Wholetoning: Synthesizing Abstract Black-and-White Illustrations

by

Jie Xu

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2009

AUTHOR'S DECLARATION

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as required by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Black-and-white imagery is a popular and interesting depiction technique in the visual arts, in which varying tints and shades of a single colour are used. Within the realm of black-and-white images, there is a set of black-and-white illustrations that only depict salient features by ignoring details, and reduce colour to pure black and white, with no intermediate tones. These illustrations hold tremendous potential to enrich decoration, human communication and entertainment. Producing abstract black-and-white illustrations by hand relies on a time consuming and difficult process that requires both artistic talent and technical expertise. Previous work has not explored this style of illustration in much depth, and simple approaches such as thresholding are insufficient for stylization and artistic control.

I use the word *wholetoning* to refer to illustrations that feature a high degree of shape and tone abstraction. In this thesis, I explore computer algorithms for generating wholetoned illustrations. First, I offer a general-purpose framework, "artistic thresholding", to control the generation of wholetoned illustrations in an intuitive way. The basic artistic thresholding algorithm is an optimization framework based on simulated annealing to get the final bi-level result. I design an extensible objective function from our observations of a lot of wholetoned images. The objective function is a weighted sum over terms that encode features common to wholetoned illustrations.

Based on the framework, I then explore two specific wholetoned styles: papercutting and representational calligraphy. I define a paper-cut design as a wholetoned image with connectivity constraints that ensure that it can be cut out from only one piece of paper. My computer generated papercutting technique can convert an original wholetoned image into a paper-cut design. It can also synthesize stylized and geometric patterns often found in traditional designs.

Representational calligraphy is defined as a wholetoned image with the constraint that all depiction elements must be letters. The procedure of generating representational calligraphy designs is formalized as a "calligraphic packing" problem. I provide a semi-automatic technique that can warp a sequence of letters to fit a shape while preserving their readability.

Acknowledgements

Thanks to my supervisor and friend, Craig S. Kaplan. It is impossible to finish this work without his kindly and fruitful direction and help. From Craig, I learned a lot about how to do research and solve problems. I am very grateful to Craig's suggestion on my future profession.

A great deal of gratitude to the members of my committee. They provided fantastic suggestions and advice about my research.

The faculty and members of CGL provided me invaluable discussion and help in my courses and research. I never forget the wonderful life in Waterloo. Thanks in general to all my friends.

Many thanks to a lot of artists and photographers for permission to use their pictures in my research.

Finally, I would like thank my parents and all members of my family for their support and love.

Part of the research work described in this dissertation has been appeared in previous publications. The general framework of wholetoning was presented at NPAR 2008 [148], the work of computer generated papercutting was presented at Pacific Graphics 2007 [149] and the calligraphic packing result was in GI 2007 [145].

Contents

List of Tables

List of Figures

Chapter 1

Introduction

If everything isn't black and white, I say, "Why the hell not?"

– John Wayne

Visual communication is one of the most important and popular ways to represent ideas and exchange information. Its origins can be traced back to more than ten thousand years ago [4]. When ancient people lived in caves, they drew figures and patterns on rock to record their activity and knowledge and convey emotion and consciousness (Figure 1.1(a)). With the progress of civilization, visual communication has developed into an effective and diverse form of communication. It includes many media such as painting, photography, film, typography, illustration and drawing.

How do people communicate visually? Chinese painter Zheng Xie (1693-1765) described his creative experience [77]:

> One autumn morning, I was absorbed by the sight of bamboo on the riverside. The branches were waving in the mist and light was blinking between leaves. The beautiful scene inspired me to record it in a drawing. But at this time, the bamboo in my mind was not the bamboo I had seen. I brought out brush and ink, and finished a painting in high spirits. But the bamboo on the paper was not the one in my mind.

(a) (b) (c)

Figure 1.1: Examples of black-and-white illustrations in different media: an ancient rock painting in Spain (between 8,000 and 3,000 BCE) in (a), a decorative pattern on a Chinese ceramic vase (between 600 and 900 AD) in (b) [33, Page 28] and a batik-dyed fabric design (modern times) in (c)[33, Page 32].

His description effectively summarizes the procedure of creating a graphic depiction of the real world. In this procedure, we manipulate acquired information and present it in a way that follows our purpose. I conceptualize it to be an iterative process of *observation*, *composition* and *depiction*. The observation step acquires external reality through an observer's visual system. The composition step creates the internal representation by our deliberate processing, manipulating, selecting and filtering of visual data. Finally, the results are depicted on real media to express and communicate our thoughts and perceptions. The depiction can itself feed back as the input of further observation activity. The composition and depiction steps are subjective. Individual bias will shape different results, and at the same time act as a catalyst for the process of abstraction.

Among depiction approaches in the visual arts, black-and-white imagery is a special and interesting technique, because the language of expression is simplified to tints and shades of a single colour. Since we see the world in colour, a monochromatic image offers a more abstract, interpretive view of reality. When viewing a black-and-white image, we are moved by its emphasis on shapes and forms, patterns and contrasts, simplicity and expressivity. Black-and-white imagery can employ different techniques to derive numerous representations ranging from a realistic style to an exaggerated or highly simplified style. It has long been a desirable medium in

decoration (Figure 1.1(b)) , pattern design (Figure 1.1(c)), advertising, book illustration and logo design.

The most prominent channels through which images communicate are colour (we refer to tone in black-and-white images), texture and shape. Tone refers to a colour's degree of lightness or darkness. It communicates the lighting and shading information of objects. Texture consists of the visual patterns with properties of homogeneity that result from the presence of repeated elements. It communicates important information about the structural arrangement and material properties of surfaces. Shape defines objects' geometric properties.

Let us restrict ourselves to the case where only two tones are used, which are called pure black-and-white illustrations (In this dissertation context, I will use black-and-white illustrations to represent those two-tone black-and-white images). In black-and-white illustration, the colour, texture and shape of the real world must be reduced to a simple vocabulary of black marks on a white background or vice versa. This reduction is the fundamental challenge of this style of visual communication. Artists have developed many techniques in black-and-white illustration to present desired visual features.

My initial inspiration arises from traditional media as shown in Figure 1.2. Artists have developed many approaches to creating the black-and-white illustrations, such as woodcutting, seal cutting, papercutting, line drawing, engraving, etching, and Batik dyeing. Many of them aim at creating black-and-white representations of continuous-tone images. The majority of these techniques are concerned with *halftoning*, the approximation of continuous tone using a distribution of black primitives on a white background [131]. Halftoning was—and continues to be—a necessary means of overcoming limitations in output technologies. For the most part, computer displays have moved beyond the need for extensive halftoning, but print technology is still highly dependent on it. Colour painting has been around for a long time, but print-based techniques offer the advantage of reproducibility, and grew in popularity with the development of the printing press. Printing is limited to a finite colour palette, meaning that it quickly became necessary to approximate intermediate tones. Halftoning has also found its way into non-photorealistic rendering research; examples include algorithms for approximating tone with ordered dithering [131], stipples [27, 115], ASCII art [123], screens [95], pen strokes [142], or the walls of a maze [99, 146]. These approaches produce image with distinct visual characters, which can evoke different aesthetic responses in viewers.

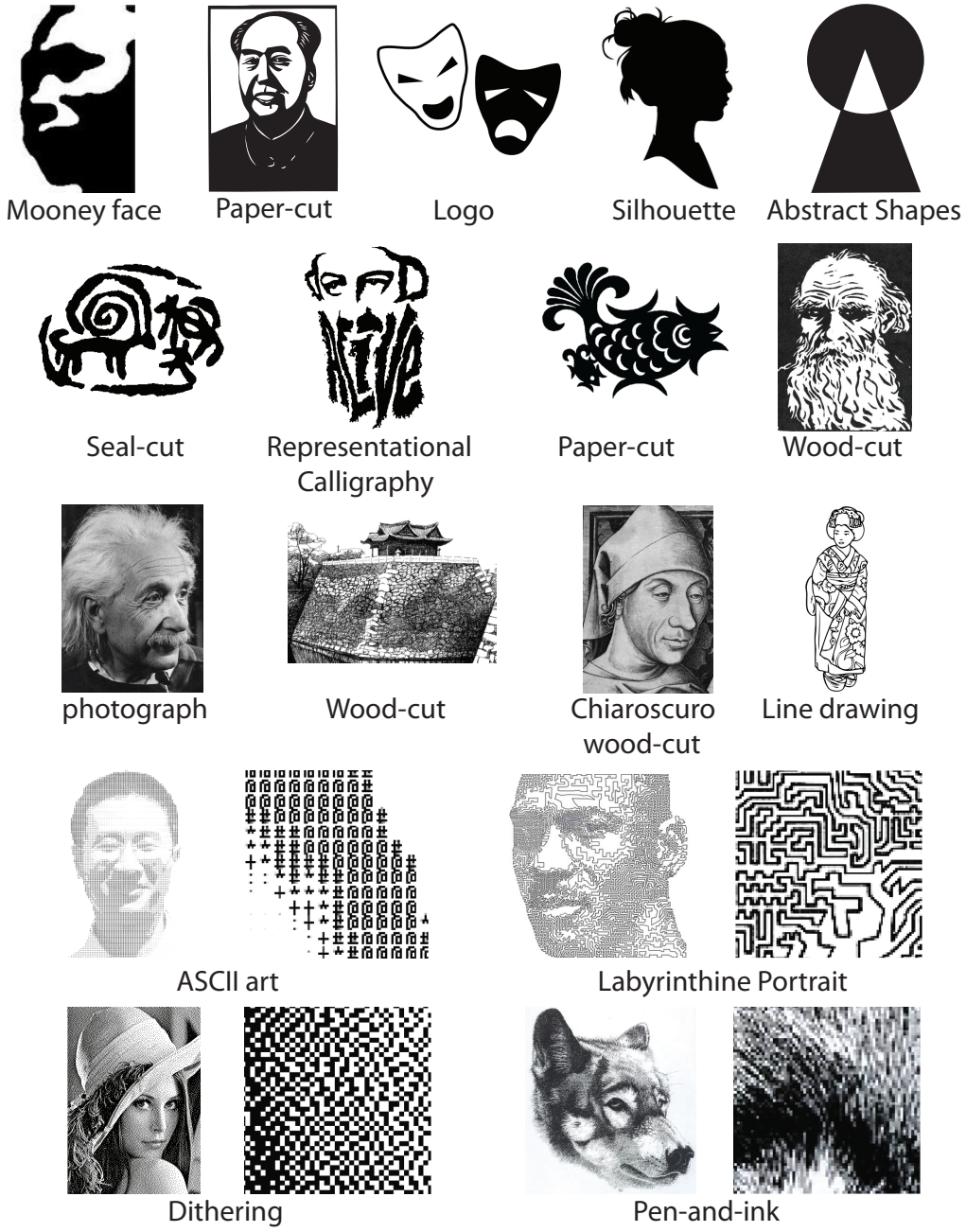


Figure 1.2: Black-and-white imaging styles. Mooney faces are low-information two-tone pictures of faces used in psychological tests [84]. Labyrinthine Portraits are line drawings made from a single labyrinthine stroke [85]. The last two rows show close-ups alongside the originals.

Figure 1.3: Classic examples of black-and-white illustration without halftoning: a simple line drawing (a), a papercutting by Susan Throckmorton (b), Felix Vallotton's woodcut *La Raison Probante* (c), and an ink painting [150, Page 11].

However, halftoning need not be the only way to create a black-and-white depiction of an image. There exists a great deal of traditional art in black-and-white that makes no attempt at halftoning. An immediate example is line drawing (see Figure 1.3(a)), which seeks to convey all information about form and texture from a few sparse visual cues embedded in contours. Aesthetically, we appreciate the "efficiency" of this encoding of a scene, and enjoy the experience of unpacking it.

Figure 1.4: Examples of two-tone images. A graffiti work by Banksy is shown in (a). A picture from Frank Miller's Sin City is shown in (b) and a page from Neil Gaiman's Sandman is shown in (c).

Other traditional techniques use large regions of black and white instead of lines. Examples are shown in Figure 1.3. Some ink paintings use large black forms to carry all the salience of an image [150]. Papercutting, by its nature a binary medium, depicts an image by cutting holes in paper. Closely related is stencilling, as in the graffiti work of the British artist Banksy (Figure 1.4(a)). Many woodblock prints ignore tone (though masters of the medium may include hatching or stippling in their blocks). Contemporary examples include Frank Miller's *Sin City* comic [81] and the computer animated film *Renaissance* [108]. Both use pure black and white to reinforce the stark setting of their stories. The story "Exiles" in Neil Gaiman's *Sandman* series also makes heavy use of black and white, in a looser brush stroke style. A photographer can reduce an image to pure black and white with no grays by copying a negative onto high-contrast

litho film and printing that negative [48]. Concrete examples in nature, such as black-and-white stripes on a zebra and the black-and-white coat of giant panda, make us admire the beauty of this style.

While the problem of producing line drawings from images has received attention in non-photorealistic rendering (NPR) research [60], depiction using large black and white regions remains underexplored. Examples like *Sin City* (Figure 1.4(b)) and *Sandman* (Figure 1.4(c)) inspired me to explore the inherent features of this style of illustration.

Figure 1.2 offers a sampling of black-and-white imaging styles. Notice that a subset of them have similar properties. For instance, the first two rows of Figure 1.2 have only two tones and abstract shapes. This visual style is further exemplified by the Calvin and Hobbes comic in Figure 1.5 and by the famous portrait of Che Guevara in Figure 1.6. In such an extreme black-and-white style, we can only observe pure black and pure white regions. The contrast between black and white is sometimes used metaphorically to express balance or opposition as in the *Yin Yang* symbol and M.C. Escher's work *Encounter* [32].

By analogy with halftoning, I use the term "wholetoning" to refer to black-and-white illustrations with the kind of abstraction in shape and tone described here. This is a distinct, recognizable style within black-and-white illustration, one that deserves to be studied in isolation. I will offer a more precise definition of this term in the next section.

In this dissertation, I examine the following problem: given a colour input image (*observation*), can I convert it into a wholetoned representation (*composition* and *depiction*)? This topic can be classified as a research area in NPR that focuses on emulating traditional styles of art and illustration and developing expressive rendering styles. By exploring this wholetoned style, I hope to deepen our understanding of black-and-white illustration and complement the already extensive published literature.

In the rest of this chapter, I offer a definition of wholetoned style and discuss the challenges of my research. In Chapter 2, I review the mathematical and computational background related to this work. The state of the art in black-and-white image synthesis is introduced in Chapter 3. Chapter 4 presents my general framework for wholetoning. Then, I investigate two specific wholetoned styles: papercutting in Chapter 5 and representational calligraphy in Chapter 6. Finally, I finish with conclusions and a discussion of future work in Chapter 7.

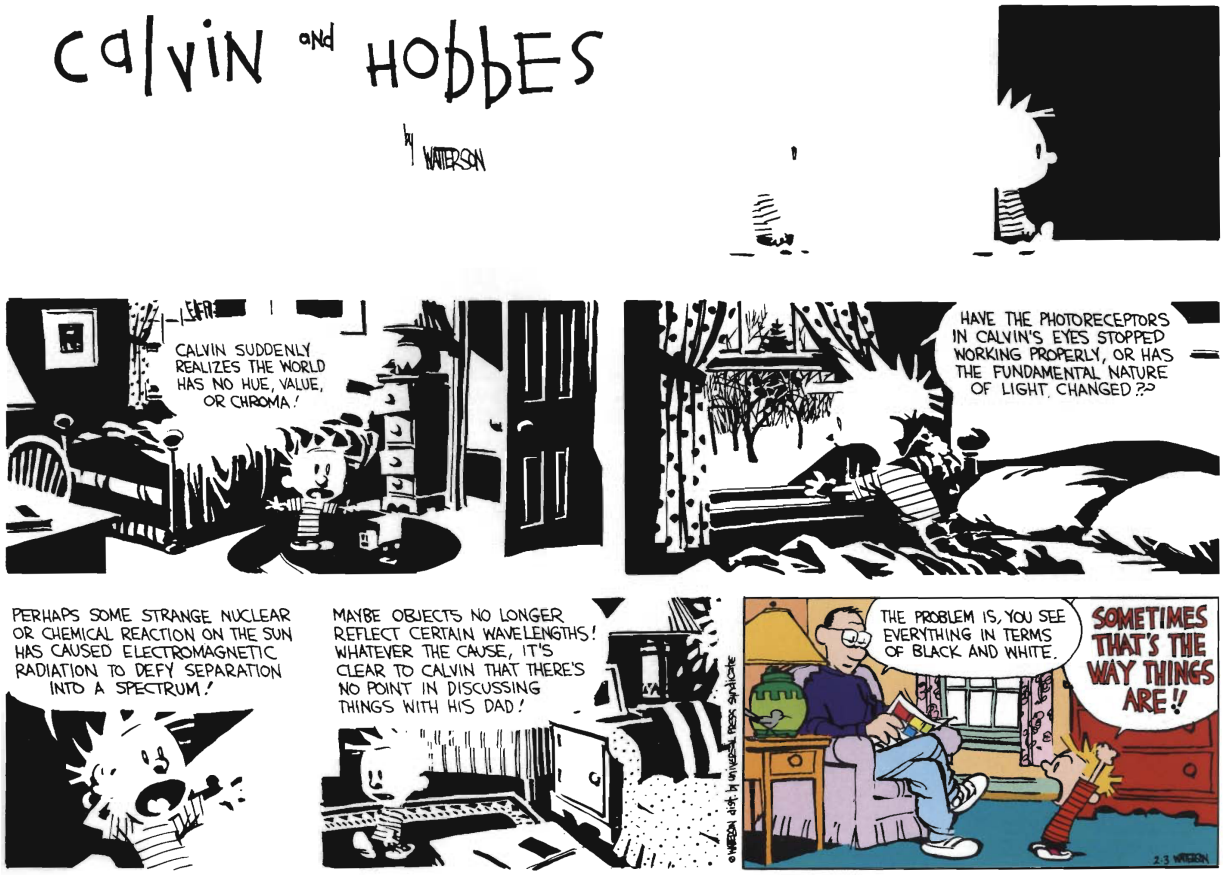


Figure 1.5: A Calvin and Hobbes comic [137, Page 57] explores the mystery of a black-and-white world. Watterson recognized it is a distinct style: "To draw this strip, I not only avoided color and halftones, I avoided outlines. As the top 'throwaway' panels illustrate, the placement of black is crucial to making the pictures comprehensible."

1.1 Definition of wholetoned style

I analyze the collection of black-and-white images in Figure 1.2 by ordering them informally for visualization purposes. I begin by studying these images in terms of the visual features described earlier: tone, texture and shape. Figure 1.7 orders images based on the range of tones they contain. Halftoned images, in which black primitives are interpreted as continuous tones by our visual systems, are intermediate between highly abstract images made of large black regions, and

Figure 1.6: One of the most iconic portraits of the twentieth century. The left portrait of Che Guevara was taken by Alberto Korda. The right stylized poster was created by Jim Fitzpatrick.

Figure 1.7: Tone variations in black-and-white visual arts. The number of tones increases along the tone axis. The images of interest to me (in the boxed region) include only two tones.

continuous-toned images such as photographs. Note that if you are reading this page in print, all examples are halftoned — The images on the right are merely halftoned at the printer's finer scale. I am interested in images that are highly abstract in tone, shown boxed in the figure. They only have two tones, the minimum value for any meaningful representation.

Some black-and-white illustrations are abstractions of realistic pictures. The abstraction can be attained by suppressing textures and simplifying geometric shapes. Figure 1.8 orders the example images by faithfulness of texture reproduction. The abstract styles lie on the left side.

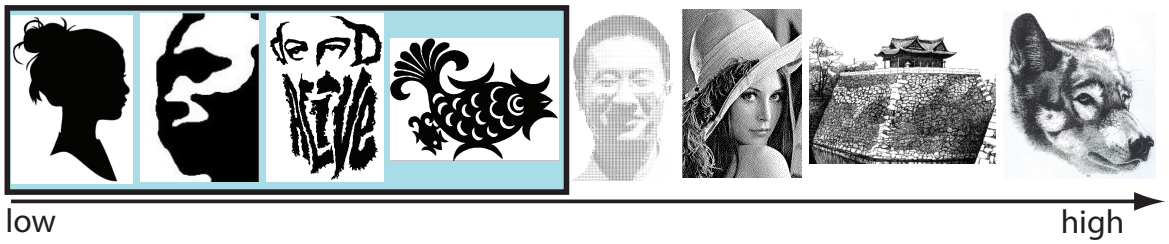


Figure 1.8: Faithfulness of texture reproduction. The images of interest to me (in the boxed region) have low texture fidelity.

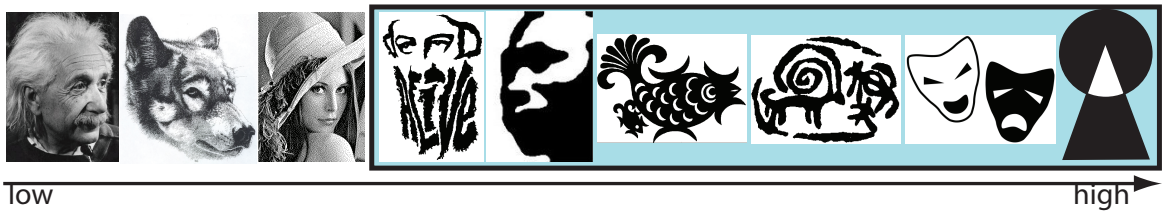
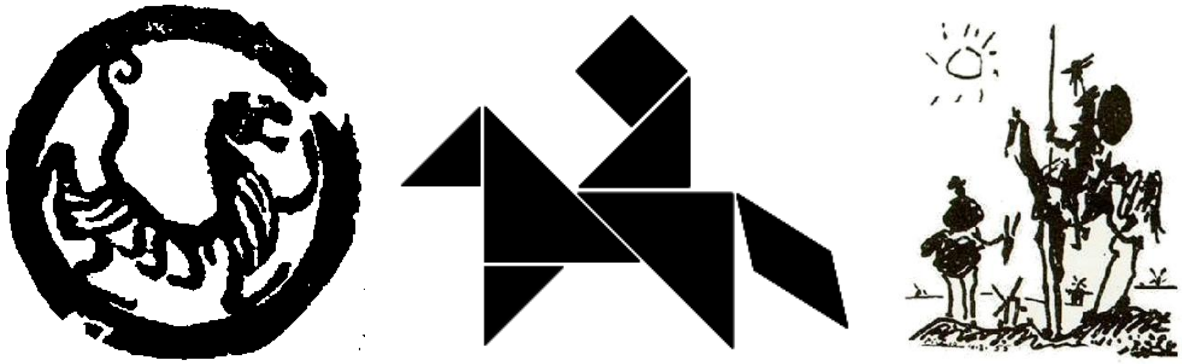


Figure 1.9: Shape abstraction in black-and-white visual arts. The images of interest to me (in the boxed region) are those with a high degree of abstraction.

Texture reproduction and abstraction are hard problems in computer graphics. Because of the importance and usefulness of texture in pattern and object recognition, there has been a fair amount of previous work on this topic [113, 6]. However there is no general approach to detect and analyze texture patterns.

Figure 1.9 demonstrates shape abstraction in black-and-white visual arts. I would like to explore the styles that lie on the right-hand side. Shape abstraction can be achieved in two ways: simplification that omits some properties or details, and idealization that represents an object by an approximate ideal form. The artist can apply shape abstraction to maximize the likelihood that the information he deems important will be received and processed. There are many real-world examples of shape abstraction. The ancient Chinese seal-cut in Figure 1.10(a) is a good example. Tangrams (Figure 1.10(b)), a traditional puzzle game, is another extreme case that can compose hundreds of objects with only seven simple geometric shapes. Some artists are also masters of stylized black-and-white drawings (Figure 1.10(c)). In these images, only a few simple lines or shapes are used, but the features of objects are depicted well. The abstraction of images empowers us to organize the essential geometric features to fit human perception and



(a) (b) (c)

Figure 1.10: Abstraction images: a Chinese seal-cut (tiger pattern) in (a) , an example of Tangram (b) and a painting (c) (Pablo Picasso's "Don Quixote") (Scan by Mark Harden http://www.artchive.com).

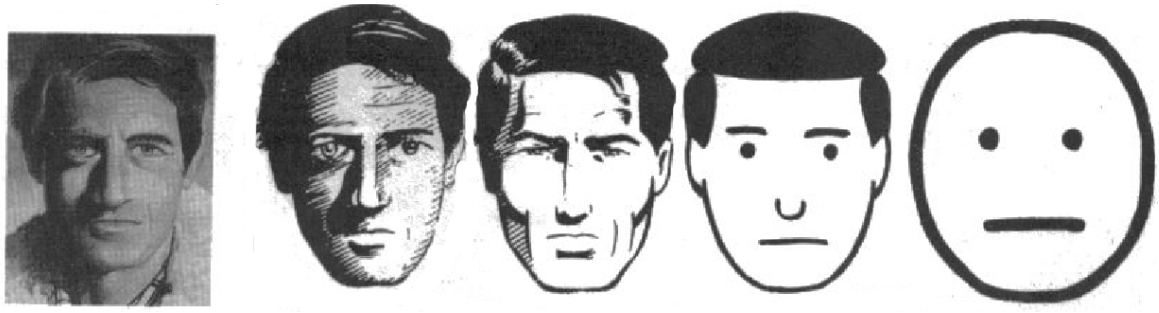


Figure 1.11: Image taken from Scott McCloud's Understanding Comics [80, Page 31]. It shows the universality of comics. While a realistic picture turns into an abstract image by simplifying, more people can identify with the face being depicted, because a more abstract image connects to the essential features more directly.

cognition more naturally. A good demonstration is the power of comics [80]. When we abstract an image through cartooning, we are not so much eliminating details as we are focusing on specific details. As shown in Figure 1.11, by stripping down an image to its essential meaning, an artist can amplify that meaning in a way that realistic art cannot offer. This aspect of depiction in comics explains one cause of our fascination with them. It helps absorb our sense of identity so that we become an active part in the comic, seeing "ourselves" instead of "others". As with

Figure 1.12: Dimensionality of primitives in visual art. Zero dimension means using points. One dimension means using lines. Two dimensions mean using regions. The images of interest to me (in the boxed region) use lines and regions, but emphasize regions.

texture abstraction, shape abstraction is also a difficult research topic [21, 38, 122]. It is closely related to cognitive psychology. The role of the human observer must be taken into account in abstraction processing.

Kandinsky analyzed the geometrical elements that compose every painting, namely point, line and plane [59]. Wucius Wong also listed four conceptual elements in pictorial design: point, line, plane (or region) and volume [144]. The elements are individual primitives that are used to depict objects in an illustration, and not the actual content. A point indicates position. It has no length or breadth. As a point moves, its path becomes a line. A line has length but no breadth. The path of a line in motion becomes a plane. A region has length and breadth, but no thickness. The path of a region in motion becomes a volume. It has position in space and is bounded by regions. Although volume primitives can be simulated, no real volume exists in two-dimensional art. The choice of primitives determines what we call the dimensionality of representation. I order images by dimensionality in Figure 1.12. I am interested in artistic styles based on two-dimensional primitives.

From the preceding discussion, I can give a working definition of wholetoning.

Definition: a *wholetoned image* is a black-and-white image in which

1. Only two tones are used;
2. Little texture information is involved;
3. A high degree of shape abstraction is used; and
4. The pictorial elements are lines and regions.

This definition is of course “loose”. It cannot be perfectly precise, and surely there will be

(a) (b)

Figure 1.13: Variants of wholetoning: A papercutting example in (a) and a representational calligraphy example by AlmapBBDO in (b).

illustrations that are on the verge of being wholetoned, but not quite. However, it clearly states the main features that I focus on and is good enough for our purposes. I use the term "wholetoning" to refer to any algorithm for constructing wholetoned images.

Wholetoning algorithms have many potential applications today, for example in printing and illustrations. People frequently use simplified images when they want to express complicated ideas. This rule is also applied to icon and logo design, where highly stylized small images are preferred [141]. Abstract logo images have the power to stimulate the imagination and recall to our minds images, experiences and feelings embedded in memory. Further, faithful reproduction of tone is not always desirable, even when possible. Some artists deliberately employ this pure black-and-white technique to obtain special effects that can be found in comic books and movies.

Moreover, we can impose some extra constraints to generate variants descended from the basic wholetoned style, such as papercutting (Figure 1.13(a)), stencilling, and representational calligraphy (Figure 1.13(b)). Notice that a paper-cut design must satisfy the constraint that it can be cut out from one piece of paper, and the depiction primitives in a work of representational

calligraphy are limited to letters.

The goal of my work is to combine the expressivity and beauty of wholetoned styles with the power and flexibility of computer graphics. I want to explore opportunities where mathematical and computational models can be applied to the synthesis of wholetoned images. The goal will be achieved by analyzing the features of wholetoned images, and presenting a general framework to define a wholetoning pipeline. After studying wholetoning in general, I focus on how to reproduce two particular derived styles: representational calligraphy and papercutting. To capture the outstanding appearance of a specific style, I must derive a specialized algorithm that captures the idiosyncrasies of that style. During my investigations of representational calligraphy and papercutting, I watch for principles and techniques that might be applied to other wholetoned styles.

1.2 Challenges of my research

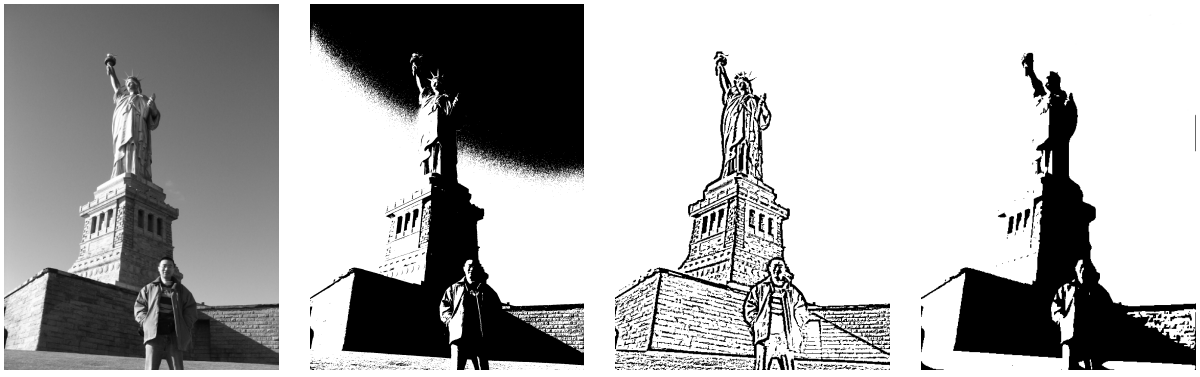
Before I discuss my research on wholetoning in greater detail, it is instructive to see the limitations of existing methods and examine some considerations face by an artist when converting a colour image into a wholetoned image. We should consider the following problems: How to deal with tone? What is the trade-off between line-based depiction and plane-based depiction? How to make use of feature homogeneity? And how to derive more wholetoned styles? All of these problems have significant implications for my approach.

1.2.1 Wholetoning via image processing

A naive way of generating a wholetoned representation of a continuous-tone image is thresholding:

$$Color(i,j) = \begin{cases} White & \text{if } I(i,j) > T \\ Black & \text{otherwise} \end{cases} \tag{1.1}$$

Here, $I(i,j)$ is the intensity value of the pixel at position (i,j), normalized to the range $[0,1]$. T is the threshold value. Unfortunately, the quality of the result is usually poor (see for example the colour gradient in the sky region in Figure 1.14(b)). The reason is that this method applies one single threshold value globally. It also disregards the difference between luminance (a physical



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

Figure 1.14: Examples of applying different thresholding methods. Given a source image (a), we can get a black-and-white image by thresholding in (b), adaptive thresholding in (c) and thresholding after bilateral filtering in (d).

property) and brightness (a product of human perception), which is highlighted by well-known contrast illusions [34].

An enhanced method is adaptive thresholding, which changes the threshold value dynamically over the image [37], as shown in Figure 1.14(c). It can do a better job, but the result has too much noise. There are some more advanced methods such as applying thresholding immediately after bilateral filtering (Figure 1.14(d)). This approach is more successful, but it still loses too many feature details and the user has little control over style variations. With respect to my wholetoning definition, these methods can satisfy the tone condition but are weak in the other properties. My challenge is to produce a wholetoning algorithm that can generate convincing results and also provide flexible control of styles.

1.2.2 Tone preservation and tone inversion

Since tone is one of the most important communication factors in wholetoning, I should deal with tone carefully. Basically, I look forward to the following property in wholetoning: the dark parts in original image should be turned into black while the bright parts should be white. I call it “tone preservation”. An image violating this rule is called a “tone inverted” image. One problem concerning tone is whether a tone inverted image is as perceptually acceptable as

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

Figure 1.15: Tone inversion and tone preservation examples. The source picture in (a) [151, Page 45] and the tone inversion result in (b) are both perceptually acceptable. But for the paper-cut design in (c) [71, Page 131], our perception will reject the inverted picture of (d).

Figure 1.16: Spatial scales of images in Figure 1.15. Each image includes a high spatial frequency data (left) and low spatial frequency data (right). Tone inversion images have similar high spatial frequency data as original images. But low spatial frequency data are totally different.

the original image. Although local tone inversion of part of an image is enough for graphic design, I will discuss some tone inversion examples of a whole image to explain my findings. I observe that some images and their inverted counterparts are equally good for human perception (Figure 1.15(a) and (b)), while others are not (Figure 1.15(c) and (d)).

This difference originates from the representation characteristic of images. Normally, an image contains variations in what neuroscientists term "spatial frequency" [106]. High spatial frequencies correspond to abrupt spatial changes in the image, such as sharp, fine lines and fine details. Low spatial frequencies correspond to global information about the shape such as blurred edges or large objects. By applying the image processing techniques of high-pass filtering and low-pass filtering, an image can be separated into different spatial scales. From Figure 1.16, we can see that the inverted images have similar high spatial frequency information as the originals while low spatial frequency information is different. For the fish pattern, the high spatial frequency information alone is enough for human perception, so both images are perceptually acceptable. But for the portrait example, we have to rely on low spatial frequency information to help us finish the recognition. Because the low spatial frequency data of inverted image differs a lot from original one, our perceptual system will reject it. The perception of some images is guided by information at high spatial frequencies (where only the edges can provide enough information), others by information at low frequencies (where shadow is usually engaged). This distinction is important because artists use different techniques to abstract tone in high- and low-frequency dominant images. I will discuss this topic more in Section 1.2.3.

Figure 1.17: A simple tone violation example. With some tone violations, the right picture encodes more shape information than the left picture.

I also notice that it is possible and even helpful to introduce some minor violations of tone

(a) (b)

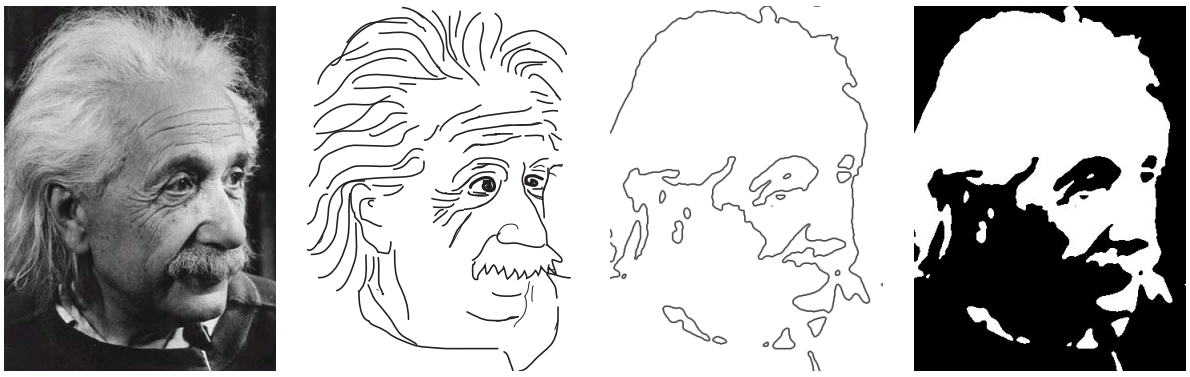
Figure 1.18: Tone inversion in wholetoned images: A papercutting (by the Zhou family) in (a) and a logo example (D'Amico & Partners by Duffy & Partners) in (b).

preservation regardless of spatial frequency. To highlight important contrasts between elements, we have to break the "tone preservation" by inverting tone. Otherwise some feature components will be lost. Figure 1.17 demonstrates the necessity of inverting tone to depict shape. Tone inversion clearly discloses how the three rectangles overlap each other (though not their relative ordering). There are some concrete examples in real art works, such as the table legs and chair legs in Figure 1.18(a) and the human heads and letters in (b). The challenge here is to depict complex scenes in black and white in a way that balances between the conflicting goals of tone preservation and shape depiction. To do this, I need to understand when tone can safely be inverted.

1.2.3 Line-based and plane-based representation

Corresponding to the dominant spatial frequency, shape depiction can be divided into two categories: line-based, which refers to high spatial frequency and plane-based, which refers to low spatial frequency. The former depicts a shape with only the outer boundary, and the latter depicts a shape with the entire shape region. The outer boundaries can be object contours, shadow contours and texture contours. Line drawings that depict shape features using only thin line strokes can create convincing representations. Figure 1.19(b) shows an example. However, the wholetoned image conflates information about object structure and illumination. Its outer boundaries

are a combination of object contours and shadow contours. In this situation, the outer boundary alone (Figure 1.19(c)) may not provide enough clues to finish the perception task, because these boundary lines correspond to the brightness discontinuities in the image. They cannot distinguish object contours from shadow contours. Shadows have great effect on recognition [12]. To accurately recover the structure of an object, it is essential to identify the cast shadow borders in an image because they are generally unrelated to the object contours and they seriously disrupt the interpretation of the image if they are confused with object contours. This explains why Figure 1.19(c) is unsuccessful. Therefore, a good line drawing should highlight object contours that are the key feature guiding our perception instead of simply tracing the brightness discontinuities. Otherwise, this type of representation is rendered meaningless by missing shadow contours and object contours. With the help of shaded regions, we can get hints to discover shadow contours (see Figure 1.19(d)). These differences highlight the need to explore style variation between line-based and plane-based depiction.



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

Figure 1.19: Given the source picture in (a), a line drawing is shown in (b). The outer boundary of a wholetoned image (d) is in (c).

1.2.4 Feature homogeneity

Visual perception interprets objects with two strategies: a bottom-up strategy that relies on the input stimulus in recognition, and a top-down strategy that is also influenced by the viewer's

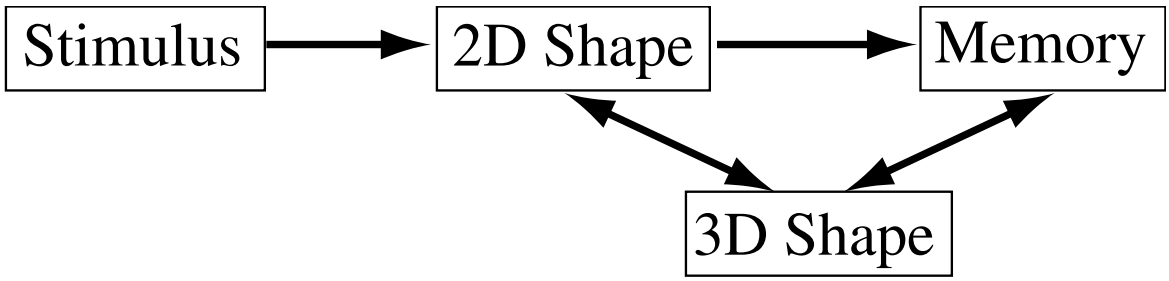


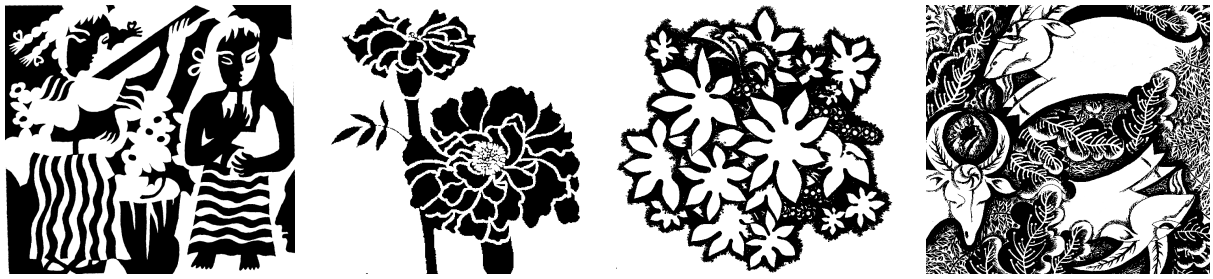
Figure 1.20: The procedure of top-down perception strategy. The input stimulus drives recognition to find a match of 2D shape information against memory prototypes, which then iteratively guide the construction of a 3D model.

prior experience [12, 13]. As shown in Figure 1.20, the top-down strategy is a feedback model. It is driven by our memory and attention, and controlled by higher level cognitive processes, which means that we must depend on our prior experience to aid recognition. Based on an input stimulus, recognition constructs a crude match of 2D views to internal prototypes in memory. The prototype guides the construction of a 3D model that can adjust the interpretation of 2D views. Finally, the completed 3D model determines our recognition. In top-down perception, grouping is important in that we group together seemingly unrelated fragments of a single object when we are interpreting it [67]. After grouping, we try to search a matched template in our memory to identify the image.

As I mentioned before, a wholetoned image is an abstraction of an input image due to its simplification and exaggeration in tone and shape. In the NPR literature, abstraction means removing details so that the main point of a presentation is perceptually salient. Research in cognitive psychology shows that we depend mainly on top-down perception in interpreting an abstract image [12]. Therefore, another important consideration in wholetoning is that it should represent the features so that they can easily arouse our memory.

1.2.5 Wholetoning style variants

As mentioned in Section 1.1, wholetoning includes many variants. The visual appeal of wholetoned images attracts me to explore how I can reproduce similar styles. To simulate papercutting (Figure 1.13(a)) and representational calligraphy (Figure 1.13(b)), the challenge that I face here



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

Figure 1.21: Artistic devices in black-and-white illustration: switching the foreground and background colour deliberately in (a) [15, Page 38], contoured silhouettes in (b) [15, Page 40], relief by shadowing in (c) [15, Page 42] and contrast between homogeneity and heterogeneity in (d) [15, Page 44].

is that some specific algorithms have to be developed. For instance, a representational calligraphy algorithm should try to warp letter forms to fit a shape while preserving their readability as well as possible. And a papercutting algorithm should guarantee that the final design consists of a single connected component.

Figure 1.22: An example of illusory contours in a papercut design by Susan Throckmorton.

Artists have developed many techniques and devices to deal with a small number of tones, which can be borrowed in wholetoned image design. A colour can appear as foreground and background in the same design (Figure 1.21(a)). Solid color regions can be outlined by contours to give a concise effect (Figure 1.21(b)). Objects can be emphasized by the effects of relief or halo that are simulated by shadow regions surrounding the contours of objects (Figure 1.21(c)). The contrast between homogeneity and heterogeneity can help highlight objects, as shown in Figure 1.21(d): the homogeneous goat figures stand out prominently from the heterogeneous background. Due to the very limited range of tones in wholetoned images, some visual features may even not be presented explicitly. But a skilled artist can make use of our perception to reconstruct those lost features. A good example is the contours of the human figures in Figure 1.22. With the help of the patterns on cloth, our brains can easily infer the missing portions of the characters' outlines. We are probably aided in this process by our perceptual ability to perceive "illusory contours" [128]. The challenge, then, is to analyze artistic techniques and devices, learn from the wisdom and experience of artists, and incorporate this understanding into wholetoning algorithms.

1.3 Contributions

In this dissertation, I define an art style—wholetoning—and develop a general framework called "artistic thresholding" to generate wholetoned illustrations from continuous-tone images. I then study how to simulate two particular wholetoned styles: papercutting and representational calligraphy. I expect that the analysis, algorithms, and results presented here will provide valuable ideas to the computer graphics community, and drive further research in abstract black-and-white illustration.

As is common in NPR research, I do not try to establish objective standards for evaluating the quality of the output of my algorithms. We are starting to see papers in the NPR literature that attempt to do evaluation [54, 18, 143], papers just about evaluation [76, 53], and new conferences [91] related to the problem. I am hopeful that future work will uncover more objective standards by which images like those shown in this thesis might be evaluated.

The principal contributions of this dissertation are as follows:

- I provide the formal concept of wholetoned images, define the style of wholetoned images,

and articulate a set of constraints that can be used to produce wholetoned images from photographs in general.

- A general framework—artistic thresholding—is presented to generate wholetoned images. I apply segmentation to a source image and user-provided high-level information to extract feature homogeneity. A planar subdivision that captures segment connectivity is constructed. My artistic thresholding algorithm is a combinatorial optimization over this graph. The optimization is controlled by wholetoning style constraints. A wholetoned image is the result of those competing forces that comprise the trade-offs in tone reproduction, depiction style, and salient features. This comprehensive approach overcomes the deficiencies of existing methods and can be tuned to achieve different artistic styles.

- I explore one variant of wholetoning: papercutting. I define the papercutting as a wholetoned image with connectivity constraints to ensure that it can be cut out from only one piece of paper. I present a technique for composing digital paper-cut designs. The elements of a design may be images, which are processed via artistic thresholding and a connectivity algorithm, or they may be procedurally generated arrangements of shapes that simulate traditional paper-cut patterns. Elements are composed using a set of boolean operators that preserve connectivity.

- I study another wholetoning variant: representational calligraphy. A representational calligraphy design is defined as a wholetoned image with the constraint that all depiction elements must be letters. The procedure of generating representational calligraphy designs is formalized to a "calligraphic packing" problem. I develop a semi-automatic solution based on dividing up a target region into pieces and warping a letter into each piece. I define an energy function that chooses a warp that best represents the original letter.

Chapter 2

Background

In this chapter, I will introduce some general background knowledge in my research. More specific knowledge will be introduced in later related chapters.

2.1 Image processing

The similarities and interrelations between computer graphics and image processing make image processing techniques widely applied in computer graphics systems. To be able to compute wholetoned illustrations from images, I apply many existing image processing approaches. So I introduce some of them here.

2.1.1 Blurring

Blurring operations are used in many visual computing applications to reduce image noise and details. One typical operation is Gaussian smoothing. It is defined as the convolution of an image with a Gaussian function. A two-dimensional Gaussian function is:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/(2\sigma^2)}, \tag{2.1}$$

where σ is the standard deviation. The width of the Gaussian increases as σ increases.

Mathematically, a convolution of two functions f and g is given by the following equation:

$$(f * g)(x,y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x-x', y-y') g(x', y') dx' dy'. \tag{2.2}$$

(a) (b) (c)

Figure 2.1: An example of blurring. The source image is shown in (a). The Gaussian blurred result is in (b) and the bilateral filtered result is in (c).

In image processing applications, convolution can be viewed as a kernel window of some finite size scanned across the image. The output pixel value is the weighted sum of all pixels within the window where weights are given by the elements in the kernel window. If I is an image, and K is the kernel function whose dimension is $M \times N$, then this discrete convolution is given by

$$(I * K)[x,y] = \sum_{x'=0}^{M-1} \sum_{y'=0}^{N-1} I[x-x', y-y'] K[x', y']. \tag{2.3}$$

Gaussian smoothing is defined as $(I * G)[x,y]$, where G is a discrete Gaussian kernel. Figure 2.1(b) shows an example of Gaussian smoothing.

Because the effect of Gaussian blurring is to compute a weighted average of pixel values in the neighborhood isotropically, it leads to the whole image being smoothed evenly. To prevent averaging across obvious edges while still averaging within smooth regions, some anisotropic blurring methods have been developed [100]. Bilateral filtering is a typical one [130]. It extends Gaussian smoothing by weighting the filter coefficients with their corresponding relative pixel intensities. It is composed of two Gaussian filters. One is in the spatial domain, named the domain filter, and the other is in the intensity domain, named the range filter. The definition is

given by the following function:

$$h(\mathbf{x}) = k^{-1}(\mathbf{x}) \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(\xi) c(\xi, \mathbf{x}) s(f(\xi), f(\mathbf{x})) d\xi, \tag{2.4}$$

$$k(\mathbf{x}) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} c(\xi, \mathbf{x}) s(f(\xi), f(\mathbf{x})) d\xi, \tag{2.5}$$

where $f(\mathbf{x})$ is the intensity value of a given pixel $\mathbf{x}$ and $c(\xi, \mathbf{x})$ is a domain filter that measures the spatial distance between the neighborhood center $\mathbf{x}$ and a nearby point ξ. It is commonly defined as

$$c(\xi, \mathbf{x}) = e^{-\frac{1}{2}(\frac{||\xi - \mathbf{x}||}{\sigma_d})^2}, \tag{2.6}$$

where σ_d is the standard deviation of the domain filter.

$s(f(\xi), f(\mathbf{x}))$ is a range filter that measures the intensity similarity between the pixels. It is commonly defined as:

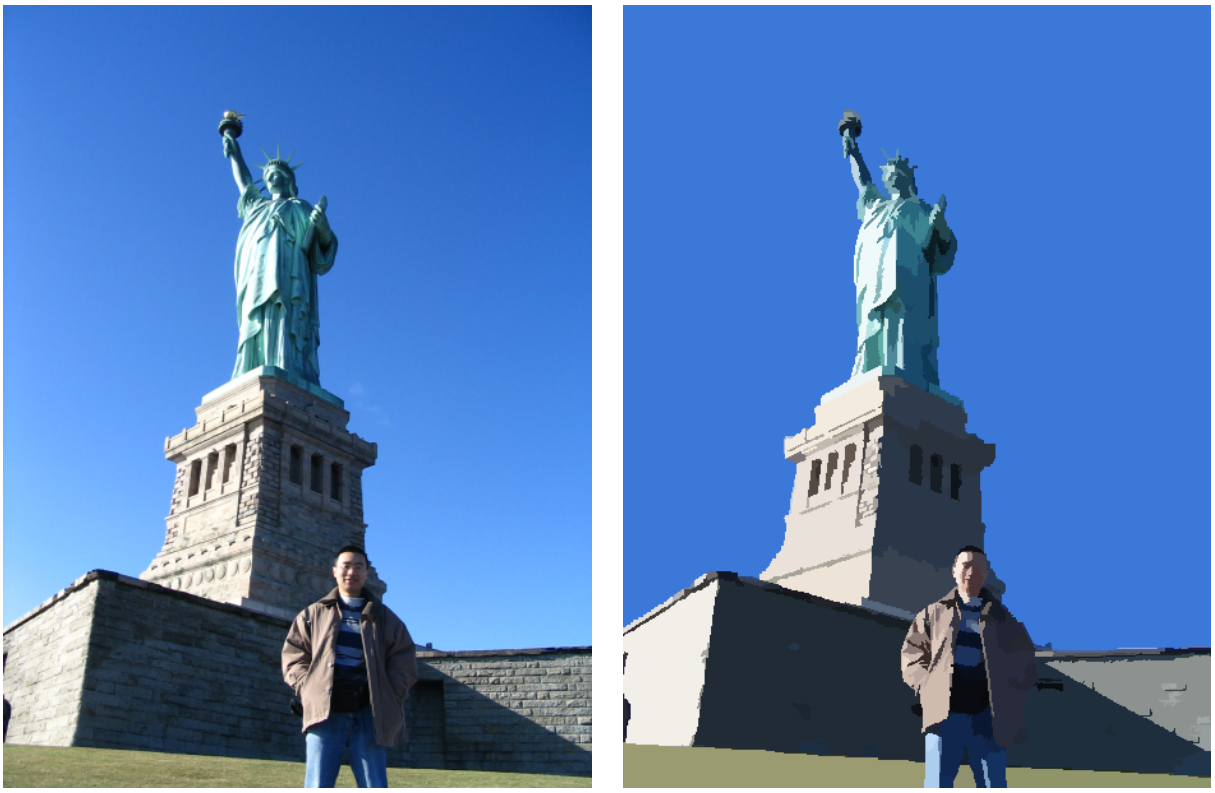
$$s(f(\xi), f(\mathbf{x})) = e^{-\frac{1}{2}(\frac{||f(\xi) - f(\mathbf{x})||}{\sigma_r})^2}, \tag{2.7}$$

where σ_r is the standard deviation of range filter. Both spatial distance and intensity distance are computed as Euclidean distance. Figure 2.1(c) gives a result using bilateral filtering.

2.1.2 Segmentation

In many applications of image processing techniques, great importance is attached to image segmentation. The goal of image segmentation is to partition an image into homogeneous and connected regions. Homogeneity of regions in image segmentation usually involves colours. In contrast to single pixels in the original image, the regions of the segmented image have more consistent features like shape, boundary, texture, and so forth.

Many segmentation methods have been developed [56]. These methods can be classified as pixel-based methods that use properties of the individual pixels, region-based methods that analyze the properties in larger areas, and edge-based methods that detect edges and follow them to isolate regions. Among these methods, mean shift segmentation is a robust and versatile region-based method that clusters pixels to segment regions [20]. Mean shift segmentation applies a kernel of influence for each pixel x_i to estimate the density. This kernel defines a measure of



(a) (b)

Figure 2.2: Mean shift segmentation example. Given a source colour image (a), the segmented image is shown in (b).

the distance between two pixels that operates both in the spatial and colour domains. The kernel function is given by

$$\hat{f}(x) = \frac{c}{n(h^s)^p(h^r)^q} \sum_{i=1}^{n} k(\left\| \frac{x^s - x_i^s}{h^s} \right\|^2) k(\left\| \frac{x^r - x_i^r}{h^r} \right\|^2), \tag{2.8}$$

where c is a normalization constant, n is the number of pixels, h^s is the spatial bandwidth and h^r is the range bandwidth. Those two parameters are given by users and define the size of kernel. $p(=2)$ is the spatial dimension size and $q(=3)$ is the colour dimension size. x^s and x^r are the spatial and range parts of a feature vector. $k(x)$ is the profile of the kernel. The Epanechnikov

kernel is sufficient for most applications. Its profile is described as:

$$k(r) = \begin{cases} 1-r & 0 \leq r \leq 1 \\ 0 & r > 1 \end{cases} \tag{2.9}$$

The mean shift procedure iteratively moves every pixel along the gradient of the density until they converge to a stationary point. The pixels that converge to the same stationary point are clustered into a single segment.

Figure 2.2 shows an example of mean shift segmentation of a colour image using EDISON's synergistic image segmenter [16].

2.1.3 Feature detection

Edge detection is one of the fundamental feature detection operations. Many other operations are based on edge detection. The edges of objects in an image hold much information including location, size, shape, and texture.

The Sobel operator is a simple difference filter that computes an approximation of the gradient of the image (Figure 2.3(a)). It defines a pair of 3×3 convolution masks corresponding to horizontal and vertical directions:

$$K_x = \frac{1}{8}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \; K_y = \frac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}. \tag{2.10}$$

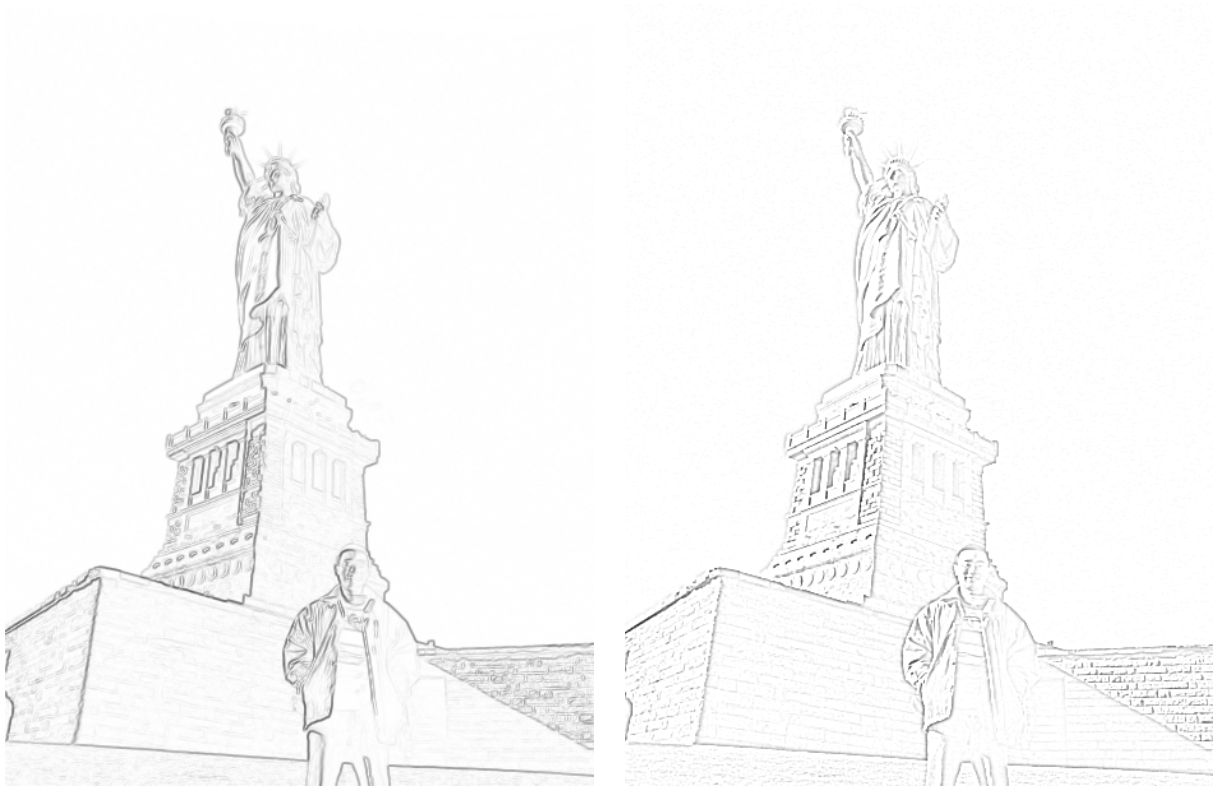
By applying convolution, the horizontal derivative G_x and vertical derivative G_y can be approximated as

$$G_x = I * K_x, \; G_y = I * K_y. \tag{2.11}$$

The resulting gradient approximations can be combined to compute the gradient magnitude G and direction θ at each pixel:

$$G = \sqrt{G_x^2 + G_y^2}, \; \theta = \arctan(\frac{G_y}{G_x}). \tag{2.12}$$

The gradient image is the edge detection result.



(a) (b)

Figure 2.3: Edge detection examples. Given the source image in Figure 2.1(a), The Sobel mask operator produces a result in (a) and a differences of Gaussians operation produces the result in (b).

Another important edge detector is the differences of Gaussians (DoG) operator. The DoG operator allows the width of a convolution mask to vary in order to adjust the detail in the output. A wider mask will eliminate more details. The masks are differences of Gaussian functions.

The two-dimensional DoG function is defined as

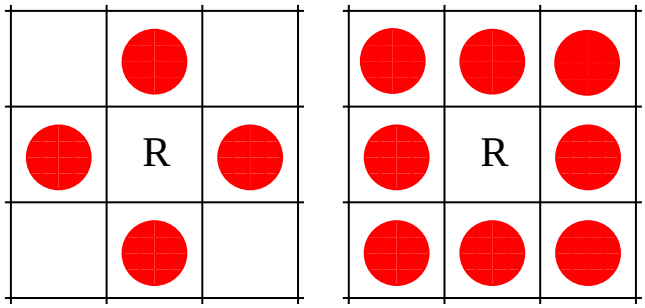
$$f(x,y,\sigma_1,\sigma_2) = \frac{1}{2\pi\sigma_1^2}e^{-(x^2+y^2)/(2\sigma_1^2)} - \frac{1}{2\pi\sigma_2^2}e^{-(x^2+y^2)/(2\sigma_2^2)}, \tag{2.13}$$

where x, y are the distances from the origin in the horizontal and vertical axes respectively, and σ_1, σ_2 are two different standard deviations of two Gaussian distributions, usually $\sigma_2 = 1.6\sigma_1$. This DoG function is used as a convolution kernel on images. Figure 2.3(b) gives an example of

DoG edge detection.

2.1.4 Morphological operations

There is a set of operators that can be described in terms of adding or removing pixels from a binary image according to certain rules, which depend on the pattern of neighboring pixels. They are called morphological operators. While most morphological operations focus on binary images, some can be applied to grayscale images.



(a) (b)

Figure 2.4: Given a reference pixel, the 4-neighborhood is shown in (a) and 8-neighborhood is shown in (b). The red circles indicate neighboring pixels.

Morphological operations are performed in a small neighborhood of a reference pixel, which includes the reference pixel itself. In many applications, the 4-neighborhood or 8-neighborhood is used (Figure 2.4). The corresponding masks are

$$M_{four} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}, M_{eight} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}. \tag{2.14}$$

There are four basic binary morphological operators: EROSION, DILATION, OPEN and CLOSE [56]. We label every pixel in a binary image by zero and one, corresponding to white and black pixels respectively. Let G be the set of all the pixels that are not zero. M_p denotes the mask (M_{four} or M_{eight}) shifted so that its center coincides with the pixel p. EROSION is defined as

$$G \ominus M = \{p | M_p \subseteq G\}. \tag{2.15}$$

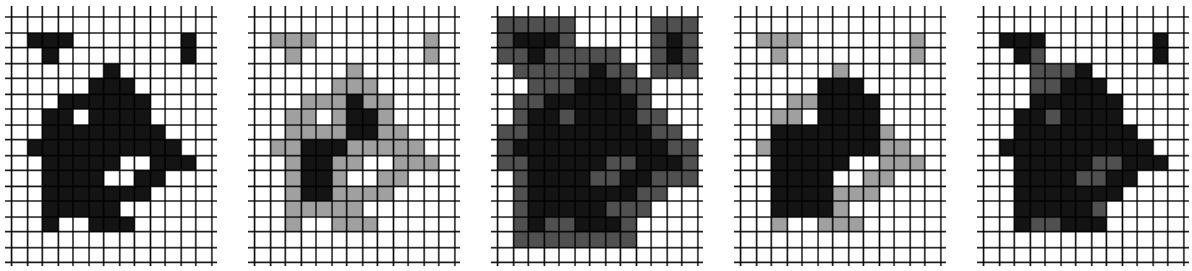


Figure 2.5: Morphological operations. The removed pixels are shown in light gray colour and added pixels are shown in dark gray colour.

This equation means that if a pixel and all its neighbors are in G, then this pixel is in the erosion result.

DILATION is defined as

$$G \oplus M = \{p | M_p \cap G \neq \emptyset\}. \tag{2.16}$$

This equation means that if a pixel is in G or at least one of its neighbors is in G, then this pixel is in the dilation result.

The EROSION operation is useful for eliminating small objects. But it has the disadvantage that all the remaining objects shrink in size. We can avoid this effect by applying DILATION to the image after EROSION. This combination of operations is called OPEN:

$$G \circ M = (G \ominus M) \oplus M. \tag{2.17}$$

Intuitively, OPEN refers to the ability to open up gaps between just-touching features. It is commonly used to remove pixel noise.

In contrast, the DILATION operator can fill holes and cracks but enlarge objects. So we can define a CLOSE operation that is the combination of EROSION following DILATION:

$$G \bullet M = (G \oplus M) \ominus M \tag{2.18}$$

CLOSE refers to the ability to close breaks in features. It is used to fill in missing pixels within features and narrow gaps between components of a feature.

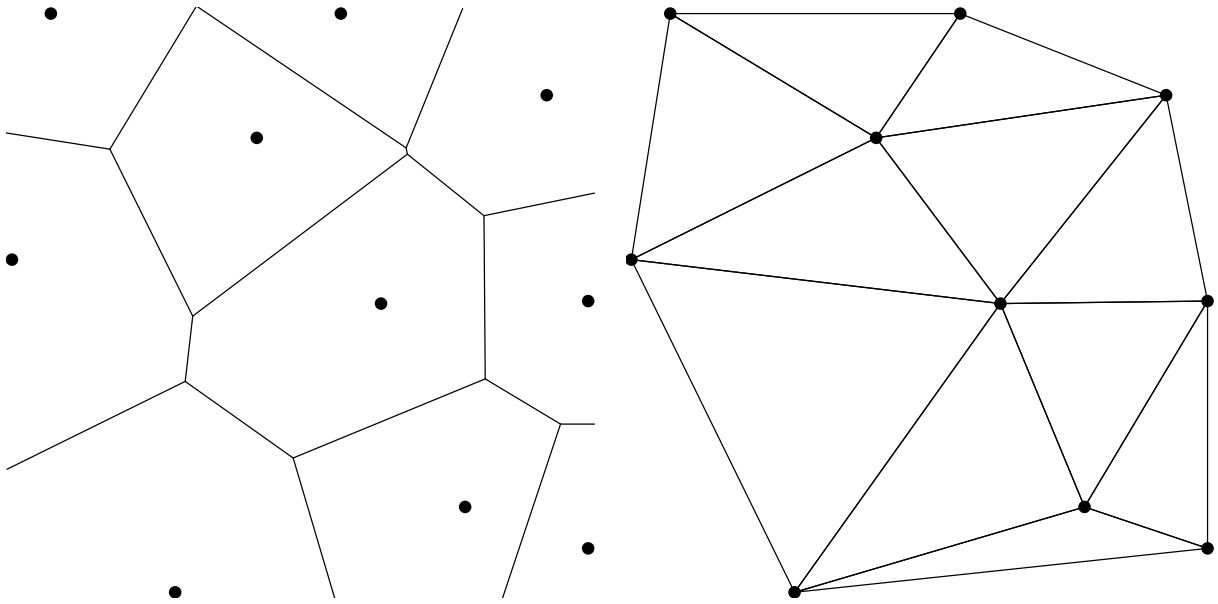
Figure 2.5 shows an example of these operators using the 8-neighborhood mask. These operators can also be viewed as a form of applying the neighborhood mask to do convolution

over a Boolean algebra. I will apply morphological operations to smooth results in Chapters 4, 5, and 6.

2.2 Geometric data structures and algorithms

I introduce some geometric data structures and algorithms used extensively in this research.

2.2.1 Voronoi diagrams and Delaunay triangulations



(a) (b)

Figure 2.6: An example of a Voronoi diagram (a) and its corresponding Delaunay triangulation (b).

The Voronoi diagram comes from the field of computational geometry, in which it is used for nearest neighbor queries, among other uses. Given a set S of points in the plane, the Voronoi region of a point P in S is defined as the set of points in the plane closer to P than to any other points in S. These Voronoi regions tessellate the whole plane and we call this tessellation a Voronoi diagram.

The dual graph of the Voronoi diagram is the Delaunay triangulation, which is a triangulation of S such that no point in S is inside of the circumcircle of any triangles. Figure 2.6 shows a Voronoi diagram and its corresponding Delaunay triangulation. Voronoi diagrams and Delaunay triangulations can be computed by computer software such as CGAL [14] and Triangle [120]. Voronoi diagram provides the basis of some optimization techniques, such as Lloyd's method (Section 2.3.1).

2.2.2 Graph structure

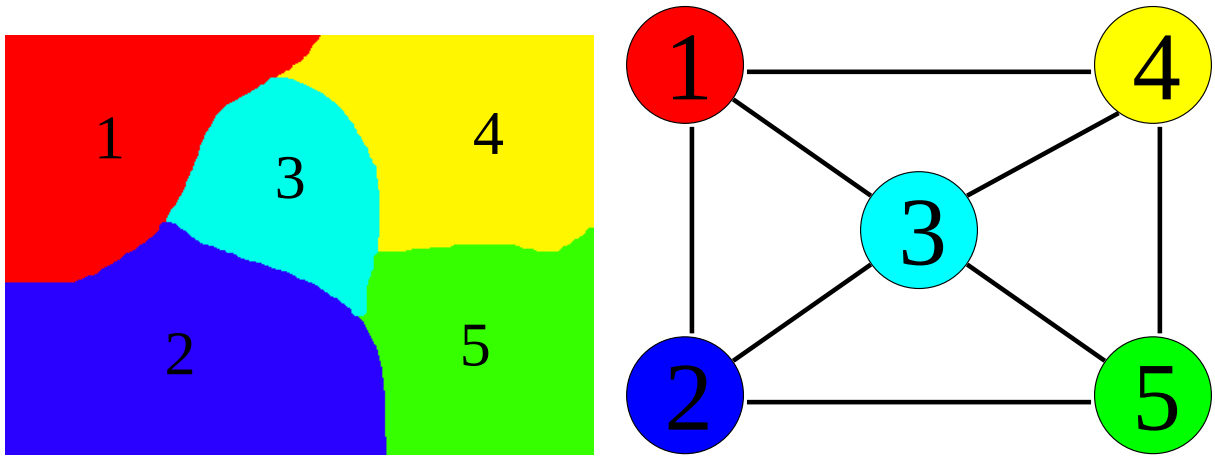
To efficiently explore the geometric and topological relationships of objects, I encode an image using a graph data structure. My representation is similar to the *region adjacency graph* used occasionally in image segmentation and labeling algorithms [50].

Assume that the source image can be subdivided into many regions. The regions can be a single pixel or a segment composed of many pixels. We can think of the image as a graph with a vertex for each region. Two vertices are connected by an edge when there are two pixels, one from each of the corresponding regions, that are adjacent horizontally or vertically (each is in the other's 4-neighborhood). Figure 2.7 gives a simple demonstration. The artistic thresholding algorithm presented in Chapter 4 will work on a graph structure. The colour and geometric properties of the regions (*e.g.*, area and boundary) can be saved in vertices and edges of this graph. I will discuss more details about the usage of the graph structure in Chapter 4.

2.3 Optimization methods

Optimization methods are applied in choosing the best solution to problems that have many possible answers, or for which it is difficult or impossible to derive a closed-form solution. Given a space S of configurations for a system and an objective function f, optimization searches for a configuration x in S for which $f(x)$ is as small as possible.

A lot of research problems posed in computer graphics are optimization problems [133]. I present several optimization methods applied in my research.



(a) (b)

Figure 2.7: An example of the region adjacency graph for an image. Every region in (a) has a corresponding vertex in (b) with the same colour and label.

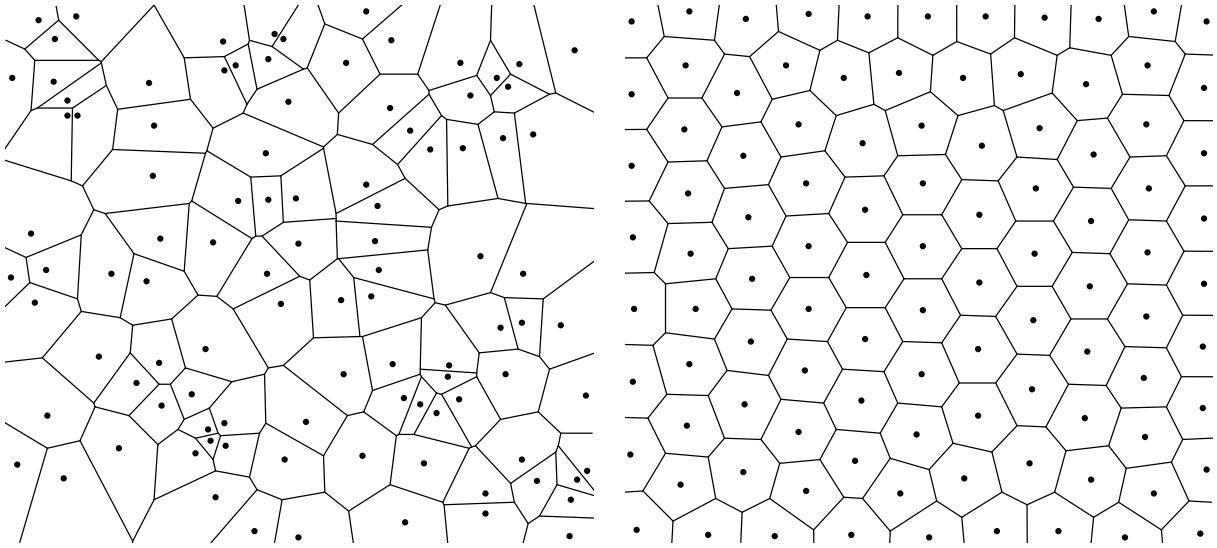
2.3.1 Lloyd's method

Lloyd's method is a relaxation method based on the Voronoi diagram [75]. It is used to distribute a set of points P evenly in a region of the plane. The basic algorithm is as follows:

Algorithm 1 Lloyd's method

Let P be an initial distribution of points.
repeat
  Compute the Voronoi diagram of P.
  Compute the center of gravity for each Voronoi region.
  Move each point to the centroid of its region.
until The sum of point movements is below a given threshold.

Figure 2.8 shows the effect of this method. A typical application of Lloyd's method is stippling [27, 115, 47]. I will apply Lloyd's method in the calligraphic packing technique of Chapter 6.



(a) (b)

Figure 2.8: Lloyd's method. Starting from the random distribution (a), we can get an even distribution (b).

2.3.2 Simulated annealing

The simulated annealing method is inspired from annealing in metallurgy, a technique involving heating and controlled cooling of a material to increase the size of its crystals and reduce their defects. In an annealing process a melt, initially at high temperature and disordered, is slowly cooled so that the configuration of atoms converges to the lowest energy state.

By analogy, simulated annealing is a generic probabilistic algorithm for global optimization. It locates a good approximation to the global optimum of a given objective function in a large search space. It is better than local search methods such as hill climbing, because simulated annealing allows you to occasionally move to a less optimal solution, which can help avoid getting caught in local minimum. It is a clever combination of local search and random walk. Each step of this algorithm replaces the current solution by a random "nearby" solution, chosen with a probability that depends on the difference between the corresponding function values and on a global "temperature" parameter T, which is gradually decreased during the process. As the temperature gradually decreases the algorithm becomes more deterministic. How to set the

initial value of T and decrease it in each step is related to the problem domain. The user has to tune them to achieve good performance. The algorithm is shown in Algorithm 2. Simulated annealing has been widely used in computer graphics [45, 1]. I will apply it in the artistic thresholding algorithm.

Algorithm 2 Simulated annealing method

Get an initial state s.
Initialize temperature T to a high value.
while T is higher than a temperature threshold T_0 and the energy $E(s)$ is greater than an energy threshold E_0 **do**
  Randomly pick a neighbor yielding a state s'.
  if $E(s') < E(s)$ **then**
    Replace s by s'.
  else
    Replace s by s' with probability $e^{\frac{E(s)-E(s')}{T}}$.
  end if
  Decrease the temperature T.
end while
return the best solution found.

Chapter 3

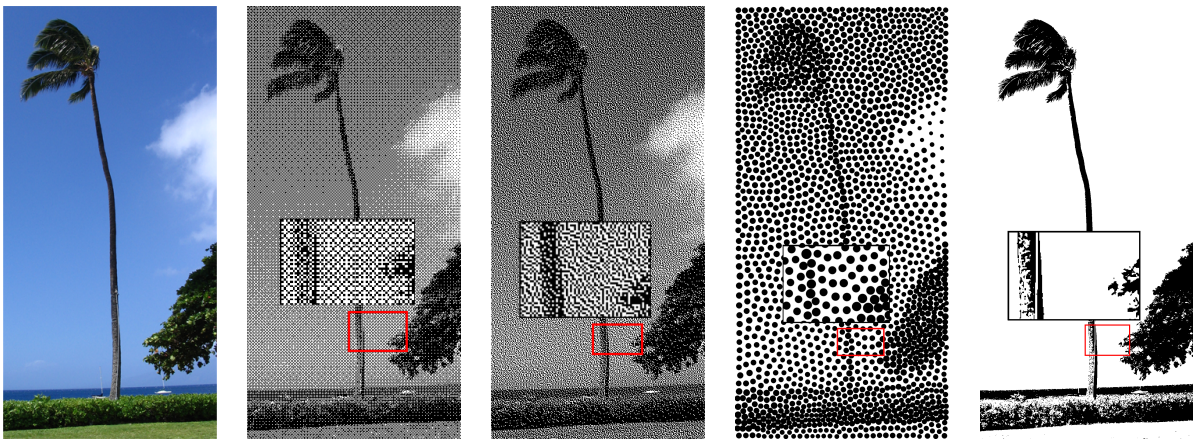
State of the Art in Black-and-White Image Synthesis

In this chapter, I survey the state of the art in black-and-white image synthesis. Based on the dimensionality of representation presented in Section 1.1, I classify related work into three types: point-based techniques, line-based techniques and plane-based techniques. Note that these terms are not intended to correspond to their geometric definitions, but to their common use in art. Thus a "point" is any small localized primitive, "lines" can include curves, and "planes" can be regions of any shape.

3.1 Point-based techniques

First, I will review the basic techniques that make use of points as elements. Kandinsky said, "the point is temporally the briefest form" [59]. Although in theory a point has no dimension or size, I regard any shapes as points if they are small compared to the image size. For instance, small rectangles and circles are considered points in our discussion. Every pictorial form can be viewed as a composition of points. This fact is verified by many techniques that are still applied today. For example, the basic element of modern raster monitors is the pixel which is a tiny point. Most image file formats are also based on assigning specific colours to a grid of points.

Halftoning is one the most popular means of converting an image into black-and-white while preserving the image tone. Among different halftoning approaches, dithering is frequently



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

Figure 3.1: Given a source image (a), the result of ordered dithering is shown in (b). Using error diffusion, a better result (c) can be acquired. The result of Weighted Voronoi Stippling is in (d) and the result of applying a Difference-of-Gaussians method is shown in (e).

used in the printing and publishing industries to reproduce image tone using only black and white [131]. It simulates continuous gray levels by varying the configuration of black dots arranged in a regular pattern (Figure 3.1(b)). However, the regularity of patterns leads to easily noticeable artifacts that detract from the quality of the rendered image. An enhanced technique is error diffusion (Figure 3.1(c)). It traverses each pixel and computes the closest intensity available. The error is the difference between input value and the nearest available intensity. The next step is to divide up the error and distribute it to nearby neighbors. When we reach to these later pixels, the distributed error is combined into image values for processing. Error diffusion is a good halftoning method. Many artifacts are suppressed, producing images with little grain.

Other more complicated dot distribution methods have been developed. Ostromoukhov demonstrated how to generate a random, smooth dot distribution by relaxing a random point set with a mass-spring model [93]. In other work, he used a Hilbert space-filling curve or a random space-filling curve to distribute the dithering dots [96]. Velho and Gomes introduced a similar digital halftoning technique that also applied space filling curves such as the Hilbert, Peano, and Sierpinski curves to generate non-repeating dithering dots [134].

Several point distribution techniques are based on computation of Voronoi diagrams. Deussen

et al. [27] applied Lloyd's method, which is an iteration between constructing Voronoi tessellations and centroids (see Section 2.3.1). Secord extended this work by introducing a weighted centroidal Voronoi diagram method [115]. Figure 3.1(d) shows an example using this technique with 2000 dots. This technique can present images with a much smaller number of dots by sacrificing resolution.

Researchers also employed results from human visual perception to seek better solutions. For example, Gooch *et al.* used a perceptually motivated technique to convert photographs of human faces into black-and-white illustrations [38]. Their method is based on a Difference-of-Gaussians technique. Two different Gaussian kernels operate at varied scales. Then the differences of the Gaussian operations are summed up and thresholded to a two-tone image. The DoG works as a band-pass filter. One effect of this technique is that areas with a constant gradient are removed. Figure 3.1(e) shows an example.

Of course, the stippling dots do not need to be small circles exactly. Ostromoukhov and Hersch [95] proposed a technique which they called "artistic screening" to generate screen dots of any shape. ASCII art is an extension technique that down-samples the image, computes the intensity value for each cell, then assigns a corresponding character to it.

Another related art style is mosaics [72]. Several attempts have been made to simulate the appearance of ancient mosaics [43, 31]. Of course, the small mosaic tiles can be extended to any shape, so this problem actually is a packing problem. It has been studied by many computer graphics researchers [65, 47]. Dalal *et al.* defined a new metric to solve the "NPR packing" problem that aims at evenly distributing tiles with the constraints of minimizing both grout and overlaps [22]. But NPR packing does not really attempt to approximate tone. Schlechtweg *et al.* presented the "RenderBots" technique to unify point-based and stroke-based rendering [114]. Their framework can handle stippling, mosaics, and hatching.

3.2 Line-based techniques

Line drawing is widely used by artists. It is a representation method of using straight and curved strokes whose width can be constant or varied to depict objects. This technique emphasizes form and outline. Many traditional media can be used to create line drawings, for example woodcut, engraving, and pen-and-ink.

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

Figure 3.2: Different feature lines: given a 3D model (a), we can get silhouettes (b), contours (c), ridge and valley lines (d) and contours with suggestive contours (e).

Lines can convey shape information efficiently. A few lines often suffice to offer us strong shape cues. NPR researchers have studied the mathematical and perceptual properties of various lines (Figure 3.2) [18]. These feature lines can be computed from 3D models [109]. Silhouettes are boundaries between object and background. Contours emphasize depth discontinuities. Ridge and valley lines are local maxima and minima of curvature. Hatching lines are repeated patterns of lines. They are useful to convey shape. DeCarlo *et al.* presented a new type of feature lines: suggestive contours, which they defined as lines that become contours in nearby views [23].

Compared with other NPR sub-fields, researchers have been exploring this area for a long time and many techniques and systems have been developed. These techniques can be categorized as image-space methods or object-space methods. For an image-space method, the input is a 2D image, while for an object-space method, it is a 3D model.

First, I review image-space methods. A direct approach is applying edge detection [11] to get lines. But the result includes too much noise and is hard to control, as I discussed in Section 1.2.1. Pearson and Robinson designed feature-extraction operators which make use of the second derivative of image [98]. Their operators are helpful to suppress noise. Instead of using general 2D images, Saito and Takahashi [110] rendered geometric properties of a 3D scene, such as depth and normal vector, into raster images they called G-buffers. They developed line drawing algorithms for discontinuities, edges, contour lines, and curved hatching based on G-buffers. Deussen extended the basic G-buffer approach [125, Chapter 6]. He used a set of clipping planes to cut the 3D scene and extracted the intersecting lines from G-buffers. These intersecting lines are simulated hatching lines. His method can be applied to generate copper

plates. Decaudin showed how to combine depth and normal maps to render outlines [25]. Lee *et al.* described a new algorithm to extract silhouettes, creases, ridges and suggestive contours from a shaded image [68].

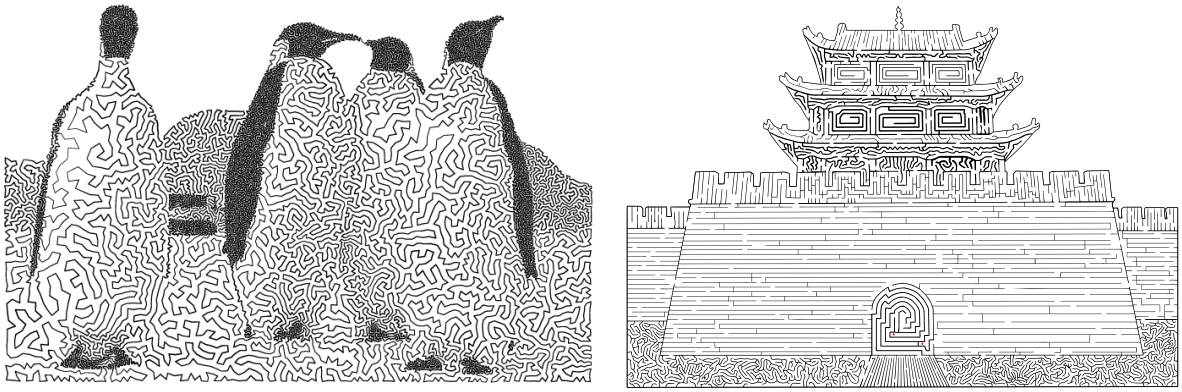
Line-based primitives can be applied in halftoning. Pnueli and Bruckstein developed a gridless halftoning technique [101]. They used the image intensity to control the evolution of a planar curve.

Many researchers generate line drawings from object-space, because it provides more helpful shape information so that we can extract meaningful lines more precisely. In Hertzmann's survey paper [44], he also summarized methods for finding silhouettes of 3D models. Ohtake *et al.* detected ridge and valley lines by estimating curvature of the mesh model [90]. As a preprocessing step, they first fitted an implicit surface to the mesh. Dooley and Cohen described an automatic illustration system that can generate line illustrations from 3D objects [28]. The lines include boundary lines, silhouette lines, discontinuity lines and isoparametric lines. Elber applied a set of isoparametric curves to cover a 3D surface [30]. The density of curves is controlled by an illumination model of this surface. In another paper, he extended this technique to process freeform surfaces based on point coverage [29]. Ostromoukhov suggested simulating traditional facial engraving by composing layers of transformed screened engraving patterns [94]. Goodwin *et al.* presented an approach to generate line drawing whose stroke thickness is computed from a shaded surface [39].

Hatching is a popular topic of research because it is an efficient way to illustrate a surface to denote its smoothness, material properties, and shading. Hertzmann and Zorin [46] inferred a direction field from principal curvature and smoothed it by optimization. Finally, they placed hatching lines following this direction field. Praun *et al.* applied a set of pre-defined "tonal art maps" to achieve real-time hatching [103]. They also maintained spatial and temporal coherence. In a later paper [138], they extended their hatching technique with hardware acceleration.

A closely related art form is pen-and-ink illustration. Many researchers have studied this topic [112, 142, 111]. The research results of hatching have direct applications in pen-and-ink illustration. Salisbury *et al.* [113] introduced "orientable textures" to convey 3D information in an image-based system. They provided a set of interactive tools that allow the user to edit direction information. It is helpful in creating compelling pen-and-ink illustrations.

Researchers also explored other interesting variants of line drawing. For instance, with the



(a) (b)

Figure 3.3: An example of TSP Art in (a) and a result of our image-guided maze construction algorithm in (b).

purpose of generating drawings made up of a single long path, Kaplan and Bosch [61] presented a technique that they called "TSP Art". They first used a point placement method such as stippling to create dots from an image, then generated a nearly optimal solution to the Euclidean Traveling Salesman Problem (TSP) to find a short closed path that visits every dot exactly once. An example is shown in Figure 3.3(a). Pedersen and Singh [99] presented a synthesis algorithm that evolves an initially simple shape into an organic labyrinthine drawing. Xu and Kaplan applied maze depiction techniques to represent geometric patterns [147] and general images [146]. Figure 3.3(b) shows an example generated by their system. I will apply some line-based techniques in papercutting (Chapter 5) and representational calligraphy (Chapter 6).

3.3 Plane-based techniques

Artists have great interest in using small shapes to compose a scene. In theory, all point-based and line-based techniques can be viewed as special cases of plane-based techniques. These special planes are small or thin. Many stippling and packing techniques [47, 65, 22, 114] can be applied with larger shapes.

Plane-based representation is preferred in wholetoning. Recently, Mould and Grant [88]

(a) (b)

Figure 3.4: Given a source image (a), the mean shift segmented result is shown in (b).

presented an algorithm based on energy minimization for abstracting input images into black-and-white images. Their goal is to preserve details while as much as possible producing large regions of solid colour. Contemporaneously, Vergne *et al.* [135] defined a new shape descriptor based on convexity and curvature information of 3D objects. It could be used to depict shape through different shading styles. They demonstrated a minimal shading result that is similar to wholetoning. Bronson *et al.* [9] described a method for creating stencils from 3D models and images through error minimization. Stencils have the same mathematical constraints as paper-cut designs (Chapter 5).

Another related technique is image segmentation. Black-and-white images are an extreme case of using only two colours to segment an image. There are many segmentation methods such as clustering, edge detection, physics based methods, and region growing [121]. One good method is mean shift (Section 2.1.2). It computes a mean value for search windows. If windows converge to the same extrema, they are merged (Figure 3.4(b)). Image segmentation has wide applications in NPR. Recently, Qu *et al.* [105] presented a technique for screening manga backgrounds from photographs. Their method is based on image segmentation and applies texture matching and tone matching to find an optimal assignment of screen types. Their method aims at the richness of screens and tone reproduction, so it can be viewed as an improved halftoning technique.

Little work in NPR is concerned with creating abstracted imagery via large regions of constant colour. DeCarlo and Santella used a hierarchical segmentation of an image to permit abstraction with spatially-varying level of detail [24]. In their work, level of detail was guided by eye fixations, leading to a very natural distribution of detail around salient image features and high abstraction elsewhere. Similar results were achieved by Orzan *et al.* [92]. They used edge information to guide the smoothing of features into large regions of constant colour. Wen *et al.* [140] allowed the user to edit a segmentation interactively, and abstracted segment shapes into attractive coloured forms. Song *et al.* presented a technique of "arty shapes" to simplify image segments [122]. A closely related art form is stained glass. Mould used morphological operators to smooth a segmentation and create stained glass tiles [87]. Brooks [10] presented an image-based approach to stained glass generation based on image segmentation. Each segment is covered by textures from real stained glass. The regions in a stained glass image are not exactly constant colour, but the principle is the same. All of these techniques focus on detail abstraction and preserve colours from the source image.

Chapter 4

Artistic Thresholding

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

Figure 4.1: Examples of a photograph (a) to which simple thresholding (b), adaptive thresholding (c) and max-flow graph cut optimization (d) have been applied.

As discussed in Chapter 1, the simplest way to convert any colour image to a wholetoned image is undoubtedly *thresholding*. Thresholding does not produce convincing abstractions of images (Figure 4.1). Even if we apply more powerful techniques such as adaptive thresholding [37] (Figure 4.1(c)) or a max-flow graph cut algorithm [8] to cluster pixels into two sets based on their intensity values (Figure 4.1(d)), the result suffers from many of the same problems and it is hard to control to generate different styles. One notable deficiency in those methods is that they cannot account for many of the wholetoning characteristics discussed in Section 1.2

such as "tone inversion", where an artist may choose to colour dark objects white or bright objects black to make them stand out from the background. These tone reversals are important for capturing all the details in a scene, and appear natural to the viewer.

In this chapter I present an optimization-based artistic thresholding algorithm to generate wholetoned images. My algorithm automatically segments a source image and constructs a graph data structure based on that segmentation (Section 4.1). I establish an energy function that measures the quality of different black-and-white colourings of the segments (Section 4.2). My system adopts a simulated annealing framework to minimize an energy function based on a weighted sum of competing aesthetic goals, such as tone preservation, relative amounts of black and white, and the depiction of edges and high-level features (Section 4.3). The user can interactively adjust the energy function's weights in real time and immediately observe the effect on the optimization process (Section 4.5). The final result is vectorized as output.

4.1 The region adjacency graph

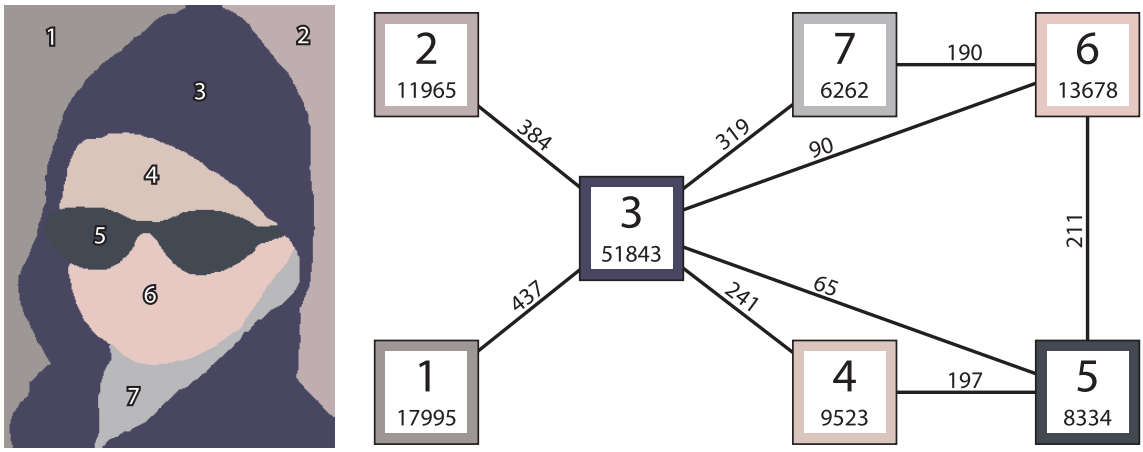
The core of my artistic thresholding framework is a graph data structure (Section 2.2.2) that encodes the geometric and topological properties of small-scale features in an image.

Given a source image, I use EDISON's synergistic image segmenter (Section 2.1.2) to subdivide it into regions. The small segments that are produced (related to the notion of "superpixels" in computer vision [107]) support a wide range of abstracted results, while eliminating the distracting details associated with operations on individual pixels. Image segmentation can group pixels homogeneously and implement shape and texture abstraction in a naive way.

Let us assume that the source image has dimensions $W \times H$ (in pixels). Segmentation yields a set of N regions that I will denote using the indices $1, \ldots, N$. Each region is a (not necessarily connected) subset S_i of integer locations (x, y) in the image. We can think of the segmentation as a graph with a vertex for each region. Two vertices are connected by an edge when there are two pixels, one from each of the corresponding regions, that are adjacent horizontally or vertically.

I augment this purely topological description of the segments with information that will allow us to develop artistic thresholding algorithms. With each region S_i I associate c_i, the average colour of the pixels in S_i, and the area A_i, the number of pixels in S_i. For every pair of distinct indices i and j, let $l_{i,j}$ denote the length of the shared boundary between S_i and S_j. This length is

measured in terms of the number of adjacencies as described in Section 2.2.2; if S_i and S_j are not connected by a graph edge, $l_{i,j}$ is zero (and hence an explicit graph data structure is not required). Figure 4.2 illustrates a simple segmentation, together with its region adjacency graph.



(a) (b)

Figure 4.2: A visualization of the region adjacency graph for a simple segmentation of the image in Figure 4.1(a). Every segment in (a) has a corresponding vertex in (b) with the same colour and label. Each vertex records the number of pixels in its segment. Each edge records the length of the shared boundary between its neighbouring segments.

This simple representation of a segmentation enables a wide variety of non-photorealistic rendering algorithms. Here I envision artistic thresholding as a technique that assigns a value $b_i \in \{\text{black}, \text{white}\}$ to every region. I freely abuse this definition, thinking of b_i sometimes as a colour defined in the same colour space as c_i, sometimes as a boolean value (where black is true and white is false), and sometimes as real-valued a grey level (where black is 0 and white is 1). Given an assignment $\{b_1, \ldots, b_N\}$, I also define $B = \{i \in 1, \ldots N | b_i = \text{black}\}$, the current set of black regions.

I would also like to allow a designer to divide the image more coarsely into high-level features, and have those features interact with the low-level segmentation. If explicit features are provided, I simply divide any segmentation regions that cross feature boundaries. I will use these features in Section 4.2.3. In principle, these two levels could be extended into a full segmentation

hierarchy, such as the one constructed by DeCarlo and Santella [24]; I found two levels adequate for my purposes.

4.2 Evaluating the quality of an assignment

A natural first approach to artistic thresholding is an energy-based optimization framework. There are 2^N possible assignments to the b_i; I search over that space of assignments for one that minimizes an objective function. In this section, I build the objective function, based on an attempt to address the challenges outlined in Chapter 1. I believe that this quality depends on several competing forces based on the following design rules that should be considered to generate a wholetoned image:

- Due to the spatial and visual coherence in an image, adjacent things should have the same colour. Image segmentation helps achieve this goal, because it respects spatial and colour similarities.
- Because of "tone preservation", dark regions should be black while bright regions should be white.
- Edges in the source image should be depicted in the wholetoned illustration. This goal may be satisfied by colouring two adjacent regions with opposite colours.
- I would like to control relative amounts of black and white. Different ratios of black to white will produce diverse styles.
- It is desirable to distinguish high-level features in some cases. To highlight a high-level feature, all segments belonging to this feature should be coloured homogeneously.

The tradeoff in these forces is manifested by an objective function that is a weighted sum of individual measurements. The designer can adjust the weights to bias the search.

4.2.1 Colour matching

One term in the objective function must measure how well the binary assignment to each region approximates the original colour of that region.

(a) (b)

Figure 4.3: An example demonstrating the use of C_{col} in isolation. The photograph from Figure 4.1 is shown segmented in (a). The optimized result is then shown in (b).

I define C_{col}, which evaluates the overall difference between the binary assignment and the source image pixels:

$$C_{\text{col}} = \left(\sum_i A_i d(c_i, b_i) \right) / (WH) \ .$$

Here, $d(c_1, c_2)$ is a function that computes the difference between two colours, producing a result in the range $[0,1]$. I divide by the total image area to normalize the cost to the range $[0,1]$ (I will seek to normalize all cost functions in a similar way). When C_{col} is used in isolation, the optimal assignment can be found easily by setting b_i to be the thresholded luminance of c_i. For reference, Figure 4.3 shows an image produced by minimizing C_{col}. The result is already more

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

Figure 4.4: Demonstrations of the area and boundary costs. The image in (a) was optimized to be 80% black. Image (b) demonstrates the use of C_{alike} in isolation. In (c), I combine C_{alike} and C_{opp} with comparable values to achieve a balanced composition. In (d) I combine all three costs, aiming for 10% black.

attractive than the pixel-based thresholding of Figure 4.1: pixels are collected into segments, resulting in a less noisy image.

4.2.2 Area matching

In some cases a designer might wish to control the overall proportion of black used in the final image. Given a user-supplied target value $T_{\text{area}} \in [0,1]$, I define

$$C_{\text{area}} = \left| \left(\sum_{i \in B} A_i \right) / (WH) - T_{\text{area}} \right| .$$

Figures 4.4(a) and (d) demonstrate the effect of optimizing C_{area}.

4.2.3 Boundary contrast

The most interesting costs associated with an assignment measure the impact of contrast (or lack of it) between adjacent segments on the quality of the final assignment. Boundaries between

segments tend to contain the image's edges as a subset. I would like to preserve the visibility of those edges by ensuring that adjacent segments with contrasting colours are assigned opposite binary values. Conversely, similarly-coloured segments should be assigned identical binary values.

I divide the set E of edges of the segment graph into two groups E_{alike} and E_{opp}. An unordered pair (i, j) is in E_{alike} if $b_i = b_j$ (i.e., the segments are both black or both white); otherwise the edge is in E_{opp}. I can now define

$$C_{\text{alike}} = \left(\sum_{(i,j) \in E_{\text{alike}}} l_{i,j} d(c_i, c_j)^{1/5} \right) / \left(\sum_{(i,j) \in E} l_{i,j} \right) \text{, and}$$

$$C_{\text{opp}} = \left(\sum_{(i,j) \in E_{\text{opp}}} l_{i,j} (1 - d(c_i, c_j)^{1/5}) \right) / \left(\sum_{(i,j) \in E} l_{i,j} \right) .$$

The cost C_{alike} measures how effectively the assignment uses similar binary values for similarly-coloured adjacent segments. On its own, this cost is theoretically minimized by letting E_{alike} be empty, corresponding to a 2-colouring of the region adjacency graph. In practice, minimization of C_{alike} produces busy assignments that approximate 2-colourings, as shown in Figure 4.4(b). Conversely, minimizing C_{opp} comes from placing every edge in E_{alike}, which can be achieved by assigning every segment the same binary value. When these two costs are given comparable weights, they cooperate to achieve a balanced use of contrast. Most interestingly, optimizing these costs can lead to a segment being assigned a binary value that contradicts its luminance, if that assignment improves the overall depiction of shapes via edges.

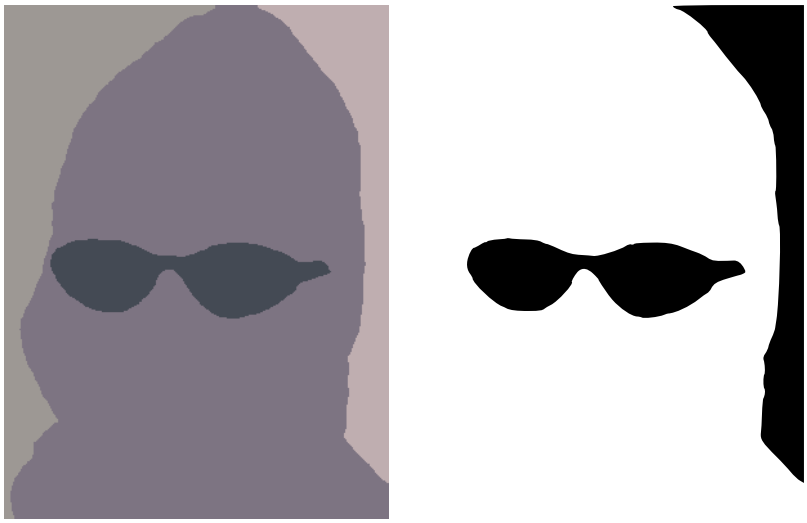
Most of the time, we expect that adjacent segments will have similar colours because of spatial coherence, which suppresses the effectiveness of these measurements. To improve the robustness of this cost function, some robust estimators can be applied [52]. Empirically, I found that taking the colour differences to the power of $1/5$ magnifies small differences and evens out the cost functions. Note also that contrasts are multipled by boundary lengths, and normalized by the total length of boundaries in the image.

When high-level features are provided by the designer, I modify C_{alike} to further penalize feature edges that are not depicted via segment contrast:

$$C_{\text{alike}} = \left(\sum_{(i,j)\in E_{\text{alike}}} l_{i,j} d(c_i, c_j)^{1/5} p_{i,j} \right) / \left(\sum_{(i,j)\in E} l_{i,j} p_{i,j} \right) .$$

Here, $p_{i,j} = 1$ when segments i and j belong to the same high-level feature, and $p_{i,j} = 100$ when they do not.

4.2.4 Feature homogeneity



(a) (b)

Figure 4.5: An example of minimizing the group homogeneity measure C_{group} in (b), based on the user-supplied features shown in (a).

When the image is divided explicitly into high-level features, we may sometimes wish to assign binary values as homogeneously as possible within each of those features. Assume that there are M features. For a given assignment, let u_k and v_k denote the number of black and white segments within feature k. Then I let

$$C_{\text{group}} = \left[\sum_{k=1}^{M} \left(1 - \frac{|u_k - v_k|}{u_k + v_k} \right) \right] / M .$$

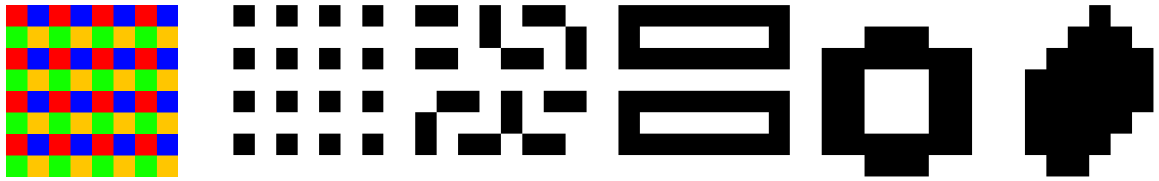


Figure 4.6: Demonstrations of the effects of neighboring cost C_{neigh}. Given the source image, I prefer 0, 1, 2, 3, 4 black neighbours respectively.

This cost is minimized by assigning all segments within the same high-level feature identical binary values. On its own, C_{group} can produce attractive images, but deceptively so: the quality arises almost entirely from the salience of the user-provided features (see Figure 4.5). This cost can be used profitably in conjunction with the others, as a way to produce a "calmer" assignment.

4.2.5 Other costs

I experimented with three other cost measurements for assignment quality. The first, C_{num}, simply counts the number of black segments. It can be combined with other costs to account for abstraction. The second, C_{comp}, counts the number of connected components of black segments. I may seek to minimize this value in some applications. If I can reduce the assignment to a single black connected component, the resulting picture would be a connected set and could, for example, function as a papercutting. Finally, I let the user define a number N_{target} that records a desired number of black neighbours for each black segment. I can then include a cost C_{neigh} that measures how far every black segment is from having exactly that many neighbours (see Figure 4.6). For example, setting N_{target} to 2 might tend to produce space-filling paths through the segments, which could ultimately be treated like mazes. These costs are disabled by default but can be enabled for specific applications.

4.2.6 Total cost

I must define a single energy function that can be minimized via optimization. I adopted an approach that has been seen before in NPR [45, 65], computing the overall energy as a weighted

sum of individual terms:

$$C_{\text{total}} = \frac{w_{\text{col}}C_{\text{col}} + w_{\text{alike}}C_{\text{alike}} + w_{\text{opp}}C_{\text{opp}} + w_{\text{group}}C_{\text{group}}}{w_{\text{col}} + w_{\text{alike}} + w_{\text{opp}} + w_{\text{group}}} .$$

The weights are non-negative real numbers, not all zero. As with the individual costs, I divide by the sum of the weights to normalize the quality measurement. This normalization becomes especially important in the next section, where C_{total} is used as the objective function in a simulated annealing optimization. If the overall scale of the weights were allowed to drift, the resulting cost function would behave differently with respect to the optimization's cooling schedule.

4.3 Optimization

The goal of artistic thresholding is to find an assignment $\{b_1, \ldots, b_N\}$ that minimizes the total cost C_{total} defined in the previous section. Except for simple cases, it is too expensive to search over all 2^N assignments for the optimum. Instead, I use an optimization framework based on simulated annealing [104, Chapter 10]. Simulated annealing is a robust, general purpose optimization algorithm. It is particularly useful when the geometry of the configuration space is difficult to characterize. Simulated annealing is one example of a Markov chain Monte Carlo method. Barbu and Zhu applied the similar Swendsen-Wang method to solve graph partition problems [5].

I initialize the b_i randomly to black or white, and keep track of the current best assignment. The optimizer repeatedly constructs perturbations of this assignment and tests their costs. If a perturbed assignment has a lower cost, it is accepted unconditionally as the new best answer; if the cost is higher, it is accepted with a small probability, temporarily degrading the solution in the hopes of avoiding local minima in the configuration space. I use a cooling schedule to decrease the acceptance probability at an exponential rate. Optimization continues until a specified number of iterations have passed with no changes to any of the b_i, at which point I report the best assignment found. A similar approach was used by Agrawala and Stolte [1] to render route maps.

The question remains of how to construct perturbations of the current assignment. A natural approach is to flip a random b_i and check whether that improves the overall assignment. However, in practice this approach is too local. There may be situations where overall cost can be decreased

by changing the values of several nearby segments in tandem, even though the cost goes up if any one of them is flipped in isolation.

I mitigate this problem by operating on subgraphs instead of individual segments. Given a vertex in the region adjacency graph, I generate a random connected subgraph containing that vertex. The number of vertices in the subgraph can be controlled by the user; I have found that three to five vertices provide a good balance between performance and quality. I construct new assignments for all possible combinations of binary values within this subgraph, and emit the lowest-energy combination as the chosen perturbation.

Note that I do not need to recompute C_{total} from scratch every time some b_i changes. Most of the energy terms depend only on the relationship between a segment and its immediate neighbours. When one segment changes colour, I can compute the effect of this change on those cost terms and add the difference to the current cost. I further exploit this fact when testing all combinations of binary assignments within a subgraph. I iterate over the combinations in an order given by a Gray Code [139], so that the combinations can be tested by flipping a single b_i at a time.

4.4 Postprocessing

Once the optimization is complete, I can simply render every pixel in S_i using colour b_i. I have also experimented with several postprocessing operations that can improve the overall quality of my results.

The designer may optionally apply the standard OPEN and CLOSE morphological operators described in Section 2.1.4, adapted here from the pixel grid to the region adjacency graph. These operators can help to eliminate small, isolated pockets of contrasting colour, smoothing out noise and increasing the level of abstraction in the thresholded image. For each segment, I control which of its neighbours participate in the computation of OPEN and CLOSE. The regions cannot be processed uniformly in these morphological operations, because we do not like big segments to be affected and we prefer to keep salient edges. Some modifications are incorporated. A user-controllable area threshold prevents small adjacent segments from having an influence. An edge threshold excludes adjacent segments that are too different in colour, helping to preserve image edges.

(a) (b)

Figure 4.7: Demonstrations of the postprocessing operations discussed in Section 4.4. I smooth out isolated pockets of black and white in (a) using graph-based morphological operators. In (b), I superimpose edges between high-contrast segments that were assigned the same binary value.

While I wish to draw a clear distinction between artistic thresholding and previous work on line drawing, a few well-chosen lines can enhance some results. I search over all pairs of adjacent segments. When two segments are given the same binary assignment but have a colour difference beyond a user-selected threshold, I draw the boundary between them as an edge. These edges can help to reinforce object boundaries in the source image that were missed by the optimizer. In practice, I found that many images are comprehensible without these edges.

Figure 4.7 demonstrates the application of the morphological operators and edges.

Finally, I vectorize the binary image to produce a high-quality scalable result. I have obtained good results using the Potrace library [117]. All results presented here have been vectorized in

this way.

4.5 Implementation

I have constructed an interactive software implementation of artistic thresholding. The interface is designed to support a continuous, dynamic interaction between the designer and the optimization process. A more complete description of the implementation, together with a screenshot, is given in Appendix A. The current best binary assignment is displayed during optimization. The designer can modify the weights in the cost function, immediately affecting the algorithm. To allow weight changes to have a significant effect on the assignment, each adjustment causes a small increase in annealing temperature. The designer can also fix binary values for individual segments when desired; such values are held constant during optimization. This design procedure can be viewed as a mixed-initiative interaction [2] in which the designer and the system collaborate to finish a design task. The designer provides intuition, evaluation, and makes high-level decisions, while the system manages details and generates immediate feedback responding to the designer's current settings.

One downside of this interface is the lack of continuity. In the processing of artistic thresholding, any change to the binary assignment will necessarily produce discontinuous feedback. During optimization, these discontinuities become even more pronounced when the annealing temperature is raised following user interaction. The assignment can change chaotically while the optimization settles down. Even at low temperatures, there might exist multiple near-optimal results corresponding to current parameter configuration. These results have nearly the same energy, meaning that the optimization will flip back and forth between them. I look forward to investigating how to provide less chaotic feedback in future work.

In my implementation, I set the initial annealing temperature to 100. Every time the temperature changes, I iterate over the vertices in the region adjacency graph in random order. For each vertex, I construct a random connected subgraph containing that vertex and optimize it as described in Section 4.3. After visiting all vertices, I set the temperature to 99% of its current value. When the current best assignment has survived 200 temperature changes, I stop the optimization process.

4.6 Results and discussion

I first compare my wholetoned results against those produced by artists. Figure 4.8 shows several experiments. In general, my technique can create similar results to artists. The principal features are also reproduced well by my system. But I also notice that artists have deliberate control over the details, so that they can capture features more precisely and direct the viewer's focus. See, for example, the eyes of Che Guevara and the teeth of Marilyn Monroe. Artists can also handle abstraction more effectively, as in the wheels of the skateboard. Figure 4.9 shows the difference between my results and images generated via naive thresholding followed by vectorization. We can notice that there are some significant differences in the results produced by my system: for example, Che Guevara's nose and Marilyn Monroe's lips are both more sharply defined because of tone inversion.

On the other hand, I found my implementation to be successful on a wide variety of source images. Figure 4.10 demonstrates different styles of Che and Marilyn generated by my system. Figures 4.11, 4.12 and 4.13 show more results produced using my artistic thresholding algorithm. A typical result requires only a few minutes of processing (disregarding the segmentation step) and very little user interaction. In most cases, it is not necessary to fix the binary values for any segments manually.

In Figure 4.14, I compare my technique to drawings that could be produced with more traditional image processing methods. I found that a combination of bilateral filtering, blurring and thresholding can produce attractive results, as in Figure 4.14(c). However, this method misses some of the edges and details brought out by artistic thresholding. Also, it is ultimately tied to traditional luminance-based thresholding, and could never produce more abstract results like the rightmost image in Figure 4.11.

The algorithm produces interesting results across a wide range of weights, and the responsive interface invites the designer to manipulate the weights interactively in a dialogue with the optimizer. My implementation is an engaging and effective tool for creative exploration of this artistic style.

As mentioned in Section 4.1, I support manually specified high-level features, which can be provided by tracing salient boundaries in a drawing program. In some cases, I can make use of pre-existing feature data. The images in Figure 4.12 all come from the Berkeley Segmentation Dataset [78], and are accompanied by human feature identification data. In Figure 4.12, high-

Figure 4.8: Evaluation of wholetoned results. The first column lists the source photographs. The second column shows the results by professionals and the last column shows my results.

Figure 4.9: Comparison of artistic thresholding and naive thresholding. The first column shows my results and the second column shows the results by naive thresholding on pixels. The red regions in the last column visualize the differences between them.

Figure 4.10: Variations of Che and Marilyn results generated by my system.

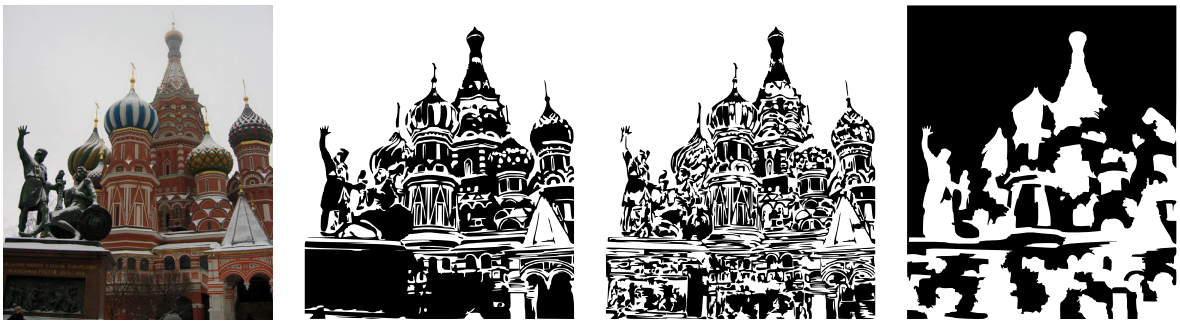


Figure 4.11: A photograph of St. Basil's Cathedral in Moscow, rendered three different ways using artistic thresholding. Photograph used with permission from CNET Networks, Inc., Copyright 2008. All rights reserved.

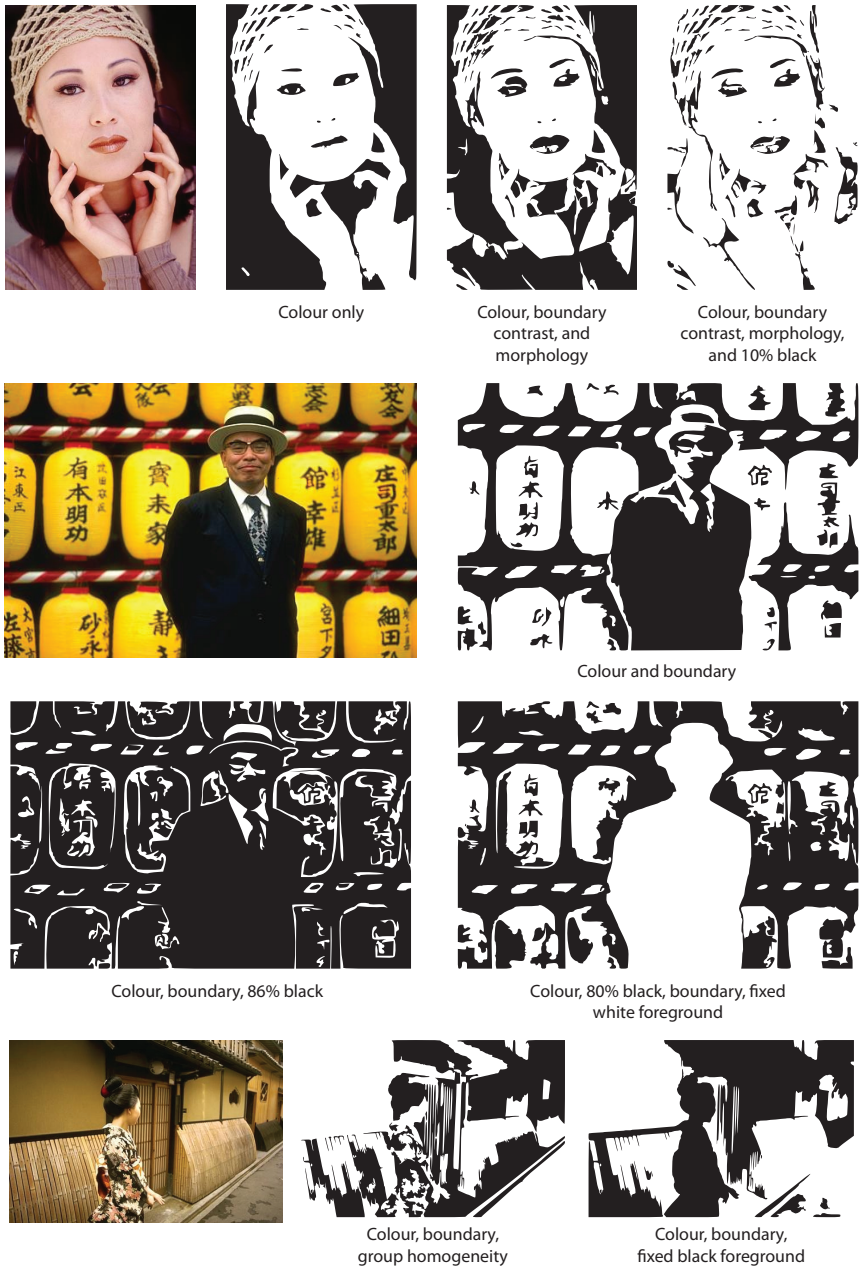


Figure 4.12: Sample results produced using my artistic thresholding algorithm. I summarize the settings used to produce each result by listing the non-zero weights and target area (if applicable).

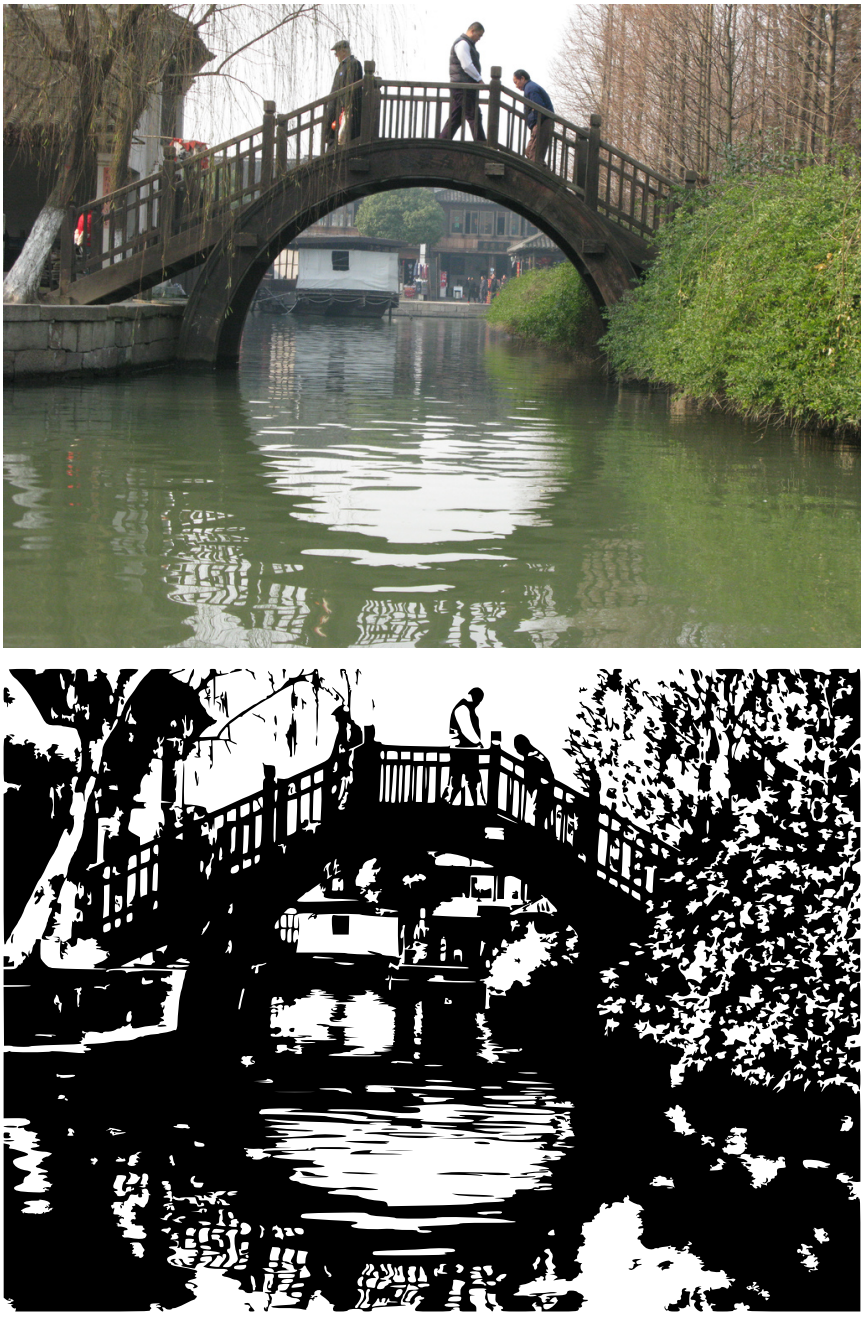
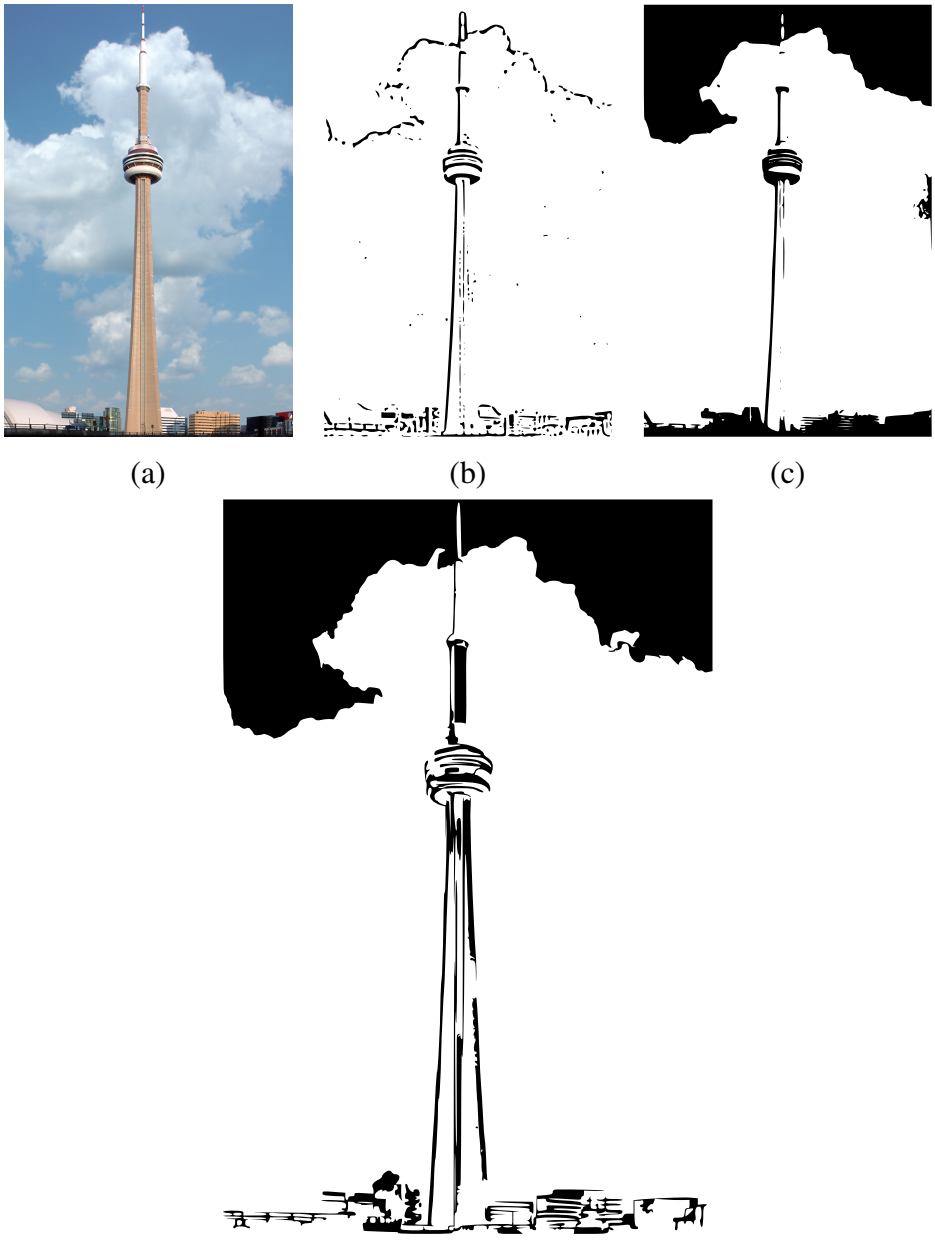


Figure 4.13: A wholetoned drawing based on a photograph of a bridge in Wuzhen. Photograph by Carsten Ullrich.



(a) (b) (c)

Figure 4.14: A wholetoned drawing based on a photograph of the CN Tower, together with related drawings produced via traditional image processing approaches and vectorization. The original photograph is shown in (a). Blurring and adaptive thresholding produces the drawing in (b). In (c), I apply bilateral filtering, blurring and thresholding, leading to a result with some similarities to the illustration produced via artistic thresholding.

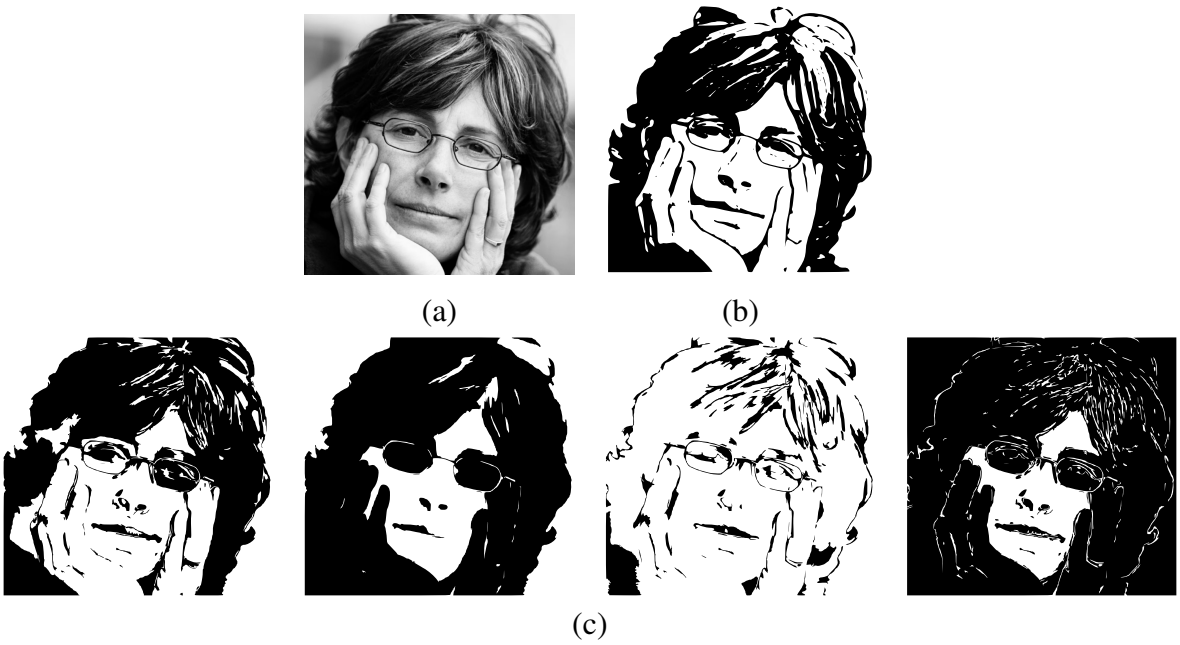


Figure 4.15: Comparison with Mould and Grant's work. A source photograph is shown in (a). Mould and Grant's result is shown in (b). Several different results generated by my system are shown in (c).

level features were used for results that have fixed foregrounds or that made use of C_{group}.

Despite the seemingly unpredictable nature of simulated annealing, I found that my implementation was quite stable: given a source image and its region adjacency graph, re-running the optimization with the same weights produced nearly identical results. Differences were minor and did not affect the visual character of the thresholded images. Quantitatively, the images differed from each other over only a few percent of their pixels.

As with many algorithms in graphics and vision, I begin with a finely segmented image and treat segments as atomic entities. Conceivably my algorithm could be modified to operate directly on the graph induced by image pixels. However, segments provide a basic level of image abstraction and noise reduction that would be difficult to achieve at the pixel level. A graph of individual pixels would probably produce less attractive results in the presence of C_{alike} and C_{opp}. Looked at another way, I suspect that for any sufficiently robust artistic thresholding algorithm, there is a need to cluster pixels together for shape abstraction and feature homogeneity.

Segmentation seems like a good way to do that. Like other researchers [10], I prefer to trust in the high quality of published segmentation algorithms.

The quality of segmentation has great impact on my result. Unlike the closely related work by Mould and Grant [88], which combined a base layer describing the global features with a detail layer describing local small features, my system relies on one single existing image segmentation technique. Their work will create abstract black-and-white images while preserving details, and because the two-layered segmentation is more robust, their results are more appealing in some cases. But their technique is hard to adapt to a wide range of styles and its control parameters are not intuitive. As shown in Figure 4.15, my method is able to generate a wider variety of wholetoned images.

Because the quality of image segmentation has a great impact on the artistic thresholding result, my algorithm might not produce satisfying results for some images that current image segmentation methods cannot process well. This can be verified in Figure 4.16. Image segmentation performs poorly in an image with a narrow range of tones, such as the cat picture in the leftmost column. Other simple thresholding methods, such as blurring plus simple thresholding or bilateral filtering plus thresholding, also cannot produce satisfactory results. Adaptive thresholding and Mould and Grant's method work better in this situation. Another case where image segmentation is unsuccessful is in areas where an image has gradients, as in the shadow on the Sphinx's face in the second column. In these areas, image segmentation cannot create sharp edges, leading to two regions that are merged falsely. This limitation also exists in the adaptive thresholding method. In this situation, we can improve the quality of results with the benefit of user specified high-level features. Of course, more robust segmentation techniques are always desirable in a system such as mine. For images that have relatively even tone distribution, artistic thresholding will work well as shown in the third and fourth columns of Figure 4.16.

4.7 Black-and-white Animation

Given the compelling appearance of the film *Renaissance*, it is natural to consider the application of artistic thresholding to video. I developed an approach that is similar to Video Tooning [136]. In my approach, I view a video sequence as a three dimensional data set. Then a high dimensional segmentation is executed on it. The next step is to find a low-cost assignment directly on the

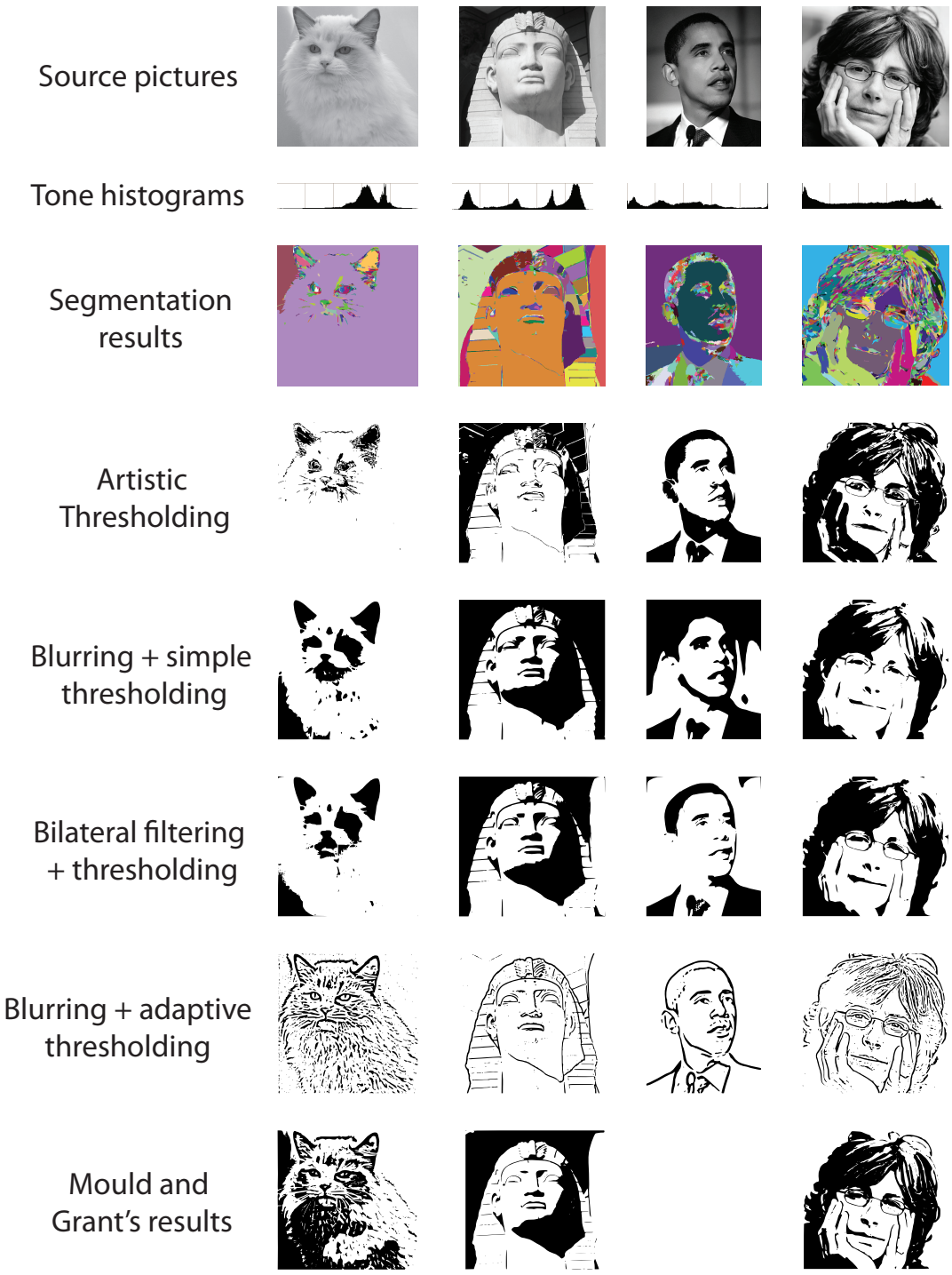


Figure 4.16: Applicability of artistic thresholding.

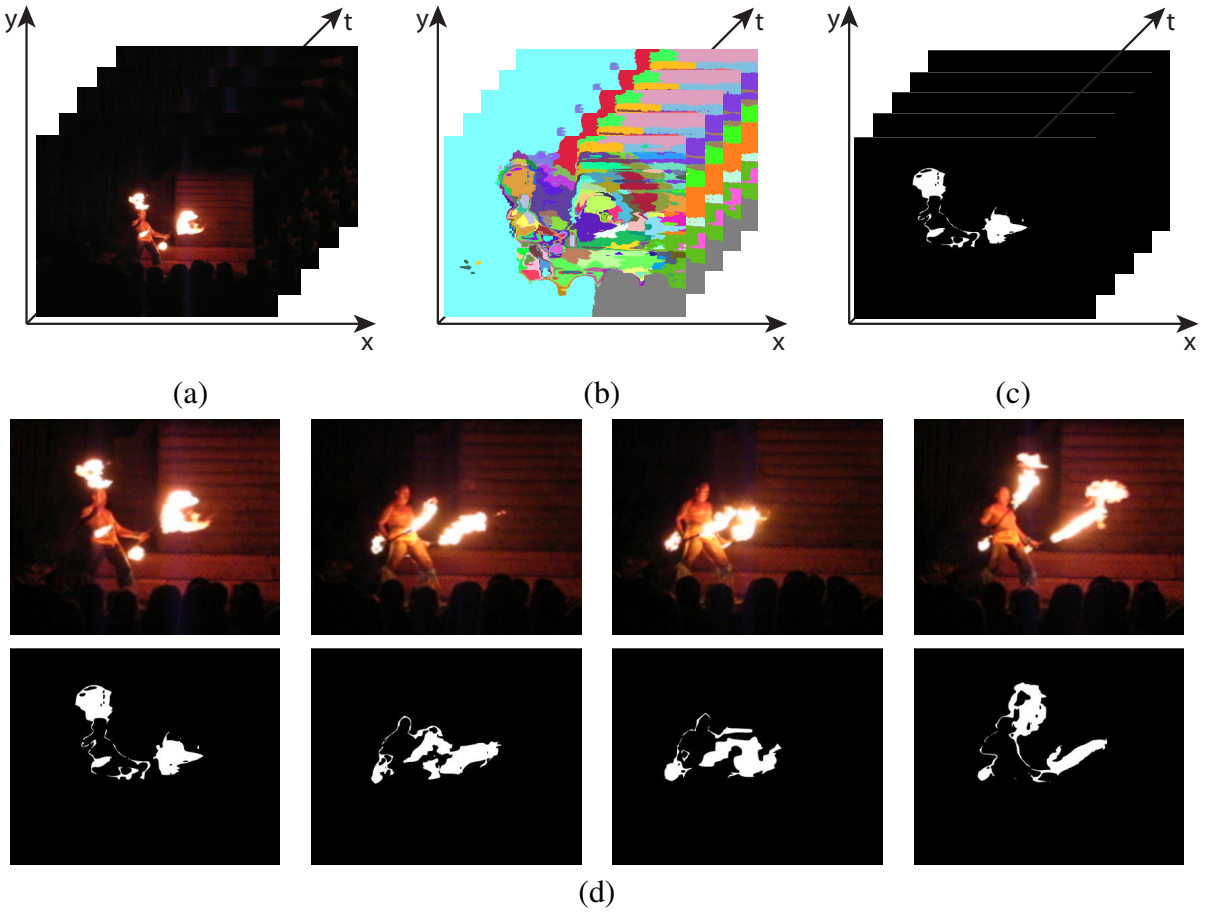


Figure 4.17: An example of black-and-white animation generation. An animation is a three dimensional data set (a). The segmented result is in (b). After optimization, I can get the bi-level animation in (c). Frames from source animation and resulting animation are in (d).

space-time volume equivalent of the region adjacency graph. The objective function is the same as my previous description. Finally, the newly generated black-and-white frames are merged into an animation. Figure 4.17 illustrates the approach.

4.8 Future work

There is an interesting relationship to be explored between artistic thresholding and line drawing. My research takes a first step by drawing missed edges explicitly after optimization is complete. I would like to investigate how an understanding of edges can be incorporated directly into the optimization. I might extend my binary assignment to include a black or white value for each edge in the region adjacency graph. I would then have to modify the cost functions to evaluate assignment quality in the presence or absence of these edges. At a minimum, I would want a term that simply minimizes the total length of all edges drawn, to encourage the binary segments to carry most of the salience in the result.

(a) (b)

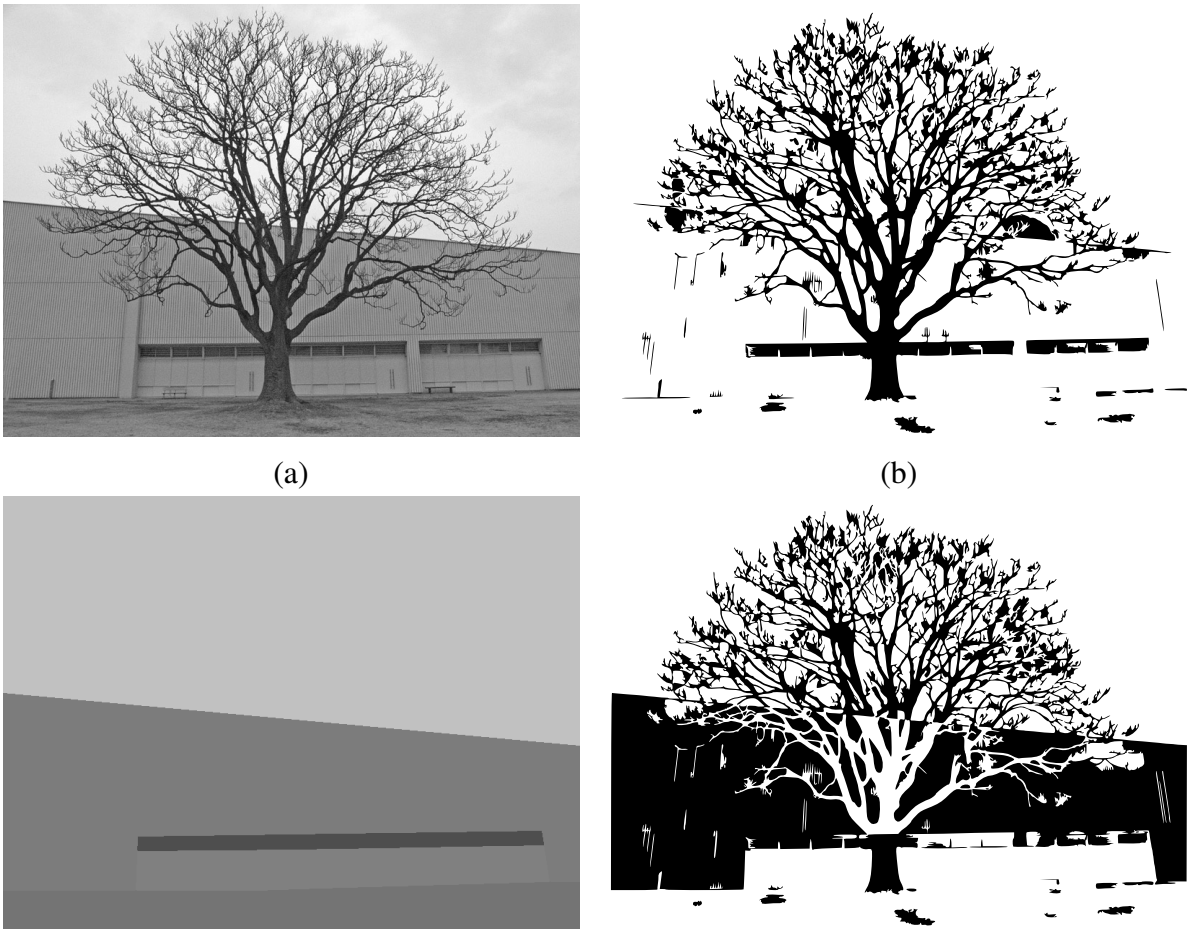
Figure 4.18: Examples of illusory contours: the Kanizsa triangle (a) and my drawing (b), inspired by ink painting [150].

Even when adjacent segments with contrasting colours are given the same binary value, it need not follow that their shared boundary must be drawn. Our perceptual systems are wired to infer portions of object contours that are obscured by lack of contrast. Perhaps the most famous demonstration of this effect is the "Kanizsa triangle", shown in Figure 4.18(a). Missing contours are inferred so strongly that we actually see an edge where none is present. These "illusory" or "subjective" contours have been demonstrated in many contexts [97], and are usually attributed to a Gestaltist explanation: a triangle occluding three circles is the simplest interpretation of the

image. This effect is used in practice, as in the boundaries between the houses in Figure 4.18(b) and the dresses in Figure 1.13(a) (Chapter 1). I would like to explore the problem of how well image edges can be represented even when partially invisible, taking inference into account. This effect is enabled uniquely by artistic thresholding: line art drawings do not exhibit illusory contours to the same degree.

Another effect not easily achieved is the use of "exclusive-or" to depict foreground objects on top of a background that varies between black and white. I have encountered many examples where thin foreground objects such as trees or table legs are drawn in white on black segments and black on white segments (see Figure 4.18(b)). The object is then visible everywhere, and can be perceived as a cohesive whole even though it varies between black and white. Because I start by segmenting a flat image, it is difficult to discover opportunities to use this effect. I can contrive to achieve it by specifying high-level features that deliberately cross through foreground objects. Those objects are then broken into multiple segments, allowing the optimizer to make independent assignment choices in the sub-segments. Figure 4.19 shows an example. An alternative would be to work from a 3D scene or $2\frac{1}{2}$D layers, in which case I can decide on an XOR-like compositing rule when the layers are flattened. This technique provides a solution to the tone inversion challenge discussed in the introduction. I discuss a similar approach in Chapter 5 in the context of papercutting.

Finally, it would be interesting to augment the objective function by taking into account further measures of salience in the source image. Salience could be painted by hand or derived from eye-tracking data [24]. It might also be computed automatically; Collomosse and Hall [19] used an automated salience algorithm to drive a genetic algorithm for painterly rendering. Salience would probably be used to annotate edges in the region adjacency graph with an importance value, which would then affect boundary contrast costs.



(a) (b)

(c) (d)

Figure 4.19: An example of how an "exclusive or" effect may be achieved via carefully constructed user features. The tree in (a) yields a single large segment, and therefore cannot contrast with both the building and the sky in (b). In (c) I force a feature for the building to cross through the tree. The boundary of this feature splits the tree into two segments that can be given opposite colours. Photograph by Flickr user atemzeit, used with permission.

Chapter 5

Computer-Generated Papercutting

The craft of papercutting is part of the folk art traditions of cultures all over the world. From the point of view of computer graphics, papercutting can be seen as a method of composing abstract bi-level images. It satisfies the definition of a wholetoned style discussed in Chapter 1. But it is not just wholetoning; papercutting is special because it has a set of geometric connectivity constraints so that a paper-cut design can be cut out from one single piece of paper. Moreover, many stylized ornamental motifs are commonly used in traditional papercutting as an efficient approach to texture and shape abstraction.

In this chapter, I present a technique for composing digital paper-cut designs. The elements of a design may be images, which are processed via artistic thresholding (Section 5.3), or they may be procedurally-generated arrangements of shapes (Section 5.4). Elements are composed using a set of boolean operators that preserve connectivity (Section 5.5). The resulting designs are well suited to being cut by a new generation of inexpensive computer peripherals.

5.1 Introduction

Papercutting, which originated in China 2000 years ago, is today part of the folk art traditions of cultures all over the world [55]. It is still a popular decorative art in China [71] (where it is known as Jianzhi), and is practised in distinct styles in Japan (Kirigami), Germany (Scherenschnitte), Poland (Wycinanki), Mexico (Papel Picado) and Jewish culture. In some of these cases, designs play a symbolic role in rituals or festivals; other uses are more purely decorative or artistic.

As a loose collection of traditions, there is no single recipe for constructing paper-cut designs. But if we restrict ourselves to a subset of all human-made examples, we begin to see some recurring mathematical features. In this work I consider the common case where the design is a connected shape formed by cutting holes into a piece of paper. If the image being depicted is represented by the paper itself, we call it a "positive paper-cut"; the image may also be represented by the holes left in the paper, which we call a "negative paper-cut". In either case the design is a connected subset of the plane which, except for the simplest silhouettes, will contain holes. Such shapes, which can feasibly be cut from a piece of paper, will be called "valid paper-cut designs".

Motivated by this simple mathematical description, in this chapter I examine the problem of constructing valid paper-cut designs with computer assistance. In particular, I develop a set of tools for constructing simple designs, each of which is a valid paper-cut, and then give a set of binary operations that allow paper-cuts to be combined while preserving validity. In my system, the user has access to high-level controls for creating and combining designs, and the computer handles the geometric details.

5.2 Related work

Figure 5.1 shows just a few examples from thousands of years of papercutting tradition. The traditional Chinese design in (a) is used to express wishes of good fortune for the new year. Cut-out silhouettes like those in (b) enjoyed significant popularity as a style of portraiture in 18^{th} century Europe. Today, artists such as Susan Throckmorton, whose work is shown in (c), produce beautiful and highly detailed paper-cut scenes that are excellent examples of both art and craft [129].

The use of computers to design cutting patterns for paper has become a popular research topic, driven in part by the simplicity and ready availability of computer-controlled cutting tools. Previous research has investigated papercraft sculpture [83], origami architecture [82] and pop-up books [35]. Within the art-math community, paper has been cut into interlocking grids depicting mathematical shapes [119] and polyhedral sculptures [42].

In computational geometry, some researchers have investigated the algorithmic aspects of cutting paper. Demaine *et al.* showed that a single straight-line cut can remove any shape from a suitably folded piece of paper [26].

(a) (b) (c)

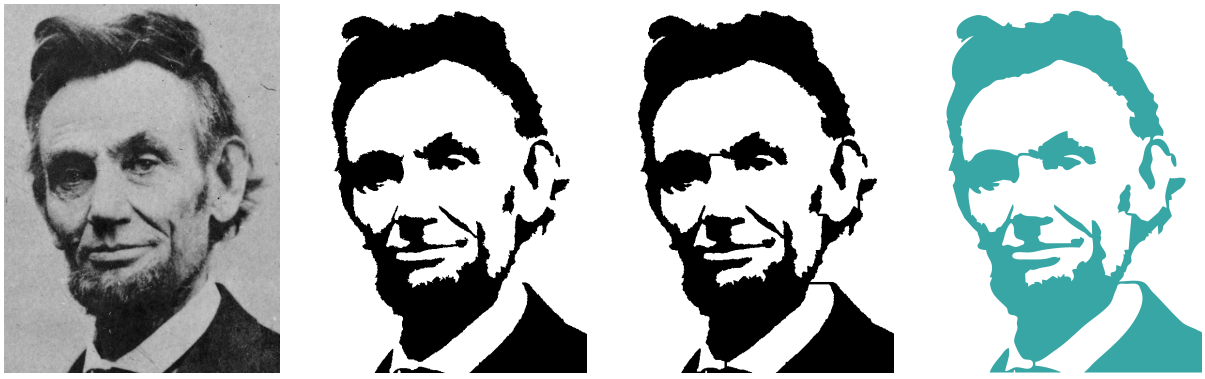
Figure 5.1: Three examples of papercutting: a traditional phoenix and flower design in (a), a silhouette portrait in (b), and a modern paper-cut by Susan Throckmorton in (c).

On the other hand, little research in computer science has addressed the problem of papercutting in the traditional folk art sense. The cut-out Islamic star patterns of Kaplan and Salesin [63] can be seen as a family of purely geometric paper-cut designs. Liu *et al.* [74] studied the cyclic and dihedral symmetries of different annuli in paper-cut designs, and showed how to synthesize new designs with different rotational orders. Recently, Li *et al.* [69] presented a design tool for annotating animated 3D surfaces with holes derived from traditional papercutting motifs.

Insofar as the designs we study in this chapter are binary images, we can see papercutting as a form of wholetoning. Existing research in non-photorealistic halftoning, such as pen-and-ink rendering [142], might therefore be seen as related. On the other hand, even if a pen-and-ink algorithm might be constrained to produce a connected final result (for example, the maze designs of Xu and Kaplan [146] are connected by construction), such designs are likely too detailed to be cut by a human, and possibly even by a machine. In my study of traditional papercutting, I found few examples with hatching or any other form of continuous tone reproduction. Papercutting tends to use a more stylized representation of shape.

5.3 Image-based paper-cut designs

I would like to have access to images as a source of paper-cut designs. Artistic thresholding might serve as a suitable conversion algorithm. However, for the purposes of papercutting, we must impose the additional constraint that the resulting representation be connected. In this section I therefore propose a two step conversion process: I apply my artistic thresholding on images, and then run an image-based algorithm to enforce connectivity.



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

Figure 5.2: Artistic thresholding applied to a portrait of Lincoln (a) to generate the black-and-white image in (b). Additional pixels are coloured in to force all regions to be connected in (c). The final vectorized paper-cut design in shown in (d).

5.3.1 Wholetoning

In my interactive application, I first provide the user with a standard set of tools for foreground extraction, based on lazy snapping [70] and intelligent scissors [86].

After background removal, we are now ready to create a bi-level version of the image. As shown in Figure 5.2(b), I apply the artistic thresholding technique presented in Chapter 4. Additionally, I provide an enhancement tool to overlay the result of edge detection on the wholetoned image. Image edges can help fill in outlines that disappear during wholetoning (see Figure 5.3).

Figure 5.3: A demonstration of adding detected edges to better depict object outlines.

5.3.2 Enforcing connectivity

The thresholding operation can produce many disconnected regions. As discussed in Chapter 4, we could attempt to force connectivity by minimizing the number of connected black components during artistic thresholding, but there is no guarantee this would work, or that the results would be visually satisfactory. I used a simple image-based algorithm to colour additional pixels black to yield a single connected component.

I first compute all the connected components in the thresholded image, and identify the smallest one. To create a valid paper-cut, I will need to connect this component to some other com-

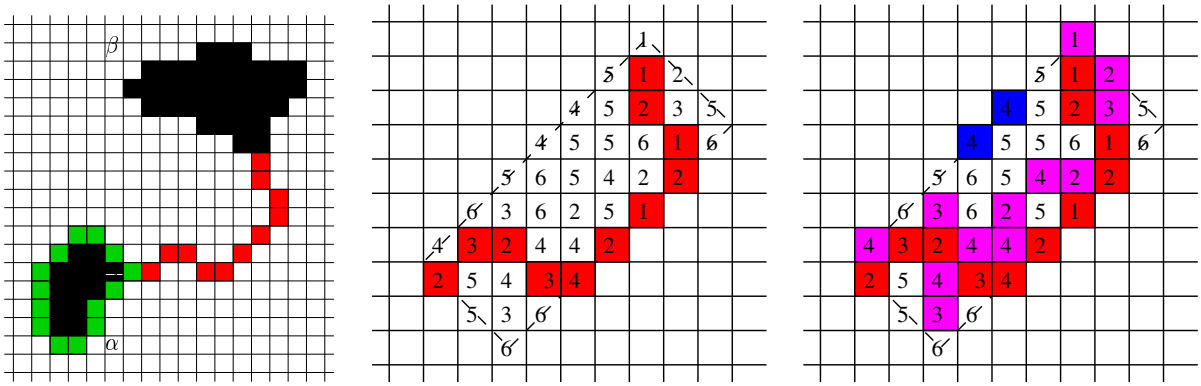
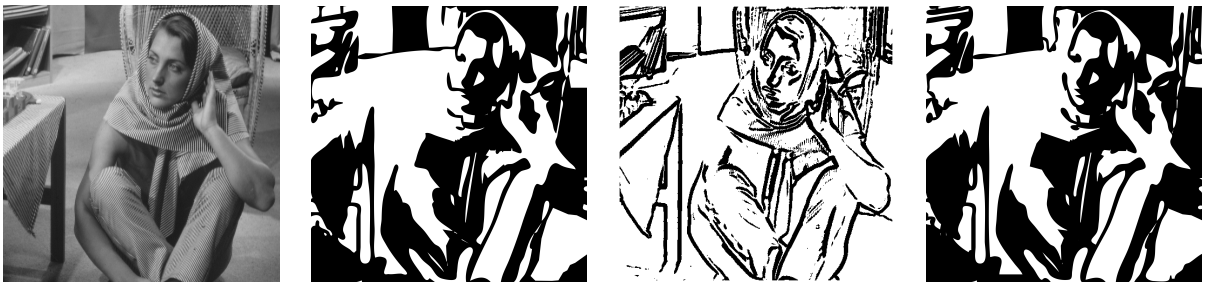


Figure 5.4: In (a), two components α and β are connected by a path of red pixels from the boundary of α (in green) to β. The path is surrounded by an oriented bounding box in (b). The numbers are the intensities of the corresponding pixels in the source image. I threshold all pixels inside the bounding box with the maximum intensity along the path (4 in this example). The result is shown in (c). The pixels connected to the path (shown in magenta) are added. The disconnected pixels (shown in blue) are discarded.

ponent via a path of pixels. I would like this path to be as dark as possible in the source image (measured as the sum of the values of its pixels). We can find such a path by running Dijkstra's algorithm outward from every pixel on the boundary of the component, using pixel values as edge weights. I treat the component as a single source vertex connected to its boundary pixels, and stop the algorithm as soon as I encounter a pixel on the boundary of another component. Figure 5.4(a) shows an example of a path discovered using this search.

Of course, a one-pixel-wide path is unattractive, and probably too narrow to be practical in a papercutting context. I therefore provide a means of thickening the path based on intensities in the source image. I compute an oriented bounding box from the 2D covariance matrix of all pixels on the path, as in the stroke analysis method of Barla *et al.* [6, Section 2.1]. The two eigenvectors of this matrix are the axes of the bounding box, and I project all pixels to these axes to get the box's dimensions (see Figure 5.4(b)). I define a threshold I as the lightest intensity of the source image pixels on the path. I then set every pixel in the bounding box to black if it

is darker than I, and white otherwise. Finally, I keep those black pixels that belong to the same connected component as the path itself, and discard any others (see Figure 5.4(c)). This process can be seen as locally relaxing the thresholding operation to let in more pixels. In practice I have found that it adequately thickens paths, making satisfactory connections between components.



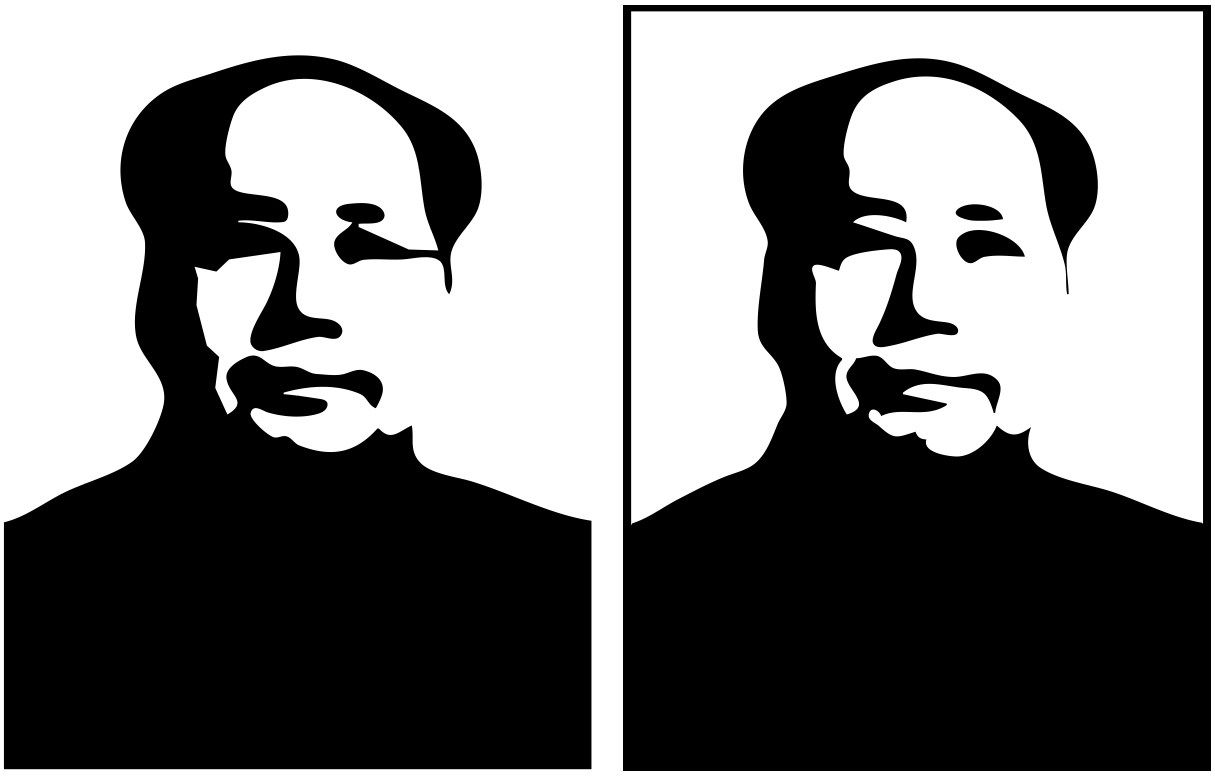
(a) (b) (c) (d)

Figure 5.5: Edge detection information is helpful in pathfinding. Given the source image (a), we can get a paper-cut design with original pathfinding algorithm in (b). Note that the cheek is poorly connected to the mouth. A better result (d) can be acquired by weighting pixels with edge information in (c). The cheek is connected to the chin by following the salient edges.

The pathfinding step merges the smallest component with some other component. I repeat this process until there is only one connected component left. I fill small holes introduced during thresholding and pathfinding by applying two rounds of the morphological CLOSE operation, and then use the well known Potrace library [117] to recover vector paths for the design.

The pathfinding algorithm can be improved by employing salience information from the image, for example, detected edges. I add an extra weight to the pixels with the edge detection information. Pixels on edges have smaller weights, which enforces the path to follow salient edges. Figure 5.5 shows the effect of this enhancement.

Note that if we invert the binary image before computing connectivity, we can construct a negative paper-cut design instead of a positive one. Connectivity can then be enforced on the negative binary image, as before. Figure 5.6 shows a comparison of positive and negative designs.



(a) (b)

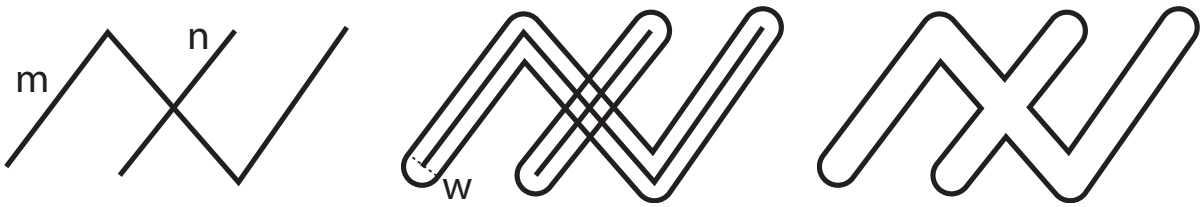
Figure 5.6: Positive (a) and negative (b) paper-cut designs of the portrait of Mao Zedong. The paper-cut in (a) is a connected black region on a white background; in (b), it is a connected white region on a black background.

5.4 Pattern synthesis

Traditional paper-cuts frequently feature decorative ornament and stylized or geometric patterns. I support an extensible set of patterns that can be combined with each other and with image-based designs.

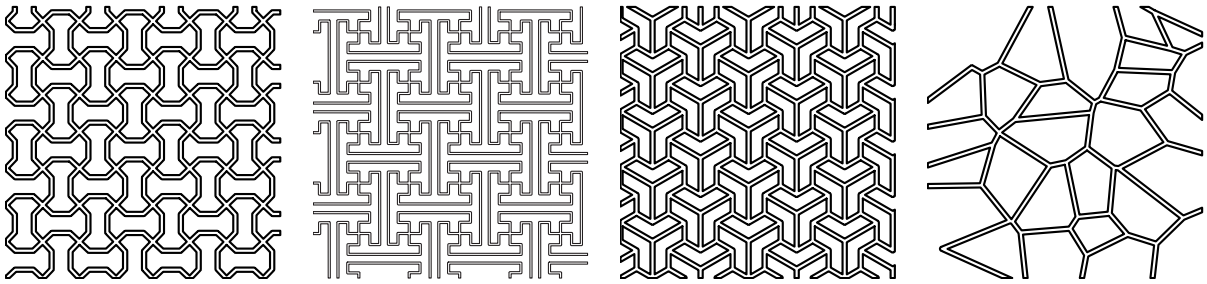
Geometric patterns reminiscent of latticework are a popular papercutting device. Aside from their decorative function, they provide a connected substrate into which other objects can be embedded. I support a wide range of geometric patterns in the form of isohedral tilings of the plane [41]. The edges in any tiling can be thickened to produce a valid paper-cut design.

Figure 5.7 shows the procedure.



(a) (b) (c)

Figure 5.7: The creation of a paper-cut design from lines. Given the two sets of lines *m* and *n* in (a), I first compute the offset lines with width *w* in (b), and for the joints between two segments, I apply round corners. Finally, all offset lines are merged into one single shape in (c).



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

Figure 5.8: Paper-cut designs based on simple geometric patterns. Isohedral tilings are shown in (a), (b), and (c); the latter two were taken from Chinese papercutting and latticework. A Voronoi diagram is given in (d) (patterns similar to Voronoi diagrams occasionally appear in Chinese latticework [124]).

Figure 5.8 shows several of my geometric patterns, inspired by traditional Chinese designs. There are many other potential sources of geometric designs for papercutting, such as Islamic star patterns [63].

I also support the synthesis of freeform arrangements of stylized ornamental motifs. I have experimented with a small set of conventionalized motifs inspired by Chinese papercutting (see Figure 5.9), though of course many others are possible. In my system, the user selects a pattern type and draws a stroke. I use the stroke pattern synthesis method of Barla *et al.* [6] to place

Figure 5.9: Predefined ornamental patterns, based on conventionalized motifs from the Chinese papercutting tradition.

Figure 5.10: An example of synthesizing the WATER and FIRE patterns along two letterforms.

motifs along the stroke, and deform the motifs to fit the stroke's path in a manner similar to skeletal strokes [51]. The user can also control the density and sizes of motifs placed along strokes. An example of my stroke-based pattern synthesis is shown in Figure 5.10.

Note that synthesized patterns are generally disconnected, and hence not valid paper-cut designs. When necessary, I consider such a pattern to be a set of holes cut out from a sufficiently large rectangle.

Procedural methods are another approach to generate patterns such as trees, flowers and mountains. I can define rules about how to "grow" such a pattern. But I must apply the constraint to ensure the all parts of a pattern are connected. Figure 5.11 gives an example.

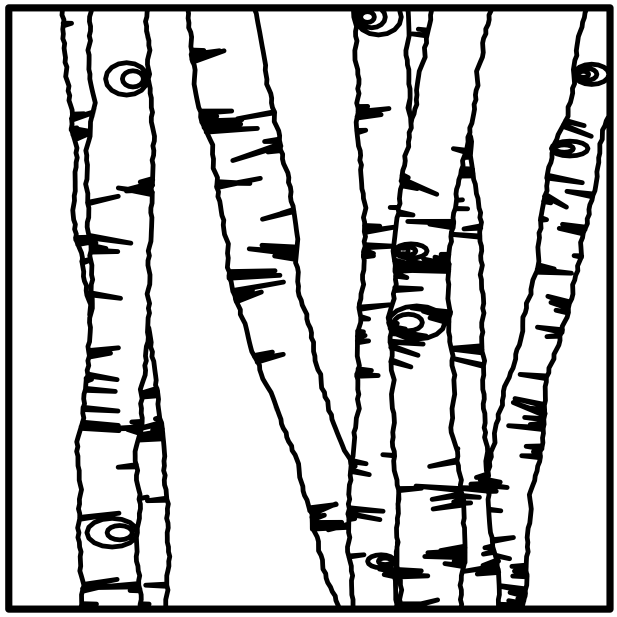


Figure 5.11: An example of procedurally generated birch trees.

5.5 Compositing Paper-cuts

The techniques of the previous two sections can be used to construct a wide variety of paper-cut designs. I would also like to be able to combine individual designs into finished scenes. Here I encounter an interesting mathematical problem: given two valid paper-cut designs, how may they be combined to produce a valid result? In other words, I wish to define a set of binary operations on connected sets that preserve connectivity.

Let A be a valid paper-cut design, i.e., a non-degenerate connected set in the plane. I may regard A as partitioning the plane into three disjoint regions: A itself, A_i, the set of holes contained entirely within the outer boundary of A, and A_o, the rest of the plane. Note that because A is connected, A_i cannot enclose further parts of A. An example is shown in Figure 5.12. Similarly,

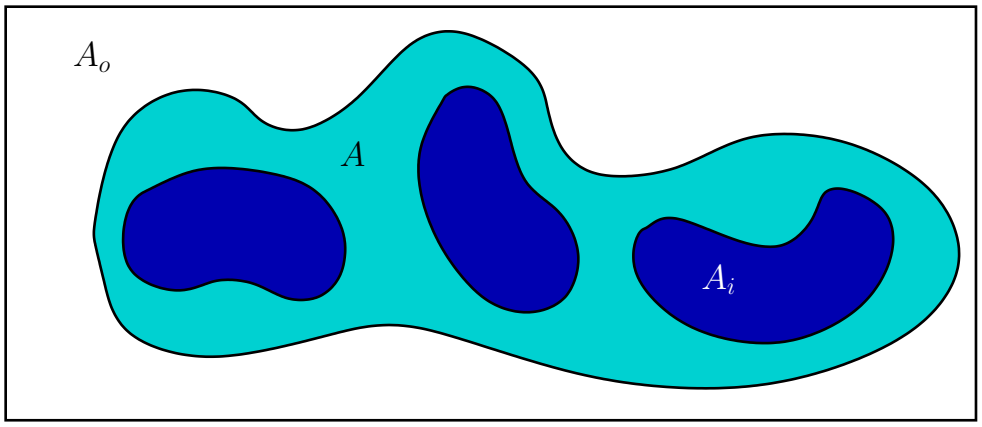


Figure 5.12: An example of shape A.

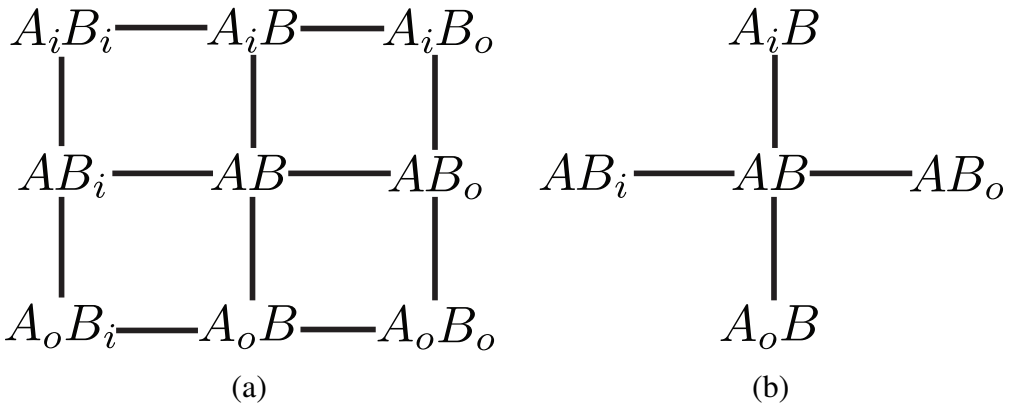


Figure 5.13: The adjacency of all nine region types is shown in (a). After removing A_oB_i, A_iB_o, A_iB_i, and A_oB_o, the remaining five regions are shown in (b).

a second design B partitions the plane into B, B_i, and B_o (see the top row of Figure 5.14). When A and B are superimposed, the pairwise intersections of these sets partition the plane into nine regions, as shown in Figure 5.13(a). It is easy to find that one region's internal holes (such as A_i) are not adjacent to its external regions (such as A_o), and if two regions are adjacent each other, then they should share one label. Figure 5.13(a) shows the relationship of the nine region types. Any binary operation on A and B can be specified by assigning a boolean value to each of these

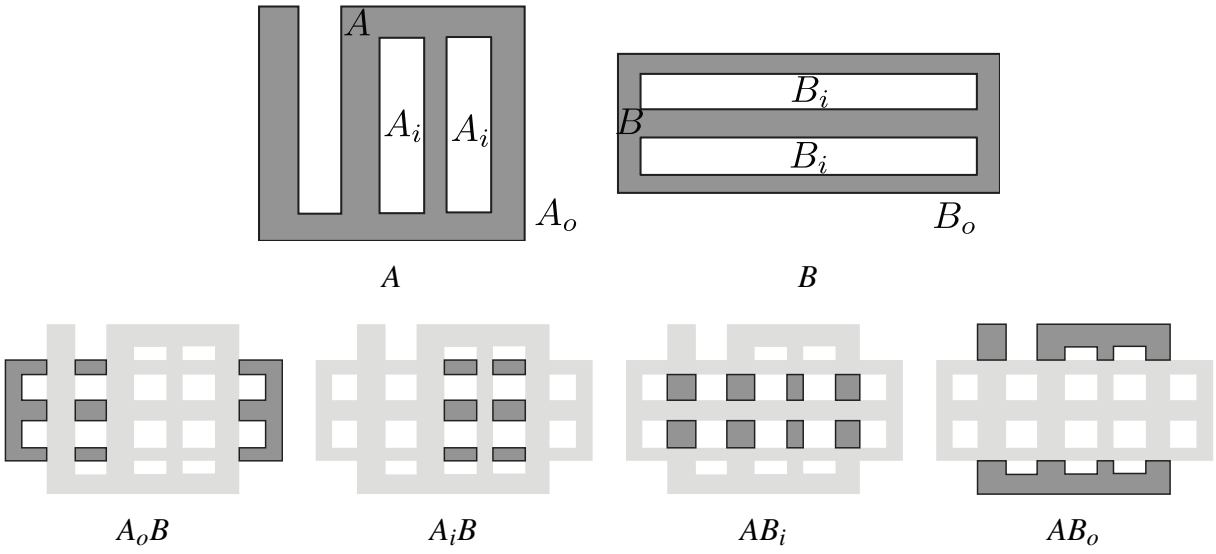


Figure 5.14: Given two valid designs A and B, A_oB, A_iB, AB_i, AB_o alone cannot guarantee validity. The two superimposed shapes A and B are illustrated in light gray colour and the combinatorial results are shown in dark gray colour.

nine regions – **true** if the region is part of the result, and **false** if it is not. Therefore, there are 512 possible operations. We can limit the possibilities by assuming that operations do not add paper that did not originally belong to A or B. In other words, the regions A_oB_o, A_oB_i, A_iB_o and A_iB_i must be set to **false**, leaving us with five boolean choices and 32 operations, as shown in Figure 5.13(b).

Figure 5.14 shows that we cannot guarantee the validity if only a single set of regions is set to **true**. From Figure 5.13(b), I can conclude that it is necessary to assign **true** to AB, if more than two sets of regions are used. In this situation, there are only 16 possible operations left.

Theorem 5.1. *Let A and B be two valid paper-cut designs, for which* $A \bigcap B \neq \emptyset$*. Their combination is also a valid design if A or B is embedded in the result.*

Proof of Theorem 5.1. Suppose A is embedded in the result. Then AB_i, AB and AB_o must be **true**. From the assumption, A is valid and AB is also not empty, so all regions in AB_i, AB and AB_o are connected. Any additional regions are either A_iB or A_oB. From Figure 5.13(b), we know

these must connect to AB, meaning that they connect to A and the new result is valid. The same argument holds for B. □

	AB	AB_i	AB_o	A_iB	A_oB
A	T	T	T	F	F
B	T	F	F	T	T
A UNION B	T	T	T	T	T
A OVER B	T	T	T	F	T
A UNDER B	T	F	T	T	T
A WITHIN B	T	T	F	T	T
A WITHOUT B	T	T	T	T	F

Table 5.1: Seven valid paper-cut operations.

Among those 16 operations, there are seven satisfying Theorem 5.1. Inspired by the compositing operators introduced by Porter and Duff [102], I summarize my operations in Table 5.1. I name each one and define it in terms of the boolean values for the five regions. Figure 5.15 gives illustrations for each.

Theorem 5.2. *There are only seven operations that guarantee validity.*

Proof of Theorem 5.2. Aside from the preceding seven operations, there are nine possible cases. Figure 5.16 shows counterexamples for each case. So I can conclude that any other operation is invalid. □

Based on observations of traditional paper-cut designs, I would also like to support an XOR operation, in which the region AB is set to false (Table 5.2), *i.e.*,the intersection of the two shapes is cut out. As I demonstrated in Figure 4.19, the XOR operation allows two shapes to coexist

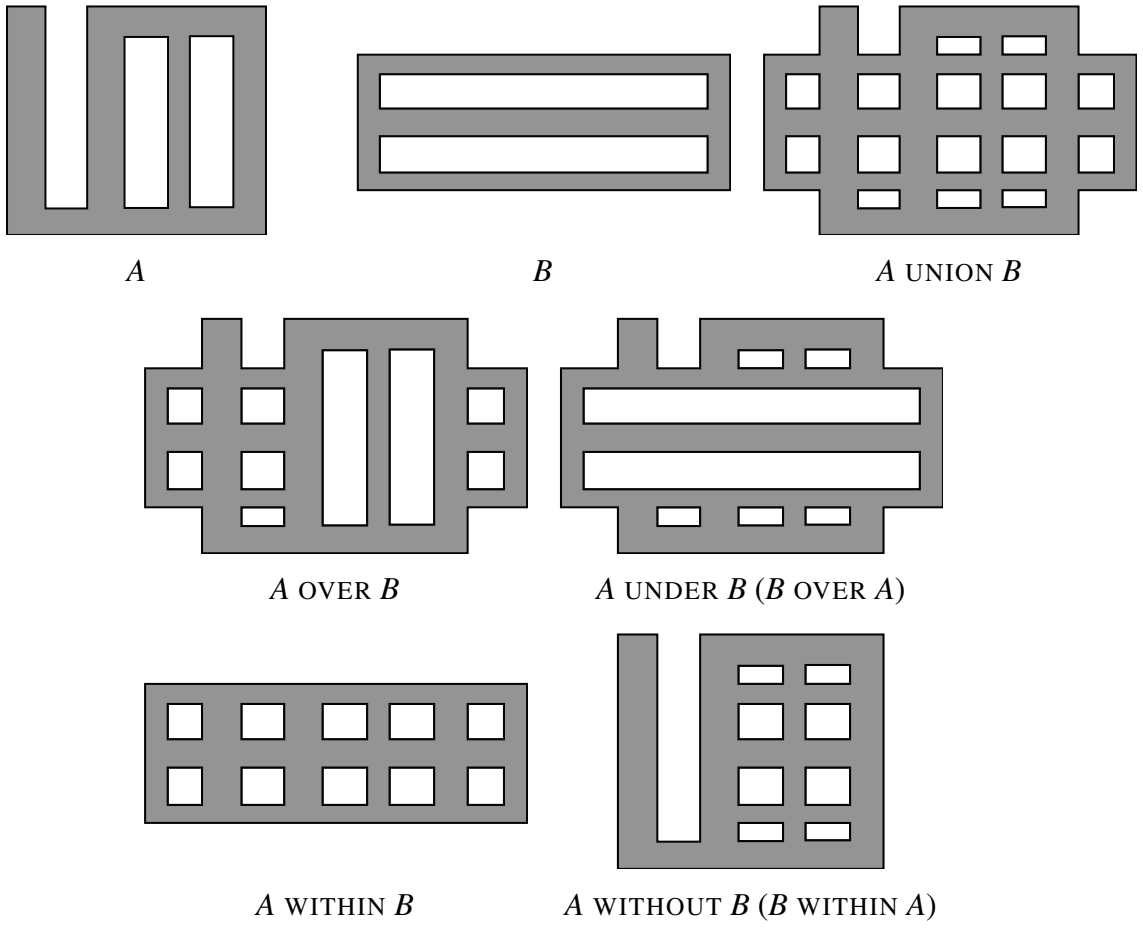


Figure 5.15: The seven binary operations on paper-cut designs that preserve validity, as explained in Section 5.5.

	AB	AB_i	AB_o	A_iB	A_oB
A XOR B	F	T	T	T	T

Table 5.2: XOR operation.

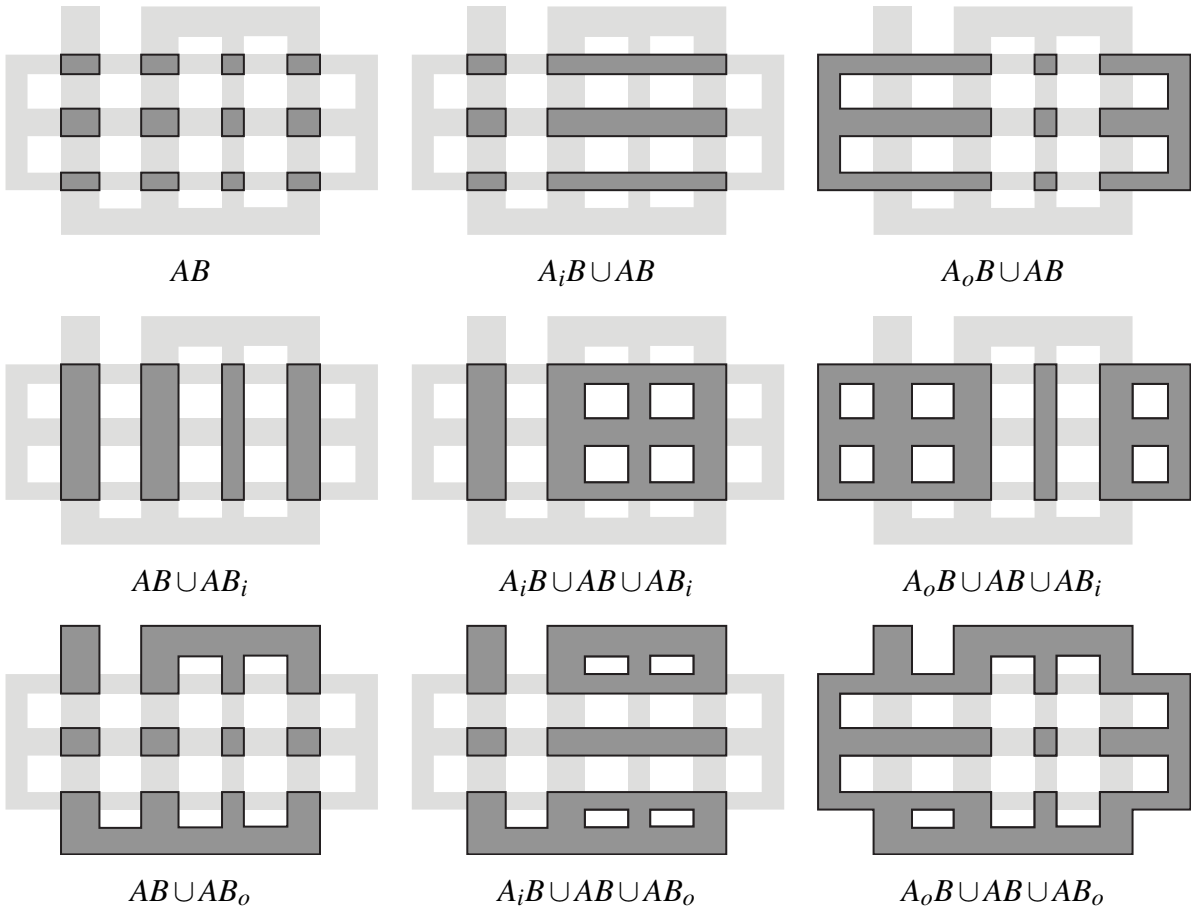
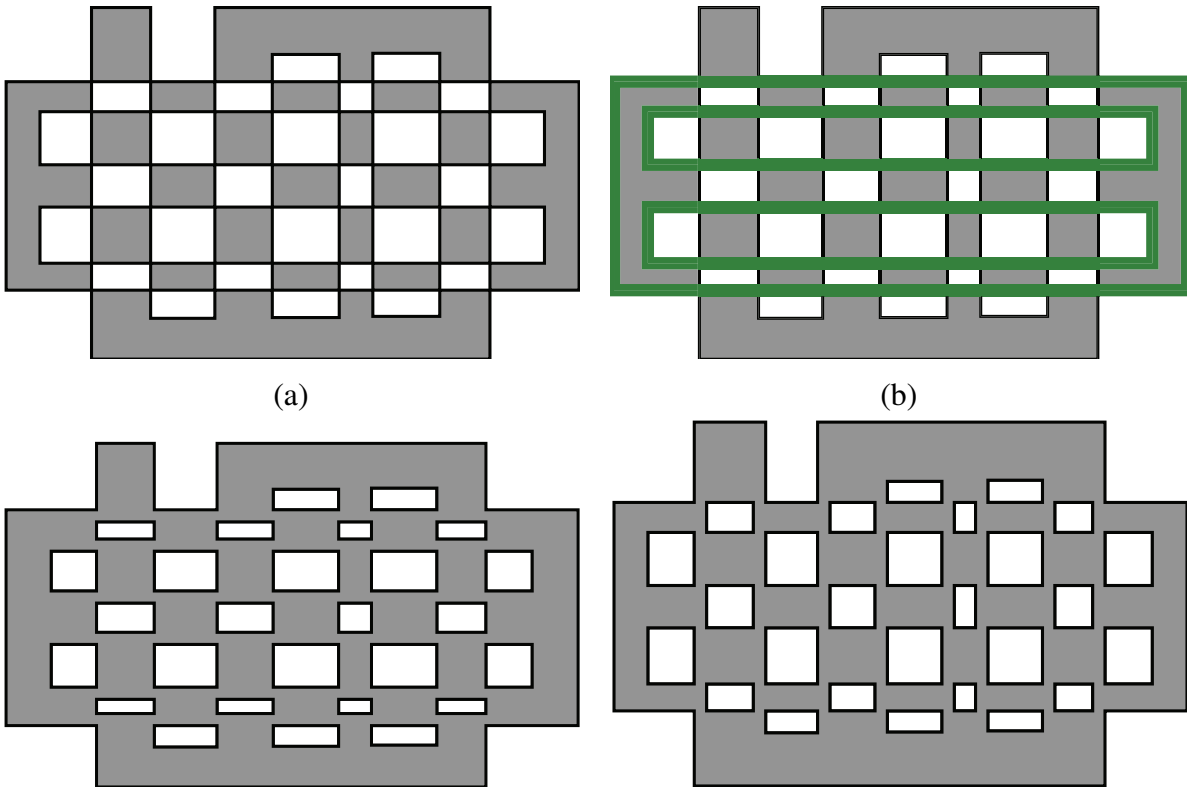


Figure 5.16: Examples of nine invalid operations, based on the shapes in Figure 5.15.

in the same space, with the edges of both discernible. It efficiently addresses the tone inversion challenge. But it is not an explicit operation in artistic thresholding, where the user's input about high-level features is need. In papercutting, it is an intuitive operation in exploring paper-cut designs to be composed in layers. Unfortunately, XOR is not guaranteed to produce connected results. I can fix this by thickening the edges of one of the shapes so that it overlaps the second, as shown in Figure 5.17. Because I can choose to thicken the edges of A or B, my XOR operation is asymmetric. Figure 5.18 shows an example of composing a complex paper-cut design from a set of primitive elements using the seven valid operations and XOR. Note that the overlapped steam cup is clearly visible because of tone inversion induced by XOR.



(a) (b)

(c) A XOR B (d) B XOR A

Figure 5.17: The XOR operation on paper-cut designs. The simple XOR in (a) does not preserve validity. If we thicken the edges of B as in (b), we can restore validity as in (c). The design in (d) thickens the edges in A instead.

Many other set-theoretic operations, such as $A \cap B$ and $A \setminus B$, produce correct results in some contexts but not others. For instance, I can cut a pattern of holes into a shape A by intersecting A with a large rectangle containing the holes, as mentioned in Section 5.4. I permit the user to perform these "unsafe" operations, but must check whether the result is connected.

5.6 Results and future work

I have created a prototype implementation of my technique using C++ and Gtkmm. I represent paper-cut designs as polygons with holes, and use the GPC library [89] to compute boolean op-

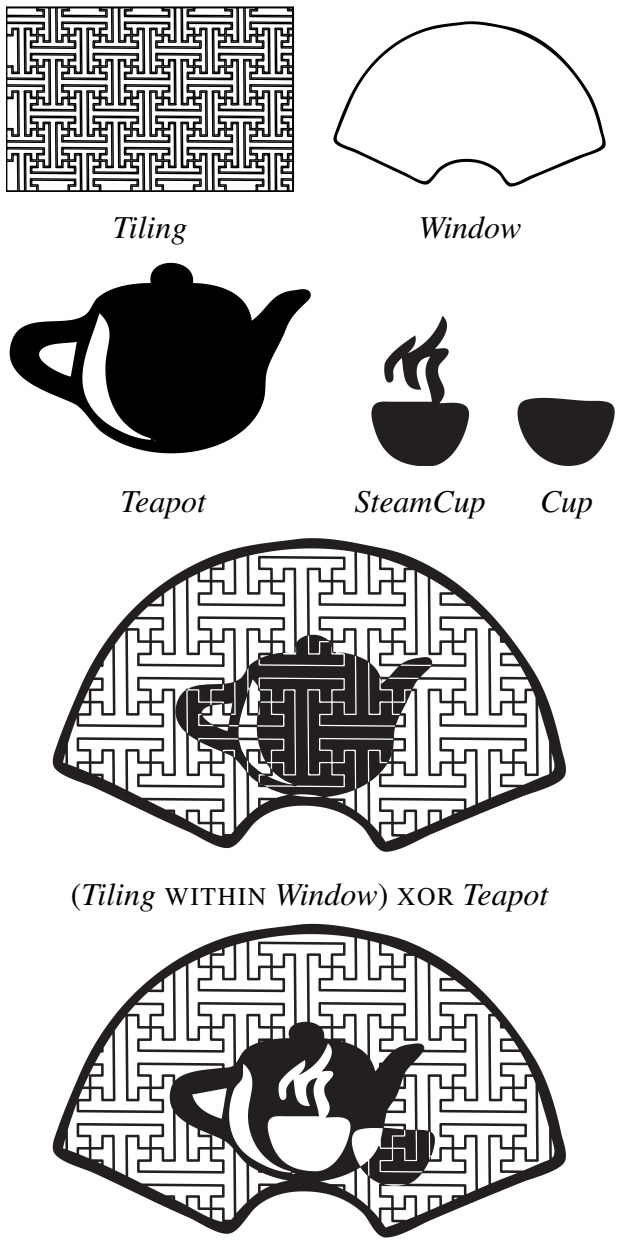


Figure 5.18: A demonstration of boolean operations on paper-cut designs. Five source designs are shown in the top two rows. In the first finished design, the XOR operation is used to make the teapot appear to be behind the screen. The teapot is placed OVER the screen in the second design, and a couple of cups are then added with XOR.

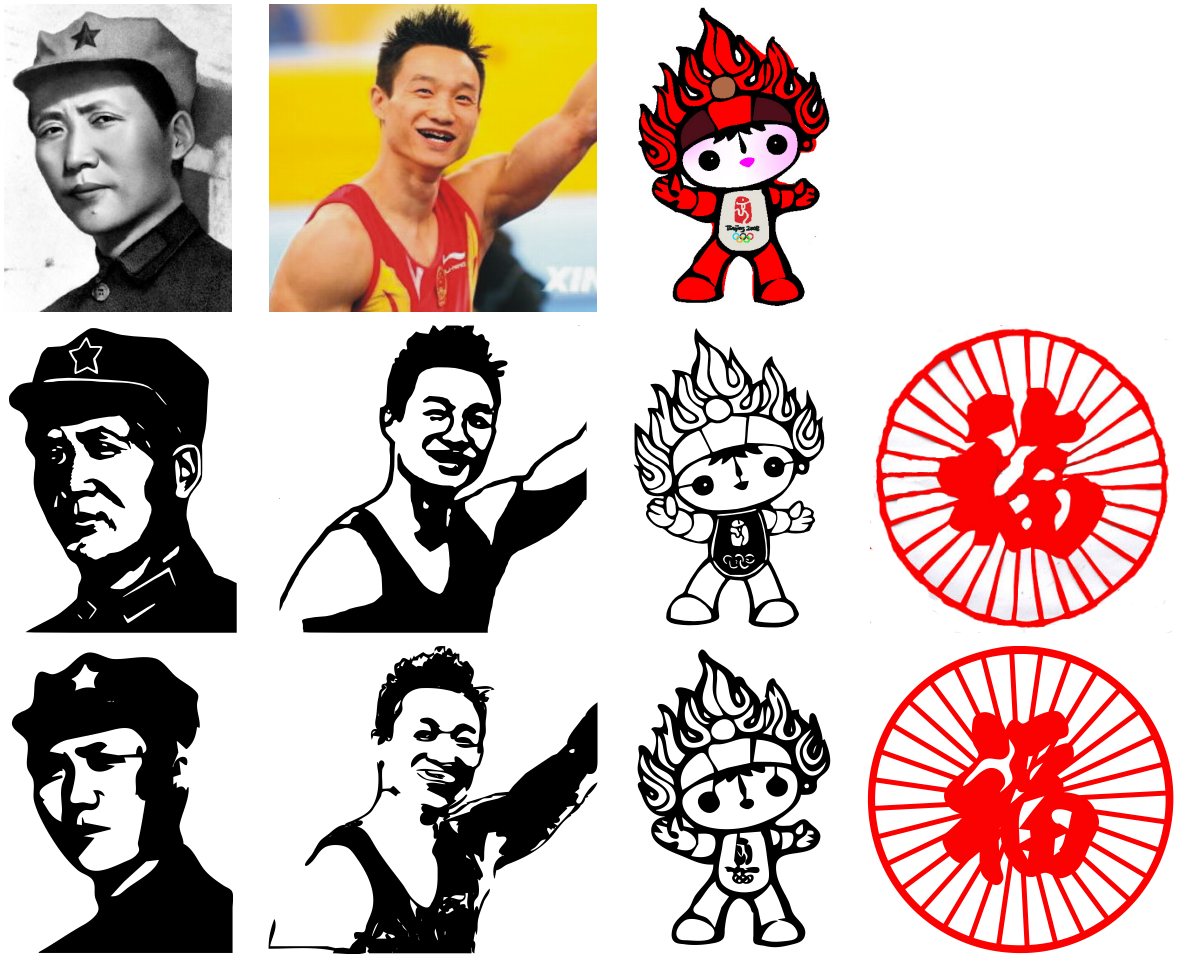


Figure 5.19: Evaluation of my paper-cut results. The first row shows source photographs. The second row shows the results created by artists. My results are shown in the third row.

erations on them. My interface lets the user process images, create stylized patterns, and perform boolean operations on designs. A screenshot of my system is shown in Appendix A. The output is a PDF file of cutting paths. The PDF can be displayed directly by filling the paths, or it can be used to create actual paper-cuts using a variety of computer-controlled manufacturing devices. I have experimented with the QuicKutz Silhouette digital craft cutter, a small, inexpensive knife cutter intended for home use (see Figure 5.20(d)). Finer and more precise results could be

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

Figure 5.20: The construction of a paper-cut based on a painting of a goose by Zhu Da (c.1626 – c.1705). The original image is shown in (a). My image-based technique extracted the design in (b), to which some synthesized patterns were added in (c). In (d), I show the design cut from cardstock using an inexpensive computer-controlled craft cutter.

obtained using a laser cutter.

(a) (b) (c)

Figure 5.21: Three additional paper-cuts. The design in (a) is based on the Stanford Dragon model, created using the technique presented in this chapter. The design in (b) is a couple dancing among procedurally generated trees, and (c) is the Chinese character “Fu” (good fortune) embedded in a geometric pattern.

Figure 5.19 demonstrates the comparison of my results and artists' works. We can see my system can generate pretty good results based on photographs and it is convenient and efficient to compose multiple designs. But the limitations are similar to those discussed in Chapter 4: artists can handle abstraction and detail much better than my system, and I need further work to implement more stylized traditional paper-cut designs.

Some additional results generated by my system are shown in Figures 5.20, 5.21, and 5.22.

There are several directions I would like to explore in the future. Many traditional paper-cuts feature a central design surrounded by annuli with different cyclic or dihedral symmetries [74]. I would like to automate the construction of symmetric designs.

It would also be interesting to extract more information from images when transforming them into paper-cut designs. It might be possible to decompose an image into overlapping regions that can then be assembled using XOR. Another challenge would be to automatically select patterns from a library to approximate details in the image, or better yet to develop conventionalized vector patterns directly from image features.

Figure 5.22: A paper-cut design based on the Big Wild Goose pagoda (Xi'an, China).

Chapter 6

Calligraphic Packing

As mentioned in Chapter 1, representational calligraphy is a derived wholetoned style. In representational calligraphy, warped letters are assembled into a composition. Due to their limited expressivity, a representational calligraphic illustration is definitely an abstract depiction. Furthermore, letters are usually rendered in black color over white background, meaning there are only two tones. Thus it satisfies my definition of a wholetoned image. However, this style presents an extra challenge on top of artistic thresholding: we would like the letters to fit the whole shape well while preserving their readability.

In this chapter, I develop a solution to the "calligraphic packing" problem. Foreground extraction and artistic thresholding are applied to a source image to produce a container region. Then, given a sequence of letters, my system uses a clustering algorithm (Section 6.3.2) to subdivide the container into subregions. For each letter, I then warp a corresponding glyph into its subregion (Section 6.3.3) in a way that minimizes distortion (Section 6.3.4). I automate the process of warping glyphs to fill the container by optimizing an energy function that chooses a warp that best represents the original letter. The energy function is composed of shape similarity, orientation and area coverage. To create more appealing results, I provide a set of interactive tools that help users adjust the details of letters. I also include some variations in rendering style (Section 6.4).

6.1 Introduction

Artists and art lovers have always been captivated by the interplay between a whole and its parts. When a complete image is seen as composed of many small cooperating elements, we can appreciate not only the image as a whole, but the ingenuity of its conception and realization. In a Roman mosaic, for example, small squares of coloured glass conspire to form a detailed scene.

(a) (b)

Figure 6.1: Two examples of representational calligraphy. The tiger on the left, by Sudanese-born artist Hassan Musa, is made up of elongated Arabic letters. The portrait on the right is from an advertisement for the magazine Veja.

Especially fascinating is the case where the parts are themselves recognizable objects. The viewer is then caught in a dynamic tension between attending to the image and to the elements that make it up. A famous example that has inspired many researchers in computer graphics

is Arcimboldo's sixteenth century portraits [79]. His subjects are assembled from small objects such as flowers, vegetables, fruits, and animals. Halsman's famous *In Voluptate Mors* is a portrait of Dalí featuring a skull assembled from nudes. In Escher's tessellations, the whole is simply the entire plane, but we are fascinated by the way lifelike characters occupy it without any gaps. Escher was also interested in the narrative possibilities that arise in having multiple different objects interact in an image.

A special case of this artform can be found in what I call "representational calligraphy". Here, a collection of letters or symbols cooperate to compose an image that is also a calligraphic inscription. Islamic calligraphers are certainly the masters of this style. A recent example by Sudanese-born artist Hassan Musa is shown in Figure 6.1(a). Today, representational calligraphy is frequently used in graphic design applications such as advertising, product design, and corporate logos. The example of Figure 6.1(b) was part of an advertising campaign for Veja, a Brazilian news magazine.

In representational calligraphy the letters are frequently distorted, and the inscription may be difficult or even impossible to read. In some examples, it is possible to recognize the letters only if one already knows (from the context) what they are. Note that there is no harm in deforming letters this way. Legibility and consistency are the goals of *typography*; both may be suppressed for artistic purposes in calligraphy. The resolution of the resulting visual puzzle can be a rewarding aesthetic experience.

Within non-photorealistic rendering there has been a great deal of work in the area of "NPR Packing", the general problem of depicting an image by automatically arranging a collection of small pictorial elements. But to my knowledge, no research has been done on the specific problem of packing letterforms. Inspired by the examples above, I therefore pose a representational calligraphy problem for computer graphics:

Problem (Calligraphic Packing): Given a region of the plane (the "container") and a sequence of letters $L = \{l_1, \ldots, l_n\}$, construct a non-overlapping arrangement of deformed glyphs $\{g_1, \ldots, g_n\}$ in the interior of the container so that

1. The glyphs fill the container as much as possible;
2. Individual glyphs are recognizable as the corresponding letters; and
3. The order of the letters is suggested by the arrangement of the glyphs in the packing.

Note that I make a distinction between a letter (an abstraction) and a glyph (its representation). A single letter may be equally well represented by any of a number of glyphs, such as lowercase and uppercase versions.

Although calligraphic packing is related to other forms of NPR packing, there are important differences that make this problem unique. First, we do not have a database of available objects, and the freedom to pack copies of objects from the database at will. We are given a specific sequence of symbols, and must include each exactly once. The relative positions and orientations of the letters are important too, as they affect overall readability. Finally, we expect to apply a significant amount of deformation to the glyphs to pack them. Unlike most NPR packing applications, the glyphs can undergo a great deal of deformation compared to ordinary images. The glyphs are not intended to be *representations*, merely *signs*. As long as the original letters can be recovered, the incidental shapes of the glyphs are unimportant.

6.2 Related work

One primary thread of NPR packing techniques has focused on the use of Lloyd's method to relax an initial distribution of objects into a final, evenly-spaced configuration. Hausner [43] simulated the appearance of Roman mosaics. He applied Lloyd's method to Voronoi diagrams generated via the Manhattan metric, giving an arrangement of oriented rectangular mosaic tiles. Hiller *et al.* [47] improved and generalized this result by using area Voronoi diagrams, allowing more complex objects to be packed together. Secord [115] used a weighted version of Lloyd's method as a form of importance sampling to create stippled depictions of images. But for arbitrary tiling shapes like the letters, these methods cannot produce tiles that resemble these shapes.

Recently, Dalal *et al.* [22] achieved excellent packing results by modifying the relaxation step in Lloyd's method. Instead of moving the object to the centroid of its Voronoi region, they define a metric based on image correlation that minimizes variation in the space around the objects (the "grout" in the mosaic). All these packing techniques focus on distributing a large number of small elements without deformation. For calligraphic packing, I wish to pack a small, fixed set of deformable shapes.

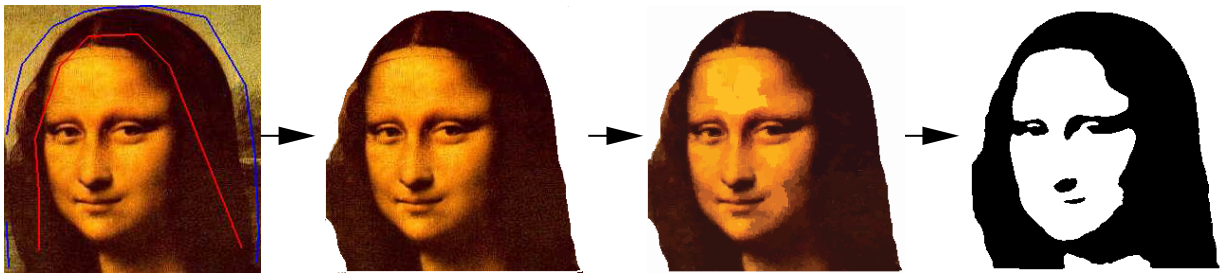
Kim and Pellacini [65] presented a more generic framework for packing shapes chosen from a database of candidates into a container. Their algorithm attempts to minimize an energy function that trades off between various measures of the packing's quality. It is similar to the weighted sum objective function in artistic thresholding (Chapter 4). Their technique deforms objects slightly to even out the irregular boundaries between them. I would like to support an even greater amount of deformation and provide some mechanism for controlling the flow of the letters to maintain readability.

Kaplan and Salesin [62] examined the problem of Escherization, in which a given shape must be converted into a tiling of the plane. Their algorithm seeks a tileable shape that resembles the original as closely as possible, thereby minimizing the amount of deformation that must be applied.

In the artistic screening method of Ostromoukhov and Hersch [95], shapes such as letter-forms were continuously deformed via interpolation from a small set representing different tones. Grids of these shapes were then used for halftoning by varying the interpolation level spatially. Surazhsky and Elber [126, 127] presented a method to arrange text along free-form parametric curves. They rendered their text with varying tone along projected parametric curves on surfaces to produce comprehensible text-based rendering of 3D shapes. Compared to these techniques, my deformations are more freeform. Also, my results are not halftoned representations; I pack a container derived from a bi-level image, producing a wholetoned illustration as a result.

6.3 Approach

I created a system that produces calligraphic packings. Given an image and a sequence of letters, I convert the image into a black-and-white representation of an object (Section 6.3.1), subdivide it into regions (Section 6.3.2), and warp the individual letters into these regions (Sections 6.3.3



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

Figure 6.2: The extraction of a container region from an image. The user marks the input image in (a) with foreground (red) and background (blue) curves. Lazy Snapping extracts the foreground object in (b). The segmented image in (c) is fed in to the artistic thresholding framework to obtain the bi-level container in (d).

and 6.3.4).

6.3.1 Container extraction

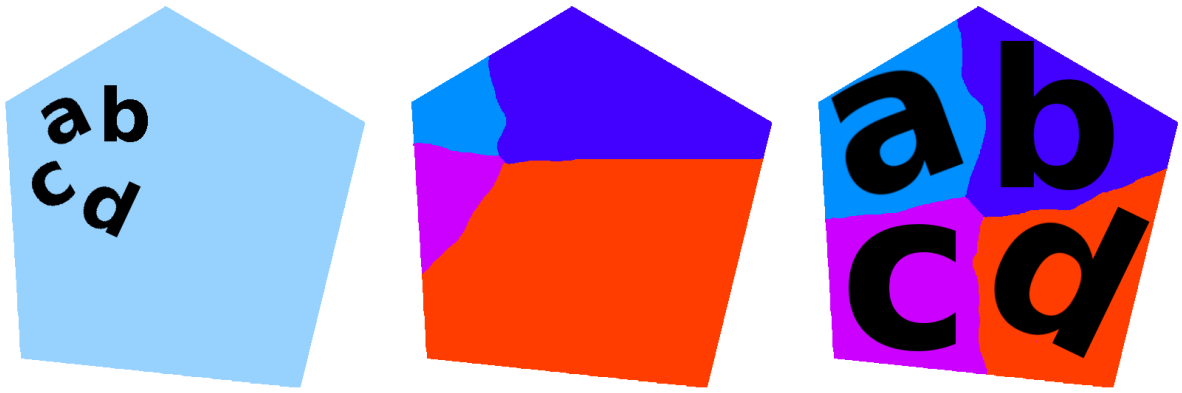
The first step in my system is to define the container region into which I will pack the letter sequence. When drawn as a black shape on a white background, the container should be a clear depiction of an object.

As with papercutting, I first extract a desired foreground object from an image using the Lazy Snapping method of Li *et al.* [70]. I then apply my artistic thresholding technique to produce a bi-level raster image of the container region. The user can control the parameters to select an appropriate container region. This procedure is illustrated in Figure 6.2.

6.3.2 Subdivision

The pixels of the container region must now be partitioned into n subregions $R_1, \ldots, R_n$, where each R_i will be assigned to l_i from the original letter sequence. This step can be seen as a data clustering problem, where pixels form into clusters around letters.

As with other NPR packing techniques, I use an iterative algorithm to refine an initial letter placement into an evenly distributed set of subregions. To begin, the user interactively creates a starting arrangement of n letters inside the container region, as shown in Figure 6.3(a). I then

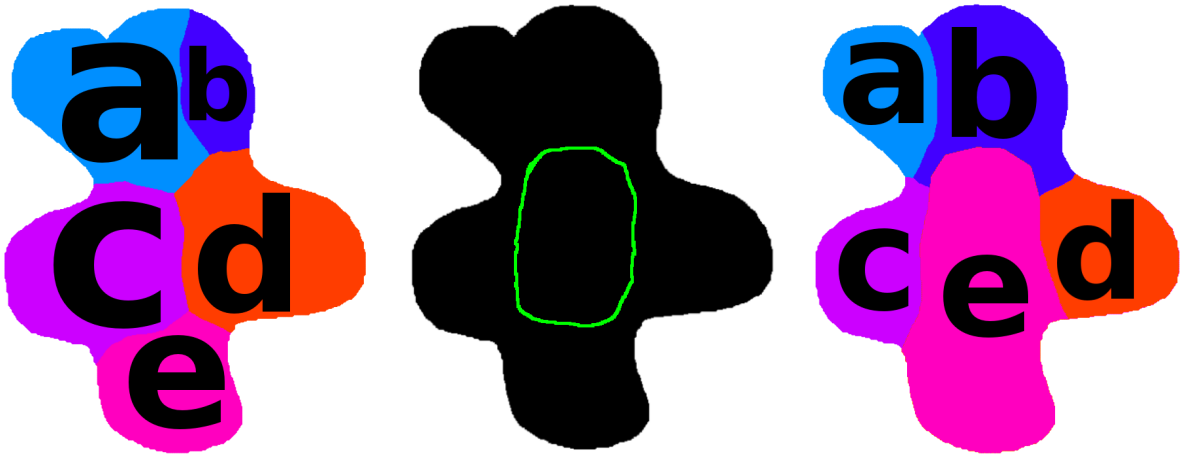


(a) (b) (c)

Figure 6.3: The user provides initial positions and orientations for a set of letters in (a). The initial clustering result is shown in (b). In (c), after iterative relaxation, I obtain an optimized clustering result.

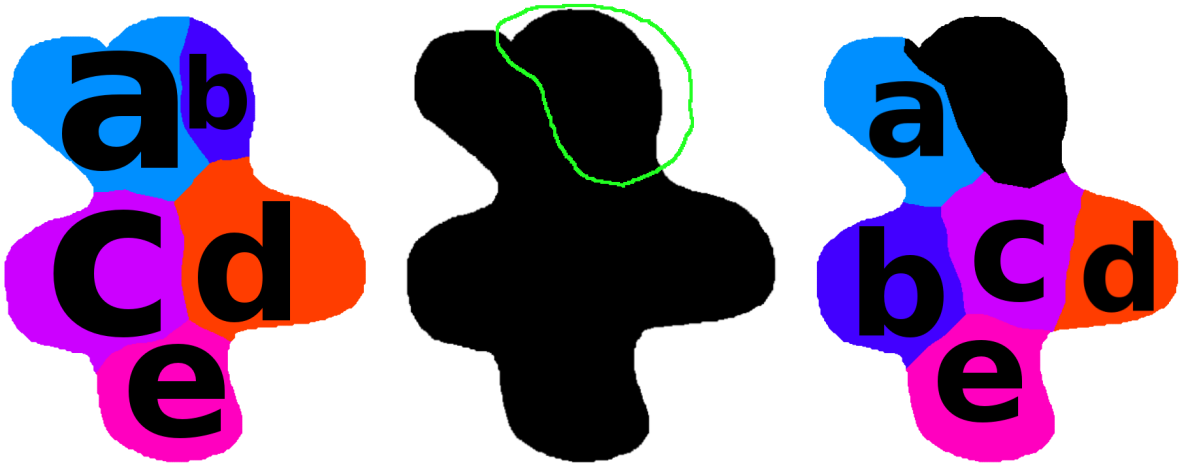
run a level-set algorithm to grow subregions associated with the letters [118], which is similar to variational shape approximation [17]. (The use of a level-set algorithm helps ensure that letters are contained in their subregions, and that the subregions are connected.) I initialize n clusters to the rasterized outlines of the letters. Then, while there exist unclaimed pixels, I iterate over the clusters. Each cluster claims the pixels adjacent to its boundary. The subregion boundaries will grow at a roughly constant rate until they encounter one another and consume the container. The resulting subdivision may be uneven, as in Figure 6.3(b). I refine it by moving each letter to the centroid of its cluster, in the style of the modified Lloyd's method of Hiller *et al.* [47]. Each letter is scaled so that it fits more snugly into its cluster. The subdivision process then repeats until the user is satisfied with the locations and orientations of the letters. Figure 6.3(c) shows an example of a final set of subregions produced by this process.

I provide the user with two additional interactive tools to modify the behaviour of the subdivision step. First, they can sketch closed curves containing pixels that must all belong to the same subregion. A demonstration is given in Figure 6.4. This tool can be used to force an area of the container to belong to a single letter when the subdivision process is trying to pull it apart. It was used in Figure 6.11 to ensure that the mouth would be depicted by a single subregion



(a) (b) (c)

Figure 6.4: The forced clustering of pixels. The clustering algorithm produced the result in (a). The user drew the green curve in (b). The pixels inside the curve are forced to belong to a single cluster in (c).



(a) (b) (c)

Figure 6.5: The forced exclusion of pixels from the subdivision process. The user sketched the green curve in (b). Its contents do not participate in the subdivision in (c).

containing the letter O. Second, the user can sketch a closed curve that excludes a portion of the container from the subdivision, as shown in Figure 6.5. In Figure 6.16(b), the propellers of the airplane were simply drawn as black shapes rather than as awkward protrusions on the letter D.

At this point, the subdivision process has produced a partition of the container region into sets of pixels. I must now convert these subregions into paths that can serve as geometric targets for image warping. I smooth the boundaries of the subregions by applying a few iterations of the OPEN and CLOSE morphological operations. I finish by applying OPEN three more times to shrink the subregion boundaries and create space between them. A library such as Potrace [117] can then convert the rasterized subregions into vector paths. If desired, the user can edit the paths manually before proceeding with the warping step. A set of subregions is shown in Figure 6.10(b).

6.3.3 Warping

Now that I have the container subregions, I must warp the corresponding glyphs into them. Warping and morphing are well studied problems in computer graphics [36]. Unfortunately, in most cases the user must provide an explicit correspondence between the source and target shapes. I would like the computer to derive this correspondence automatically based on the quality of the warp it produces. I provide a geometric warping technique that incorporates an energy function based on shape matching to measure the success of the warp. I can then try a large set of possible correspondences and choose the one that minimizes the energy function. I discuss the warp in this subsection, and the energy function in the next one.

Denote by C_i the convex hull of glyph g_i, which will be warped into subregion R_i. An example is given in Figure 6.6(a). I will define a warp from C_i to R_i, and apply it to the glyph. To begin, I place the same number of sample points evenly around both C_i and R_i (I have found that 100 points is an adequate trade-off between efficiency and accuracy). Any given assignment of a point on C_i to a point on R_i induces a correspondence between the two shapes.

Previous research by Hormann and Floater [49] and Ju *et al.* [58] has shown how to warp between two arbitrary simple polygons, based on a generalized notion of barycentric coordinates [49, 58]. I found that their technique performs well when the source and target polygons are convex, but produces unacceptable distortion in general. While C_i is convex by definition, R_i need not be. Therefore, I partition R_i into convex pieces using the approximation algorithm of

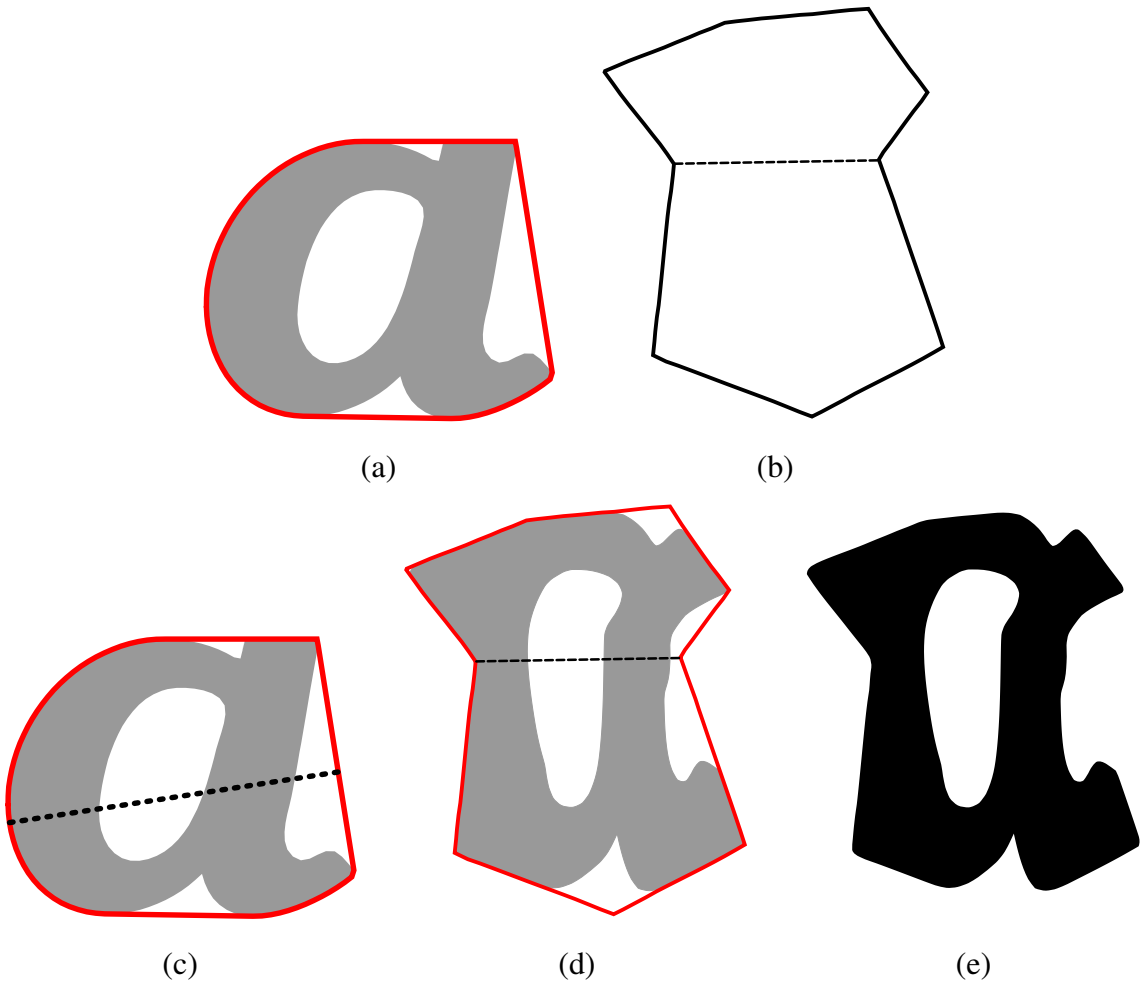


Figure 6.6: A visualization of the warping process. A source letter and its convex hull are shown in (a). A container subregion is shown in (b), partitioned into convex pieces. The corresponding subdivision is applied to the convex hull in (c), and the letter is warped piece-by-piece into the convex partition in (d). The finished warped letter is shown in (e).

Greene [40]. Figure 6.6(b) shows an example of a partitioned subregion. Given a correspondence between the sample points on C_i and R_i, I can then map this partition back through the correspondence and create a convex partition of C_i, shown in Figure 6.6(c). Finally, I apply the geometric warp of Hormann and Floater [49] inside each convex piece to obtain a warp of the entire glyph. The outlines of the glyph are approximated by piecewise-linear paths with m ver-

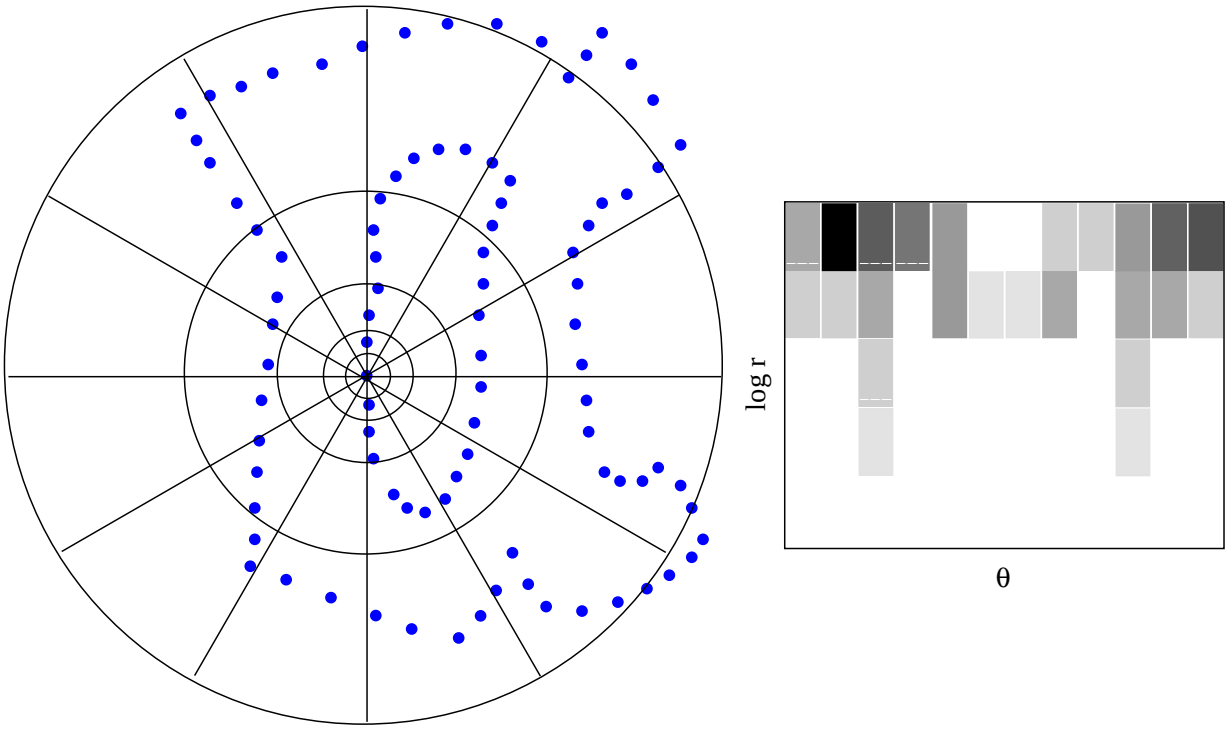


Figure 6.7: A visualization of the shape context for a single reference point on a warped letter "a". The plane is divided into log-polar regions on the left, and the sample points are collected into histogram bins on the right.

tices $\{v_1, \ldots, v_m\}$, and the glyph is warped by computing new positions $\{w_1, \ldots, w_m\}$ for each of the vertices in those paths. A finished example is given in Figure 6.6(e).

6.3.4 Shape Matching

As was mentioned in the previous section, if I fix a sample point on C_i and assign it in turn to each sample point on R_i, I get a sequence of correspondences between the two shapes. To decide which is best, I need a measurement that evaluates the quality of the warp resulting from a given correspondence. My measurement is based primarily on geometric similarity between the warped glyph and the original. I also penalize warps that rotate the glyph too much or fail to

cost=0.328 cost=0.369 cost=0.371 cost=0.429 cost=0.512

Figure 6.8: The costs of warping letters. Smaller values indicate a closer match to the original letter of Figure 6.6.

fill the container subregion.

The measurement of similarity between two shapes is an important and active area of research in computer vision. I found that the "shape context" method of Belongie *et al.* [7] is a good fit to my needs. In this method, the shapes to be compared can have arbitrary topology (unlike, for example, the method of Arkin *et al.* [3]). They need only have the same number of sample points, which mine do by construction.

A shape context for a single reference point is a log-polar histogram of the locations of the other points in the shape. The histogram is discretized by logarithmic distance from the reference point and by angle. An example is given in Figure 6.7.

Belongie *et al.*explain how to compute $\delta(v_i, w_j)$, the distance between the histograms associated with reference points v_i and w_j on the original and warped copies of the glyph. I can then define a measurement of geometric similarity as $C_g = \frac{1}{m}\sum_{i=1}^{m}\delta(v_i, w_i)$. Note that the original use of shape contexts minimized over all pairs of reference points; here, I exploit the known correspondence between the v_i and w_i.

To preserve readability, it is important to avoid rotating glyphs too much. In the worst case, a rotated letter may turn into a different letter entirely, as an "m" becomes a "w" under a half-turn. Therefore, I define a penalty C_o based on the effect of the warp on the glyph's orientation. I use a least-squares estimate [132] of the rigid motion between the v_i and w_i to compute the change θ in orientation induced by the warp. I then define $C_o = \theta/\pi, \theta \in [0, \pi]$.

I would also like to force warped glyphs to occupy as much of their container subregions as possible. I introduce an additional penalty based on area coverage: $C_a = 1 - A_w/A_r$, where A_w is

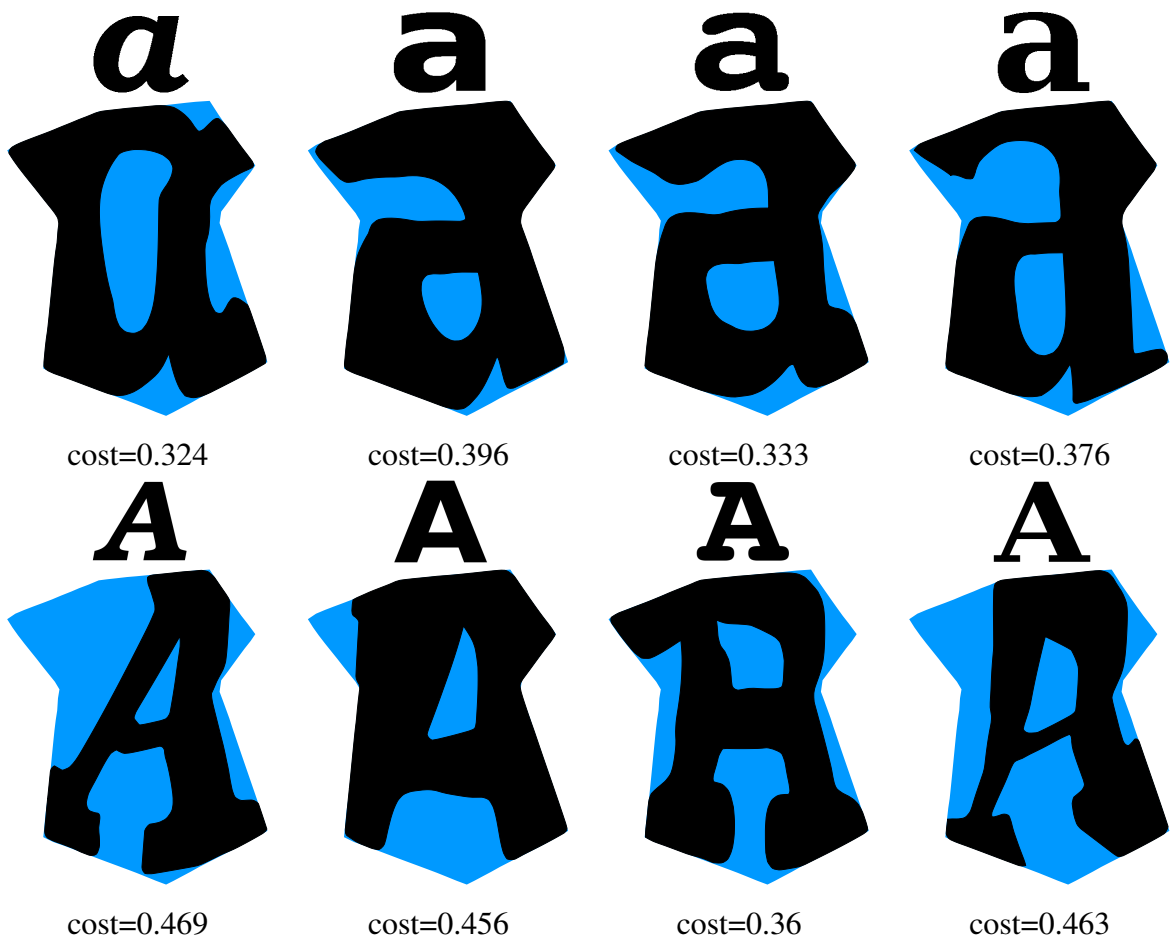


Figure 6.9: The best-fit costs of lowercase and uppercase versions of the letter "a" for four different typefaces. The original letters are shown on the top, and are packed into the blue container subregions.

the area of the warped glyph and A_r is the area of its container subregion.

Finally, I define the overall quality of a matching as $C_{total} = w_g C_g + w_o C_o + w_a C_a$. The values w_g, w_o and w_a are weights chosen to trade off between the relative importance of these three terms. In practice, I use $w_g = 0.6, w_o = 0.2$ and $w_a = 0.2$. Figure 6.8 shows some warped letters and their associated costs. I compute C_{total} for every possible correspondence between C_i and R_i, and choose the warp that produces the lowest cost.

I extend this cost function in a simple but important way by trying multiple glyphs for each

letter. I iterate over a few different typefaces, and try both lowercase and uppercase versions of letters. Notice, for example, the use of mixed case in Figure 6.17(b). The examples in this chapter (Figure 6.9) were constructed from the four typefaces *URW Bookman*, *Bitstream Vera Sans*, *Courier*, and *DejaVu Serif*. The additional variety of multiple letterforms helps to ensure a better match.

A finished example of calligraphic packing, based on the Mona Lisa, is shown in Figure 6.10.

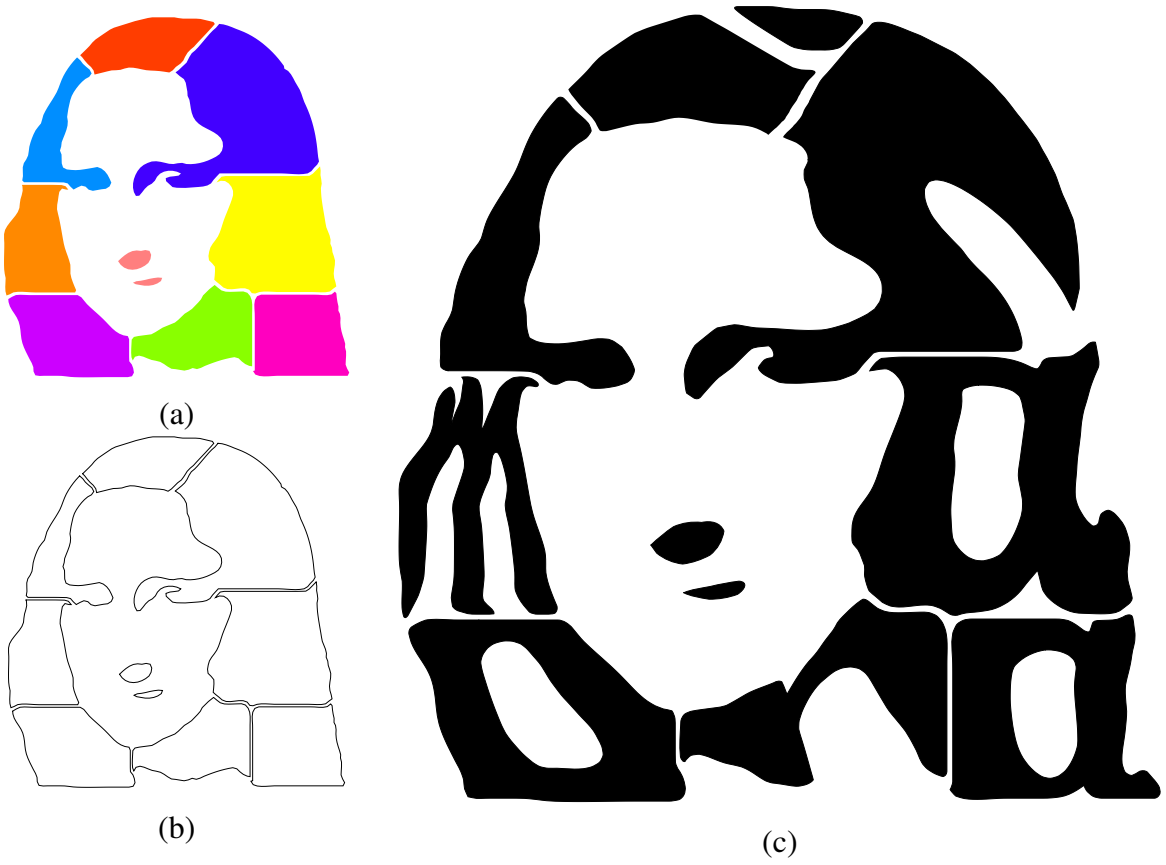


Figure 6.10: A calligraphic packing made from the painting and words "Mona Lisa". After I subdivide the image (a) and extract the outlines (b), my system will produce the finished result in (c). The details of the nose and mouth were excluded from the packing process, and taken directly from the container.

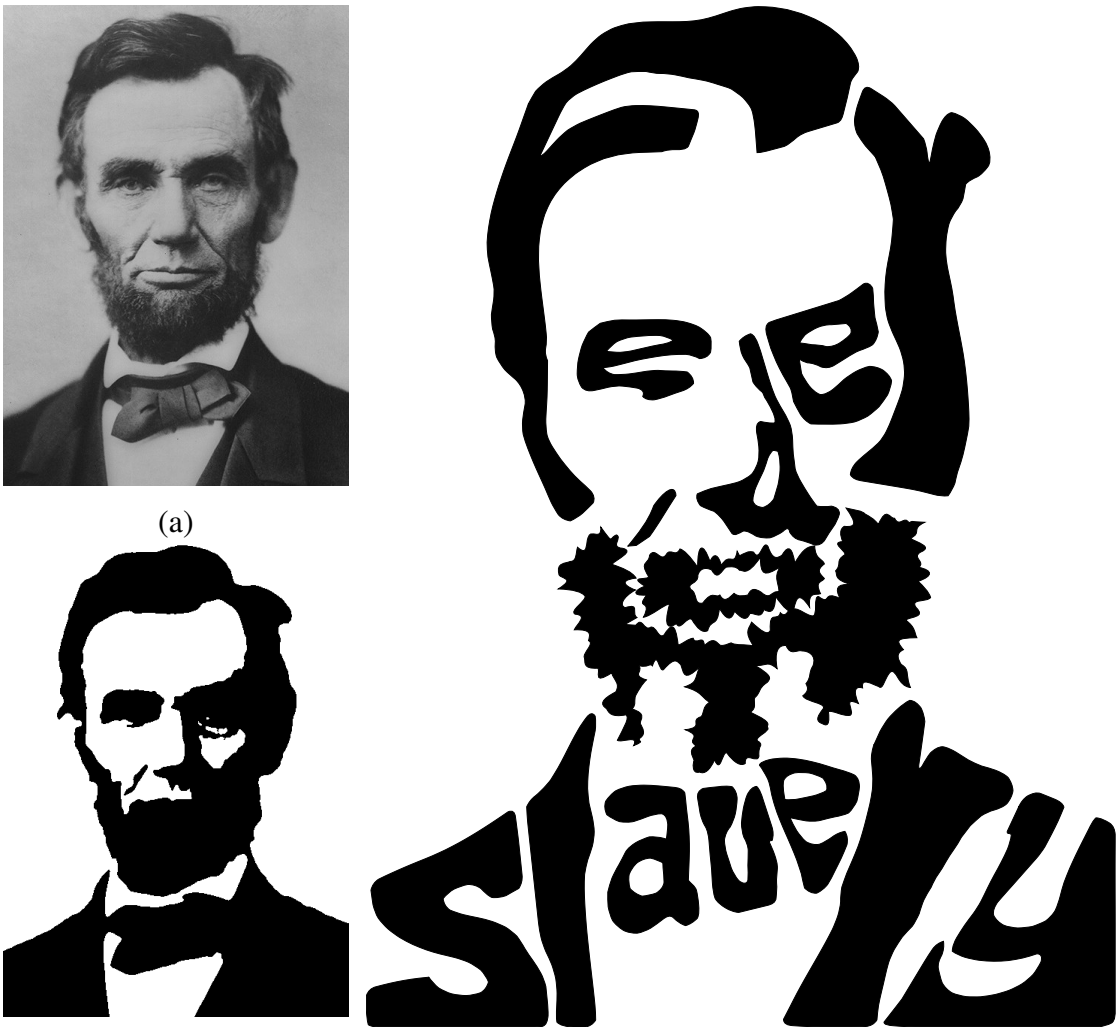
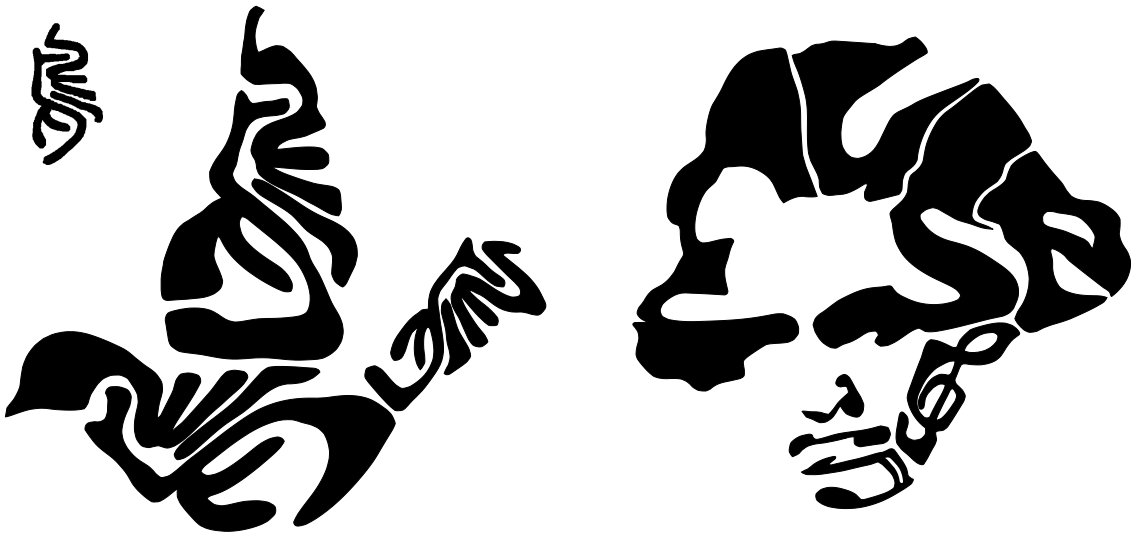


Figure 6.11: Abraham Lincoln (a) was faced with the conflict between "freedom" and "slavery". The container image is shown in (b). In the result (c), I added noise to the outlines of the letters "O" and "M" to give the appearance of Lincoln's facial hair.

6.4 Rendering

I usually render the warped letters as solid black shapes. I have also experimented with a few additional rendering styles to increase the aesthetic appeal of my results.

Figure 6.12: The word “successful” made into a packing of a boat and its reflection.



(a) (b)

Figure 6.13: Three copies of the Chinese character for “bird” are used to fill a bird in (a). The source character is shown on the upper left. “Muse” and some musical notation are used to create a portrait of Ludwig van Beethoven in (b).

Inspired by the rough outlines in Figure 6.1(b), I allow boundaries of letters to be perturbed geometrically before rendering. This style is useful for capturing the look of rough or irregular objects, such as Lincoln's facial hair in Figure 6.11.

Referring to the dimensionality of representation in introduction (Chapter 1), solid colour filled letters are two dimensional regions. I also experiment with a one dimensional representation. I allow the interior of a letter to be filled with strokes instead of a solid colour. The user is able to draw a few sample curves to define the principle stroke direction in a letter. I interpolate from points sampled on those curves using radial basis functions to infer a vector field, in the manner of Litwinowicz [73]. The streamline placement algorithm of Jobard and Lefer [57] draws long, curved strokes that follow the vector field. These strokes give me an opportunity to experiment with texture and tone in my designs. For example, Figure 6.12 shows streamlines used to give the look of a reflection in water and Figure 6.16(a) demonstrates a beard simulated by streamlines.

6.5 Implementation and results

My system is implemented in C++, and generates Postscript output. I use the CGAL library [14] to compute the convex hull and convex partition. I implemented a prototype user interface in Gtkmm that lets the user load in an image, create the binary container image, fill it with letters, and modify the rendering parameters. The automated packing process usually takes from a couple of seconds to a couple of minutes to complete on a standard PC, depending on how many letters are used.

As mentioned throughout the chapter, I provide a wide range of interactive tools that permit the user to intervene in the automatic construction of a packing. They can edit the curves representing the subregions and the shapes of the warped letters. For example, I added some detail to Chaplin's eyes in Figure 6.17(b).

My system is not restricted to the Roman alphabet; it can process more complex symbols. Figure 6.13(a) shows a result in which three copies of the Chinese word "bird" (written in the traditional zhuànwén, or "seal script") are used to fill the outline of a bird. Figure 6.13(b) includes some musical symbols in a portrait of Beethoven.

Some additional results are shown in Figures 6.14, 6.15, 6.16 and 6.17.

(a) (b) (c)

Figure 6.14: A calligraphic packing created by my system. A horse is extracted from a photograph (a) and converted into a container region (b) using artistic thresholding method. The word "horse" is then packed into the container (c).

(a) (b) (c)

Figure 6.15: An elephant example. The source photograph is in (a). The word "elephant" is packed into container region (b) to get the result (c).

(a)

(b)

Figure 6.16: The “Da Vinci Code” packed into a portrait of Leonardo in (a). and an airplane flying “like a bird” in (b).

(a) (b)

Figure 6.17: A “graceful” dancer in (a) and Charlie Chaplin depicted using the words “laugh” and “cry” in (b).

6.6 Discussion

Figure 6.18: Evaluation of my calligraphic packing results. The first row shows works by artist. The corresponding container regions are derived in the second row. My results are shown in the last row. There is no user modification included.

In this chapter, I introduced the calligraphic packing problem and proposed a solution. My system uses a level-set approach to subdivide a container region into subregions, and a best-fit

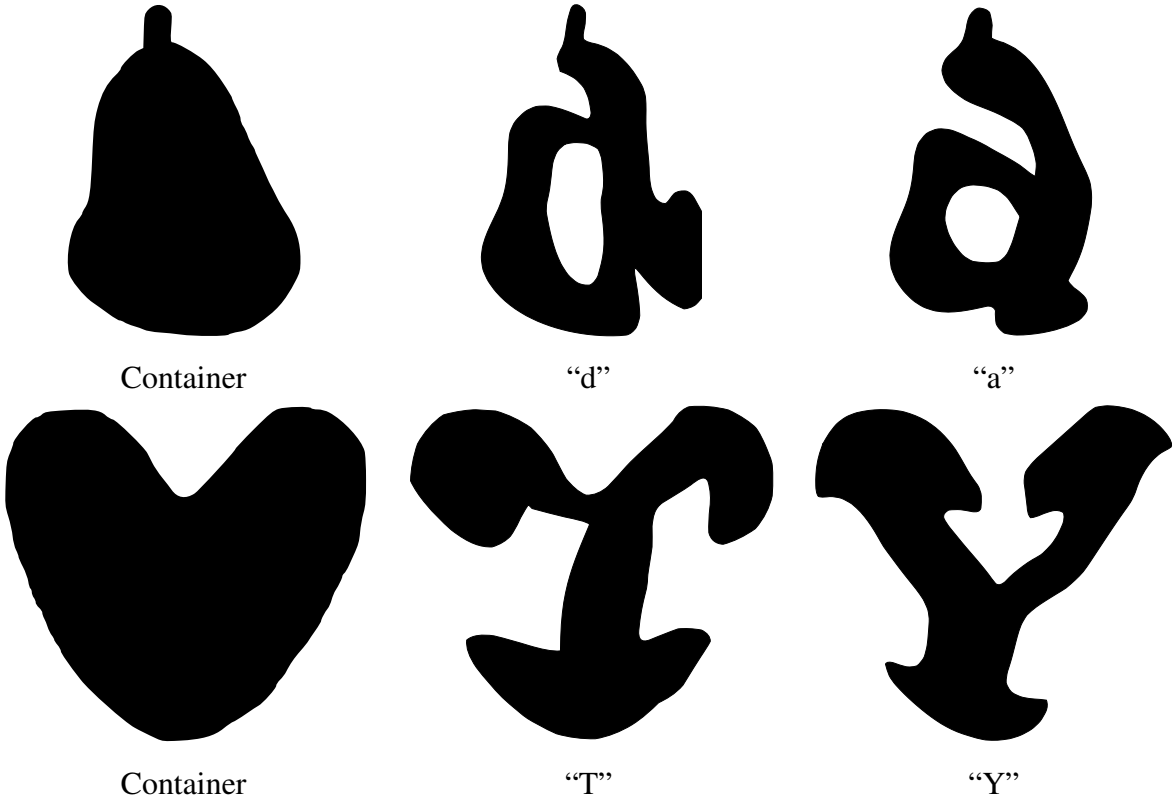


Figure 6.19: Comparisons of similar letters. The first example shows that we can get similar results using letter “d” and “a” to tile a pear shape. The second example shows the results of using letter “T” and “Y” to tile a heart shape.

warping algorithm to map letters into subregions. I also developed some variations in rendering style that provide the user with additional aesthetic opportunities.

I evaluate my results in Figure 6.18. To compare with artists’ works, I derive the container regions from artists’ results, then create my versions in those regions. In my technique, the letters can be warped to fit the contours of shapes pretty well but my system cannot handle the details inside each letter, such as Osama’s eyes. To acquire more appealing results, I provide a set of interactive tools to help user adjust every parts of letters elaborately to depict shape features. However, these tools cannot match the subtlety and stylization achieved by the original designs.

Like other problems in non-photorealistic rendering, calligraphic packing is a challenging

topic because a successful solution relies on facts from both mathematics and human perception. Calligraphy is an especially interesting medium in this regard, because it relies on the cognitive and perceptual mechanisms of reading, about which far too little is understood. Figure 6.19 shows that different letters could be warped to create similar results. These letters might be confused in isolation, though we can distinguish them in the context of whole words depending on high-level cognition.

I note two important directions in which the technique in this chapter can be improved. First, I would like to solve the third requirement of my calligraphic packing problem, namely that the arrangement of glyphs respect the order of the original sequence of letters. I would like to find a way to distribute letters automatically so that the eye is naturally drawn across them in the order in which they are intended to be read. I would have to extend my energy function to account not only for the quality of each individual warp, but for the layout of the glyphs as a whole.

Second, I would like to improve the legibility and fit of the rendered letters. I cannot simply reduce the convex hull to the boundary of the glyph, because some of the negative space adjacent to the letter is essential in identifying it (consider, for example, the spaces separating the arms of an uppercase "E"). However, the convex hull also contains incidental features such as serifs that reduce the quality of the packing. It may be necessary to move to a spine-based model of letter construction, where I pack zero-thickness drawings of letters and expand outward from them to fill the container. A parameterized model of letterforms like that of METAFONT [66] could be used to control the shapes of letters during this process.

In the long run, I am interested in ways that the computer can be used as a tool for calligraphic design. Computers are long established as a powerful technology for typography, but more research needs to be done to achieve the same level of power and flexibility with letterforms that non-photorealistic rendering grants us with images.

Chapter 7

Conclusions and Future Work

Wholetoned images, which depict objects with two high contrast tones, hold tremendous potential to enrich art, decoration, advertisement and entertainment. However, there is little work that specifically focuses on this area. Related research typically aims at simulating typical art styles such as pen-and-ink, engraving and halftoning, which pursue reproduction of continuous image tone with black and white. With minimal tone information, wholetoned images are a higher abstraction that arouses a viewer's interest to interpret them. The work presented in this dissertation unifies and eases the wholetoned image generation process by offering a novel framework and opportunities for extensions. I review my contributions in Section 7.1 and 7.2. In Section 7.3, I describe several areas for future work. Finally, I conclude with some observations in Section 7.4.

7.1 General wholetoning framework

This thesis presents an interactive technique, artistic thresholding, which converts an original colour image to a wholetoned image. It allows the user to control a set of intuitive, high-level parameters. These parameters determine the result of a continuous optimization process. The user can adjust the weights on the fly and get a corresponding result interactively. The desired result can be obtained in an intuitive way. This optimization-based framework is not restricted to one specific style. Driven by different parameters, it can generate a variety of styles and different levels of abstraction.

Artistic thresholding provides more powerful and flexible control of wholetoned image gen-

eration than other image thresholding methods. The output satisfies my definition of wholetoning in Chapter 1. Only two tones appear in result images. Artistic thresholding applies image segmentation to do initial shape and texture abstraction. Those two-dimensional segments are the representation elements. Further abstraction can be obtained by adjusting parameters. The user can control the coverage of black area in the image, the faithfulness of tone, the strength of boundary contrast, and the effect of feature homogeneity. With the aid of these parameters, we can achieve trade-offs between tone inversion and tone preservation, between feature homogeneity and heterogeneity, and style variations. My experiments show that this system can generate results with the essential features of wholetoned images. I also explored the combination of lines from edge detection and wholetoned images using inversed tone to highlight boundaries. Depending on user specified high-level features, my technique can generate more convincing feature homogeneity and tone inversion results.

7.2 Extended wholetone styles

This research enables us to make the advantages of wholetoned style available to a wide variety of applications. Starting with the artistic thresholding results, I explored two styles to enhance the complexity and aesthetics.

The first extended style is a computer-generated papercutting technique that provides an approach to simulate papercutting, a traditional folk art. I applied artistic thresholding to generate motifs. Then all parts of a motif were connected to form a valid paper-cut design corresponding the special constraint of papercutting. Following the traditional style of papercutting, I also developed ways to construct other abstractions of shapes such as synthesized stylized patterns, geometric patterns and procedurally generated patterns. These paper-cut patterns are composed of region-based representations (such as artistic thresholding generated motifs) and line-based representations (such as the decorative latticework and procedurally generated patterns). By employing a set of binary operations, any two such paper-cut designs can be composed while preserving connectivity. Compared to artistic thresholding, my papercutting technique makes use of binary operations on layers of paper-cut patterns, which solves the tone inversion issue more intuitively. The results demonstrate that they achieve the goals of wholetoned style and overcome some of the challenges in Section 1.2. My technique efficiently constructs paper-cut

designs that are suitable for outputting to a craft cutter to produce real papercutting.

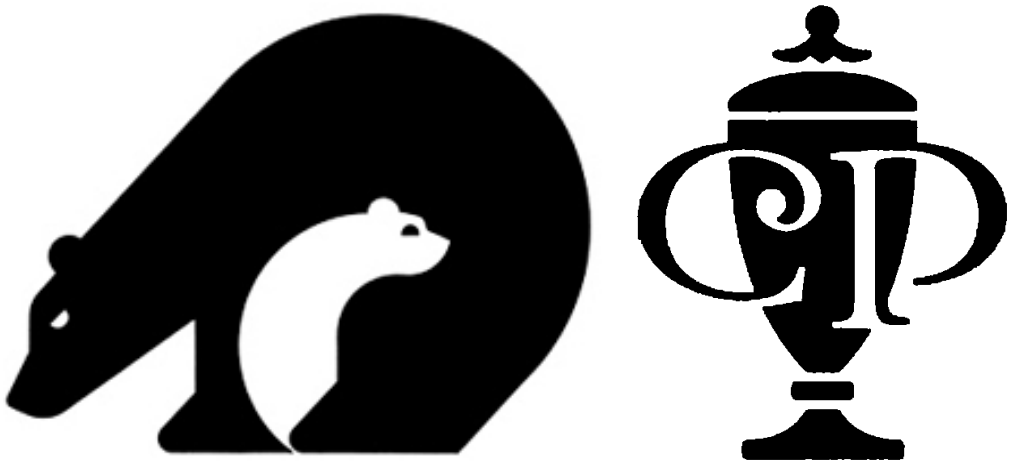
In my second application, I presented a system for calligraphic packing. This technique allows the user to perceive a shape that is composed of a set of letters. It is a semi-automatic approach. I first apply artistic thresholding to create container regions. The user can then set the initial configuration of letters in container regions. After running a clustering algorithm, my system can generate a set of subregions. Each subregion corresponds to a letter. The letters are warped in subregions so that they try to match the shapes of subregions as well as possible while preserving readability. This technique is distinguished from other existing packing algorithms since it formalizes the warping process to a minimization problem by evaluating a cost function that measures the geometric similarity, orientation and area coverage. The results of calligraphic packing satisfy the definition of wholetoned images. Furthermore, using warped letters to depict an image is a specific abstraction approach. I also explored line-based rendering as an alternative to solid filled regions. In calligraphic packing, tone inversion is trivial because there are no overlapping shapes. Calligraphic packing is also related to papercutting. If the stencil letterforms are used, then the result is a valid paper-cut design.

7.3 Future directions

I have mentioned some future work for each wholetoned style in Chapters 4, 5 and 6. I want to discuss other more general future directions here.

Although wholetoned image generation is a simple idea, it is surprising that graphics researchers seem to have overlooked this topic for a long time and there is little previous research about it. I am happy to find that this situation is changing. Some researchers have noticed this research area and observed its extensive applications in practice [9, 88]. Unlike this related work which only focuses on specific styles or applications, I summarize several applications and present a general framework, one that I believe establishes a new independent field of study for NPR. The output images generated by my system have demonstrated the power and effectiveness of my research.

My wholetoning framework could be extended in a number of ways. First, I can find more uses for it. For example, graphic design such as logo design could benefit from it. Another novel use would be in animation and games [116]. My discussion in Section 4.7 would be helpful in



(a) (b)

Figure 7.1: Examples of special aesthetic effects: tone inversion can be applied to depict two embedded animal figures in (a) (California conservation Corps logo by Vanderbyl Design) and tone inversion can be applied to distinguish letters and simulate 3D-like effects in (b) (Crestview Plaza logo by Gardner Design).

these areas.

Other improvements extend the framework's architecture. Future work can tune my framework to adapt it to specific purposes and extend it by importing more constraints. For example, there is no explicit representation of salient lines. The lines in my results are a by-product of image segmentation and very noisy and imprecise. My framework also cannot handle textures. Further research about how to combine texture properties into segmentation and optimization should provide better results. It should be significant to explore "distortion" or "symbolism" in shape. Distortion occurs when there are discrepancies between the representation of an object and the properties that the object actually possesses. Deliberate distortion can produce more absorbing aesthetic objects. Skillful designers can create some appealing aesthetic effects using wholetoned illustration. Figure 7.1(a) shows an example of two animal figures embedded together with inversed tones. Applying tone inversion, it even can produce 3D-like effects as shown in Figure 7.1(b). It is an interesting research direction to simulate those effects.

As discussed in Chapter 1, tone inversion is unacceptable for images in which low spatial frequencies dominate perception while it has little effect when high spatial frequencies dominate.

I observed that images of human figures are typically dominated by low spatial frequencies, perhaps because of the importance of lighting and shadows. It could be helpful to apply computer vision techniques such as face recognition algorithms to distinguish different frequency dominant regions, and adjust optimization weights accordingly in those regions.

It is also possible to exploit machine learning techniques. The system might be able to "learn" a specific style by training it with user provided examples.

I also look forward to extending my framework to multi-colour illustrations so that I can deal with colourful styles.

7.4 Observations

I conclude with a few observations. Currently the criterion for evaluating my results is that it is right when it looks good. This is acceptable for the prototype algorithms presented in this thesis. But some more novel and sophisticated approaches should be developed. We should work with artists to find out how they do a design and what are their considerations. my systems should generate results according to each artist's intention and distinctive preference.

In Chapter 1, I presented the definition of wholetoning. My qualitative definition is based only on the visual appearance of images. I would like to explore the inherent relationships between wholetoned images. I want to know whether there exist some fundamental computational and psychological properties that distinguish wholetoning from other styles. There are many possible tools that could be used to answer these questions, such as algorithmic complexity, information theory, group theory and geometric analysis. I did several experiments with the algorithmic complexity of black-and-white illustration styles. First, a black-and-white image is converted into a one-dimensional binary sequence in row-major or column-major order. Then I compute an approximate Kolmogorov Complexity measure of this sequence following an explicit algorithm by Kaspar and Schuster [64]. In my experiments, I choose eighteen styles and include 10 sample images from each. These styles are Mooney faces, Calligraphic Packing, Artistic Thresholding, Icon, Seal-cut, Ink Painting, Paper-cut, Wood-cut, ASCII art, Etching, Line Drawing, Comics, Dithering, TSP Art, Engraving, Pen-and-Ink, Maze Illustration and Stippling. As mentioned in Chapter 1, the first nine styles can be classified into wholetoning. Since wholetoning is an abstract representation, I anticipate that wholetoned images should have lower complexity than

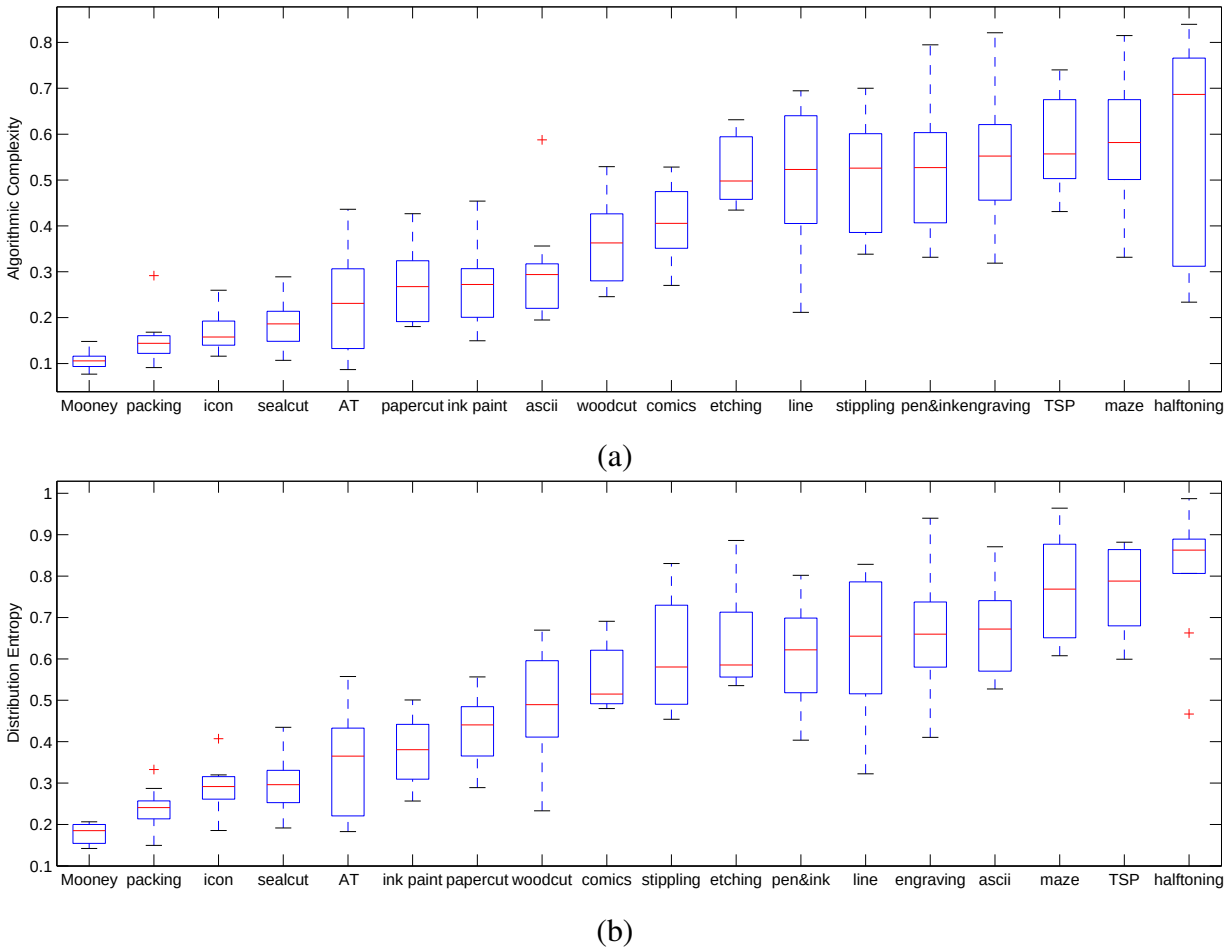


Figure 7.2: The algorithmic complexity of eighteen black-and-white visual art styles is in (a). Their distribution entropy measurements are in (b). They are sorted in increasing order of median values.

other styles. Figure 7.2(a) demonstrates a boxplot of the result. In this figure, a smaller value corresponds to lower complexity. I observe that approximate Kolmogorov Complexity is doing a decent job of sorting images according to an intuitive measure of abstraction or complexity. Most wholetoned styles have lower complexity than non-wholetoned styles, except that ASCII art has low complexity. The reason is that ASCII art applies a small number of repeated patterns which decreases the complexity. To verify the dimensionality of representation, I compute

the distribution entropy of each image, as shown in Figure 7.2(b). Higher distribution entropy means that black and white pixels distribute more evenly. Because wholetoning images emphasize regions, black and white pixels are unevenly distributed. Therefore, they should have lower distribution entropy value, which is clearly verified in Figure 7.2(b). These experiments provide interesting views to my research. These measurements are necessary, but not sufficient, for the creation of wholetoned images. We cannot expect to produce art simply by computing random images whose measurements are comparable to those of wholetoned illustrations. Furthermore, there exist some depiction styles, such as ASCII art that satisfy these measurements but which should not be classified as wholetoned images subjectively. I hope to see future research explore other metrics and discover more hidden facts among different visual arts.

Appendix A

Screenshots of wholetoning systems

Figures A.1, A.2, and A.3 show the interfaces of my artistic thresholding system, papercutting system and calligraphic packing system respectively.

In the interface of artistic thresholding (Figure A.1), the widgets in the top right panel are used for image segmentation and help the user to specify high-level features. In the next panel, the region size slider sets the size of the random subgraphs used during optimization (Section 4.3). The sliders labeled w_color, w_area, w_alike, $w\_opposite$, w_number, $w\_component$, $w\_neighbor$, and w_group control the corresponding weights in the energy function, as described in Section 4.2. The buttons below that help the user to operate the image generation procedure. They can restart the optimization, pause current processing and output a vector image file. The *areaT* and $edgeT$ fields define the threshold values for morphological operators. The $edgethresh$ defines the threshold for superimposing salient edges. The T slider shows the current temperature of simulated annealing optimization. The user can adjust it interactively to control the degree of random perturbation.

In the main window of papercutting system (Figure A.2), the first widgets provide parameters for image processing. The $front$ and *back* colour buttons set the colours of the paper-cut design and background. The next set of widgets is used to in the generation of stylized geometric patterns. In the design window, the user can rotate, scale and reflect paper-cut designs. In the operator window, the user can load candidate designs and apply the composition operators to generate new designs.

In the interface of the calligraphic packing system (Figure A.3), the topmost controls allow

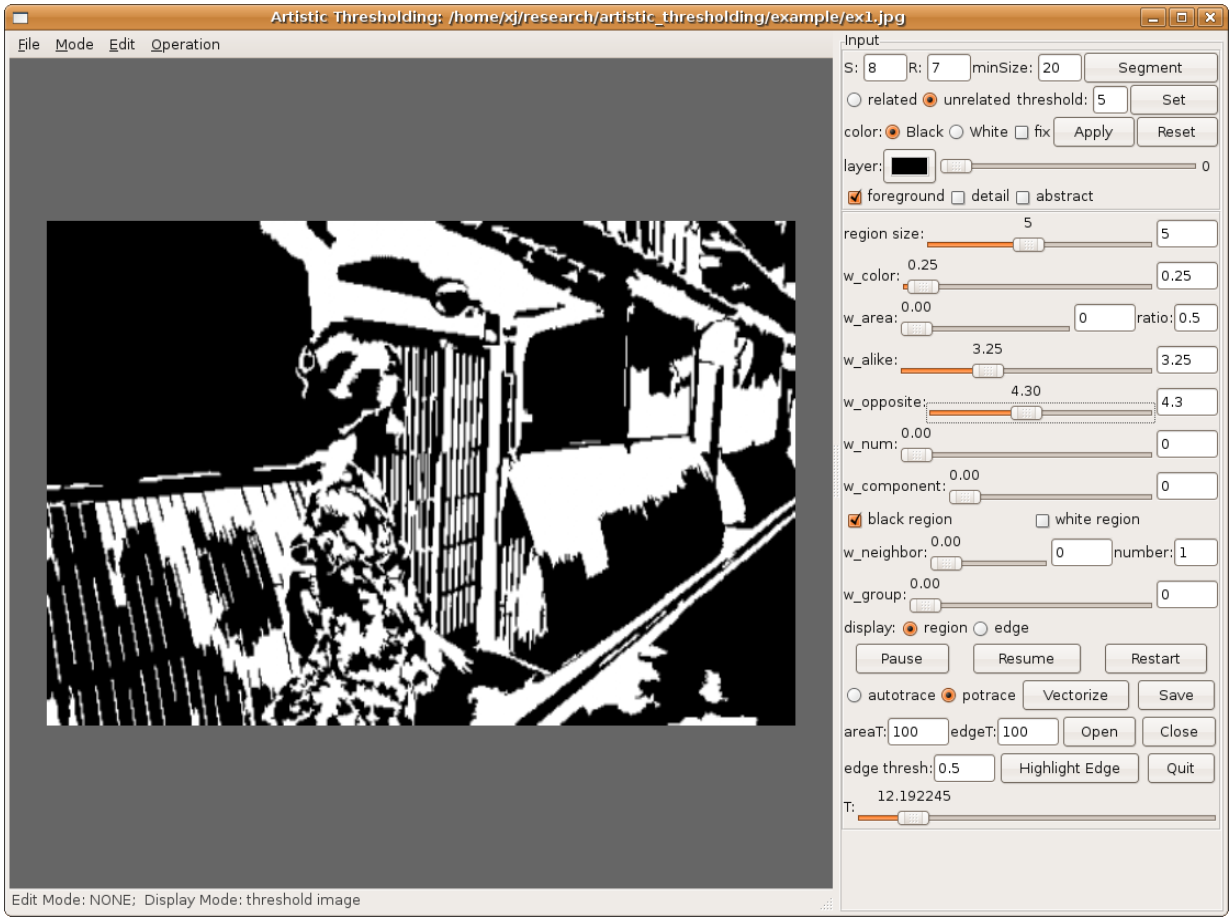


Figure A.1: Screenshots of the artistic thresholding system. By setting parameters interactively, the user can see the result instantly.

the user to set the initial arrangement of letters, cluster pixels and generate subregions. *Mixletter* indicates whether the system should consider both lowercase and uppercase glyphs. *oriweight* sets the weight of the orientation cost, and *areaweight* sets the weight of the area cost (Section 6.3.4). The controls in the "Setting" panel help the user set and review one specific warped result. The "edit" panel helps the user to manipulate letter details.

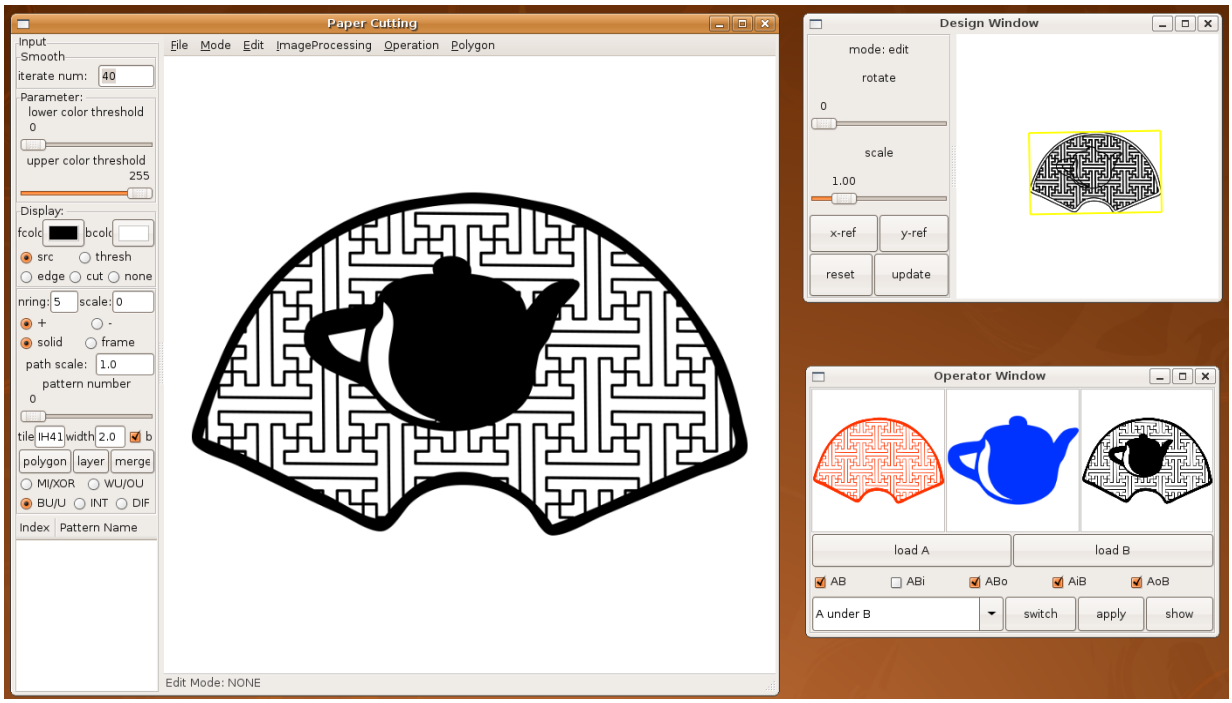


Figure A.2: A screenshot of papercutting system. The interface is composed of three windows: the main window, the design window and the operator window. In the main window, the user can set control parameters and review the results. In the design window, the user can design stylized patterns. In the operator window, the user can compose multiple designs by applying predefined operators.

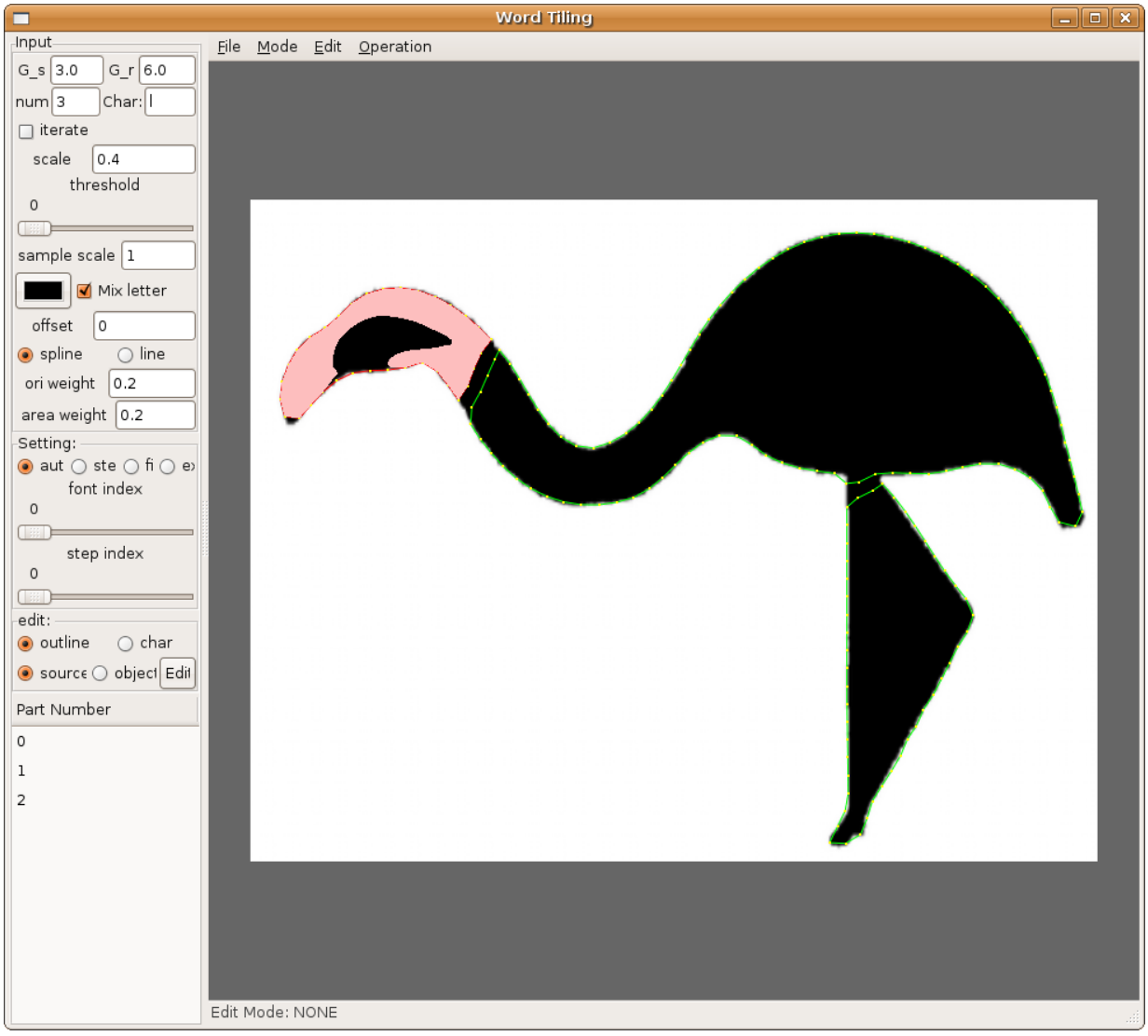


Figure A.3: A screenshot of the calligraphic packing system. My system iteratively warps letters and selects the best results.

Bibliography

[1] Maneesh Agrawala and Chris Stolte. Rendering effective route maps: improving usability through generalization. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 241–249, New York, NY, USA, 2001. ACM.

[2] James F. Allen. Mixed initiative interaction. *Proc. IEEE Intelligent Systems*, 14(5):14–16, 1999.

[3] Esther M. Arkin, L. Paul Chew, Daniel P. Huttenlocher, Klara Kedem, and Joseph S. B. Mitchell. An efficiently computable metric for comparing polygonal shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(3):209–216, 1991.

[4] Elif Ayiter. The history of visual communication. `http://www.citrinitas.com/history_of_viscom/index.html`, 2008.

[5] Adrian Barbu and Song-Chun Zhu. Generalizing Swendsen-Wang to sampling arbitrary posterior probabilities. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(8):1239–1253, 2005.

[6] Pascal Barla, Simon Breslav, Joëlle Thollot, François Sillion, and Lee Markosian. Stroke pattern analysis and synthesis. In *Computer Graphics Forum (Proc. of Eurographics 2006)*, volume 25, 2006.

[7] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *Pattern Analysis and Machine Intelligence, IEEE Transaction on*, 24(4):509–522, 2002.

[8] Member-Yuri Boykov and Member-Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(9):1124–1137, 2004.

[9] Jonathan Bronson, Penny Rheingans, and Marc Olano. Semi-automatic stencil creation through error minimization. In *NPAR '08: Proceedings of the 6th international symposium on Non-photorealistic animation and rendering*, pages 31–37, New York, NY, USA, 2008. ACM.

[10] S. Brooks. Image-based stained glass. *Visualization and Computer Graphics, IEEE Transactions on*, 12(6):1547–1558, Nov.-Dec. 2006.

[11] J Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, 1986.

[12] Patrick Cavanagh. What's up in top-down processing? In *A. Gorea (ed.), Representations of Vision: Trends and tacit assumptions in visio research*, pages 295–304. Cambridge University Press, 1991.

[13] Patrick Cavanagh. Top-down processing in vision. In *MIT Encyclopedia of Cognitive Science*, pages 844–845. MIT Press, 1999.

[14] CGAL Editorial Board. *CGAL-3.2 User and Reference Manual*, 2007. `http://www.cgal.org`.

[15] Hui Chen. *Black-and-white, pattern and art*. Anhui Fine Arts Publishing House, Hefei, Anhui, China, 1999.

[16] Christopher M. Christoudias. Synergism in low level vision. In *ICPR '02: Proceedings of the 16 th International Conference on Pattern Recognition (ICPR'02) Volume 4*, page 40150, Washington, DC, USA, 2002. IEEE Computer Society. EDISON code available at `http://www.caip.rutgers.edu/riul/research/code/EDISON/`.

[17] David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. Variational shape approximation. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 905–914, New York, NY, USA, 2004. ACM.

[18] Forrester Cole, Aleksey Golovinskiy, Alex Limpaecher, Heather Stoddart Barros, Adam Finkelstein, Thomas Funkhouser, and Szymon Rusinkiewicz. Where do people draw lines? *ACM Trans. Graph.*, 27(3):1–11, 2008.

[19] J. P. Collomosse and P. M. Hall. *Genetic paint: a search for salient paintings*, volume 3449 of *Lecture Notes in Computer Science (Proc. EvoMUSART)*, pages 437–447. Springer-Verlag, 2005.

[20] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[21] Cassidy J. Curtis. Loose and sketchy animation. In *SIGGRAPH '98: ACM SIGGRAPH 98 Electronic art and animation catalog*, page 145, New York, NY, USA, 1998. ACM.

[22] Ketan Dalal, Allison Klein, Yunjun Liu, and Kaleigh Smith. A spectral approach to npr packing. In *NPAR '06: Proceedings of the 4th international symposium on Non-photorealistic animation and rendering*, pages 71–78, 2006.

[23] Doug DeCarlo, Adam Finkelstein, Szymon Rusinkiewicz, and Anthony Santella. Suggestive contours for conveying shape. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 848–855, New York, NY, USA, 2003. ACM.

[24] Doug DeCarlo and Anthony Santella. Stylization and abstraction of photographs. *ACM Trans. Graph.*, 21(3):769–776, 2002.

[25] Philippe Decaudin. Cartoon looking rendering of 3D scenes. Research Report 2919, INRIA, June 1996.

[26] Erik D. Demaine, Martin L. Demaine, and Anna Lubiw. Folding and one straight cut suffice. In *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'99)*, pages 891–892, Baltimore, Maryland, January 17–19 1999.

[27] Oliver Deussen, Stefan Hiller, Cornelius van Overveld, and Thomas Strothotte. Floating points: A method for computing stipple drawings. *Computer Graphics Forum*, 19(3):40–51, 2000.

[28] Debra Dooley and Michael F. Cohen. Automatic illustration of 3d geometric models: lines. In *SI3D '90: Proceedings of the 1990 symposium on Interactive 3D graphics*, pages 77–82, New York, NY, USA, 1990. ACM.

[29] Gershon Elber. Line art illustrations of parametric and implicit forms. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):71–81, 1998.

[30] Gershon Elber. Line art rendering via a coverage of isoparametric curves. *IEEE Transactions on Visualization and Computer Graphics*, 1(3):231–239, Sep 1995.

[31] Gershon Elber and George Wolberg. Rendering traditional mosaics. *Visual Computer*, 19:67–78, 2003.

[32] M. C. Escher. *Escher on Escher: Exploring the Infinite*. Henry N. Abrams, Inc., 1989. Translated by Karin Ford.

[33] Yang Gang. *Black and white decorative painting*. Henan Medical University publishing house, Zhengzhou, China, 2004.

[34] Alan Gilchrist. *Seeing Black and White*. Oxford University Press, USA, 2006.

[35] Andrew Glassner. Interactive pop-up card design, part 1. *IEEE Computer Graphics and Applications*, 22(1):79–86, 2002.

[36] Jonas Gomes, Lucia Darsa, Bruno Costa, and Luiz Velho. *Warping and Morphing of Graphical Objects*. Morgan Kaufmann, 1999.

[37] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Prentice Hall, Upper Saddle River NJ, 2002.

[38] Bruce Gooch, Erik Reinhard, and Amy Gooch. Human facial illustrations: Creation and psychophysical evaluation. *ACM Trans. Graph.*, 23(1):27–44, 2004.

[39] Todd Goodwin, Ian Vollick, and Aaron Hertzmann. Isophote distance: a shading approach to artistic stroke thickness. In *NPAR '07: Proceedings of the 5th international symposium on Non-photorealistic animation and rendering*, pages 53–62, New York, NY, USA, 2007. ACM.

[40] D. Greene. The decomposition of polygons into convex parts. In *In Advances in Computing Research*, pages 235–259. JAI Press, 1983.

[41] Branko Grünbaum and G C Shephard. *Tilings and patterns*. W. H. Freeman & Co., New York, NY, USA, 1986.

[42] George W. Hart. Modular kirigami. In *Proceedings of Bridges 2007*, 2007. See also `http://www.georgehart.com/kirigami/Hart-Modular-Kirigami.html`.

[43] Alejo Hausner. Simulating decorative mosaics. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 573–580, New York, NY, USA, 2001. ACM Press.

[44] Aaron Hertzmann. Introduction to 3D Non-Photorealistic Rendering: Silhouettes and Outlines. In Stuart Green, editor, *ACM SIGGRAPH 99 Course Notes. Course on Non-Photorelistic Rendering*, chapter 7. ACM Press/ACM SIGGRAPH, New York, 1999.

[45] Aaron Hertzmann. Paint by relaxation. In *CGI '01: Proceedings of the International Conference on Computer Graphics*, page 47, Washington, DC, USA, 2001. IEEE Computer Society.

[46] Aaron Hertzmann and Denis Zorin. Illustrating smooth surfaces. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 517–526, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[47] Stefan Hiller, Heino Hellwig, and Oliver Deussen. Beyond stippling - methods for distributing objects on the plane. *Computer Graphics Forum*, 22(3):515–522, 2003.

[48] Henry Horenstein. *Black & white photography: a basic manual*. Little, Brown and Company, 1983.

[49] Kai Hormann and Michael S. Floater. Mean value coordinates for arbitrary planar polygons. *ACM Trans. Graph.*, 25(4):1424–1441, 2006.

[50] Steven L. Horowitz and Theodosios Pavlidis. Picture segmentation by a tree traversal algorithm. *J. ACM*, 23(2):368–388, 1976.

[51] Siu Chi Hsu and Irene H. H. Lee. Drawing and animation using skeletal strokes. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 109–118, New York, NY, USA, 1994. ACM Press.

[52] Peter J. Huber. *Robust Statistical Procedures, 2nd edition*. Society for Industrial Mathematics, 1987.

[53] Tobias Isenberg, Petra Neumann, Sheelagh Carpendale, Mario Costa Sousa, and Joaquim A. Jorge. Aesthetics of hand-drawn and computer-generated illustrations. In Bruce Gooch, Lszl Neumann, Werner Purgathofer, and Mateu Sbert Casasayas, editors, *Dagstuhl Seminar 06221 on Computational Aesthetics in Graphics, Visualization and Imaging*. 2006.

[54] Tobias Isenberg, Petra Neumann, Sheelagh Carpendale, Mario Costa Sousa, and Joaquim A. Jorge. Non-Photorealistic Rendering in Context: An Observational Study. In Doug DeCarlo and Lee Markosian, editors, *NPAR2006*, pages 115–126, New York, 2006. ACM Press.

[55] Ramona Jablonski. *The Paper Cut-Out Design Book*. Stemmer House Publishers, Inc., Owings Mills, Maryland, 1976.

[56] Bernd Jähne. *Digital Image Processing, 6th revised and exthended edition*. Springer, 1997.

[57] Bruno Jobard and Wilfrid Lefer. Creating evenly-spaced streamlines of arbitrary density. In *Visualization in Scientific Computing '97. Proceedings of the Eurographics Workshop in Boulogne-sur-Mer, France*, pages 43–56. Springer Verlag, 1997.

[58] Tao Ju, Scott Schaefer, and Joe Warren. Mean value coordinates for closed triangular meshes. *ACM Trans. Graph.*, 24(3):561–566, 2005.

[59] Wassily Kandinsky. *Point and Line to Plane*. Van Nostrand Reinhold, New York, 1947.

[60] Henry Kang, Seungyong Lee, and Charles K. Chui. Coherent line drawing. In *NPAR '07: Proceedings of the 5th international symposium on Non-photorealistic animation and rendering*, pages 43–50, New York, NY, USA, 2007. ACM.

[61] Craig S. Kaplan and Rober Bosch. TSP art. In *Bridges 2005: Mathematical Connections in Art, Music and Science*, pages 301–308, 2005.

[62] Craig S. Kaplan and David H. Salesin. Escherization. In Kurt Akeley, editor, *Siggraph 2000*, pages 499–510. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[63] Craig S. Kaplan and David H. Salesin. Islamic star patterns in absolute geometry. *ACM Trans. Graph.*, 23(2):97–119, 2004.

[64] F. Kaspar and H. G. Schuster. Easily calculable measure for the complexity of spatiotemporal patterns. *Phys. Rev. A*, 36(2):842–848, Jul 1987.

[65] Junhwan Kim and Fabio Pellacini. Jigsaw image mosaics. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 657–664, New York, NY, USA, 2002. ACM.

[66] Donald E. Knuth. *The METAFONT book*. Addison-Wesley, 1986.

[67] Wolfgang Kohler. *Gestalt Psychology: An introduction to new concepts in modern psychology*. Liveright Publishing Corporation, 1992.

[68] Yunjin Lee, Lee Markosian, Seungyong Lee, and John F. Hughes. Line drawings via abstracted shading. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 18, New York, NY, USA, 2007. ACM.

[69] Yan Li, Jinhui Yu, Kwan-Liu Ma, and Jiaoying Shi. 3D paper-cut modeling and animation. *The Journal of Computer Animation and Virtual Worlds*, 18:to appear, 2007. (Also published in Proceedings of CASA 2007).

[70] Yin Li, Jian Sun, Chi-Keung Tang, and Heung-Yeung Shum. Lazy snapping. *ACM Trans. Graph.*, 23(3):303–308, 2004.

[71] Chunlan Liang. *The Techniques of Traditional Chinese Papercutting*. Gansu People's Publishing House, Lanzhou, Gansu, China, 2006.

[72] Roger Ling. *Ancient Mosaics*. Princeton University Press, Princeton, New Jersey, 1998.

[73] Peter Litwinowicz. Processing images and video for an impressionist effect. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 407–414, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[74] Yanxi Liu, James Hays, Ying-Qing Xu, and Heung-Yeung Shum. Digital papercutting. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Sketches*, page 99, New York, NY, USA, 2005. ACM Press.

[75] S. Lloyd. Least squares quantization in pcm. *Information Theory, IEEE Transactions on*, 28(2):129–137, Mar 1982.

[76] Ross Maciejewski, Tobias Isenberg, William M. Andrews, David S. Ebert, and Mario Costa Sousa. Aesthetics of hand-drawn vs. computer-generated stippling. In Douglas W. Cunningham, Gary Meyer, Lszl Neumann, Alan Dunning, and Raquel Paricio, editors, *Proceedings of Computational Aesthetics in Graphics, Visualization, and Imaging 2007 (CAe 2007, June 20–22, 2007, Banff, Alberta, Canada)*, Eurographics Workshop Series, pages 53–56, Aire-la-Ville, Switzerland, 2007. Eurographics Association.

[77] Jianbo Mao and Yin Jiang. *Banqiao's painting inscription*. Xilingyingshe Publishing House, Hangzhou, Zhejiang, China, 2006.

[78] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[79] O Mataev and H Mataev. Olga's gallery. giuseppe Arcimboldo. `http://www.abcgallery.com/A/arcimboldo/arcimboldo.html`, 2006.

[80] Scott McCloud. *Understanding Comics*. Kitchen Sink Press Inc., Northampton MA, 1993.

[81] Frank Miller. *Frank Miller's Complete Sin City Library*. Dark Hore, 2005.

[82] Jun Mitani and Hiromasa Suzuki. Computer aided design for Origamic Architecture models with polygonal representation. In *Proceedings of Computer Graphics International 2004*, pages 93–99. IEEE Press, 2004.

[83] Jun Mitani and Hiromasa Suzuki. Making papercraft toys from meshes using strip-based approximate unfolding. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 259–263, New York, NY, USA, 2004. ACM Press.

[84] C. M. Mooney. Age in the development of closure ability in children. *Canadian Journal of Psychology*, 11:219–226, 1957.

[85] J. Eric Morales. Virtual Mo. `http://www.virtualmo.com`, 2006.

[86] Eric N. Mortensen and William A. Barrett. Intelligent scissors for image composition. pages 191–198. ACM Press, 1995.

[87] David Mould. A stained glass image filter. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering*, pages 20–25, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[88] David Mould and Kevin Grant. Stylized black and white images from photographs. In *NPAR '08: Proceedings of the 6th international symposium on Non-photorealistic animation and rendering*, pages 49–58, New York, NY, USA, 2008. ACM.

[89] Alan Murta. GPC: General Polygon Clipper library. `http://www.cs.man.ac.uk/~toby/alan/software`, 2007.

[90] Yutaka Ohtake, Alexander Belyaev, and Hans-Peter Seidel. Ridge-valley lines on meshes via implicit surface fitting. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 609–612, New York, NY, USA, 2004. ACM.

[91] Symposium on Computational Aesthetics. Computational aesthetics in graphics, visualization and imaging. `http://computational-aesthetics.org`, 2008.

[92] Alexandrina Orzan, Adrien Bousseau, Pascal Barla, and Joëlle Thollot. Structure-preserving manipulation of photographs. In *NPAR '07: Proceedings of the 5th international symposium on Non-photorealistic animation and rendering*, pages 103–110, New York, NY, USA, 2007. ACM.

[93] Victor Ostromoukhov. Pseudo-random halftone screening for color and black & white printing. In *9th congress on advances in non-impact printing technologies*, pages 579–582, 1993.

[94] Victor Ostromoukhov. Digital facial engraving. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 417–424, 1999.

[95] Victor Ostromoukhov and Roger D. Hersch. Artistic screening. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 219–228, New York, NY, USA, 1995. ACM.

[96] Victor Ostromoukhov and Roger D Hersch. Stochastic clustered-dot dithering. *Journal of Electronic Imaging (Special issue on Color Imaging: Device-Independent Color, Color Hardcopy, and Graphic Arts)*, 8(5), October 1999.

[97] Theodore E. Parks. Illusory figures: A (mostly) atheoretical review. *Psychological Bulletin*, 95(2):282–300, 1984.

[98] D.E. Pearson and J.A. Robinson. Visual communication at very low data rates. *Proceedings of the IEEE*, 73(4):795–812, April 1985.

[99] Hans Pedersen and Karan Singh. Organic labyrinths and mazes. In *NPAR '06: Proceedings of the 4th international symposium on Non-photorealistic animation and rendering*, pages 79–86. ACM Press, 2006.

[100] Pietro Perona and Jitendra Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639, 1990.

[101] Yachin Pnueli and Alfred M. Bruckstein. Gridless halftoning: A reincarnation of the old method. *Graphical Models and Image Processing*, 58(1):38–64, 1996.

[102] Thomas Porter and Tom Duff. Compositing digital images. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 253–259, New York, NY, USA, 1984. ACM Press.

[103] Emil Praun, Hugues Hoppe, Matthew Webb, and Adam Finkelstein. Real-time hatching. In *SIGGRAPH 2001, Computer Graphics Proceedings*, pages 579–584, 2001.

[104] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, second edition, 1992. ISBN 0-521-43108-5.

[105] Yingge Qu, Wai-Man Pang, Tien-Tsin Wong, and Pheng-Ann Heng. Richness-preserving manga screening. *ACM Transactions on Graphics (SIGGRAPH Asia 2008 issue)*, 27, 2008. 155:1-155:8.

[106] Vilayanur S. Ramachandran and Diane Rogers Ramachandran. Cracking the Da Vinci Code. *Scientific American reports, special edition on perception*, pages 78–81, 2008.

[107] Xiaofeng Ren and Jitendra Malik. Learning a classification model for segmentation. In *Proc. 9th Int'l. Conf. Computer Vision*, volume 1, pages 10–17, 2003.

[108] Renaissance. Movie: Renaissance. `http://video.movies.go.com/renaissance/`, 2008.

[109] Szymon Rusinkiewicz, Doug DeCarlo, and Adam Finkelstein. Line drawings from 3d models. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Courses*, page 1, New York, NY, USA, 2005. ACM.

[110] Takafumi Saito and Tokiichiro Takahashi. Comprehensible rendering of 3-d shapes. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 197–206, New York, NY, USA, 1990. ACM.

[111] Michael P. Salisbury, Corin Anderson, Dani Lischinski, and David H. Salesin. Scale-dependent reproduction of pen-and-ink illustrations. In Holly Rushmeier, editor, *SIGGRAPH96*, pages 461–468, New York, 1996. ACM Press/ACM SIGGRAPH.

[112] Michael P. Salisbury, Sean E. Anderson, Ronen Barzel, and David H. Salesin. Interactive pen-and-ink illustration. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 101–108, New York, NY, USA, 1994. ACM.

[113] Michael P. Salisbury, Michael T. Wong, John F. Hughes, and David H. Salesin. Orientable textures for image-based pen-and-ink illustration. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 401–406, 1997.

[114] Stefan Schlechtweg, Tobias Germer, and Thomas Strothotte. *Computer Graphics Forum*, 24(2):137–148, June 2005.

[115] Adrian Secord. Weighted voronoi stippling. In *NPAR '02: Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*, pages 37–43, New York, NY, USA, 2002. ACM.

[116] SEGA. Platinum games: Madworld. `http://www.sega.com/platinumgames/madworld/EnglishUK/index2.html`, 2008.

[117] Peter Selinger. Potrace: a polygon-based tracing algorithm. `http://potrace.sourceforge.net/potrace.pdf`, September 2003.

[118] J. A. Sethian. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press, 1999.

[119] John Sharp. *Sliceforms: Mathematical Models from Paper Sections*. Tarquin, 1999.

[120] Jonathan Richard Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In Ming C. Lin and Dinesh Manocha, editors, *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer-Verlag, May 1996.

[121] Wladyslaw Skarbek and Andreas Koschan. Colour image segmentation — a survey",. Technical report.

[122] Yi-Zhe Song, Paul L. Rosin, Peter M. Hall, and John P. Collomosse. Arty shapes. In *Computational Aesthetics in Graphics, Visualization, and Imaging*, pages 65–72, June 2008.

[123] Joan G Stark. The history of ascii text art. `http://www.geocities.com/SoHo/7373/history.htm#typewrite`, 2008.

[124] George Stiny. *Shape: Talking about Seeing and Doing*. The MIT Press, 2006.

[125] Thomas Strothotte. *Computational Visualization: Graphics, Abstraction, and Interactivity*. Springer-Verlag, Berlin, Germany, 1998.

[126] Tatiana Surazhsky and Gershon Elber. Arbitrary precise orientation specification for layout of text. In *PG '00: Proceedings of the 8th Pacific Conference on Computer Graphics and Applications*, page 80, Washington, DC, USA, 2000. IEEE Computer Society.

[127] Tatiana Surazhsky and Gershon Elber. Artistic surface rendering using layout of text. *Computer Graphics Forum*, 21(2):99–110, 2002.

[128] susan Petry. *The perception of illusory contours*. Springer, 1987.

[129] Susan L. Throckmorton. Wycinanki: original papercuttings by Susan L. Throckmorton. `http://www.papercuttings.waw.pl`, 2007.

[130] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Proceedings of the 1998 IEEE International Conference on Computer Vision*, pages 839–846, Bombay, India, 1998.

[131] Robert Ulichney. *Digital Halftoning*. The MIT Press, Cambridge, Massachusetts, 1987.

[132] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(4):376–380, 1991.

[133] Luiz Velho, Paulo Cezar Pinto Carvalho, Luiz Henrique de Figueiredo, and Jonas Gomes. *Mathematical Optimization in Computer Graphics and Vision*. Morgan Kaufmann Publishers, 2008.

[134] Luiz Velho and Jonas de Miranda Gomes. Digital halftoning with space filling curves. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 81–90, New York, NY, USA, 1991. ACM.

[135] Romain Vergne, Pascal Barla, Xavier Granier, and Christophe Schlick. Apparent relief: a shape descriptor for stylized shading. In *NPAR '08: Proceedings of the 6th international symposium on Non-photorealistic animation and rendering*, pages 23–29, New York, NY, USA, 2008. ACM.

[136] Jue Wang, Yingqing Xu, Heung-Yeung Shum, and Michael F. Cohen. Video tooning. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 574–583, New York, NY, USA, 2004. ACM.

[137] Bill Watterson. *Calvin and Hobbes: Sunday Pages 1985-1995*. Andrews McMeel Publishing, 2001.

[138] Matthew Webb, Emil Praun, Adam Finkelstein, and Hugues Hoppe. Fine tone control in hardware hatching. In *NPAR '02: Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*, pages 53–59, New York, NY, USA, 2002. ACM.

[139] Eric W. Weisstein. Gray code, 2008. From MathWorld – A Wolfram Web Resource. `http://mathworld.wolfram.com/GrayCode.html`.

[140] Fang Wen, Qing Luan, Lin Liang, Ying-Qing Xu, and Heung-Yeung Shum. Color sketch generation. In *NPAR '06: Proceedings of the 4th international symposium on Non-photorealistic animation and rendering*, pages 47–54, New York, NY, USA, 2006. ACM.

[141] Julius Wiedemann. *LOGO Design*. Taschen, 2007.

[142] Georges Winkenbach and David H. Salesin. Computer-generated pen-and-ink illustration. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 91–100. ACM Press, 1994.

[143] Holger Winnemöller, David Feng, Bruce Gooch, and Satoru Suzuki. Using npr to evaluate perceptual shape cues in dynamic environments. In *NPAR '07: Proceedings of the 5th international symposium on Non-photorealistic animation and rendering*, pages 85–92, New York, NY, USA, 2007. ACM.

[144] Wucius Wong. *Principles of Form and Design*. Van Nostrand Reinhold, New York, 1993.

[145] Jie Xu and Craig S. Kaplan. Calligraphic packing. In *GI '07: Proceedings of the 2007 conference on Graphics Interface*, pages 43–50, 2007.

[146] Jie Xu and Craig S. Kaplan. Image-guided maze construction. *ACM Trans. Graph.*, 26(3):29, 2007.

[147] Jie Xu and Craig S. Kaplan. Vortex maze construction. *Journal of Mathematics and the Arts*, 1(1):7–20, March 2007.

[148] Jie Xu and Craig S. Kaplan. Artistic thresholding. In *NPAR '08: Proceedings of the 6th international symposium on Non-photorealistic animation and rendering*, pages 39–47, New York, NY, USA, 2008. ACM.

[149] Jie Xu, Craig S. Kaplan, and Xiaofeng Mi. Computer-generated papercutting. In *PG '07: Proceedings of the 15th Pacific Conference on Computer Graphics and Applications (PG'07)*, pages 343–350, Washington, DC, USA, 2007. IEEE Computer Society.

[150] Chao Zheng. *New Decorative Landscape Pattern Design*. Zhejiang People's Art Publishing House, first edition, 2000. ISBN 7-5340-1002-0.

[151] Hanzhong Zuo. *Fan Xiaomei's papercutting technique*. Hunan Fine Arts Publishing House, Changsha, Hunan, China, 2006.