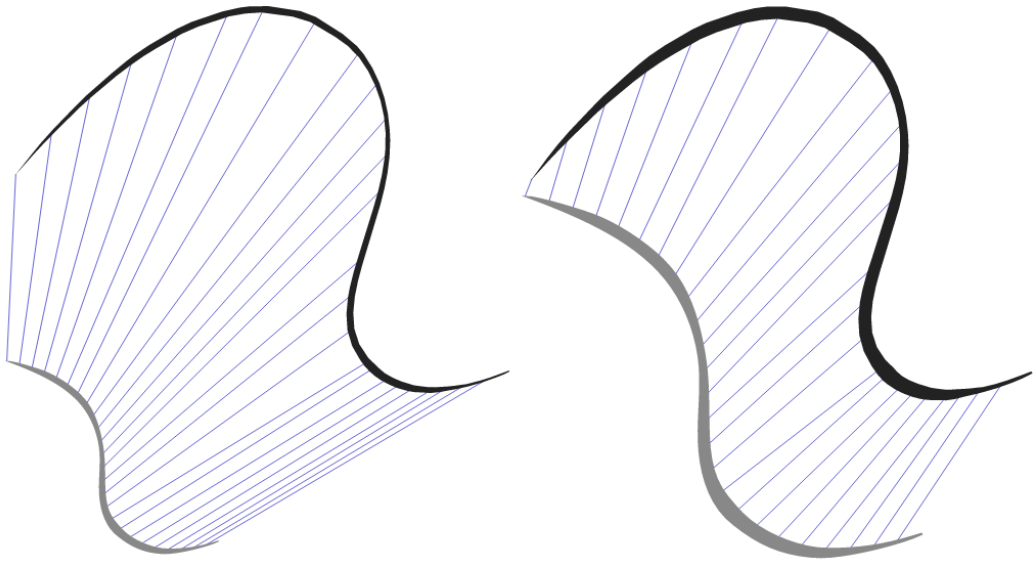

Figure 31. Result of stroke transfer (black stroke). On the right, the sample stroke (gray) is scaled up before deformation. Correspondence is rendered in blue.

4. Implementation

4.1 Vector graphics processing pipeline

Vector graphics format: SVG

Scalable Vector Graphics (SVG) is a vector graphics standard [20]. The SVG file format is XML-based which is human readable and also easy to process programmatically. It is an open standard, meaning that it is widely adopted and various tools are available to work with the SVG format. Several open source software mentioned below were the key parts of the vector graphics processing pipeline. SVG was used as the format of input and output in all implemented programs.

Vector graphics editor: Inkscape

Inkscape [21] is an open source vector graphics editor with excellent support of the SVG standard. It was used for creating and preparing shapes for input to the system, and inspects the output shapes.

Development platform: HTML5

There are two major open source implementations of HTML5, namely Chromium [22] and Firefox [23]. SVG is part of the HTML5 standard and browsers had excellent support to SVG. Development was mainly done in Chromium. The browser was leveraged to parse, render and export SVG graphics. The SVG API, particularly the functions `getTotalLength()` and `getPointAtLength()` were used extensively.

Support library: Raphael

Raphael [24] is a wrapper library over the browser's native SVG API to make it easier to render SVG graphics. It had some important functions, like point-in-shape test, shape intersection and shape interpolation which were used in the implementation.

Vectorization utility: Potrace

Potrace [25] is a polygon-based tracing library. It was especially designed for tracing line and solid-filled graphics and can produce high quality vector output. It was used to vectorize scans of illustrations in the first step of the processing pipeline.

4.2 User interface

The software implementation in this project was meant to be a prototype. Its primary purpose was to demonstrate the functionality and feasibility of the proposed system and individual algorithms. Multiple interfaces were built, one for the integrated system and others for individual algorithms. The interface for testing skeletal deformation is interactive, that the skeleton can be dragged around to test the dynamic behavior of the algorithm. Other interfaces are non-interactive. At the end of this project, an interactive sketching interface will be implemented, to allow users to sketch freely and see the result of stylization in real time.

4.3 Conventions

A lower-right-origin coordinate space was used. Outer contours were encoded clockwise and inner contours are encoded anticlockwise. All measurements were dimensionless. Angles were from $-\pi$ to π, and always wrapped during computation. All shapes were resized to fit within the {xmin:0, ymin:0, xmax:1, ymax:1} bounding box, such that the longest distance from any two points is sqrt 2. Path lengths were normalized to the 0 to 1 range.

4.4 Optimal assignment

The classic Hungarian algorithm was used for optimal assignment. The complexity is $O(N^3)$ but its efficiency was not a major concern, because in most cases there were only less than 5-5 elements to assign.

5. Testing

Unit testing was carried out to ensure that each component of the system works as specified. The test shapes were sequenced from simple to complex. Specific shapes were created to test particular weakness of the algorithms. Unit testing is especially important during development to ensure that an enhancement does not cause a previously passed test case to fail.

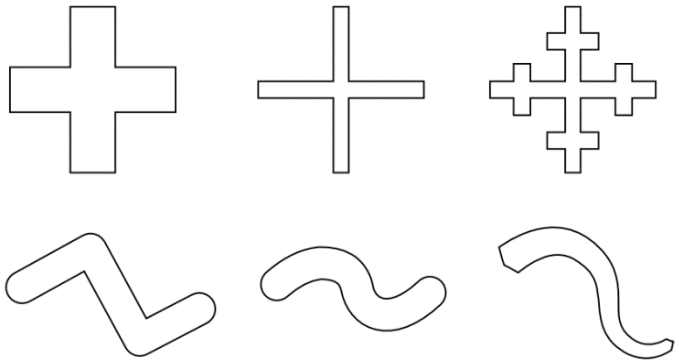

Figure 32. Test shapes for decomposition (top row), test shapes for thinning (bottom row). Complexity increases to the right.

Figure 33. Test curves for correspondence

Skeletal deformation was tested by hand using the interactive interface and observing its dynamic behavior. Other subsystems were tested using real world data.

6. Evaluation

The automated inking system was evaluated by processing inked illustrations drawn by artists. The objective of this project is to develop a general automated inking system. Two very different forms, cartoon and Chinese calligraphy were chosen.

Cartoon was chosen because it is a very prevalent form of inked illustrations. And indeed the animation making industry needs automated inking the most. Chinese calligraphy was chosen because the strokes are thick, have high variability and many intersections. These characteristics create problems which existing methods cannot tackle well. They are thus theoretically interesting to study.

All evaluation was done on a development machine with a 1.00 GHz AMD C-60 CPU.

6.1 Artwork used

These illustrations were collected and used to evaluate the system:

Image 1. A Cartoon. Artwork courtesy AnarchyWulf, reproduced without permission.

Image 2. A frame extracted from the animation ***Toy Tinkers***, 1949 by Walt Disney Productions, reproduced without permission.

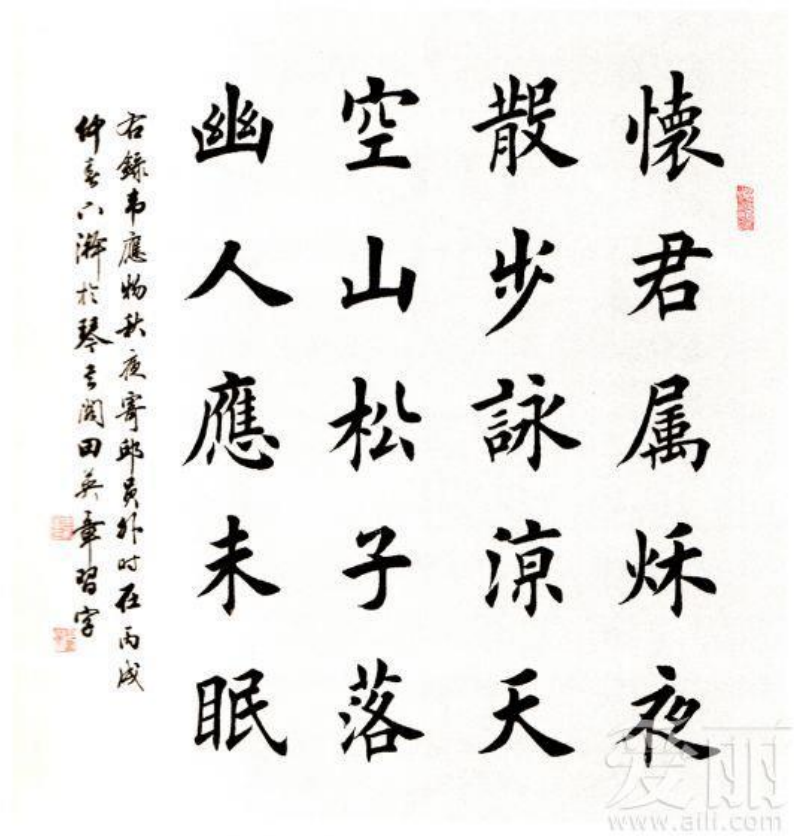

Image 3. A Chinese calligraphy. Artwork courtesy 王彬, reproduced without permission.

Image 4. A Chinese calligraphy. Artwork courtesy 萬承毅, reproduced without permission.

Image 5. A Chinese calligraphy. Written by unknown, reproduced without permission.

6.2 Decomposition and thinning

5 characters were extracted from Image 3 to use as input to the decomposition system. 2 other characters were drawn using a real brush, scanned and vectorized to use as input.

Figure 34. Input shapes extracted from Image 3 (top row), decomposition result (middle), thinning result (bottom row).

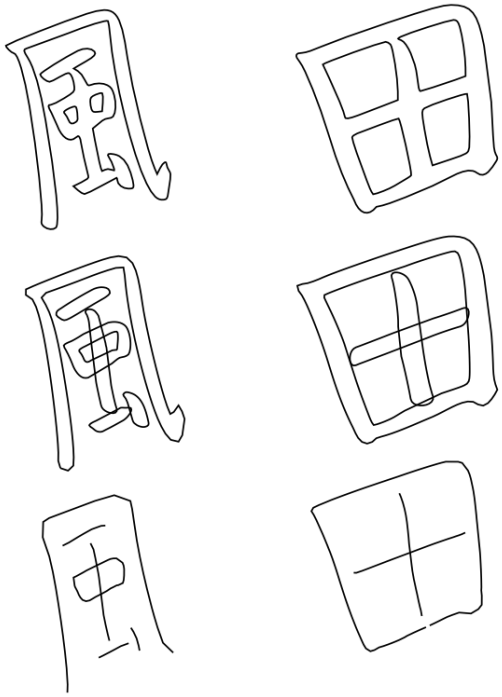

Figure 35. Decomposition and thinning of 2 hollow shapes

This showed that our decomposition and thinning algorithms effectively tackled shapes from real world. However, the implementation was sensitive to noise and produced non-optimal results.

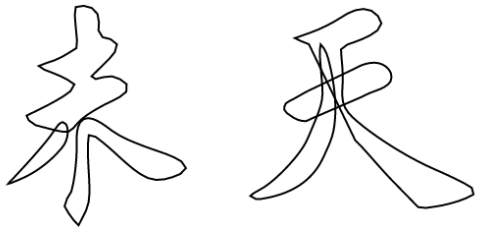

Figure 36. Non-optimal decomposition when perturbation is added.

The thinning algorithm tackled most real world strokes nice and fast, but did occasionally fail. It produced slight artifacts on regular strokes when compared to image based thinning, but generally, it produced smoother and more sensible results on natural strokes.

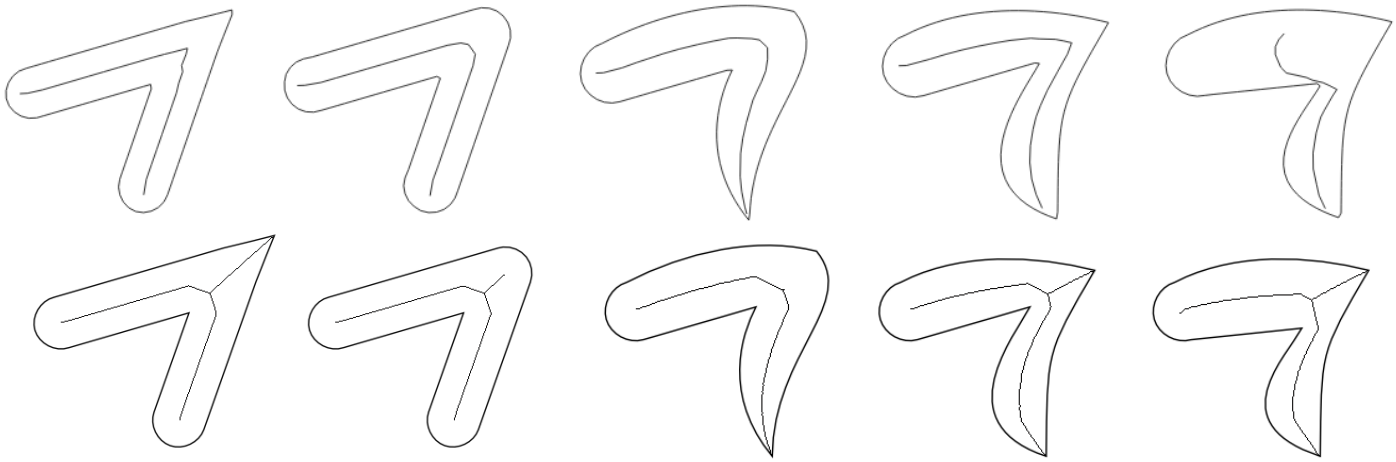

Figure 37. Some thinning results. Our method (top row), image based method (bottom row).

6.3 Skeletal stroke representation and deformation

No major problem was encountered when using the skeletal representation generation algorithm. The skeletal deformation algorithm worked very nice for large as well as small number of sample points. Its dynamic deformation was “elastic” and smooth, making it suitable for use in animation.

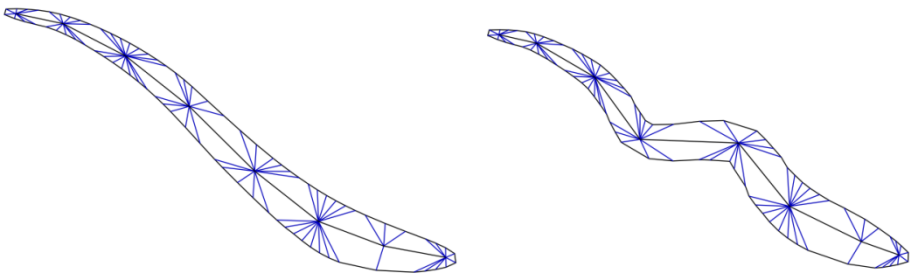

Figure 38. Deformation on a shape with small number of sample points.

6.4 Curve interpolation

The curve correspondence algorithm lent itself to curve interpolation. It was shown that our algorithm behaved better than the naive path-length-corresponded interpolation algorithm found in common vector graphics software. In Figure 39, the naive result was generated by Raphael the vector graphics library. In particular, our method did not disturb the curve’s characteristics.

Combining curve interpolation and stroke deformation, our system can generate high quality intermediate strokes. This is particularly useful for interpolation between animation keyframes, a functionality that can hardly be implemented in a raster pipeline.

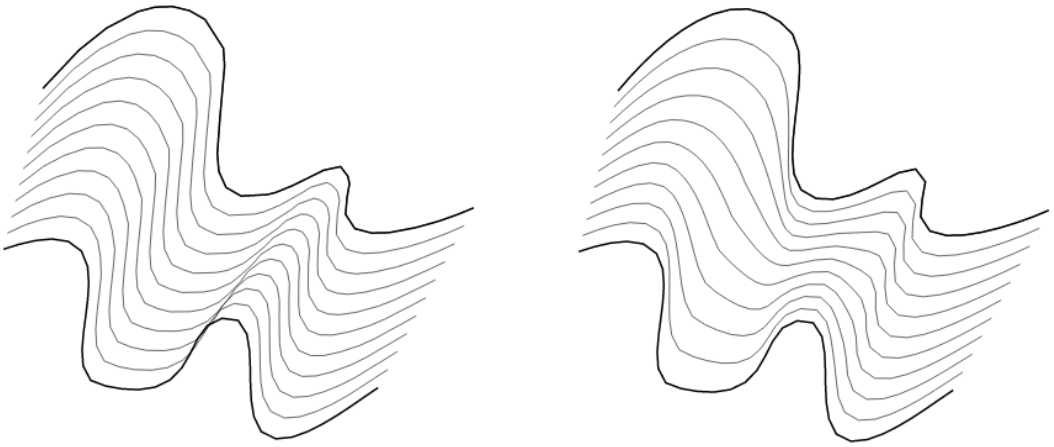

Figure 39. Path interpolation. Our method (left), naive (right).

Figure 40. Stroke interpolation.

6.5 Stylization of Chinese calligraphy

Preparation

A stroke library of Chinese calligraphy was constructed from the characters extracted from Image 4. The decomposition step required quite some manual fix, but the reason was that the image resolution was not high enough, i.e. each character has only 85x85 pixels. The recommended resolution was at least 200x200 pixels for one Chinese character. Some extra components were added, so there were totally 34 characters. Decomposition took 50 seconds. The strokes were grouped by characters, and it took 4 seconds to build the library subsequently. The criterion for choosing characters to form the library was to cover the largest variety of component forms.

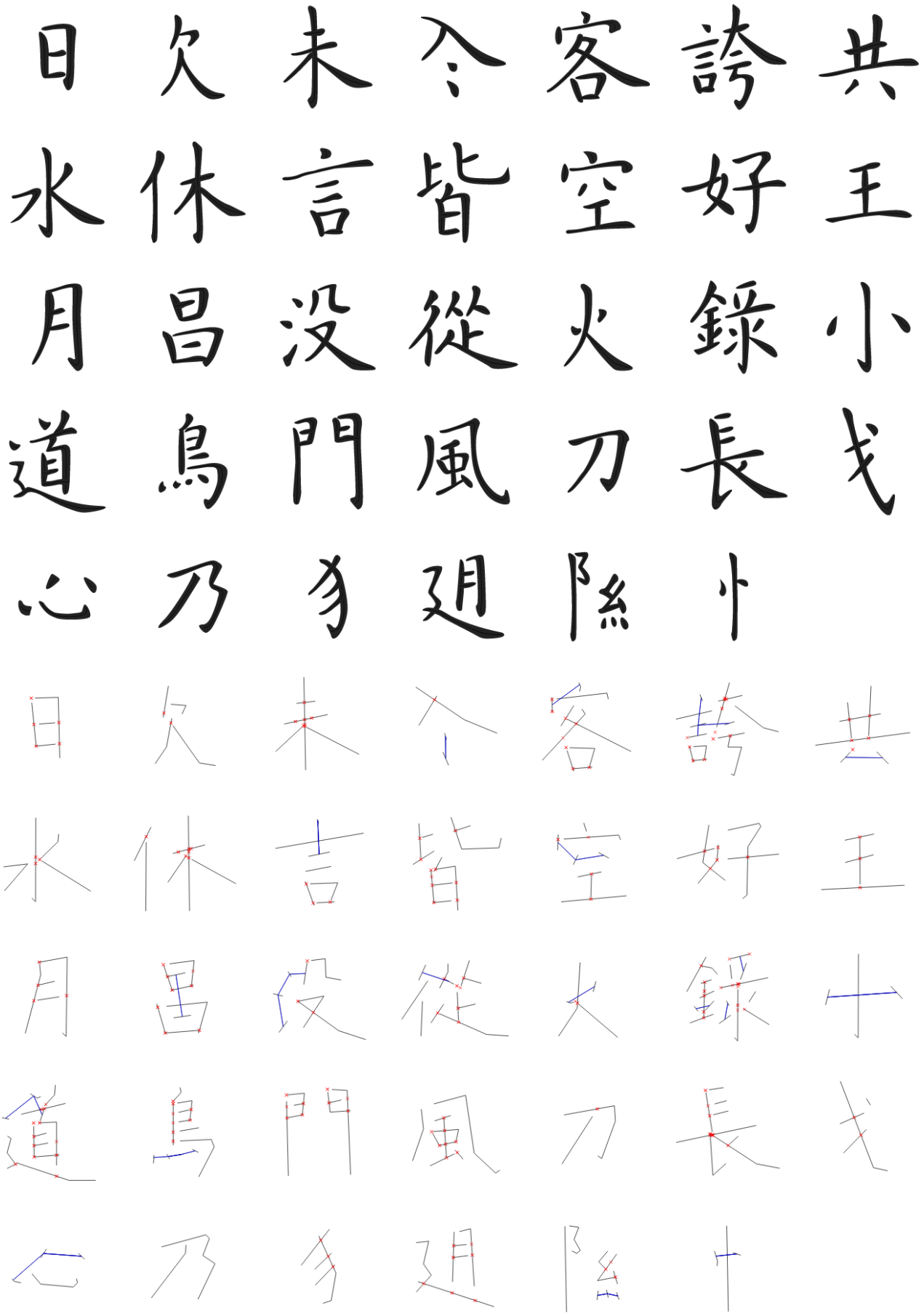

Figure 41. Stroke library of Chinese calligraphy (top) and the corresponding component profiles (bottom)

Test input from Image 4

5 characters were traced by hand from Image 4, and were used as input to the stylization system. It took 2.6 seconds to run which averaged to about 0.5s per character. The input characters (except 水) did not exist but they were structurally similar to some in the library. The stylization result was then compared to the original characters.

Figure 42. Input (top row), stylized result (middle), original characters (bottom).

Evaluation

Starting from the left in Figure 42, the first two (詩 and 俠) were nearly indistinguishable from their originals. The third and fourth (黃 and 秋) were okay but noticeably thinner. The last one (水) had one wrongly matched stroke, even though it had an exact character match in the library.

Test input from Image 5

12 characters were traced by hand from Image 5, and were used as input to the stylization system. It took about 8 seconds to run which averaged to about 0.66s per character. None of the input characters existed in the library and they were structurally very different.

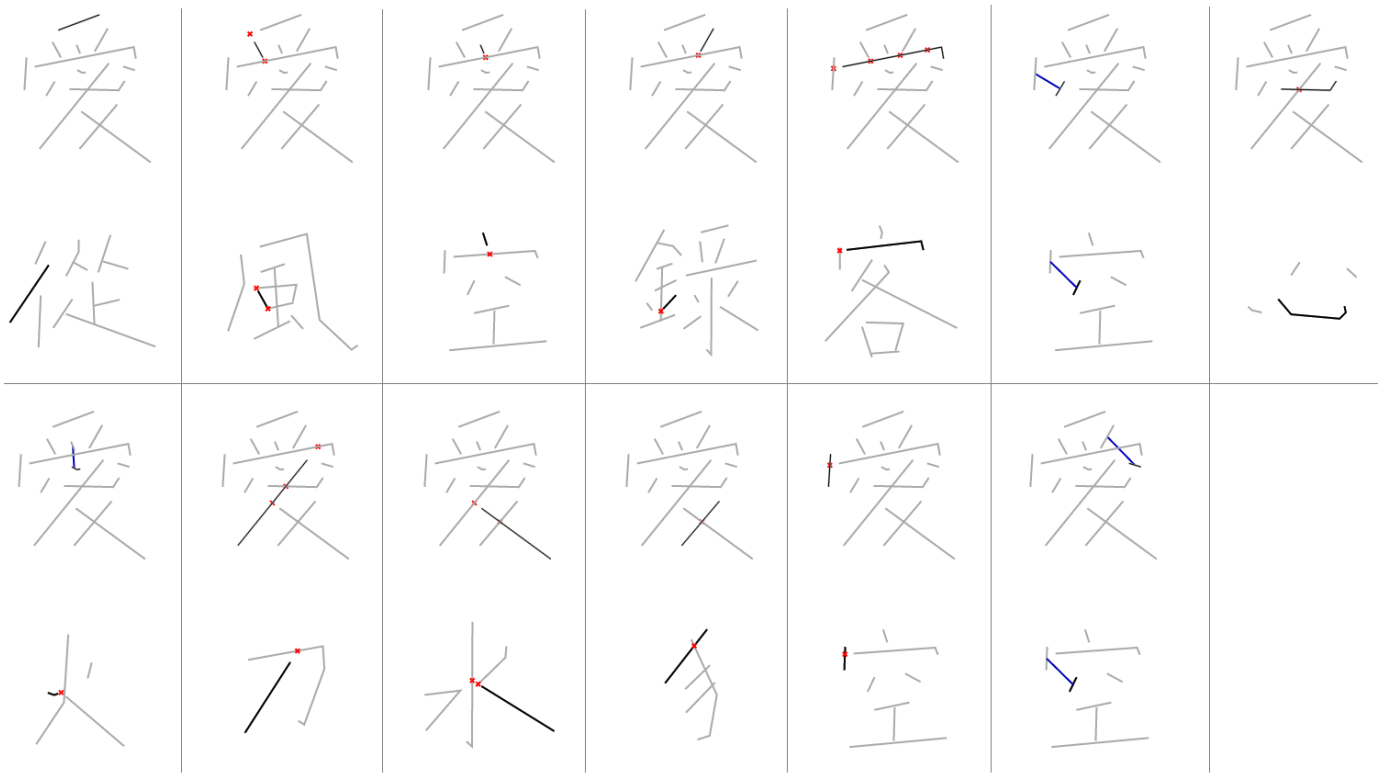

Figure 43. Component matching. Query components (first & third row from top) and their respective matched components (second & forth row).

Figure 44. Input curves

Figure 45. Stylization result

Figure 46. The same characters extracted from Image 5

Evaluation

The stylization result was indeed very nice and plausible. Image 4 and Image 5 were written by different calligraphers, that they should be stylistically different. Indeed, the result was stylistically more similar to Image 4 than to Image 5.

6.6 Stylization of cartoons

Preparation

Image 1 was vectorized to create a stroke library and was decomposed manually. Image 2 was traced by hand to use as input to the stylization system. It took 5 and 20 seconds to build the library and run the stylization respectively. There was a problem during component matching. In the cartoon, curves were quite flexible. But the current matching scheme was not rotation, scale or reflection invariant, such that components were not matched very well. This was mitigated by adjusting the weights of different terms in the similarity equation. Additionally, several transformed copies of a component were added during library construction. A basic stylized result was also produced. There was only one stroke in the basic library.

Figure 47. Cartoon stroke library.

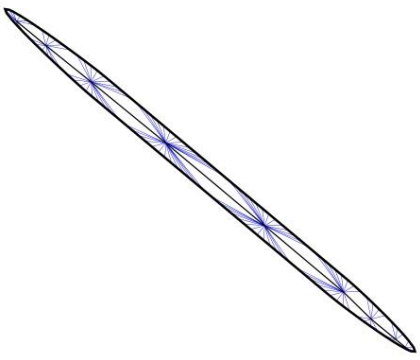

Figure 48. Basic stroke library. There was only one stroke.

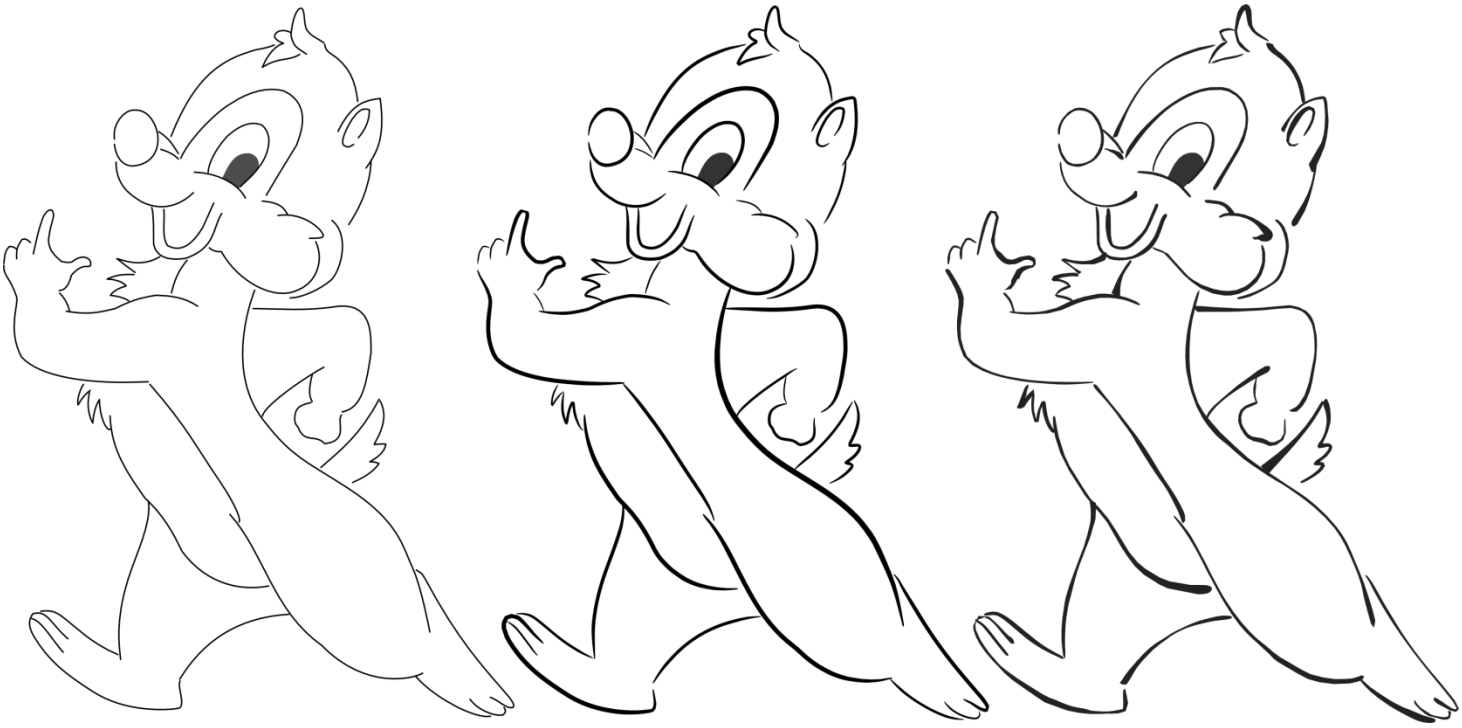

Figure 49. Input (left), basic stylization (middle), cartoon-stylized result (right). The solid eyes were added afterwards for aesthetic purpose.

Evaluation

The basic stylization was indeed good-looking, but it was overly clean and feels artificial. The cartoon-stylized result had more variation and feels more natural and looks plausibly hand-drawn. It inherited many stroke characteristics from the sample illustration and looked as if it was drawn by the same artist.

6.7 Subjective evaluation

We conducted a micro survey. A vectorized version of Image 1 was put side by side to the stylization result. Similarly, stylized and scanned Chinese characters were put together. The subjects were asked to tell which side was “computer generated ink”. They knew the background of the project was about “stylization”, but they have no expert knowledge in computer graphics. Among the 12 participants, 4 had artistic background and all of them can read and write Chinese.

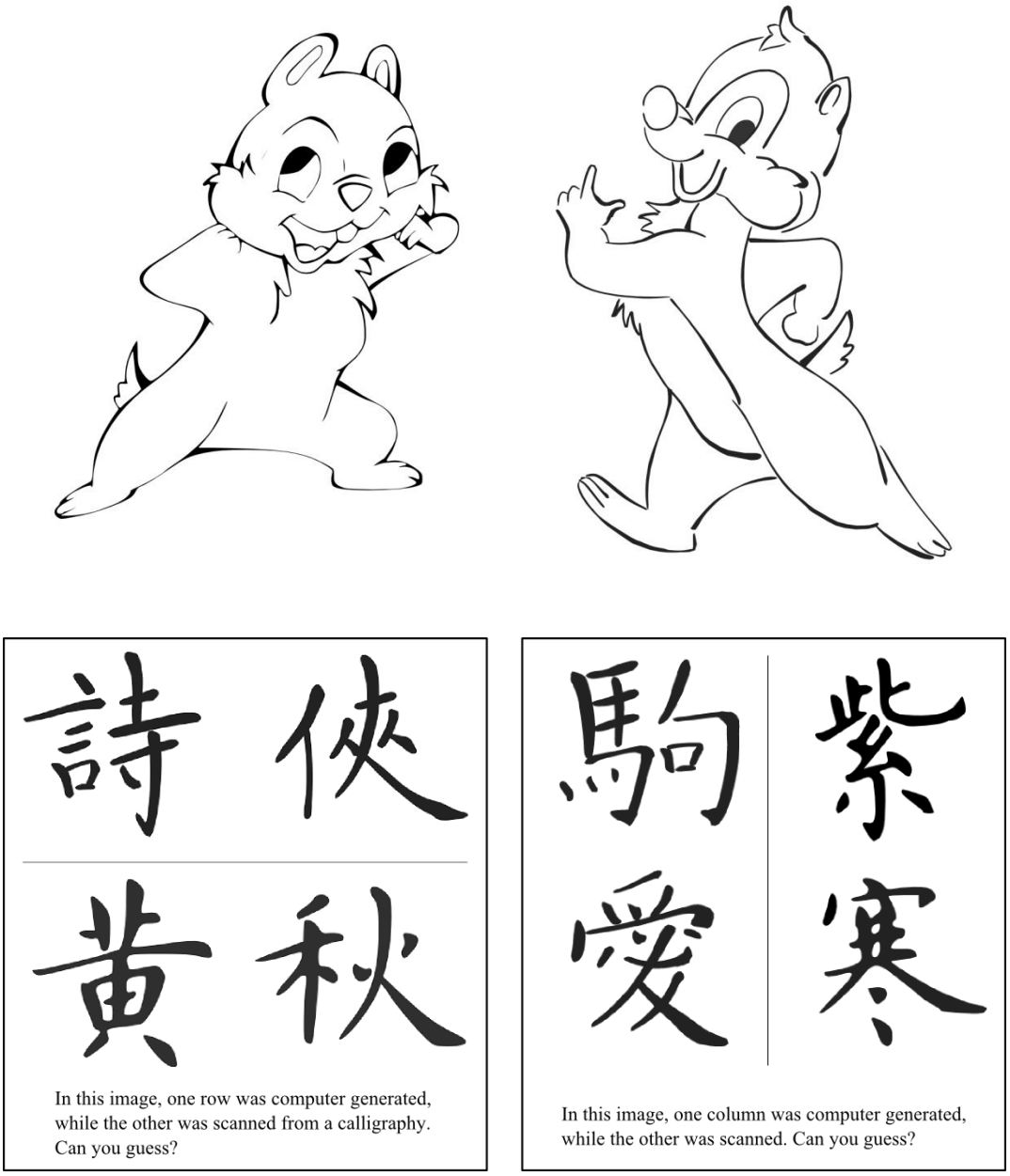

Figure 50. Graphics used in survey.

Result

The participants were asked to state the quality of each artwork. The average rating of the stylized work was between “good” and “very good”. Then, they were asked to judge which artworks were computer generated. Participants were unable to judge which cartoon was computer generated. Their first answers were “unsure” and the correct rate is close to 50%, which was simply wild guessing. Surprisingly, participants were able to judge correctly which Chinese characters were generated, up to 90%. They were told the truth, and asked to rate the plausibility when compared to that drawn by artists. The average rating was “plausible”.

Quality	Extremely bad	Very bad	Bad	Quite bad	Average	Quite good	Good	Very good	Excellent

Plausibility	Unquestionably fake	Obviously fake	Fake	Questionable	Okay	Quite plausible	Plausible	Believable	Unbelievably true

Table 2. Subjective rating table

6.8 Summary

The automated inking system as a whole showed the potential to automate artists' production workflow. The quality of the generated artwork was "good and plausible" as evaluated by human subjects.

The decomposition system worked well on a limited set of shapes but did not perform well on real world data. There were occasional artifacts. It was evaluated as a semi-automatic tool that provided aid to the artist workflow.

The thinning system worked well on test cases as well as real world data. Failure was rare. No manual tweak from the user is required. It was fully-automatic.

The stroke transfer (representation computation, correspondence and deformation) system worked well on real world data. There was no failure. The strokes it generated were of very high quality. The same stroke can be transferred to a wide range of curves. Good result was obtained even by applying the same stroke to a whole illustration. It was also demonstrated that the system was able to tackle animation.

The component matching system worked well for Chinese calligraphy. It required some tweaks to work on cartoons. It was fast enough for interactive use. It involved a linear search, which should be reasonably fast if performed on larger libraries.

7. Discussion

7.1 Shape theory

The shape processing pipeline

The problem of automated inking should not be viewed solely as a problem of stylization, because the limitations of a stroke representation are inherent to a stylization system. The stylization model proposed in this project required a completely new design of shape processing pipeline. There was no coincidence that all algorithms developed were vector based.

They can be used in several ways:

- stroke interpolation: curve correspondence + stroke deformation
- stroke transfer: thinning + curve correspondence + stroke deformation
- character recognition: decomposition + thinning + component analysis
- font generation: component analysis & matching + stroke transfer
- automated inking: full stack

which are very hard to achieve in a raster processing pipeline.

Stroke model

The stroke model employed in this project gave no clue how a stroke should be created in the first place. We were unable to work out a definition of stroke. Traditionally, it can be defined as the area swept by a moving brush along a trajectory, in which the shape of the brush must be fixed. But given a very tiny brush, any shape can be produced by simply filling an area of interest exhaustively. A reasonable fix is to disallow a brush to intersect with its previous footprint, effectively disallows a self-intersecting trajectory, and thereby disallowing hollow shapes. This limitation can be raised by performing decomposition as a preprocessing step. Extending the definition, a stroke is allowed to intersect itself provided the resulting shape is decomposable. By assuming the brush is indecomposable, stroke can be defined as indecomposable shape under a composition model. To complete the theory, it is required to prove that moving a brush along a trajectory is an indecomposable shape evolution process. Strokes that are indecomposable are called atomic strokes. Strokes that can be decomposed into **one** atomic stroke are called hollow strokes.

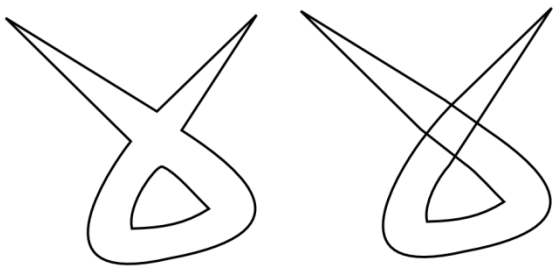

Figure 51. Decompose of a self-intersecting stroke. Hollow stroke (left), atomic stroke (right).

The above is one possible direction to formalize the underlying theory involved in this project. It transforms the problem of stroke definition into a problem of the theory of shape composition and evolution.

The thinning algorithm

The validity of the thinning algorithm relied on the **Slope Theorem** mentioned in the work [8] by G. Pijush et al., as stated here:

> THEOREM 2 (Slope Theorem). Let B be the representation of the outline of a brush and T the representation of a trajectory. The boundary S of the generated shape when the brush is moved along the trajectory can be expressed as a subset of the set of all points b + t such that b ∈ B, t ∈ T, and the tangents to B and T at these points are parallel.

For an elliptic brush, there exists exactly two such points b on every t. The pairing process is essentially to recover those b-pairs. Considering a circular brush with radius r, the pairs are exactly (t+*b* and t-*b*), where *b* is a vector with length r and normal to the slope at t. Which means for circular brushes, the search for best pair criterion should only be (2) the angle difference. General, convex and elliptic-like brushes are more complicated to analyze, but is essentially what the equation stated in section 3.3 is trying to model. The final contour is only a subset of all b+t, which explains why either A or B can stay during the pairing process.

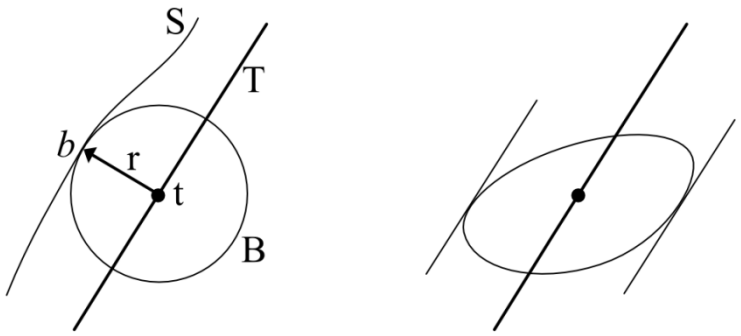

Figure 52. Diagram of stroke generation. Circular brush (left), general convex brush (right).

Thus, it has to be understood the thinning algorithm only works for convex and elliptic-like brushes, which, showed by S.H. Chu et al. in [1], is sufficient for modeling real brushes. This might also explain why the thinning algorithm failed for some strokes: the brush footprint deformed too seriously at a turn.

7.2 Artisticity

Brush and ink

The essence of inking is in the careful control of strokes, as seen in its variability and fluidity. Choosing a brush with high expressiveness is thus very important to an illustrator. Brushes in raster graphics packages like Photoshop have acceptable expressiveness, but they were still far from a physical brush or pen. One way to solve this problem is to do physically-modeled brush simulation, another way is to encourage the use of physical pens and assimilate the high quality scan. This project chose the latter.

The essence of inking style

The non-technical significance of this project was it modeled the underlying rules of inking. Inking and calligraphy require years of practice to perfect until the subtleness of motion control became programmed in the artist's motor neurons. The subtleness is consistent within an illustration and across different illustrations by the same artist. After studying hundreds of inked illustrations carefully, we concluded that the key is in ***interaction***. The model presented in this project did not cover all the factors and they vary highly among types of illustration. For example, in Chinese calligraphy the aspect ratio and size of a component are very important but they are not concerned in cartoon inking.

Artificial intelligence?

It is interesting to ask: to what extend can a mechanical system achieve artisticity? The automated inking system presented in this project was completely mechanical. There is no element of machine learning or stochastic process. It is literally a "Chinese room" that processes and generates Chinese characters with completely no knowledge of Chinese characters. Fortunately, it failed *the test*. Nearly all human participants were able to point out the computer generated ones immediately. We were unable to explain, the feedback had no consensus. Some said it was the corner that leaked the truth, some said fluidity and some said speed and control.

7.3 Future applications

2D Animation

2D animators do deserve automated tools. Many animation tools had keyframe and tweening functionalities, but they were able to satisfy artists' needs. As a result, in the current state of the 2D animation industry, every frame, stroke and detail was artists' hard work. In fact, there is typically high coherence between animation frames. The problem was that the graphics pipeline was not able to assimilate and reuse the information. While it is much more difficult to structurally extrapolate lines and curves across frames, it should be feasible, as demonstrated by this project, to extrapolate strokes and styles. Hopefully, the ideas and techniques developed in the project could be further developed into a system that can truly help artists.

Font characters generation

There is substantially lower number of Chinese fonts available when compared to Latin-based languages. It is because Chinese fonts have to support as much as 50,000 characters. Chinese characters are combinations of a limited set of basic components. The problem is the components show variation when combined differently. Regular fonts, which show fewer variations, have been generated in the past. Yet there was no successful example of calligraphic font generation known to us. The automated inking system could be an integral part of a calligraphic font generation system.

8. Conclusions

This project highlighted the limitations of existing shape processing techniques and presented a set of algorithms that could address them. The algorithms have many applications, and they were integrated to construct an automated inking system. It was able to generate high quality and plausible inking. A model that could capture the style of inking was designed and the importance of stroke interactions was raised. Finally, potential applications to 2D animation and font generation were proposed.

8.1 Further work

Some suggested further work:

- study the algorithms formally and investigate their theoretic limitations as well as room of improvement
- introduce machine learning techniques into the automated inking system to train the various parameters instead of tweaking manually
- brainstorm ways to incorporate the raster and vector processing pipeline, to produce textured and colored strokes ultimately

9. References

[1] S.H. Chu and C.L. Tai, An Efficient Brush Model for Physically-Based 3D Painting, Proc. of Pacific Graphics 2002, 2002.

[2] S. Xu, M. Tang, F. Lau, and Y. Pan, "Virtual hairy brush for painterly rendering", Graphical Models, 66(5), 263-302, 2004.

[3] S. DiVerdi, Stephen, A. Krishnaswamy and S. Hadap, "Industrial-strength painting with a virtual bristle brush", Proc. of the 17th ACM Symp. on Virtual Reality Software and Technology, 119-126, 2010.

[4] W. Baxter and M. Lin, "A versatile interactive 3D brush model", Proc. 12th Pacific Conf. on Computer Graphics and Applications, 319-328, 2004.

[5] J. Fekete, E. Bizouarn, E. Cournarie, T. Galas, and F. Taillefer, "TicTacToon: a paperless system for professional 2D animation", Proc. of the 22nd annual conf. on Computer graphics and interactive techniques, 79-90, 1995.

[6] T. Pudet, "Real Time Fitting of Hand-Sketched Pressure Brushstrokes", Computer Graphics Forum, 13(3), 205-220, 1994.

[7] H. Seah, Z. Wu, F. Tian, X. Xiao and B. Xie, "Artistic brushstroke representation and animation with disk b-spline curve", Proc. of the 2005 ACM SIGCHI International Conf. on Advances in computer entertainment technology, 88-93, 2005.

[8] G. Pijush, and S. Mudur, "The brush-trajectory approach to figure specification: some algebraic solutions", ACM Trans. on Graphics, 3(2), 110-134, 1984.

[9] S.C. Hsu and I. Lee, "Drawing and animation using skeletal strokes", Proc. of the 21st annual conference on Computer graphics and interactive techniques, 109-118, 1994.

[10] J. Lu and C. Barnes and S. DiVerdi and A. Finkelstein, "RealBrush: Painting with Examples of Physical Media", ACM Transactions on Graphics, 32(4), 117, 2013.

[11] J. Lu, F. Yu, A. Finkelstein and S. DiVerdi, "HelpingHand: example-based stroke stylization", ACM Transactions on Graphics, 31(4), 46, 2012.

[12] S. Saito, A. Kani, Y. Chang, and M. Nakajima, "Curvature-based stroke rendering", The visual computer, 24(1), 1-11, 2008.

[13] Y. Okabe, S. Saito and M. Nakajima, "Paintbrush rendering of lines using HMMs", Proc. of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia, 91-98, ACM, 2005.

[14] Adobe Systems, Inc. (2013). Illustrator. [Online]. Available: www.adobe.com/illustrator/.

[15] Adobe Systems, Inc. (2013). Photoshop. [Online]. Available: www.adobe.com/photoshop/.

[16] S. Loncaric, "A survey of shape analysis techniques", Pattern recognition, 31(8), 983-1001, 1998.

[17] P. Ghosh, "A unified computational framework for Minkowski operations", Computers & Graphics, 17(4), 357-378, 1993.

[18] B. Jang, R. Chin, "Analysis of thinning algorithms using mathematical morphology", Pattern Analysis and Machine Intelligence, 12(6), 541-551, 1990.

[19] J. Liu, W. Cham and M. Chang, "Stroke order and stroke number free on-line Chinese character recognition using attributed relational graph matching", Proceedings of the 13th International Conference on Pattern Recognition, 3, 259-263, 1996.

[20] The World Wide Web Consortium. (2004). Scalable Vector Graphics. [Online]. Available: www.w3.org/Graphics/SVG/About.html.

[21] Inkscape. (2014). [Online]. Available: inkscape.org/en/.

[22] The Chromium Authors. (2014). The Chromium Projects. [Online]. Available: http://www.chromium.org/.

[23] Mozilla. (2014). Firefox - Free Web Browser. [Online]. Available: www.mozilla.org/en-US/firefox/.

[24] Dmitry Baranovskiy. (2014). Raphael 2.1.2 - JavaScript Vector Library. [Online]. Available: raphaeljs.com/.

[25] Peter Selinger. (2003). Potrace: a polygon-based tracing algorithm. [Online]. Available: potrace.sourceforge.net/.

10. Appendix A: Meeting Minutes and progress reports

1st project meeting

Date: Friday, Mar 15, 2013

Time: 4.30pm

Place: Sander's office

People: Chris Tsang, Prof. Sander

1. Approval of minutes

This was the first formal group meeting, so there were no minutes to approve.

2. Report on progress

Chris had a preliminary idea on the project topic and written a draft proposal.

3. Discussion items

Chris and Sander agreed that 2d graphics rendering will be a good topic. Sander gave Chris a stylization project video that was published in NPAR.

4. Goals for the coming week

Look for a concrete idea to work on.

5. Meeting adjournment and next meeting

The meeting was adjourned at 5pm.

1st progress update

Date: July 25, 2013

Chris had worked more in stroke rendering, refining an implementation employing old-schooled techniques. Chris saw the need to have a more general and robust stroke rendering algorithm. Chris researched on stroke representation, skimming through many academic papers. He discovered that there is not a widely accepted general representation. Chris read Nelson Chu's work on Moxi and contacted him for discussion.

A meeting with Nelson Chu

Date: July 29, 2013

Time: 6.15pm

Place: HKUST LG7 canteen

People: Chris Tsang, Nelson Chu

Chris and Nelson discussed on the stroke representation used in Moxi. Nelson said that raster graphics can be scale to higher resolution still preserving fidelity by using image processing techniques. Chris argued vector graphics has the advantage of precise manipulation. Chris was concerned of the difficulty to control

and reproduce simulation-based rendering. Nelson talked about the life of a graphics researcher and the difference between academic and commercial research. The meeting was ended at around 9pm. This was an informal meeting. This meeting is recorded here because Chris had learnt and gained inspirations from Nelson.

2nd progress update

Date: Aug 15, 2013

Chris had worked in vector graphics, and successfully set up a software framework for parse and process SVG vector graphics in HTML5. The capabilities include Bezier spline manipulation and matrix transform.

2nd project meeting

Date: Mon, Aug 26, 2013

Time: 2.30pm

Place: Sander's office

People: Chris Tsang, Prof. Sander

1. Approval of minutes

The minutes of last meeting were approved without amendment.

2. Report on progress

Chris had researched and read more academic papers of stroke rendering and stylization.

3. Discussion items

Chris wanted to work out a general stroke representation and he argued that a lot of existing work can be improved with a better representation. Chris stated that existing stroke rendering work often have limitation in class of shapes they can produce. Sander suggested that in order to aim for publication, the project idea has to be new and able to create novel renderings. Sander pointed out a particular example of work done by Jingwan et al. in stylization.

4. Goals for the coming week

To construct a vision on what rendering effects to produce in this project.

5. Meeting adjournment and next meeting

The meeting was adjourned at 3.10pm.

3rd project meeting

Date: Tuesday, Sep 3, 2013

Time: 3pm

Place: Sander's office

People: Chris Tsang, Prof. Sander

1. Approval of minutes

The minutes of last meeting were approved without amendment.

2. Report on progress

Chris read about a new technique of using machine learning model in stroke rendering. Chris saw the possibility to improve upon this work.

3. Discussion items

Sander and Chris discussed on the practicability to work on a new representation. Sander pointed out that this direction will involve more foundational work. Sander asked whether rendering in pure vector has the advantages over the increased complexity. Chris argued that vector manipulation is more intuitive and precise.

4. Goals for the coming week

To work out the exact project proposal.

5. Meeting adjournment and next meeting

The meeting was adjourned at 3.30pm.

3rd progress update

Date: Sep 15, 2013

Chris encountered an important book "Mathematics of Shape Description: A Morphological Approach to Image Processing and Computer Graphics" by Pijush K. Ghosh and Koichiro Deguchi. This book helped Chris to formulate the stroke representation presented in this paper. Chris learnt the properties of a good theoretic representation and that it should oversee discrete and continuous implementation. Chris also learnt the power of morphological image processing.

4th progress update

Date: Sep 20, 2013

Chris had set up a framework in HTML5 for image processing to prepare for acquiring useful data from images to augment stroke rendering. Chris had nearly finished composing the proposal report.

4th project meeting

Date: Sep 26, 2013

Time: 3pm

Place: Sander's office

People: Chris Tsang, Prof. Sander

1. Approval of minutes

The minutes of last meeting were approved without amendment.

2. Report on progress

Chris had just submitted the proposal report.

3. Discussion items

Sander went over the proposal report with Chris and gave his comments on individual sections.

4. Goals for the coming week

To experiment on various stroke generation techniques.

5. Meeting adjournment and next meeting

The meeting was adjourned at 3.30pm.

5th progress update

Date: Oct 15, 2013

Chris was impressed by the work of Pijush K. Ghosh and believed in a Minkowski sum-based representation. However, he spent so much time in it but still failed to extend it to support a “changing brush” model- all the elegant properties with Minkowski sum do no work anymore. Chris was confused. He thought, may be, a Minkowski sum based representation is not suitable for that purpose.

6th progress update

Date: Nov 11, 2013

Chris had opened his mind for other theories. He had a sudden interest in shape analysis and researched into that topic instead. He had some inspirations. He “invented” a theoretic framework for shape analysis. Though, it turned out later, that was highly similar to technique known as the Zahn and Roskies method, well, originally published in 1972.

7th progress update

Date: Dec 31, 2013

Chris had finished his final exam and enjoyed his holidays. He had to work on his FYP again. He wanted to apply the techniques he developed some time earlier to the FYP, but without changing the topic. He came up with a newly designed automated illustration system that met these requirements. He began working on it immediately.

8th progress update

Date: Jan 15, 2014

Happy new year! After intense period of working, Chris had got the 2 algorithms, decomposition and thinning, functional! He was excited and reported to Sander about the progress immediately.

9th progress update

Date: Feb 18, 2014

After another period of intense working, Chris also got the stroke deformation algorithm working. But writing a program is different from writing a report. Chris spent a lot of effort in trying to describe the algorithms concisely. He spent much time in preparing the diagrams. He met with Ted Spaeth, and received grammatical and formatting corrections from Ted. He worked overnight before handing in the progress report.

5th project meeting

Date: Feb 25, 2014

Time: 10am

Place: Hangout

People: Chris Tsang, Prof. Sander

1. Approval of minutes

The minutes of last meeting were approved without amendment.

2. Report on progress

Chris had just submitted the progress report.

3. Discussion items

Sander said the progress report was generally good. He raised concern about the robustness of the various algorithms and suggested Chris to investigate more in that direction. Chris agreed and included discussions on robustness in the final report.

4. Goals for the coming week

Finish the stylization part of the system.

5. Meeting adjournment and next meeting

The meeting was adjourned at 10.30am.

10th progress update

Date: Mar 15, 2014

Chris just extended the decomposition algorithm to support the processing of hollow shapes. He was still puzzling on how to actually perform stylization. He looked into a maximal sub-graph matching algorithm and thought it was too complicated to implement.

Another meeting with Nelson

Date: Mar 17, 2014

Time: 6.30pm

Place: HKUST LG7 canteen

People: Chris Tsang, Nelson Chu

Chris and Nelson talked about graphics research. Chris showed Nelson his progress report, and Nelson expressed his concern that how well a stylization system could perform without specific knowledge of Chinese calligraphy. Nelson also pointed out the problems of skeletal strokes and how it is unable to tackle complex strokes in Chinese calligraphy. Chris largely agreed and that was the reason he sought a new stroke model.

11th progress update

Date: April 1, 2014

Chris abandoned the idea of doing maximal sub-graph match optimally. He bet that heuristics will do well for the purpose of stylization.

12th progress update

Date: April 10, 2014

Chris has successfully implemented a stylization system! It worked for some Chinese characters. Chris loved Chip and Dale. He decided to also test the system on a drawing of Dale.

6th project meeting

Date: April 15, 2014

Time: 2pm

Place: Sander’s office

People: Chris Tsang, Prof. Sander

1. Approval of minutes

The minutes of last meeting were approved without amendment.

2. Report on progress

Chris had just finished the stylization system and gave Sander a sample of a stylized Chinese character.

3. Discussion items

Sander said that a stylization system should be able to support different styles, and suggested Chris to prepare more. Chris told Sander he was also testing on cartoons. Sander suggested Chris to make an interactive demo that is capable of stylizing sentences.

4. Goals for the coming week

Finish the final report, and prepare for the presentation.

5. Meeting adjournment and next meeting

The meeting was adjourned at 2.20pm.

7th project meeting

Date: April 23, 2014

Time: 3pm

Place: Sander's office

People: Chris Tsang, Prof. Sander

1. Approval of minutes

The minutes of last meeting were approved without amendment.

2. Report on progress

Chris had just finished most of the final report.

3. Discussion items

Sander said that the stylized renderings look great. He commented that the technical and application aspect of the project was very good, and Chris had a good chance of publication. He mentioned that the presentation should be largely technical, while the poster should be more graphical.

4. Goals for the coming week

Finish the final report, and prepare for the presentation.

5. Meeting adjournment and next meeting

The meeting was adjourned at 3.20pm.

11. Appendix B: Project Planning

11.1 Tasks

Component
Stage A. Reference data set construction
1. Input images of stroke samples
2. Perform thresholding, edge extraction and vectorization
3. Decompose the shapes into individual strokes
4. Thinning: compute the central axis of each stroke
5. Compute the skeletal representation of each stroke
6. Build a relational graph of curves
Stage B. stylization
1. Input digitized unstroked curves
2. Build a relational graph of curves
3. Perform a maximal sub-graph search in the data set
4. Compute correspondence between input curves and reference curves
5. Compute the skeletal stroke deformation for each input curve
6. Output the stroked set

Table 3. Working status of various parts of the system.

11.2 Work breakdown and scheduling

This is a one-man-project and thus all work is expected to be carried by the sole project member. Here is a detailed work breakdown and scheduling:

1. prepare a vector graphics processing framework
2. research in shape analysis and processing
3. design the automated illustration system
4. design, implement and test the decomposition algorithm
5. design, implement and test the thinning algorithm
6. design, implement and test the skeletal representation and deformation algorithm
7. design, implement and test the relational representation algorithm
8. design, implement and test the curve correspondence algorithm

9. integrate different parts into one system
10. collect calligraphy and illustrations for testing
11. test the system
12. fix problems and enhancements
13. evaluation
14. report writing

tasks\month	Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr
1								
2								
3								
4								
5								
6								
7								
8								
9								
10								
11								
12								
13								
14								

Table 4. GANTT Chart of the project schedule

12. Appendix C: Required Hardware & Software

Hardware

Desktop computer

Drawing tablet for sketching

Software

Operating system	Linux, Windows or Mac
Vector graphics editor	Inkscape
HTML5 browser	Firefox or Chrome

Table 5. Required software.