A novel line hatching style through conformal mapping

by

Heng Sun Chao, B.Eng.

A thesis submitted to the
Faculty of Graduate and Postdoctoral Affairs
in partial fulfillment of the requirements for the degree of

Master of Computer Science

Ottawa-Carleton Institute for Computer Science
The School of Computer Science
Carleton University
Ottawa, Ontario
September 2021

©Copyright
Heng Sun Chao, 2021
The undersigned hereby recommends to the
Faculty of Graduate and Postdoctoral Affairs
acceptance of the thesis

A novel line hatching style
through conformal mapping

submitted by Heng Sun Chao, B.Eng.

in partial fulfillment of the requirements for the degree of

Master of Computer Science

Professor David Mould, Thesis Supervisor

Professor Anil Somayaji, School of Computer Science

Professor Robert Laganiere, OCICS

Professor Michel Barbeau, Chair,
School of Computer Science

Ottawa-Carleton Institute for Computer Science
The School of Computer Science
Carleton University
September, 2021
Abstract

Pen-and-ink line hatching is a non-photorealistc rendering (NPR) image style typically characterized by monochrome, locally near-parallel lines. Our motivation was to generate line hatching images inspired by the Grendel image by Emil Antonucci. Antonucci’s images are composed of many cells containing relatively short near-parallel lines. We develop an automatic NPR system that converts a photo into an output image portraying a novel line hatching style. One key characteristic of our approach is its use of conformal mapping to reshape segmented regions into a standardized shape. After drawing the lines on the standardized shape, we then transfer the lines back to the segment region. The transformation automatically distorts the lines to follow the segment region contours. Our process generates line hatching images that have short near-parallel lines, lines that change direction to represent feature boundaries, line density and line width variations to represent tone intensity, edge lines, and line perturbations.
Acknowledgments

I would like to acknowledge my supervisor Professor David Mould. He has been my sounding board, my devil’s advocate, and most of all my teacher. Without his direction and inspiration, this thesis would not have been possible. Any problems I encountered, he was my sounding board: What is the problem? What are potential solutions? Who has solved this problem before? His wealth of knowledge consistently got me unstuck. From his experience he exposed me to weird approaches and processes. Without him, I would not know about Barycentric coordinates, Conformal mapping, nor realize there is a difference between classical dithering and ordered dithering. Finally he always pushed for 100 percent in everything we did. I cannot say that I ever achieved this goal, however, I did learn a lot about graphics and myself in the pursuit.

Many times I would feel lost or hit a wall. Sometimes I became extremely frustrated, failing to see the light at the end of the tunnel. In these situations Professor Sonia Chiasson, my wife, was there to drag me out of these ruts, to show me that there was an end to this journey, and that it was not as far as I thought. Without all her support, the thesis would not have been completed.

Thank you to the defense committee members: Professor Anil Somayaji, Professor Robert Laganiere, and Professor Lianying Zhao. I thank you for spending your precious time reading through my thesis and for providing feedback. Each of you brought a unique perspective and your contributions improved the end result.
# Table of Contents

Abstract iii

Acknowledgments iv

Table of Contents v

List of Tables viii

List of Figures ix

1 Introduction 1
  1.1 Goal ................................................. 3
  1.2 Solution overview .................................. 4
  1.3 Contribution ........................................ 4
  1.4 Organization ....................................... 5

2 Background 6
  2.1 Non-Photorealistic Rendering ................................. 6
    2.1.1 Stroke based approach ................................... 7
    2.1.2 Region-based abstraction ................................. 8
    2.1.3 Line Hatching ........................................... 9
  2.2 Technical Operations .................................. 14
    2.2.1 Segmentation .......................................... 14
    2.2.2 Poisson Disc Sampling .................................. 16
    2.2.3 Boundary First Flattening ............................... 17
    2.2.4 Barycentric coordinates ................................ 18
  2.3 Summary ............................................. 19
3 Algorithm

3.1 Process overview ................................................. 20
3.2 Phase 1: Segmentation ........................................... 23
3.3 Phase 2: Mesh Generation ....................................... 24
3.4 Phase 3: BFF transformation .................................... 24
3.5 Phase 4: Drawing hatching lines ............................... 25
  3.5.1 Line angle .................................................. 25
  3.5.2 Exterior vertex line seed .................................. 26
3.6 Phase 5: Transferring lines ..................................... 27
3.7 Line Refinement Phase .......................................... 27
  3.7.1 Line density and Line weight ............................... 28
  3.7.2 Reorienting lines .......................................... 28
  3.7.3 Eroded hatching lines ..................................... 30
  3.7.4 Edge lines .................................................. 31
  3.7.5 Line extension .............................................. 32
  3.7.6 Intersection detection ..................................... 33
  3.7.7 Perturb Lines .............................................. 34
3.8 Summary ....................................................... 35

4 Results and Discussion ............................................ 36

4.1 Evaluation Criteria .............................................. 38
4.2 Isolation variants ................................................. 39
  4.2.1 Core version ................................................ 41
  4.2.2 Core + Line density (Ld) .................................. 43
  4.2.3 Core + Line width (Lw) .................................... 43
  4.2.4 Core + Extension (Ex) ...................................... 45
  4.2.5 Core + Erosion (Er) ........................................ 47
  4.2.6 Core + Perturbation (Pt) ................................... 47
  4.2.7 Core + Reorienting lines (Re) .............................. 50
  4.2.8 Core + Edge lines (Ed) ..................................... 52
  4.2.9 Summary of isolation variants ............................. 52
4.3 Complex variants ................................................. 58
  4.3.1 Complex base version (Core + Re + Ld + Lw) .............. 59
  4.3.2 Complex base + Er + Ex ................................... 63
  4.3.3 Complex base + Er0 + Ex + Pt ............................. 72
4.3.4 Complex base + Er200 + Ex + Ed + Pt .......................... 74
4.3.5 Summary of complex variants ................................. 76
4.4 Comparison to other methods ................................. 79
   4.4.1 Wei and Mould ........................................ 79
   4.4.2 Rosin and Lai ........................................ 81
4.5 Success and Failure Cases ................................. 83
   4.5.1 What makes an ideal image .............................. 83
   4.5.2 Failure conditions .................................. 86
4.6 Performance ............................................ 89
4.7 Limitations ............................................ 90
4.8 Discussion ............................................. 92

5 Conclusion ............................................... 95
   5.1 Contribution ........................................... 95
   5.2 Discussion ............................................. 96
      5.2.1 BFF limitations .................................. 96
      5.2.2 Segmentation considerations ....................... 97
      5.2.3 Drawing lines .................................... 97
   5.3 Future Work ......................................... 98

List of References ....................................... 100

Appendix A .............................................. 104
   A.1 Additional isolation operation comparison grids ................. 104
   A.2 Additional complex variant comparison grids ...................... 112
   A.3 Python Libraries ..................................... 116
List of Tables

4.1 Assigned scores for isolation line refinements . . . . . . . . . . . . . . 39
4.2 Assigned scores for complex variants . . . . . . . . . . . . . . . . . . 61
A.1 Libraries . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 116
A.2 Custom Functionality . . . . . . . . . . . . . . . . . . . . . . . . . . . 117
### List of Figures

1.1 Hand drawn Line hatching examples ....................................... 1  
1.2 Antonucci’s Grendel image ......................................................... 2  
1.3 Doré’s Don Quixote image ......................................................... 3  
2.1 Lines expressing different tones and textures .......................... 10  
2.2 Using line waviness to exaggerate wood grain .......................... 10  
2.3 Example copperplate engraving .................................................. 11  
2.4 Example hatching and crosshatching output ............................... 12  
2.5 Example Op Art ................................................................. 12  
2.6 Example Op Art Curved Lines .................................................. 13  
2.7 Example user input to produce oriented textures ...................... 13  
2.8 Coordinates in a triangle ......................................................... 18  
3.1 Grendel image by Antonucci ..................................................... 20  
3.2 Example transformation from our process ................................... 21  
3.3 Process Flow ................................................................. 22  
3.4 Example segmentation 400 vs 1600 .......................................... 23  
3.5 Equal line spacing on standardized shape .................................. 26  
3.6 External vertex line seed ........................................................ 27  
3.7 Line reorientation example ...................................................... 29  
3.8 Example effect of erosion ....................................................... 30  
3.9 Example effect of extension ..................................................... 32  
3.10 Example intersection detection ................................................. 33  
3.11 Line perturbation example ...................................................... 34  
4.1 original seven representative images ....................................... 37  
4.2 Overview of isolation line refinements ..................................... 40  
4.3 Core version - seven representative images ............................... 42  
4.4 Core + Line density ............................................................ 44  
4.5 Core + Line width .............................................................. 46
A.6 Core + isolation operations: Athletes .......................... 110
A.7 Core + isolation operations: Skyscraper ....................... 111
A.8 Complex base + Er10 + Ex + Pt ............................... 113
A.9 Complex base + Er200 + Ex + Pt ............................. 114
A.10 Complex base + Er10 + Ex + Pt + Ed ......................... 115
Chapter 1

Introduction

Figure 1.1: Sample of hand drawn line hatching images. Images reproduced from Guptill [16]

In computer graphics, pen-and-ink line hatching is a non-photorealistic rendering (NPR) image style typically characterized by monochrome, locally near-parallel lines. Figure 1.1 illustrates hand drawn line hatching images which computer-generated images emulate. Notice from the figure that this style represents tone through varying line density and line thickness. The lines may be straight, or curved as seen on the barrel hoops. To represent texture, the lines can be perturbed, as seen on the barrel planks. Features have silhouettes emphasized through edge lines, as seen in the pitcher.

The style is still relevant today. Requiring only monochrome lines, it is easy
and cheaper to replicate in print than colour images. Unlike colour photos, the reduced palette has relatively simple hardware printing requirements. The style itself is an interesting artistic representation resembling hand drawn sketches. With the proliferation of mobile devices incorporating image capturing technology, hatching line style could be applied as a filter to real world images, giving end-users the ability to generate versions of their photos with this visual effect.

In this thesis, we were inspired by images produced by Emil Antonucci in 1971 for John Gardner’s Grendel [12] novel. As illustrated in Figure 1.2, the images use a unique Pen and Ink line hatching drawing style. Antonucci’s images are composed of many cells that contain relatively short near-parallel lines, rather than using long continuous lines as seen in the background of Figure 1.3. The lines are constant thickness with minor variations in density to impart some degree of intensity. From Antonucci’s images, we wanted to capture these characteristics:
CHAPTER 1. INTRODUCTION

Figure 1.3: Don Quixote illustration by Gustave Doré [9]

- Near parallel straight and contour lines
- Short hatching lines
- Line direction changes exclusively delineate feature boundaries
- Monochrome lines

We were also inspired by more typical line hatching drawings, as in Figure 1.1, and wanted to include the following additional properties:

- Tone intensity conveyed through line thickness
- Line perturbations convey small structure information
- Edge lines provide silhouette definition

1.1 Goal

Our objective is to create an automated process capable of producing line hatching images that combine the characteristics of both the Grendel image and more typical hatching styles from an input photo. Our process should generate line hatching
images that have short near-parallel lines, lines that change direction to represent feature boundaries, line density and line width variations to represent tone intensity, edge lines, and line perturbations. The process should allow variation of the line characteristics to produce different visual effects.

1.2 Solution overview

Our overall process has five main phases. First we create regions using a segmentation algorithm. For each region, we generate a triangle mesh to encode the region boundary. Each region is distorted using conformal mapping into a standardized shape. Once we have the standardized shape, we draw straight lines on it. These lines are then transferred back to the original region. The transferred lines will conform to the region contours and be simultaneously distorted by the standardized shape. The regions are then combined to create our core version output image.

Our process includes an optional refinement phase where we can modify the core version by applying the following line refinements:

1. Line density
2. Line thickness
3. Line erosion
4. Line reorientation
5. Line perturbation
6. Line extension
7. Edge lines

In our process, we apply the line refinements in various combinations to produce different visual effects.

1.3 Contribution

Our main contribution is an automatic system that produces a novel line hatching style from input photos. As a combination of our two inspiration styles, our core version output images have short near-parallel lines that are bound to their respective regions. This version successfully meets three of the seven characteristics mentioned above. Augmenting the core version, we further make line alterations to create various
visual aesthetic effects. For example, one alteration results in cellular structures that resemble “features”. The hatching lines are able to differentiate large scale features, foreground and background are clearly delineated, and the hatching lines vary characteristics to capture image tone intensity. The variant that we deem most successful meets six of the seven characteristics; in it, we intentionally ignore the “parallel lines” characteristic to support line extension, though we can also achieve this characteristic in other variants.

We devised a workflow that uses existing processes in a novel combination; our process transforms segment regions to a standardized shape, allowing us to elegantly transfer strokes that conform to region contours. Additional standardized shapes can be created in the future and then substituted into our workflow. Each standardized shape would contribute its own distortion pattern to the hatching lines. Similarly, more complex stroke types can be constructed and used in our workflow. In our current implementation, we use straight lines. Varying the stroke type will alter the visual effect of the hatching lines.

Although we have applied our workflow to line hatching, our transformation process could be incorporated into a variety of other stroke based techniques. For example, one modification would be to change from hatching lines to painterly strokes.

1.4 Organization

The thesis is organized as follows. We present relevant background about NPR and line hatching in Chapter 2. Chapter 3 provides details of our workflow and implemented system. Chapter 4 presents our output images and interpretation of our results. Chapter 5 offers our concluding remarks and discusses future work directions.
Chapter 2

Background

In this thesis, we develop an automated NPR system that produces an image in line hatching style based on an input photo. In the following sections we present the related work and background approaches used in the system.

2.1 Non-Photorealistic Rendering

Non-photorealistic rendering (NPR) is an area of computer graphics that has the goal of “generat[ing] imaging that does not aspire to realism” [13]. In contrast, photorealistic approaches attempt to produce images that closely resemble the output of a photographic camera. Images generated in NPR can be generated for different purposes, including for artistic or aesthetic purposes, or to enhance comprehensibility and clarity [13]. NPR covers many different output styles, e.g., water colour [7, 49], painterly (simulating paintings) [17, 35], mosaic [20, 26], technical or architectural illustrations [54], and line hatching [28, 46]. Kyprianidis et al. [30] survey the field of NPR and categorize the various techniques and styles. The survey provides a good introduction to the various styles in NPR and serves as a common reference point for comparing and discussing past and future works in NPR.

The traditional graphic generation pipeline uses 3D models (i.e., object-space or geometry-based models) of the objects being represented as input, which is then used to produce 2D projections as output images [54]. In contemporary NPR, the input can be interactive user input, object-space models, or image-space information (i.e., photographs). In this thesis, we concentrate on object-space models and image-space information. Both approaches have advantages and disadvantages.

Because of their 3D models, geometry-space systems can generate images from
different perspectives (as opposed to image-space systems which are based on a single, static view). These systems can extract the actual contour lines that are appropriate for the objects [51] in the scene, they can add or remove light sources, and they can render shadows based on light source and depth of field information. Unfortunately, when geometry information is unavailable, these systems are rendered useless. Often, obtaining 3D models of outdoor scenes or real life events is still difficult and may not be available at all.

In contrast, image-space systems process 2D input images, which are ubiquitously available from consumer image-capture devices. For example, a wedding may be captured in many 2D photos, but it is unlikely that there are geometry-based models of the event. Image-space systems process the information available from an input image, e.g., intensity and colour. The system may extract features by applying transforms to the original image [51], e.g., for gradient, for edge detection, or for segmentation. It can then use the extracted features to emphasize certain characteristics of the image, e.g., the edge detection transform may indicate object silhouettes.

Most NPR systems need some level of human interaction, for example in choosing styles, parameters, and input imagery, which all contribute to the aesthetic of the output image. Systems can be differentiated by their degree of user intervention. Automatic systems require no or very limited input from users to complete their tasks once the parameters are defined [15]. Interactive systems may rely on user guidance to decide the placement of strokes, the stroke type, and the stroke colour.

In this thesis, we developed an automatic NPR image-space system. It uses photos as input to generate images in pen-and-ink line hatching style.

2.1.1 Stroke based approach

In NPR, stroke based rendering (SBR) approaches create “non-photorealistic imagery by placing discrete elements called strokes, such as paint strokes, or stipples” [23]. SBR algorithms modify a base image by placing strokes according to some goal [22], usually to make the original image take on a new appearance (e.g., to look like an oil painting, or to look like a hand-drawn sketch).

An early influential paper by Winkenbach and Salesin [54] is among the first to introduce strokes in context of NPR. Strokes have various properties, such as spacing and weight. Varying these properties can express tone and texture. For example, crisp straight line imply smooth surfaces while perturbations, such as wavy lines, can
convey an aged texture, like old wood.

Most relevant to this thesis are pen-and-ink images, which approximate hand-drawn illustrations and are usually monochromatic. Pen-and-ink illustrations are formed from parameterized strokes [46] which can be used to illustrate tone or textures, and to form outlines. Multiple strokes can be combined to achieve different tone levels by varying stroke weight and spacing [54]. Common stroke types include hatching (roughly parallel lines), crosshatching (overlapping hatching in multiple orientations), stippling (small dots or very short lines). Textures can be expressed by varying stroke orientation, spacing, and weight [54]. Outlines are long strokes that define the outer contours of an object as well as expressing interior boundaries (e.g., a leaf has an outside contour and interior veins) [54].

In our work, we vary line thickness and weight to express region tone, and we use outlines to create silhouettes. To capture texture, we add line perturbations where appropriate.

2.1.2 Region-based abstraction

Mould [38] provides a thorough explanation of region-based abstraction. In this section, we briefly summarize the most relevant aspects.

A challenge to stroke based rendering (SBR) is generating output that matches the user’s semantic understanding of the original image. One approach to address this is to divide the image into regions that are distinct from each other and apply styles based on these regions. Regions are defined as “continuous areas of the image plane, often corresponding to semantic divisions of the image content” [38]. Individually, the regions may not have much meaning, but the overall arrangement allows meaning to emerge.

Once created, regions can serve two different purposes: (i) to act as containers by imposing structure, or (ii) to act as rendering primitives themselves. When imposing structure, region boundaries can act as containers for the rendering primitives. The region boundary provides contrast by clipping primitives, such as strokes, and preventing them from spanning multiple regions. Typically, the primitives in each container region share similar properties, e.g., stroke colour and orientation, providing coherence to the region. When acting as a rendering primitive, the region shape itself is the rendered artefact, e.g., a region acting as a stained glass tile.

As an example of using regions for structure, Gooch et al. [14] segment images
based on intensity and then draw paint strokes in each region. The medial axis of
each region determines the orientation of the paint strokes. Stroke economy is an
important goal of their work, consequently they explore merging strokes within a
region to reduce the total number of strokes. As another example, Li and Huang [34]
automatically generate pencil drawings based on features extracted from a photo.
They use colour-based segmentation to divide the photo into regions and then draw
pencil strokes in each region.

Considering regions as primitives instead, Mould [37] used regions as the rendering
primitive for stained glass effect. They apply segmentation to the input image, modify
the regions by applying erosion and dilation morphology operators, and then assign
colours to each resulting region. The regions are then merged back together to create
images that have the stain glass effect. They render the region boundaries to resemble
the lead solder typically used in stained glass.

In our system, we use regions for structure, to contain our rendering primitives. The bounding of hatching lines allows us to mimic the cellular containers found in
the Grendel images.

2.1.3 Line Hatching

Line hatching is a technique used in pen-and-ink illustrations [24]. It uses a collection
of monochromatic parallel strokes that fill an area to convey tone, shape, and
texture [24,54,56]. For example, Figure 2.1 (a) illustrates different tone intensities,
Figures 2.1 (b) and (c) illustrate shapes combined with varying tone and texture, and
Figure 2.1 (d) illustrates different densities of fur. Varying line thickness, density,
or spacing can communicate different tones [54]. The character of the strokes can
also be used to present different visual effects, e.g., adding waviness to line strokes
can depict different textures [54]. Figure 2.2 contrasts two different densities of wavy
lines; due to the difference in wavy lines, the wood panels on the right appear older.

Crosshatching is a method of conveying darker tones [54]. Multiple layers of hatch-
ing lines with unique orientations are overlaid to create crosshatching. In Figure 2.1
(a), the third to fifth cells contain crosshatching illustrating different tone intensities.

Although most techniques for generating hatching lines are concerned with the
intensity and orientation of the lines, they take different approaches to generating
As shown in Figure 2.3, Ostromoukhov [41] converts 2D facial images into copperplate engraving images. Their process requires users to select region boundary shapes to act as containers for the hatching lines. The system creates a dither screen for each region while maintaining contours of the region when rendering the lines.
CHAPTER 2. BACKGROUND

Figure 2.3: Example copperplate engraving by Ostromoukhov. Image reproduced from Ostromoukhov [41]

The final output image is created by overlaying multiple dither screen layers.

Hertzmann et al. [24] use hatching and crosshatching to suggest object contours. Their system determines hatching line orientation from surface geometry information while line thickness is determined by region lighting. Their focus is on detecting and illustrating the internal curvatures of surfaces with hatching and crosshatching lines. They attempt to reduce the number of hatching lines for economy and rely on indications to fill in areas. Indications are areas that are left empty, allowing the viewer to fill in the space. Figure 2.4 demonstrates output from their work containing hatching, crosshatching, and indications.

Inglis and Kaplan [28] generate Optical Art (Op Art) line drawings where changes in line orientation are used to indicate the feature boundaries. Although this form of Op Art is not strictly considered line hatching, we include it here because it has some relevant characteristics, namely that feature boundaries are denoted through
Figure 2.4: Example hatching and crosshatching output from Hertzmann et al. on a Klein bottle. Image reproduced from Hertzmann et al. [24]

Figure 2.5: Example Op Art illustrating a skull. Image reproduced from Inglis and Kaplan [28]

Inglis and Kaplan [27] extend their work and introduce curved lines. The curves
suggest depth in the images as seen in Figure 2.6.

Figure 2.6: Example Op Art illustrating a skull with curved lines. Image reproduced from Inglis and Kaplan [27]

Salisbury et al. [47] develop an interactive system where users guide the system in generating a pen-and-ink illustrations. The user selects the line orientation and the stroke or texture type from a library. The system uses this input in its rendering of output lines for the selected region, saving the user time over having to manually draw the individual strokes. Figure 2.7 illustrates how the base image is combined with user selected orientation and texture to produce the output image.

Figure 2.7: Example user input to produce oriented textures. Image reproduced from Salisbury et al. [47].

In our work, we draw pen-and-ink images using hatching lines. We vary hatching
line density and thickness based on region density, and we try to retain region contours in the final output image.

2.2 Technical Operations

Our approach uses several mathematical techniques and operations. We first describe our approach to segmenting the image. Next we describe the operations necessary to position our hatching lines within the segments, using triangle meshes. Each of these will be used as part of our process in Chapter 3.

2.2.1 Segmentation

Segmentation algorithms divide an image into discrete, non-overlapping regions where the pixels belonging to a region share some common properties based on a distance criteria, e.g., they may share similar colours [38], proximity, orientation, or intensity [55]. Segmentation is the basis for many image processing applications, such as object detection for machine vision, analysis of medical imaging, content-based image retrieval, or recognition tasks. In NPR, segmentation of an input image enables the application of abstractions based on features or objects within the image.

Haralick and Shapiro [19] provide a good survey of early segmentation approaches while Zaitoun and Aqel [55] provide a more recent overview. According to Haralick and Shapiro [19], good segmentation should have the following properties: i) uniform and homogeneous regions, ii) simple regions without many small holes, and iii) region boundaries that are spatially accurate and simple, not ragged. They further note that it is difficult to fulfill all of these requirements simultaneously. There are many approaches to segmentation, but no single algorithm is optimal for all applications. Each approach has strengths and weaknesses, and may prioritize one or more characteristics, such as speed, edge detection, boundary adherence, well shaped regions, or region count control.

Automatic vs user-guided: Segmentation methods range from fully automatic to user-guided semi-automatic systems. Automatic systems save valuable user time, however, these systems can be brittle; they may work well for specific images and poorly on others, or they may fail to create regions that correspond to semantic features. Features caused by light and shadow may create spurious segments. Thin and
small features may be incorrectly merged with other regions and be lost altogether.

Semi-automatic systems accept user input and often produce better results than fully automatic systems. Typically, users provide additional context by identifying important regions by adding markup (e.g., by clicking or otherwise highlighting parts of the image). Semi-automatic systems use this additional information to correct or amend the segmentation. DeCarlo and Santella were among the first researchers to implement semi-automatic segmentation where user gaze identified salient content [8].

In our system, we use an automatic segmentation system to create regions as containers for our primitives. We accept reasonable but non-optimal region generation in favour of speed.

**Simple Linear Iterative Clustering:** Simple Linear Iterative Clustering (SLIC) [1] is a segmentation algorithm that implements a variation of K-means clustering to create regions. In this thesis, we first use SLIC to divide the original image into discreet regions. We selected SLIC because existing implementations were readily available, and it allowed us to easily specify the number of desired regions. Our process is agnostic to choice of segmentation algorithm. However, the segmentation algorithm must produce regions that can be transformed by Boundary First Flattening (BFF – Section 3.4). We found that BFF had difficulty transforming certain region shapes. As a result, we had to use SLIC0, which generates more regular shapes; see Section 5.2.

When given an input K, the algorithm will produce K segments. As described by Achanta et al. [1], pixels are assigned to the closest cluster site at each iteration based on the pixel distance equation 2.3. Once all pixels have been assigned to a cluster, each cluster center is moved to the average location based on all pixels that belong to that cluster. The calculated cluster centers become the sites for the next iteration. This cycle repeats a set number of iterations or until the error falls below a threshold.

The algorithm uses a pixel distance measure, $D$, that incorporates both the colour distance in CIELAB colour space (equation 2.1) and the Euclidean distance (equation 2.2). Parameter $m$ in the pixel distance equation (equation 2.3) allows control over the spatial distance contribution, where $N$ is the total number of pixels, $k$ is the number segments, and $S = \sqrt{N/k}$ - is the size of the segment. Increasing $m$ increases the spatial distance contribution and consequently produces more compact clusters.
\[ d_c = \sqrt{(l_k - l_i)^2 + (a_k - a_i)^2 + (b_k - b_i)^2} \quad \text{Colour distance (2.1)} \]

\[ d_s = \sqrt{(x_k - x_i)^2 + (y_k - y_i)^2} \quad \text{Euclidean distance (2.2)} \]

\[ D = \sqrt{d_c^2 + \left( \frac{d_s}{S} \right)^2 m^2} \quad \text{Pixel distance (2.3)} \]

\[ D = \sqrt{d_c^2 + \left( \frac{d_s}{m_s} \right)^2} \quad \text{SLIC0 pixel distance (2.4)} \]

Normally SLIC requires the user to specify the \( m \) parameter when starting the segmentation. A variant called SLIC0 removes the need to select a static \( m \) value. SLIC0 retains the maximum Euclidean distance, \( m_s \), for each cluster at the end of each iteration and then uses it in the next iteration to normalize the Euclidean distance contributions as seen in SLIC0 pixel distance equation (equation 2.4). SLIC0 tends to consistently produce compact regions and is nearly as fast as SLIC. For these reasons, we chose SLIC0 variant for our segmentation.

### 2.2.2 Poisson Disc Sampling

Poisson Disc Sampling [6] is a process that generates random points that have a minimum distance, \( r \), between neighbouring points. Additionally, the points are tightly packed, i.e., no additional points can fit into any of the spaces without violating the minimum distance. The set of points produced is a Poisson Disc Distribution.

The Poisson Disc Distribution has interesting properties when applied to graphics. For example, using Poisson Disc Distribution for sampling of pixels can reduce the aliasing effects commonly encountered with uniform sampling [6]. Among other uses, Poisson Disc sampling patterns have been used in medical MRI to improve image speed and quality [10].

The Poisson Disc Distribution can be generated with many different algorithms [31]. Bridson’s [3] implementation is fast and iterative, probabilistically improving the point distribution quality with increased iterations. The algorithm starts
with a random point within the region and adds it to the retained list, $L_{\text{retained}}$. The algorithm then iterates through all points $P_i$ in the retained list. For each $P_i$, a new random point $P_{\text{new}}$ is then selected such that the point is at least $r$ distance from $P_i$ and no more than $2r$ distance away. If $P_{\text{new}}$ is closer than $r$ to any point in $L_{\text{retained}}$, it is discarded. Otherwise, $P_{\text{new}}$ is appended to $L_{\text{retained}}$. $P_{\text{new}}$ is generated and processed $K$ times per $P_i$. The algorithm then proceeds to the next point in $L_{\text{retained}}$, terminating when the retained list has no unprocessed points.

The algorithm allows a distribution quality compromise in favour of additional speed. Increasing $K$ increases execution time and improves the distribution quality while reducing $K$ improves execution time and reduces distribution quality. We configured the implementation in favour of additional speed with some loss in point distribution quality. These points will eventually be used as vertices in a triangle mesh over a specific region. The resulting triangles are are roughly the same size and relatively regular in shape, i.e., no triangle has one edge that is significantly longer than the other edges.

### 2.2.3 Boundary First Flattening

Parameterization involves computing a one-to-one mapping from a piecewise-linear triangulated surface to another domain [36]. Although there are special shapes that can be transformed without distortion, for example flattening a cylinder, this is not the general case. The main challenge for parameterization algorithms is that they will distort angles, areas, or both [2]. It is unlikely that a parameterization can maintain both the original angles and areas simultaneously.

Boundary First Flattening (BFF) is a linear method for conformal mapping [48]. It is a fast algorithm that can support real time editing applications. It also provides control over boundary length or boundary angle of the flattened mesh. BFF can produce output conforming to a system-defined flattened disc shape or conforming to a user-defined shape.

We approximate our region surfaces with a mesh and then use BFF to reshape each region’s triangle mesh. BFF transforms the region mesh to a mesh with a circular shape (i.e., the flattened disc shape). The reshaping operation maintains a one-to-one triangle mapping between the two meshes and ensures that the local angles of the triangles are preserved as well as their orientation. We use the mesh in subsequent transformations, and at a later stage, we use barycentric coordinates to transfer lines...
from one mesh to the other.

2.2.4 Barycentric coordinates

In our process, we draw hatching lines on the circular shape’s triangle mesh and transfer the lines to the region’s triangle mesh. Since the lines are composed of points that reside inside the triangle mesh, we use barycentric coordinates to transfer the points from one mesh to the other.

As stated by Floater [11], barycentric coordinates are used to represent the position of a point $X$ in triangle $T$, identified in the barycentric equation (equation 2.5). The coordinates are the ratios of the areas of the sub-triangles, $A_i$, to the area of the whole triangle, $A$, as illustrated in Figure 2.8. The resulting barycentric coordinate components $\phi_i$ are represented by barycentric components equation (equation 2.6). Since barycentric coordinates are based on ratios of areas, when the triangle vertices are stretched, the coordinates will still provide a valid point position in the stretched triangle. We rely on this property when transferring lines from the circular shape mesh back to the region mesh.

\[
X = \phi_1 V_1 + \phi_2 V_2 + \phi_3 V_3 \quad \text{Barycentric equation (2.5)}
\]

where

\[
\phi_i = \frac{A_i}{A}, \; i = 1, 2, 3 \quad \text{Barycentric components (2.6)}
\]
2.3 Summary

Pen-and-Ink drawings can be produced in many different artistic styles, but to our knowledge, no NPR approach has produced images resembling the Grendel image [12]. The Grendel image is drawn in the Pen-and-Ink style, composed of identifiable feature regions that contain hatching lines. Drawing from various NPR techniques, we develop an automatic approach to generate Pen-and-Ink output images inspired in part by the Grendel image style and in part by more typical line hatching styles. We discuss the details our process in the following chapter.
Chapter 3

Algorithm

3.1 Process overview

The Grendel illustrations by artist Emil Antonucci for John Gardner’s 1971 novel Grendel [12] are black and white images that use only black hatching lines to show form. As seen in Figure 3.1, the shape of the region, orientation of the region, and orientation of the lines within each region work together to produce identifiable features of the Grendel face. The lines span a single identifiable region and they are confined to that single region. Some lines are straight while some are curved,

Figure 3.1: Grendel image by artist Emil Antonucci. Reproduced from John Gardner’s novel [12]
CHAPTER 3. ALGORITHM

(a) Original Image  
(b) Final Image

Figure 3.2: Example transformation from our process

however, all lines within a region are near parallel. Though we were inspired by the Grendel image, we are not trying to reproduce it faithfully. As demonstrated more fully in Chapter 4, we create pen-and-ink line hatching images with the following characteristics:

- Near parallel straight and contour lines
- Short hatching lines
- Line orientation changes differentiate features and objects
- Monochrome lines
- Regions that conform to identifiable objects in the image
- Tone intensity conveyed through line thickness and density
- Line perturbations convey small structure information
- Edge lines provide silhouettes

Our novel approach allows us to draws straight hatching lines on a standardized shape (e.g., a circle) and then transfer the lines back to the segment region. The transfer process can handle segment regions of arbitrary shape and distorts the hatching lines based on the segment region contours and the standardized shape. Though we implemented the standardized shape as a circle, in theory, we could implement different standardized shapes, e.g., a square or triangle, to achieve different visual effects. Figure 3.2 demonstrates an example input image and the resulting output from our process. Note the output image exhibits the characteristics listed above, e.g., it has short near parallel lines, tone intensity is represented through line density
and line thickness changes, and feature boundaries are highlighted by changes in line characteristics.

We now provide a general overview of our process, which is illustrated in Figure 3.3. We take an input photo, and we first segment the image into regions (Phase 1). For each region, we generate a triangle mesh over the region (Phase 2). The vertices of the mesh provide reference points for later transformations. We approximate the segment region surface with a triangle mesh. We then apply conformal mapping to the mesh and transform each region’s mesh by reshaping it to a standardized shape (Phase 3) and then draw straight lines on the standardized shape (Phase 4). The lines on the standardized shape are then transferred back to the original region using barycentric coordinates (Phase 5). The transfer causes the lines to be distorted to follow the contours of the region’s original shape. Once all regions have been processed, we reassemble all the regions to produce the final hatching image. The resulting image after reassembly will be called the *Core version*. Our process also has a number of optional refinements that can be applied to the core version. These
refinements are discussed in Section 3.7.

3.2 Phase 1: Segmentation

The Grendel image from Figure 3.1 has different identifiable regions, for example the nose, cheek, chin, and eyes. We wanted our procedure to produce images that have similarly identifiable regions. One approach to producing identifiable regions is to use image segmentation.

Among the segmentation alternatives, we selected Simple Linear Iterative Clustering (SLIC) [1] as discussed in Section 2.2.1. We observed that BFF had problems transforming irregularly shaped regions generated by SLIC. Regions with significant deviation from convexity, e.g., starfish-like appendages, posed problems. However, more convex shapes were consistently transformed by BFF.

As a result we used a SLIC variant called SLIC0 from the Scikit-image library. This variant created more uniformly shaped regions with smoother boundaries than the regions created by SLIC. The variant retains the maximum Euclidean distance for each region from the previous iteration. The previous maximum distance is then used in the next iteration normalize the distance equation 2.4. The more compact convex shaped segment regions reduced BFF transformation failure to below 1 percent. Failed regions are left blank.

Figure 3.4 provides sample segmentation results with 400 and 1600 segments. The higher segment count effectively provides “higher resolution”, as a result, smaller
objects from the original image are successfully represented by a region. For example, some of the windows on the upper portions of the building are better represented by the 1600 segment version. However, with 400 segments, the larger windows are more likely to be represented by one (or a small number of) regions, increasing chances that the eventual hatching lines are continuous over the window region.

We calculate the average region intensity per region. This information is conveyed in Figure 3.4 using a greyscale overlay. Region intensity will be used later in Section 3.7.

3.3 Phase 2: Mesh Generation

To represent the surface of each segment region, we generate a triangle mesh covering each region. We later transform each segment region by reshaping the corresponding mesh. When the mesh is reshaped, there will be a one-to-one triangle mapping between the original mesh and the reshaped mesh. Using this mapping, we can identify corresponding points between the two meshes. We then reshape the mesh through stretching and compression to represent a different perimeter. The reshaping process will occur in Phase 3.

The first step in creating the mesh is to create vertices for the mesh. In our process, we populate the region perimeter with points. The points along the perimeter are positioned such that the distance between neighbours is greater than $r$ and less than $2r$, where $r$ is the minimum distance in the Poisson distribution. This ensures that we have full coverage of the region shape and minimizes triangle gaps along the perimeter. We then use Poisson disc sampling [3] to cover region interior with points. Lastly, we use a Scipy library to compute a Delaunay triangulation [32] over this point set to create the mesh. See Phase 2 in Figure 3.3 for illustration of the Poisson disc sampling points and the resulting generated mesh.

3.4 Phase 3: BFF transformation

We use conformal mapping to reshape the segment region into a predefined standardized shape as shown in Phase 3 of Figure 3.3. We approximate each segment region surfaces with a triangle mesh. We then use the code from Sawhney and Crane to transform each segment region mesh to a standardized shape mesh using Boundary
CHAPTER 3. ALGORITHM

First Flattening (BFF) [48]. Our transformation approach is region-shape agnostic. It will work as long as we can transform the region mesh into the standardized mesh using BFF.

Reshaping the segment region mesh from Phase 2 to the standardized shape retains a one-to-one triangle mapping between the two meshes. We will use the one-to-one triangle mapping later to transfer hatching lines from the standardized shape back to the segment region in Phase 5. The hatching lines will be distorted according to the original region boundary contours in the transfer step. In the current implementation, we use the circle as the standardized shape. In theory, any shape can serve as the standardized shape. Each standardized shape would change the characteristics of the line hatching and produce a different visual style. In this thesis, limitations with BFF prevented us from generating additional standardized shapes.

3.5 Phase 4: Drawing hatching lines

We draw straight lines on the standardized shape. The first point for each line starts with on a perimeter vertex. We walk across the standardized shape until we intersect a triangle edge. The intersection of the line with the triangle edge is retained as a point. The lines are then composed of this retained point sequence. The red dots in Figure 3.6 illustrate the points that intersect triangle edges for a single line. These retained points will be used in Phase 5. In theory, we could draw additional types of lines, e.g., sinusoidal, square waves, bulls eye pattern, jagged lines, or circles. Introducing these additional lines would not alter our main operations of mesh generation, mesh reshaping, and line transfer but each would result in different visual effects. These additional line types could be combined to produce combined effects. When transferred back to the original region, the additional line types would retain some of their characteristics and would conform to the region contours.

3.5.1 Line angle

When we draw straight lines on the standardized shape, we want the line orientation to coincide with the major direction of that segment region. To achieve this, we determine the direction of the original segment region by calculating the structure tensor and extracting its predominant direction. This direction is then converted into
the equivalent angle on the standardized shape mesh. We then draw straight lines on the standardized mesh to align with this calculated angle.

3.5.2 Exterior vertex line seed

We want segment region line spacing to be approximately uniform on the original regions. We encountered a challenge in our process where the standardized shaped mesh demonstrated stretching in certain areas, resulting in sparse triangle coverage, and compression of other areas, resulting in dense triangle coverage, as seen in Phase 3 of Figure 3.3. If we draw lines with constant spacing on the standardized mesh then the line spacing on the segment region would be variable when they are transferred back to the segment region, due to the varying triangle density (Figure 3.5).

To overcome this behaviour, we use the exterior triangle vertices as the line starting points. When the lines are transferred back to the segment region (Figure 3.6), the line spacing will be driven by density of the exterior triangle vertices. Since the triangle mesh was initially created on the segment region using Poisson disc sampling [3], the distance between the points along the mesh perimeter are uniform. Although this approach produces variable line spacing on the Standardized shape mesh, it is only an intermediate state.

Figure 3.5: Equal line spacing on standardized shape.
3.6 Phase 5: Transferring lines

After the hatching lines have been drawn on the standardized shape, we transfer the lines to the segment region. We accomplish this by using barycentric coordinates to transfer the points for a given line (See Section 2.2.4). Remember that each line is a sequence of points that intersect a triangle edge (Figure 3.6). We calculate the barycentric coordinates for each point relative to the intersecting triangle on the standardized mesh. These allows us to calculate the point positions on the corresponding segment region triangles. When we transfer the lines to the original segment region they are automatically distorted according to original region contours. After we generate all the hatching lines and transfer them back to the segment regions, we construct the final image by reassembling all the segments. The resulting image at this point is referred to as the *Core version*. We have the option of applying line refinements discussed in Section 3.7.

3.7 Line Refinement Phase

The core version is a line hatching image. We explore additional line refinements that produce different visual effects that may be more pleasing to the observer. These modifications do not change our base process, rather they are optional steps applied
to the lines before the final assembly.

In the following subsections, we describe several possible line refinements that may be used to impact the output image style: line density and line weight, reorienting hatching lines, adding edge lines, line erosion, extending lines, and line perturbation. We first introduce the refinements in isolation to focus on the details of each effect, however, we have found that using them in combination is more effective. In our results, we will first present output images with single line refinements applied to the core version (Section 4.2). We will then present results of adding combinations of line refinements to the core version (Section 4.3).

### 3.7.1 Line density and Line weight

Line hatching communicates region intensity by varying the line density and line weight [56]. We calculate each region’s average greyscale intensity and use this value to determine its line density and line weight. We use an inverse linear scale to determine line spacing; higher intensities have a larger minimum distance between the lines while darker regions have a smaller minimum distance between the lines. Similarly, higher intensities were assigned lower line weights and lower intensities were assigned higher line weights. Once we determined the desired minimum line distance, we culled lines that were too close together.

To process lines in a given region, we start with the longest line in the region and add it to the retain list. We then traverse the remaining lines in adjacent order. We calculated the distance from the next line to all lines in the retain list. If the distance between the next line and any line in the retain list was less than the minimum desired retain list was less than the minimum desired distance, we removed the next line. Otherwise, the next line was added to the retain list. Since each region has its own intensity value, we repeat the whole process for each region.

### 3.7.2 Reorienting lines

Ideally the segmentation algorithm would consistently create segments matching all semantic features of the input image. However, this is not currently feasible for automated processes.

In cases where a feature is represented by multiple segments, we want the line orientation across the segment regions to have the same orientation. To achieve this,
we first calculate the average greyscale intensity difference between the current region and all neighbouring regions. If the intensity difference between two regions is below the threshold $T\alpha$ then the line orientations for the two regions attract, i.e., the line orientation of the first region will adjust to be more colinear with the second region. The attraction operator determines the average line orientation between the two regions and assigns this new direction to the first region. Figure 3.7 (2) illustrates region A adjusting lines to be more colinear with region B while Figure 3.7 (3) illustrates region A readjusting to be more colinear with region C.

However, if the intensity difference between two regions is above or equal to $T\alpha$ then the line orientations for the two regions repel, i.e., the line orientation of the first region will adjust to become more perpendicular to the second region. The repel operator determines the average between the line orientation of the first region and line orientation perpendicular to the second region. This new direction is assigned to the first region. Figure 3.7 (5) illustrates region D adjusting to be more perpendicular with region B.

The number of iterations will impact the final line orientations, but each iteration increases the execution time. We experimented with 10, 50, and 100 iterations. The orientation change between 10 and 50 iterations was significant. Going from 50 to 100 still resulted in changes; however, the changes were subtle. For the results presented
in Chapter 4, we run the reorientation for 50 iterations.

### 3.7.3 Eroded hatching lines

When we generated hatching lines for each segment region, these lines cover the region from edge to edge. We introduce the line erosion operation to shrink the length of the lines. The erosion operation accepts one parameter that indicates how many points we remove from both ends of the lines. Since we stored points in a list data structures, removing individual points from a line was made easy. An alternative approach would have been to remove a set distance from each end of the line and recalculate the position of the end point.

Recall that lines are made up of points. Figure 3.8a illustrates a line consisting of six points. Applying erosion of 1 would remove one point from each end of the
line and only four points would remain, as seen in Figure 3.8b. Figure 3.8c illustrates erosion of 2, where we would remove two points from each end of the line, leaving the line with only two points.

As we increase erosion, the line loses more points and loses more curvature. In the extreme case, the line loses all curvature, as seen in Figure 3.8c. The erosion operation introduces a margin around the regions. The visual effect is a variation on line hatching and may be desirable in some cases.

3.7.4 Edge lines

Edge lines contribute a lot of information about an object to the viewer [24, 29, 45]. We draw edge lines on the final image to add object silhouettes. We use Canny edge detection [4] to find object edges.

The performance of edge detection is highly image dependent. In some cases, if an image has many small details, edge detection can introduce too many small edge details. These small edge elements add too much noise to the output image and actually detract from the hatching line visibility. We set a percentage parameter that allows us to control the number of Canny edge pixels to retain. Reducing the percentage typically reduces the number of small edge lines returned by the Canny edge process.

We first apply Canny edge with its default double threshold values specifying the threshold range. We count the number of pixels that make up the Canny edges. We divide the number of Canny edge pixels by the total number of pixels in the original image to obtain a percentage. If the percentage is equal to or greater than the desired percentage, we exit and use the Canny edge pixels. Otherwise, we modify the values for the double threshold; we decrease the lower threshold and increase the upper threshold. We regenerate the Canny edges and check the pixel percentage again. The modification of the thresholds and checking of pixel percentage continue until we obtain the desired percentage or reach a maximum number of iterations.

From our testing, percentages in the range of two to five percent produced outlines without adding too much noise. In our results section, we use two percent for our edge lines setting.
Figure 3.9: Example effect of extension. Dotted lines indicated added elements.

3.7.5 Line extension

At the end of Phase 5, the hatching lines will span their segment regions and will not cross over into adjacent regions. We introduce the *Extend Lines* operation as an option to increase line length and to allow hatching lines to cross into adjacent regions, which will create a different visual effect. The extend lines operation is similar to the work by Li and Mould [33] where they produce artistic works through growing curve lines.

Figure 3.9a is a line that consists of six points. During the extend lines operation, we use the end segment of a line to linearly extrapolate a new point. Figure 3.9b illustrates the line with one single extension. We calculate the vector between the two existing points, $A$ and $B$. Since the length of the calculated vector will vary based on the distance between points $A$ and $B$, we normalize the vector length to one unit pixel distance. Taking a unit step in the direction of the calculated vector to point $B$, we add a new point $C$. Because we use linear extrapolation, the new line segments will always be straight. In one iteration, all lines are extended by one point on each end. This allows all lines to add the same number of points in each iteration. We repeat adding points to a line until we detect an intersection with an edge line or
another hatching line. Each line endpoint extends until it encounters an intersection. We discuss intersection detection further in Section 3.7.6.

### 3.7.6 Intersection detection

![Diagram](image)

Figure 3.10: Example of intersection detection. Dotted lines indicate elements added in this iteration. Red dotted elements indicate that an intersection is detected.

We implement intersection detection to prevent hatching lines from crossing edge lines during the extend lines operation. Since edge lines are intended to visually highlight the object silhouettes, any hatching lines that cross an edge line diminish the silhouette’s distinctiveness.

When hatching lines are extended they may cross edge lines if intersection detection is not applied. The *extend lines* operation relies on intersection detection to prevent hatching lines from crossing edge lines and from crossing other hatching lines. Figure 3.10a illustrates the starting state of two lines that each have three points. The extend lines operation lengthens both ends of hatching lines (Section 3.7.5). Figure 3.10b illustrates the result of one iteration of extend lines; the ends of both lines lengthen by one additional point, signified by dotted elements. At this point, the lines do not intersect so extension continues. Figure 3.10c illustrates the result of a second iteration of extend lines. The red point indicates a line intersection where the horizontal line would intersect the existing vertical line if the red point was added.
In this case, we do not add the intersecting red point to the horizontal hatching line and we stop the extend operation for that end of the line.

Although hatching lines are initially confined to each segment region, they may still overlap an edge line because SLIC regions do not always align with object silhouettes. To highlight the edge lines in these cases we combine line erosion and line extension on the core version of the image. The erosion ensures that the hatching lines are shrunk and reduces the likelihood of overlap with the edge lines. The line extension lengthens the lines until they encounter an intersection. We present and discuss this combination in Section 4.3.2.

### 3.7.7 Perturb Lines

Figure 3.11: Line Perturbation example. Point two has been shifted to the left. Dotted lines indicate original point and lines.

We added line perturbation to capture some small scale features that would not otherwise be visible through segment regions. We perturb points along the line based on the intensity of the point, $I_j$, as compared to the average intensity the neighbourhood, $I_{nj}$. We calculate the point intensity, $I_j$, by applying a Gaussian filter with a kernel size of $J$ centered on the point of interest. We use a value of 3 pixels for $J$ in our implementation. The neighbourhood intensity, $I_{nj}$, is calculated by applying a Gaussian filter of kernel size $N$ centered on the point of interest, where $N$ is larger than $J$. For our implementation, we use a value of 11 pixels for $N$.

If the $I_{nj}$ is greater than $I_j$, then point $j$ is shifted to the right, perpendicular to the direction of the line. If the $I_{nj}$ is less than $I_j$, then point $j$ is shifted to the left, perpendicular to the direction of the line as seen in Figure 3.11. Since lines in a segment region are roughly parallel, we traverse the points on each line in the same general direction so that the left and right orientations are the same for all lines.

**Ensuring sufficient point density:** During the implementation of line perturbation, we observed that in some cases the distances between the points were further
apart. Recall that lines are created from points intersecting the edges of mesh triangles (Phase 4). The distances between the intersecting points will vary depending on the size of the triangles and the trajectory of the lines. We reduce inter-point space variability by adding points to the line. The process finds the distance between two consecutive points in a line and compares the distance to a threshold $K$. If the distance between the two points is greater than $K$, we add points to guarantee the maximum distance between points is less than $K$. We perform this operation prior to line perturbation to increase line point density and ensure a more consistent distribution.

### 3.8 Summary

In this chapter, we presented an overview of our process and then looked at each step in detail. Our workflow is a combination of many other techniques in sequence. Where appropriate, we provided insight into our decisions.

In the next chapter we present the results. We look at the core version plus each line refinement in isolation. Section 4.2 presents how each line refinement contributes to the output image. In contrast, Section 4.3 presents the results of combining the line refinements. We present the combinations that improved the image recognizability or increased the image aesthetics.
Chapter 4

Results and Discussion

In this chapter, we present the rendering results of the line refinements in isolation and in combination. We discuss our evaluation criteria and then use it to assign a rating for each operation. We provide a rationale for our ratings and provide commentary about image output features that are not necessarily captured by the evaluation ratings. We explore the “Success” cases to highlight image categories and features where our process functions well. We look at the “Failure” cases to explore under what circumstances our process functions poorly and discuss our interpretation of why failures occurred. We discuss the performance of the process and look at the execution time and memory requirements. Lastly, we explore limitations of our process.

To provide continuity and facilitate comparisons, we have selected seven representative images to evaluate throughout the chapter. The images were selected because they have both small and large features, which we consider in our evaluation criteria. In our sample, we include variety in terms of types of textures, tonal and colour gradients, and contrast levels, either within an image or across multiple images. These seven input images are available in Figure 4.1. In Sections 4.2.9 and 4.3.5, we provide further examples with different input images to show how our process performs for other images.
Figure 4.1: The seven representative images used to compare line refinement effects.
4.1 Evaluation Criteria

Assessing the results of our process is challenging because there are no clear objective measures to use. As stated by Hall and Lehmann [18], there is no universal measure of art in terms of “good” and “bad”. Any assessments are highly dependent on culture and on the artist’s intentions, and they will likely evade definition in the field of NPR. Assigning numerical scores doesn’t always make sense, however, the intent of our scores is to provide some means of comparison. We provide our subjective assessment of the output images. The scores we assign may not align with others who view and assess these images.

We have defined two main criteria for our assessment: Recognizability and Aesthetics. We present the criteria and sub-criteria in Table 4.1. For each, we assign a score from 1 to 10, where 1 is indicates the poorest performance and 10 is the best. Note however, that no measurements were made, we simply use the scores as a way to discuss our relative rankings of the images.

**Recognizability:** The criteria recognizability looks at how well the process transfers features from the original input image while adhering to hatching line style. Recognizability consists of two components: Small scale structure and large scale structure. Small scale structure focuses on how well small features can be identified in the output image, e.g., the eyeballs in the Baby image (Figure 4.1c) or the windows of the Skyscraper (Figure 4.1g).

Large scale structure considers how well large features can be identified in the output image. We include main subjects and significant features of the main subject as large scale structures. For example, the silhouette of the Stop sign in Figure 4.1g and Bathtub in Figure 4.1a are considered large features. A qualitative way to consider large features is to ask whether the viewer can identify the main subject in the output image. In some cases, large scale features include gradients of significant spaces from the original image, e.g., the sky from the Elephant image (Figure 4.1f). Sometimes the silhouette of an object helps identification, while in other cases the hatching lines alone are sufficient to illustrate the object.

Although we distinguish between small scale and large scale features, there will likely be objects and features that fall somewhere in between. For simplicity, we will categorize these features as being small, for example the letters of the Stop sign (Figure 4.1d) and the arms of the Athletes (Figure 4.1f).
## Recognizability and Aesthetics: Small and Large Scale Structures

<table>
<thead>
<tr>
<th></th>
<th>Recognizability</th>
<th>Aesthetics</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Small</td>
<td>Large</td>
</tr>
<tr>
<td>Core</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>+ Line density</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>+ Line width</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>+ Reorienting lines</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>+ Erosion</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>+ Edge lines</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>+ Extension</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>+ Perturbation</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

Legend: Poorest 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 Best Performance

Table 4.1: Assigned subjective scores for each isolation line refinements based on our evaluation criteria. Small = Small scale structure and Large = Large scale structure.

### Aesthetics:

It is difficult to gauge aesthetics since there is no universal assessment criteria. This aspect is quite subjective and our score reflects our assessment of the visual appeal of the output image. These are the general characteristics I prefer:

- **Cohesion**: Regions that are similar should have similar line orientation and line characteristics.
- **Novelty**: The core version represents the images in a certain way. If the line refinements provide a different way to viewing the image, I generally prefer it.
- **Non-random**: Although random can be considered novel, I do not appreciate random images very much.
- **Interesting**: If the image made me go “Ooooyyyyy”. I tended to want to look at these images for more than 30 seconds upon initial viewing.

### 4.2 Isolation variants

In this section, we first explore the output images resulting from the core version. Recall that our core version segments the image into regions, populates the regions with hatching lines, and then reassemble all the regions to produce the output image.
Figure 4.2: Overview of the core version and each line refinement in isolation applied to the Stop sign image
We subjectively assess the output images based on our evaluation criteria (Section 4.1) and discuss the assigned scores. Since the core version produces an output image without applying any line refinements, it is the starting point for applying all other line refinements and generating the other image variants. Ideally, line refinements improve the output image as compared to the core version output.

Next, we isolate the effect of each line refinement by combining the core version with each line refinement in turn, and then we re-evaluate the resulting output images. All the line refinements applied in isolation improve the recognizability of the images and most also improve aesthetics, however, the changes are subtle.

Figure 4.2 provides a panel of eight images representing the core version and each of the line refinements for the Stop Sign image. We present this panel to allow a side-by-side comparison of the isolated line refinements for a single input image. We selected the Stop sign for this initial comparison because it has a little bit of everything: large scale features, e.g., the silhouette of the stop sign is clearly defined, small scale features, e.g., the letters on the sign, the image background is simple and uniform, and there are some additional fuzzy details in the bottom left corner.

From the panel, we can see that each line refinement results in a different visual effect, which we will discuss in the following sections. In some cases, the line refinement improves certain aspects, such as contrast or provides aesthetic improvements. In other cases, the line refinements worsen certain aspects, such as making the image too light or too dark, or blurring certain edges. When discussing each of the isolated line refinements in later sections, we present panels of the seven representative images to demonstrate the results across different images. Panels similar to Figure 4.2 for each of the other six representative images are available in Appendix A.1.

4.2.1 Core version

Figure 4.3 displays the output of the core version for our seven representative images.

We assign the core version images a recognizability score of 1 and aesthetic score of 2 (Table 4.1). We observe that the core version does poorly at transferring small scale features from the input image, e.g., the eyes of the Baby are not apparent (Figure 4.3c) and the limbs of the Athletes cannot be identified (Figure 4.3f). Some large scale structure information is transferred, allowing us to see the feature outlines in some of the images, e.g., the Stop sign's octagon (Figure 4.3d) and the Skyscraper's exterior...
Figure 4.3: Core version output of the seven representative images.
shape (Figure 4.3g), but other images remain ambiguous. We note that most of the images transfer enough structural information to indicate the presence of features, e.g., Elephant (Figure 4.3b), Stop sign, and Skyscraper. However, some of the images are entirely abstract, e.g., the Baby and Athletes.

### 4.2.2 Core + Line density (Ld)

Figure 4.4 presents the line density operation applied to the seven representative images. This variation improves recognizability by transferring more large structure information than the core version. We subjectively assign this variation a recognizability score of 2, predominantly due to the large scale structure improvements, and an aesthetics score of 3.

In this variation we modify the hatching line spacing based on each region’s average intensity; lighter regions will have lower line densities while darker regions will have higher line densities. Stylistically, the line density variation reduces the amount of ink coverage on the canvas because it can only reduce the number of lines within a region; it cannot increase the number of lines beyond the initial set created in the core version. Varying the line density improves boundary region separation, allowing us to pick out the large scale silhouettes of the Baby, Stop sign, and Hands (Figures 4.4c, 4.4d, 4.4e).

Changes in line density help to differentiate darker regions from lighter regions. As an example, consider the Elephant image (Figure 4.4b); we see that the sky is a lighter tone than the elephant and the ground. As we move upward, the sky progressively lightens.

However, this variation does not improve small scale structure transfer very much. Although the eyes of the Baby are more pronounced (Figure 4.4c) and the Athletes image now presents darker regions (Figure 4.4f), neither conveys much more information than the core version.

### 4.2.3 Core + Line width (Lw)

Figure 4.5 presents the line width operation applied to the seven representative images. The Line width variation modifies line width based on average region intensity;
Figure 4.4: Line density operation in isolation on the seven representative images.
lighter regions have thinner lines while darker regions have wider lines. We subjectively assign this variation a similar score to the line density variation: a recognizability score of 2 and an aesthetics score of 3.

Contrary to the line density operation, line width does not reduce the number of lines within a region; instead it modifies the thickness of the existing lines, which may increase or decrease the effective ink coverage on the canvas. The resulting line thickness can be less than or greater than the line width produced in the core version.

This variation does a reasonable job of differentiating darker regions from lighter regions. The feature boundaries of the Baby (Figure 4.5c), Stop sign (Figure 4.5d), Hands (Figure 4.5e), and Skyscraper (Figure 4.5g) are more pronounced and, as a result, highlight the large scale structures. In the Hands image (Figure 4.5e) in particular, the silhouette is accentuated effectively.

It is also able to show intensity gradients better than the core version, as can be seen in the Elephant image (Figure 4.5b), where the hatching lines capture the sky intensity gradient. We observe the regions lightening as you proceed from the middle of the image to the top. In the Bathtub image, we observe that the wall intensity variation (Figure 4.5a) is captured by the hatching lines. The background of the Skyscraper also demonstrates an intensity variation (Figure 4.5g) by presenting thinner lines in the lighter regions of the sky.

However, in some cases the additional line width detracts from the image by making regions too dark. The silhouette of the Elephant is quite dark, merging with the foreground grass and making the legs disappear (Figure 4.5b). As a comparison, we note that the legs of the elephant are better defined in the line density variation, Figure 4.4b. In the Stop sign image (Figure 4.5d), the additional line thickness cause the letter boundaries to be somewhat lost.

4.2.4 Core + Extension (Ex)

Figure 4.6 presents the line extension operation applied to the seven representative images. In our core version (Figure 4.3), the hatching lines approach region boundaries, but never cross them. As a result, there is typically a visible whitespace gap between the regions, due to pixel quantization. The line Extension operation lengthens the lines beyond their region boundaries and removes the whitespace gap (Figure 4.6). The line extension operation in isolation does not lengthen the lines by very much
Figure 4.5: Line width operation in isolation on the seven representative images.
because they quickly intersect with lines from neighbouring regions. When the lines meet, they form closed shapes.

The line extension operation makes certain surface gradients more continuous, e.g., the left wall of the Bathtub (Figure 4.6a) and the sky in the Elephant image (Figure 4.6b) are uninterrupted by whitespace. In many cases, the line extension gives an impression of melted lines because they blend into one another. Otherwise, the line extension operation produces results similar with the core version. We therefore assign the line extension variation a recognizability score of 2 and aesthetics score of 2.

4.2.5 Core + Erosion (Er)

Figure 4.7 presents the line erosion operation applied to the seven representative images. We implement the Erosion operation to shorten hatching lines in each region. In the Erosion operation, we can specify the number of points to erode from each line endpoint. For example, applying erosion of 10 will remove 10 points from both ends of each line. The additional white space give the images a distinct visual style, almost impressionistic. The limit of the erosion operation is to remove all but the two middle points, but this degree of erosion is only useful when combined with the extension operation, which we will present later in Section 4.3.

We subjectively assign the erosion variation a recognizability score of 2 and an aesthetics score of 4. It is on par with line density (Section 4.2.2) and line width (Section 4.2.3) in terms of large scale structure transfer. For example, we can identify the silhouette of the Elephant (Figure 4.7b) and the outline of the Stop sign (Figure 4.7d). However, the Baby is still hard to identify (Figure 4.7c) and the Athletes do not have recognizable structures (Figure 4.7f).

4.2.6 Core + Perturbation (Pt)

Figure 4.8 presents the line perturbation operation applied to the seven representative images.

In our core version, we draw straight and contour hatching lines. Since the hatching lines span an entire segment region, the lines cannot represent features smaller than the segment size. We observe that line perturbation can help identify significant
Figure 4.6: Line extension operation in isolation on the seven representative images.
(a) Bathtub  (b) Elephant  (c) Baby  
(d) Stop sign  (e) Hands  
(f) Athletes  (g) Skyscraper  

Figure 4.7: Line erosion operation in isolation on the seven representative images.
boundaries from the original image. For example, the line perturbations at the outer boundary of the Stop sign mimic an edge line (Figure 4.8d), mostly visibly along the top of the octagon. However, this effect can get lost if line hatching density is too high, as seen along the right edge of the octagon. In the Athletes image (Figure 4.8f), the line perturbations suggest face outlines when observed closely but we cannot determine what the shapes are because the perturbed lines are engulfed by all the other lines.

We subjectively assign the line perturbation operation a recognizability score of 3 and aesthetics score of 2. The line perturbations improve the boundaries of the large scale structures by highlighting them, e.g., the Elephant (Figure 4.8b), the Hands (Figure 4.8e), and the Skyscraper (Figure 4.8g) all have their silhouettes enhanced. The perturbations also add small scale structures in some cases, however, they are typically not enough to make a meaningful difference. The Bathtub (Figure 4.8a) and the Athletes (Figure 4.8f) have visible line perturbations, but they are not enough to indicate the actual feature. In some cases, the line perturbations detract from the image. In particular, the Skyscraper windows are not representative of the original image (Figure 4.8g).

4.2.7 Core + Reorienting lines (Re)

Figure 4.9 presents the reorienting lines operation applied to the seven representative images.

We introduce the reorienting lines operation to group similar regions and contrast dissimilar regions. The reorienting lines operation causes hatching lines in adjacent regions with similar properties to align their directions. In contrast, adjacent regions with dissimilar properties will approach perpendicular alignment.

This refinement is particularly noticeable when the background of an image is different from the foreground objects. The backgrounds of the Elephant (Figures 4.9b), Stop sign (Figures 4.9d), and Hands (Figures 4.9e) images all exhibit improved levels of cohesion. The lines in the background are not all colinear, however, their orientation makes them appear intentionally placed. As a consequence, the boundaries of the larger structures are highlighted. In these cases, the image backgrounds appear to have motion, such as in the Elephant, the Stop sign, and the Hands images. This effect is introduced by the reorienting lines algorithm, and it typically occurs in large
Figure 4.8: Line perturbation operation in isolation on the seven representative images.
homogeneous connected regions. Conversely, images without large homogeneous areas do not benefit as much from the reorienting lines operation. For example, the Baby (Figures 4.9c) and Athletes (Figures 4.9f) images are no more recognizable than their core versions.

Although the resulting images do not have improved recognizability from reorienting lines, the improved background cohesion makes them appear intentional and adds visual interest. We subjectively assign a recognizability score of 2 and aesthetics score of 3.

4.2.8 Core + Edge lines (Ed)

Figure 4.10 presents the edge lines operation applied to the seven representative images. Recall that the edge lines operation is intended to outline the object silhouettes (Section 3.7.4). The operation adds edge lines at high contrast boundaries, enhancing the large scale structures by accentuating the boundaries, e.g, the Bath-tub (Figures 4.10a), Stop sign (Figures 4.10d), and Hands (Figures 4.10e) images have significantly more apparent silhouettes. In some cases, small scale structure representation is also improved. The letters in the Stop sign (Figure 4.10d) are encased in contoured boundaries, making them more apparent. Close inspection of the Skyscraper reveals that the window structure has been accentuated (Figure 4.10g). Face details in the Athletes image (Figure 4.10f) are also transferred over, but they are almost concealed by the density of surrounding hatching lines.

Overall the edge lines improve small scale and large scale structure transfer. They accentuate boundaries and hint at small scale features. We subjectively assign a recognizability score of 3 and aesthetics score of 3.

4.2.9 Summary of isolation variants

At the beginning of the thesis, we were looking to generate images with the following characteristics:

- Near parallel straight and contour lines
- Short hatching lines
Figure 4.9: Line reorientation operation in isolation on the seven representative images.
Figure 4.10: Edge line operation in isolation on the seven representative images.
(a) Cat + Lw  
(b) Cat + Pt  
(c) Star shape + Lw 

(d) Brick + Ed  
(e) Horse + Lw  
(f) Horse + Er10 

(g) Headlight + Lw  
(h) Droplet + Lw  
(i) Droplet + Pt 

(j) Field Hockey + Ld  
(k) Field Hockey + Lw  
(l) Field Hockey + Pt 

(m) Yield sign + Re  
(n) Butterfly + Lw  
(o) Butterfly + Pt 

Figure 4.11: Additional sample isolation operation images (1 of 2).
CHAPTER 4. RESULTS AND DISCUSSION

(a) Spiral + Pt  (b) Skateboard + Lw  (c) Parrot + Lw

(d) Barn + Pt  (e) Abstract + Lw  (f) Abstract + Re

(g) Gymnast + Ld  (h) Subway + Ld  (i) Subway + Pt

(j) Beach + Ld  (k) Brush + Pt

Figure 4.12: Additional sample isolation operation images (2 of 2).
• Line direction changes exclusively delineate feature boundaries
• Monochrome lines
• Tone intensity conveyed through line thickness
• Line perturbations convey small structure information
• Edge lines provide silhouette definition

The core version generates output images that achieve only three of these criteria: near parallel straight and contour lines, short hatching lines, and monochrome lines. Clearly there is room for improvement.

In this section, we discussed the results of the line refinement operations when applied in isolation on top of the core version. We subjectively scored each line refinement based on our evaluation criteria: recognizability and aesthetics. We identified how the operations improved over the core version, and when appropriate, we compared operations against each other. Generally the line refinements in isolation showed small improvements over the core version. In most cases, these visual changes were subtle. In no cases were the results of the line refinements worse than the core version, based on our subjective assessment.

The line density and line width are fairly effective at showing the boundaries of tonal changes. However, since they both rely on the greyscale intensity, they sometimes fail to show colour variation. The edge lines operation provides silhouettes based on feature boundaries from the original image. However, with the full density of hatching lines, the edge line effects are somewhat diminished. The line extension and line erosion operations each provide a unique style variation, but they may provide better results when used in combination, as described below. The reorienting lines operation tends to force regions with similar intensities to align the orientation of their hatching lines, making them near colinear and highlighting boundaries in the original image.

For recognizability, we increased our scores of only two line refinements for the small scale structure transfer: edge lines and line perturbation. In general, we found everything improved large structure transfer; edge lines and line perturbation were the best in this regard. With respect to aesthetics, we note mild improvement in most instances. In particular, line width, reorienting lines, and line erosion add distinctive visual interest. However, we found none of the line refinements in isolation satisfying. They were able to improve only one aspect of the output image, but this was usually not enough to make a significant change.
Figures 4.11 and 4.12 provide additional characteristic output images with various line refinements applied in isolation. They illustrate that the isolation line refinements do work on a wide range of images with reasonable levels of success. We include multiples versions of some images to additionally contrast different line refinement effects.

**Combining line refinements** We consider whether combining line refinements could achieve better result and briefly discuss likely pairings. Line density and line width both rely on average region intensity, and both can be used to convey tone. Combining these two refinements will amplify the visual effect. Darker areas will have thicker lines and retain most of their hatching lines. Lighter areas will have fewer hatching lines and the lines will be thinner.

The combination of line density and line extension is another interesting pairing. Its effect is a little harder to assess and will be highly dependent on the input image. The line density operation will tend to reduce the number of hatching lines in lighter areas. As a result, the line extension operation will have more space to lengthen lines before they intersect with another line. Darker regions will retain most of their hatching lines, therefore, line extension will have less an impact in these areas.

Line erosion and line extension is another likely pair. Line erosion will remove points from the ends of hatching lines, with higher the levels of erosion removing more points. The ultimate erosion will reduce the line to its middle two or three points, effectively removing most of the line’s curvature. Recall that line extension linearly lengthens the lines. When combined, line erosion and line extension will tend to produce straighter hatching lines. With enough erosion, all hatching lines will become straight. However, the effect will be distinct from drawing straight lines directly on the segment regions, as will be seen in later sections.

Combining the line refinements will likely be more effective at producing recognizable output images that are aesthetically pleasing and at meeting our desired image characteristics. We look at different combinations in the next section.

### 4.3 Complex variants

In this section, we present the results of chaining the line refinements. The number of possible combinations is quite large. We selected a subset that we thought were
likely to produce desirable output results based on our experience with the outputs in isolation. We present the results for each combination and discuss their performance relative to our evaluation criteria (Section 4.1).

Recall that we use the following abbreviations to shorten the names of the combinations. For example, our Complex base consists of the core version in addition to Reorienting lines + Line density + Line width operations, which we abbreviate to Re + Ld + Lw.

Ld: Line density
Lw: Line width
Er#: Line erosion; # indicates the degree of erosion. A value of zero indicates no erosion while a value of 200 indicates we remove 200 points from each end of lines.
Ex: Line extension
Re: Reorienting lines
Pt: Line perturbation
Ed: Edge lines

In Table 4.3 we present our subjective scores for each combination. Figure 4.13 presents sample results of the selected complex combinations applied to the Hands image, to allow comparison across combinations. Each of these combinations will be further explored in the following sections.

4.3.1 Complex base version (Core + Re + Ld + Lw)

Through experimentation, we identified that the combination of Reorienting lines + Line density + Line width (Re + Ld + Lw) applied to the core version produced consistent improvement over the isolation variants. We therefore use it as the starting point for further refinement.

Figure 4.14 presents the complex base operation applied to the seven representative images. All the images exhibit improved feature boundary definition, e.g., the window edge of Bathtub (Figure 4.14a), the Stop sign silhouette (Figure 4.14d), and the finger edge boundaries in the Hands image (Figure 4.14e). Tonal gradients are better expressed; these are most visible in the left wall of Bathtub image, the sky of
Figure 4.13: Overview of the complex base and each combination of complex line refinements applied to the Hands image.
## CHAPTER 4. RESULTS AND DISCUSSION

### Recognizability

<table>
<thead>
<tr>
<th></th>
<th>Small</th>
<th>Large</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complex base (Re + Ld + Lw)</td>
<td>4</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>+ Ed</td>
<td>5</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>+ Er0 + Ex</td>
<td>3</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>+ Er0 + Ex + Ed</td>
<td>5</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>+ Er200 + Ex</td>
<td>3</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>+ Er200 + Ex + Ed</td>
<td>6</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>+ Er0 + Ex + Pt</td>
<td>4</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>+ Er200 + Ex + Ed + Pt</td>
<td>6</td>
<td>10</td>
<td>8</td>
</tr>
</tbody>
</table>

Legend: Poorest 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 Best Performance

Table 4.2: Assigned scores for each complex operation combination based on evaluation criteria. Small = Small scale structure and Large = Large scale structure.

the Elephant image (Figure 4.14b), the child’s chest in the Baby image (Figure 4.14c). Smaller scale features are easier to see in some cases, as seen in the eyes and mouth of the Baby, the trunk and legs of the Elephant, the windows of the Skyscraper, and the fingers of the Hands. The line orientation continues to help differentiate the background from the foreground when these have different characteristics, e.g., Elephant body has different line orientation from the background sky and the foreground grass, the Stop sign has different line orientation from the background sky, and the hatching lines composing the Hands have different orientation from the lines composing the background. However, not all images were improved enough to make them clearly recognizable. Despite additional information, such as the head being darker, there is still not enough information to clearly see what is in the Athletes image. Overall, we assign a recognizability score of 5 and aesthetics score of 6 to the complex base version.

### Adding edge lines (Complex base + Ed):

Figure 4.15 presents the complex base + Ed operation applied to the seven representative images.

In this variation, we overlay the edge lines on top of the complex base. The edge lines provide additional silhouette information, highlight large and small scale
Figure 4.14: Example complex base combination (Core version + Re + Ld + Lw) applied to the seven representative images.
features, as well as transfer some internal edges lines that were not present before. In Figure 4.15a, we see that the main object in the image is an oval Bathtub and not a table, the edge of the tub has a faucet, the tub has legs, and the window may have other features such as window panes. The Baby has identifiable arms (Figure 4.15c), chin, nose, mouth, ears, better defined eyes, and hints of hair. The Stop sign (Figure 4.15d) has a white strip along its perimeter and has two visible screw holes, one at the center near the top and one near the bottom of the sign. The fingers of the Hands (Figure 4.15e) are clearly defined and the hands have identifiable palm creases. For the first time, we see definitive human shapes in the Athletes image (Figure 4.15f); we see they have arms, feet, and heads, and that there is clearly an interaction between the two people. Finally, we see a second building in the Skyscraper image (Figure 4.15g) and the windows are better defined.

We identified a weakness in that the edge lines and segmentation regions do not always coincide. We see that the edge lines cross some of the hatching lines, e.g., the Stop sign’s outer boundary has hatching lines that cross the edge lines, and the hatching lines cross the window edges in the Skyscraper image. Using the current edge line parameter setting (i.e., two percent), we also noted that some edge lines do not fully enclose a feature. For example, the right arm silhouette of the baby is not present and the legs of the Athletes are not completely defined.

Considering these results, we note an overall improvement over the complex base variant. We subjectively assign this variant a recognizability score of 7 and aesthetics score of 7.

4.3.2 Complex base + Er + Ex

In this section, we apply the erosion operation and extension operation in sequence after the complex base.

We produce results with three different erosion values. First we look at erosion of zero (Er0) which effectively means no erosion. We introduce this as a baseline so that we can compare with the higher erosion levels, i.e., erosion of 10 and erosion of 200. When we apply the extension operation, the hatching lines are extended through linear extrapolation. The lines will retain all their contours and have the endpoints lengthened in a straight line.

We look at the impact of increasing erosion to 10 (Er10), which removes ten
Figure 4.15: Example complex base + Ed combination applied to the seven representative images.
points from each end of all lines. By removing points from the hatching lines, we remove contours that would exist in that portion of the lines. When we apply the extension operation, we see a slight difference from the $E_{r0} + E_{x}$ variant; with $E_{r10}$, the hatching lines will have lost some of their contours.

Finally we compute erosion with a value of 200 ($E_{r200}$) to show the effect of removing virtually all points from the line. Since our lines have fewer than 400 points, $E_{r200}$ means that we effectively remove all points except for the middle points (which must be retained). If the number of points in the line is even, we retain two middle points, while lines that have an odd number of lines retain the middle three points. When we apply the extension operation, the result is that most lines have lost all their contours and are very straight.

**Complex base + $E_{r0} + E_{x}$**

Figure 4.16 presents the complex base + $E_{r0} + E_{x}$ operation applied to the seven representative images.

We see that the extension operation extends the hatching lines. The lines are no longer relatively parallel, but instead form feather-like shapes. The feather shapes create cell structures, effectively causing the images to take on a different style and resembling feather mosaic tiles.

In some cases, the feathers hint at structure that is not actually present, e.g., a rectangular box near the top portion of the Elephant image (Figure 4.16b). We also note that the fingers appear to have odd cell structure in the Hands image (Figure 4.16e). In some cases, the feathers suggest certain areas are solid, e.g., the opening of the Bathtub looks solid (Figure 4.16a) and the background of the Skyscraper appears to have substance beyond what is in the original image (Figure 4.16g). Edges that were straight now appear frayed, e.g., the Bathtub window edge, Stop sign edges (Figure 4.16d), the silhouette of the Hand, and the silhouette of the Skyscraper. The cell structures reduce the readability of text, as seen by the letters on the Stop sign. Despite this, the cell structure gives the word a different character; they look more disorganized and almost evil.

Despite all these downsides, many of the large scale features are transferred and the images are almost as recognizable as those from the complex base versions. We subjectively assign this variant a recognizability score of 5 and aesthetics score of 8.
Figure 4.16: Example complex base + Er0 + Ex combination applied to the seven representative images.
Adding edge lines (Complex base + Er0 + Ex + Ed): Figure 4.17 presents the complex base + Er0 + Ex + Ed operation applied to the seven representative images.

We apply edge lines to the complex base + Er0 + Ex variant. Other than cell structures, the results resemble those from the complex base + Ed seen in Figure 4.15.

Similar to complex base + Ed, we note that the edges lines and segmentation regions do not always coincide. We see that the edge lines cross some of the hatching lines, e.g., on window edges of the Bathtub image (Figure 4.17a) and on the Stop sign’s outer boundaries (Figure 4.17d).

As before, the edge lines provide additional silhouette information, transfer large and small scale features, and transfer some internal edge lines. The edge lines also resolve the frayed edges seen in complex base + Er0 + Ex by confining some of the hatching lines, as seen in the Bathtub rim and its window edge, the Elephant silhouette (Figure 4.17b), the Baby’s chin and eyes (Figure 4.17c), the Stop sign edges, the finger silhouettes in Hands image (Figure 4.17e), and the silhouette of buildings from the Skyscraper image (Figure 4.17g). Based on these improvements, we assign this variant a recognizability score of 7 and aesthetics score of 8.

Comparison of Er0, Er10, and Er200

We compare the effect of three levels of erosion (0, 10, and 200). In Figure 4.18 (b), we see a close-up of the Stop sign image with + Er0 + Ex applied. Figure 4.19 and Figure 4.20 similarly show the effect with Er10 and Er200, respectively.

Using Er0 as a baseline, the Er10 operation causes the hatching lines to lose some contour and appear straighter, but the difference is subtle. With larger erosion levels, the change would gradually become more pronounced until Er200, where the lines are completely straight, because none of our lines have over 400 points.

Complex base + Er200 + Ex

Figure 4.21 presents the complex base + Er200 + Ex operation applied to the seven representative images.

Erosion 200 removes virtually all points from a line except for the middle two
Figure 4.17: Example complex base + Er0 + Ex + Ed combination applied to the seven representative images.
or three points. The extension operation linearly lengthens the hatching lines, most of the time producing completely straight lines, and generating cell structures that
appear jagged.

Similar to the complex base + Er0 + Ex, some of the hatching lines cross feature boundaries, making them looked frayed. For example, the hatching lines along the Stop sign boundaries (Figure 4.21d) looked frayed, particularly the bottom left and the top borders where the hatching lines cross into the adjacent regions. Especially noticeable, the Skyscraper image (Figure 4.21g) has a single errant line that extends quite far into the sky, near the top of the building. Some smaller features no longer retain their original shapes; we see that the eyes of the Baby are warped (Figure 4.21c), letters on the Stop sign are misshapen, the fingers on the Hands (Figure 4.21e) appear to be made out of rectangular blocks, and the window gaps on the Skyscraper image look like teeth.

However, overall, the large scale structure is still present. Other than the Athletes, we can still identify the main components of each image. We find the misshapen features interesting because it seems to offer a warped perspective on the real world. We subjectively assign this variant a recognizability score of 5 and aesthetics score of 8.

**Adding edge lines (Complex base + Er200 + Ex + Ed)** Figure 4.22 presents the complex base + Er200 + Ex + Ed operation applied to the seven representative images.

In this variant, we add edge lines to the previous variant. The edge lines fix some of the small scale shapes that were damaged in complex base + Er200 + Ex variant, e.g., the eyes of the Baby (Figure 4.22c) now have the proper shape, the letters on the Stop sign (Figure 4.22d) better conform to expected shapes, the fingers on the Hands (Figure 4.22e) are not as jagged, the window gaps on the Skyscraper (Figure 4.22g) no longer resemble jagged teeth.

The edge lines also resolve the frayed edges seen in complex base + Er200 + Ex by confining some of the hatching lines. We see that the Bathtub rim and window edge are well-defined (Figure 4.22a), the Elephant silhouette is smooth and well formed (Figure 4.22b), the Baby’s chin and eyes have clean edges, the Stop sign edges mostly conform to an octagon, the finger silhouette in Hands once again resemble complete fingers, and the silhouette of buildings in the Skyscraper image are straight and smooth.

Of particular interest is that this variant creates additional white space. Recall
Figure 4.21: Example complex base + Er200 + Ex combination applied to the seven representative images.
that the extension operation lengthens lines until it encounters an intersection (Section 3.7.5). When we extend the hatching lines, they will stop lengthening when they encounter an edge line. In output images from this variant, we can see the edge lines stopping some of the hatching lines to highlight small features, e.g., the faces of the Athletes (Figure 4.22f) more clearly show that they have ears, noses, and lips. Compare this with the result of the complex base + Er0 + Ex + Ed version (Figure 4.17f) where the hatching lines overlap with the edge lines and the facial features are not as apparent.

For the same reason, the windows of the Skyscraper are also better defined than the windows in complex base + Er0 + Ex + Ed version (Figure 4.17g); the windows are generally darker while the window borders are lighter. We observed two other important instances where small features were made apparent by this variant. First, the holes at the top and bottom of the Stop sign (Figure 4.22d) are visible. Second, the Elephant’s (Figure 4.22b) eye is revealed. Overall, we assign this variant a recognizability score of 7 and aesthetics score of 8.

### 4.3.3 Complex base + Er0 + Ex + Pt

Figure 4.23 presents the complex base + Er0 + Ex + Ed + Pt operation applied to the seven representative images.

Line perturbations were introduced to transfer textures and small scale structures that are smaller than region sizes.

In our output images, the line perturbations are visible in some cases, but the effect is subtle. Due to the large number of surrounding hatching lines, the effect may be muted or lost. For example, in the Bathtub image (Figure 4.23a), the hatching lines around the faucet have been perturbed. Although the perturbations do not provide enough information to indicate the actual feature, it provides a visual cue that something is present. Also, along the far edge of the tub, the line perturbations have resulted in an outline of the tub rim that resembles an edge line.

The hatching lines of the Elephant image (Figure 4.23b) have some perturbations that indicate non-smooth surfaces on the elephant body. The perturbations have also highlighted the silhouette of the elephant along the perimeter of its stomach. Upon close inspection, we can pick out the hint of tusks from the line perturbations,
Figure 4.22: Example complex base + Er200 + Ex + Ed combination applied to the seven representative images.
however, this can be easily missed. In the Stop sign image (Figure 4.23d), the hatching lines around the letters and along the edge of the Stop sign border all exhibit perturbations that produce features resembling edge lines.

In other cases, the perturbations make the objects a little less clear. The Skyscraper image (Figure 4.23g) has many perturbed lines throughout. The window structure of the Skyscraper is messy, making it harder to identify the actual window structures.

The line perturbations add texture in places, however, they are not successful in providing enough information to provide a definitive advantage over the complex base variant. For example, in the Elephant image, the line perturbations are not enough to indicate grass in the foreground. Similarly, the hair is not sufficiently indicated in the Baby image. Although the line perturbations do transfer some small scale details, it is generally not enough to express the texture or indicate the feature. Overall, we subjectively assign this variant a recognizability score of 5 and aesthetics score of 6.

4.3.4 Complex base + Er200 + Ex + Ed + Pt

Figure 4.24 presents the complex base + Er200 + Ex + Ed + Pt operation applied to the seven representative images. This variant applies all of the line refinements to the complex base and inherits most of the characteristics from the previously discussed variants. For example, similar to the other Er200 combinations, the hatching lines are mostly straight. Similar to the other Ex combinations, the hatching lines produce cell structures that appear jagged. The edge lines provide the large scale structure and small scale transfer while confining hatching lines from extending beyond their feature boundaries.

The line perturbations provide some hints of unevenness in a surface, but generally do not provide enough information to indicate the actual texture. The line perturbations contribute small hints about the existence of features and in some cases provide an indication of a feature boundary. However, the edge lines also highlight feature boundaries and will typically overshadow the line perturbations when they overlap. For example, the silhouette of the Elephant (Figure 4.24b) is fully defined by the edge lines. The tusks and the stomach perturbations are completely overshadowed by the edge lines. However, the hatching lines across the body of the elephant still exhibit some perturbations that indicate a non-smooth surface. The edges of the Stop
Figure 4.23: Example complex base + Er0 + Ex + Pt combination applied to the seven representative images.
sign's outer boundaries are completely represented by the edge lines that completely overshadow the line perturbations (Figure 4.24d). The letters of the Stop sign, on the other hand, clearly exhibit both the edge lines and the perturbations because the refinements do not overlap.

Overall, we think this is the most successful variant based on our evaluation criteria. The images from this variant are the most recognizable and we subjectively assign a recognizability score of 8 and aesthetics score of 10.

### 4.3.5 Summary of complex variants

We investigated how combining the line refinements would affect the output results. Through experimentation, we found that the combination of Reorienting lines + Line density + Line width (Re + Ld + Lw) applied to the core version led to consistent improvement over all line refinements in isolation. We called this variant the complex base. We built on the complex base and added other line refinements.

We found that variants with the edge lines operation consistently offered better recognizability. The edge lines transferred large scale and small scale features well. In some cases, edge lines produced more whitespace, further highlighting feature boundaries. The erosion operation combined with the extension operation changed the character of the hatching lines. Increasing levels of erosion reduced the contour retained by hatching lines. At the limit, the hatching lines lost all contour and become straight lines, producing images with longer, more jagged lines. The line perturbations transferred textures, however, they were generally not sufficient to identify the type of texture seen in the original image. In combination with other refinements, line perturbations helped to accentuate high contrast boundaries.

We subjectively assigned the complex base + Er200 + Ex + Ed + Pt variant the best score. We found that it produced images with the highest recognizability and the best aesthetics, but these are subjective assessments and other viewers may prefer other variants.

Figures 4.25 and 4.26 provide additional characteristic output images with various complex line refinements applied. They illustrate that these combinations of line refinement are effective on a wide range of images and all show reasonable success. We include multiples versions of the some images to additionally contrast different
Figure 4.24: Example complex base + Er200ExEdPt combination applied to the seven representative images.
CHAPTER 4. RESULTS AND DISCUSSION

Figure 4.25: Additional sample complex operation images.
combinations of line refinements.

4.4 Comparison to other methods

To our knowledge, few other methods produced results similar to ours. In this section, we compare our results with two approaches that are closest in style.

4.4.1 Wei and Mould

Wei and Mould [52] generate line drawings using particle tracing. They trace particle paths from the left side of the canvas to the right side. Rules define when the particles split and when they are allowed to merge. The resulting images are formed by nearly horizontal lines. Additional tonal information is added through cross hatching.

Although both styles are monochrome line hatching drawings, they have very different feels. Their approach has extremely long wavy lines and we note the presence of feather-like structures where lines intersect. Most of the lines in their images trace a path from the left side of the image to the right side. Their process also has good line economy, particularly in the background portion of their images. In contrast, our process tends to produce shorter lines that dynamically change direction in almost every segment region. Economy of lines was not one of our goals, and, as a result, we produce by far more distinct lines.

In comparison, our results have better edge definition. Since we implement edge lines, we produce good boundary separation, e.g., the branches in the bird image 4.27c, the silhouette of the boat 4.28c, and the silhouette of the mountain 4.29c are much more distinct. Our images also have better tonal separation. For example, the left border of the bird image is better defined 4.27d, we are better able to separate sky from water in the boat image 4.28c, and we improved contrast between the darker castle against the lighter fields 4.29d.

Based on our evaluation criteria, Wei and Mould’s results score well for large scale recognizability. However, our results outperform in terms of small scale recognizability. The aesthetics score is difficult to compare because the image styles are sufficiently different.
Figure 4.26: Additional sample complex operation images (2/2).
4.4.2 Rosin and Lai

Rosin and Lai use an ordered dithering process that produces engraving style images of human faces. They construct a dither matrix by combining a horizontal dither matrix with a vertical dither matrix. The combined dither matrix is applied to the original image to create a cross hatching image. They use face detection to provide rough proxy geometry for the head, which they use to add curves to the cross hatching lines.

The resulting engraving images are clean and normally track from left to right, but allows for breaks in the engraving lines to produce different effects. The process can rotate the engraving lines to match the orientation of the face as shown in Figure 4.30b. Their process allows for generation of monochrome or colour output images.

Comparing images in Figure 4.30 and Figure 4.31, we see that their style is quite different from ours. On the scale of realism, their process effectively filters original
image, as a result, their images will be closer to photorealistic than ours. From their engraving images, the original subject could be identified. Our style is more artistic and closer in appearance to a hand-drawn picture, particularly because different image features result in different strokes orientations. For comparison to the high fidelity results from Rosin and Lai, we generated images with 1600-segments and note that these images are able to capture facial features better than the 400-segment versions. Figure 4.31c, with 400 segments, indicates that there is a human face but the smaller details are missing, such as the eyes and the shape of the nose. Figure 4.31d, with 1600 segments, provides more small scale facial detail. However, neither the 400 or 1600 segment versions of our images are as recognizable as Rosin and Lai’s results.

Based on our evaluation criteria, we would score Rosin and Lai’s images very highly in large scale and small scale recognizability. The aesthetics score is more subjective and will depend on the viewer.
4.5 Success and Failure Cases

In this section, we provide a list of image characteristics that produce good output images using our process. We then discuss the failure conditions we have observed in the images.

4.5.1 What makes an ideal image

From our experience, we observed that images with the following characteristics performed best.

- High contrast
- Well defined large features
- Not too many small features because these may appear as noise.
(a) Original Young girl image from Rosin and Lai [44]

(b) Rosin and Lai’s output image [44]

(c) 400 segment Complex base + Er0 + Ex + Ed + Pt

(d) 1600 segment Complex base + Er0 + Ex + Ed + Pt

Figure 4.30: Comparison of our work to Rosin and Lai’s results for the Young girl image
Figure 4.31: Comparison of our work to Rosin and Lai’s results for the Sports fan image

(a) Original Sports fan image from Rosin and Lai [44]
(b) Rosin and Lai’s output image [44]
(c) 400 segment Complex base + Er0 + Ex + Ed + Pt
(d) 1600 segment Complex base + Er0 + Ex + Ed + Pt
High contrast images tended to have well defined feature boundaries. These well defined regions increases the chance of edge lines being defined along the boundary. Recall that edge lines will prevent hatching lines from lengthening in the case of an intersection. Edge lines defined over a high contrast boundary will enforce that boundary and enhances its visibility.

4.5.2 Failure conditions

In our process, we observed a few failure cases. We explore these failure cases and discuss if we can remediate these issues or whether the problem is inherent with our process. In particular, we explore conditions leading to missing regions in the output images, how our process operates on low contrast images, and output results of images that have poorly defined structures.

Missing regions

In our output images, we occasionally see that a region does not contain any hatching lines. These blank regions are caused by the BFF process failing to create a mapping between the region mesh and its distortion to the standardized shape.

Recall that in our core version, we draw lines on the standardized shape and then transfer the lines back to the region mesh (Section 3.6). The transfer mechanism relies on the mapping created between the region mesh and the standardized shape. Very rarely, BFF will fail to create the transformation and we are unable to transfer the lines from the standardized shape back to the region mesh.

We observed that regions with irregular shapes tended to cause BFF to fail in this way. For example, the barn image in Figure 4.32a contains a blank region near the middle of the image. Figure 4.32c is a close-up of the empty region. Notice that the region has an appendage that sticks out from the main body of the region. Similarly, in the beach image, Figure 4.32b, the region beside the left arm of the person is blank. Figure 4.32d is a close up of the empty region. Notice that the region has three narrow appendages extending in different directions beyond the main body.

We noticed blank regions in both 400 and 1600 segment region images. In most cases, there is only one blank region per image. In the 1600 images, the regions were more regularly shaped, so some factor other than irregularity may also be at play.

One potential solution is to detect when these blank regions occur and change our
pseudorandom seed for processing the image. We have noticed that this changes the resulting segmentation, and, in most cases, the blank region does not reoccur.
CHAPTER 4. RESULTS AND DISCUSSION

Low contrast

Our process relies on image greyscale intensity for multiple line refinements (line density, line width, and line orientation). When images have low contrast, the line refinements do poorly: line density will not change over multiple regions, line width will not vary, and line orientation will tend to align across all the regions.

Figure 4.33a presents an image with relatively low contrast. When we apply the complex base variant to the image, we produce Figure 4.33b. Although certain features do stand out, e.g., the arms and the shadows, it is hard to make out any other features. If we apply one of the better scoring variants, complex base + Er0 + Ex + Ed + Pt, the edge lines provide silhouettes of the arms, wings, and head (Figure 4.33c). However, the hatching lines themselves do not cohesively identify these features, and contribute little to recognizability of the image.

This weakness is inherent in our approach and we do not have an easy fix. It is possible that other measures such as using separate colour channels or vector fields may resolve this problem but would need significant further exploration.

Poorly defined structure

Images that have poorly defined structures cause our process some difficulty. The water caustics in Figure 4.34a fail from the very first step of the process: Segmentation (Figure 4.34b). The segmentation fails to capture the constantly changing structure of the light caustics. Applying complex base to the image produces effectively random shapes (Figure 4.34c). Applying the complex base + Er0 + Ex + Ed + Pt variant produces Figure 4.34d. Although the edge lines actually capture some of the light
caustics boundaries, most of the region boundaries do not coincide with the edge lines, resulting in a conflict between the edges lines and the hatching lines.

## 4.6 Performance

**Execution time:** In our process, we only need to generate the core version of each output image once. The output images for each isolation line refinement are generated by building on the core version instead of starting from scratch. Similarly we generate the complex base only once and then use it as the starting point for the other complex variants. The time required to create the core version and complex base can be amortized through reuse. Our results were generated with 400 segments per image.

For example, the Stop sign image is $722 \times 481$ pixels (347282 pixels total). It took a total of 22 minutes to generate the 8 images from Figure 4.2, one image for the core version and one for each of the isolation line refinements. The core version output image took seven minutes to complete while images for the seven isolation variants took about one minute each to generate.

The complex variants took a total of 15 minutes to generate. The complex base output image took 8 minutes while all the other variants took less than one minute to complete. The following list provides all the complex variants generated during our execution cycles.
• complex base
• complex base + Er0 + Ex
• complex base + Er0 + Ex + Ed
• complex base + Er0 + Ex + Ed + Pt
• complex base + Er10 + Ex
• complex base + Er10 + Ex + Ed
• complex base + Er10 + Ex + Ed + Pt
• complex base + Er200 + Ex
• complex base + Er200 + Ex + Ed
• complex base + Er200 + Ex + Ed + Pt

**Programming language:** We used Python to build our utility. The language has many useful libraries that we used for rapid prototyping. When executing the code, we allowed the built-in memory manager and garbage collector to perform their intended tasks. We did not modify their priority or schedule in any way. It is likely possible to improve overall performance through using a different programming language and changing the memory manager priority or scheduling.

**Higher segment counts:** We also experimented with generating images using higher segment counts. For example, when working with the Stop sign at 1600 segments, the isolation line refinements took 29 minutes and the complex variants took 15 minutes to complete. Given that this was an exploratory iterative process, these execution times became prohibitive. We chose to focus on the 400 segment images.

**Hardware:** We executed our code on a 2012 Macbook Pro with 16 GB of memory. Although most applications were shut down for the executions, this was not a dedicated machine. Other applications may have impacted the execution times for image generation. For example, a scheduled backup may have occurred, an application update may have run, and other miscellaneous jobs may have been scheduled. As a result, our execution times are rough averages.

### 4.7 Limitations

Over the course of the project, we made several design decisions. Although these decisions provided benefits, they also had trade-offs. We discuss these trade-offs now.
Reliance on segmentation: In order for our process to function well, the segmentation portion must be able to produce meaningful segments, e.g., the segment boundaries align with feature boundaries. If this does not occur, e.g., a low contrast image, then our process generates images that do not have good recognizability, as discussed in Section 4.5.2.

Slow performance: The current implementation of our process is relatively slow. If performance is a consideration, such as near real time or interactive applications, then our process may not be the right choice. Additionally, our process does not scale well with image size. Most our images were around 500k pixels in size. When tested against a 900k pixel image, execution times for isolation refinements went from 22 minutes to 1 hour. The complex variants went from 15 minutes to 36 minutes.

Reliance on high quality input images: Our process relies on images that have high contrast and well defined structures. As described in section 4.5.2, images not meeting these requirements tend to produce poor quality output images.

Transformation engine: In the current implementation, we use openly available utilities, BFF being one of them. During our testing, we found that BFF has certain limitations and cannot transform certain input region shapes. Ultimately, this limits what shapes our regions can be and, by extension, what types of segmentation algorithms we can use on the input image. Further investigations may find alternatives to BFF that overcome these limitations.

Reorienting considerations: The reorienting lines operation will change the line directions in each segment region to ensure that similar regions will have roughly the same orientation. However, this process does not consider the original major axis of the segment regions. As a result, the final line orientation may not follow the actual orientation of the feature. One potential solution is to additionally account for the major axis direction of the segment regions when reorienting lines.
4.8 Discussion

We developed a workflow that has five phases: segmentation, mesh generation, reshaping the segment regions to standardized shapes, drawing hatching lines on standardized shapes, and transferring hatching lines back to segment regions. We also have an optional line refinement phase. In theory, these phases should combine together to generate images in our desired output style. In practice, the underlying technology for each phase had quirks and limitations that restricted some of our design choices. We discuss the merits of our workflow and the most impactful limitation.

Plug and Play: In theory, each of the workflow phases are implementation agnostic. We chose the specific implementations because we wanted to use established or peer-reviewed algorithms as much as possible.

Currently we use SLIC0, which produces relatively compact regularly shaped regions. We could use a different segmentation algorithm. By changing the segmentation algorithm or changing the parameters of the segmentation algorithm, the resulting output image characteristics would change.

For reshaping of the segment region to a standardized shape, we use BFF. At the time, BFF was the only good alternative and writing our own conformal mapping tool was out of scope. However, we could theoretically use any tool that accepts a triangle mesh, applies conformal mapping to the region mesh, and then produces an output mesh representing the new shape. We anticipate that eventually additional tools will become available.

Impact of using BFF: One of the most important components of our process is the ability to reshape the segment region mesh into a standardized shape. With this ability, we can draw relatively simple lines on the standardized shape. When we transfer the lines from the standardized shape back to the segment region, the transformation process automatically introduces distortions that follow the segment region contours.

Our initial goal was to use BFF to transform regions with arbitrary boundaries to a standardized shape. We imagined that segmentation would create complex boundaries; imagine a ‘T’ shape region, and then transform that boundary into something simpler, e.g., an oval or rectangle. We realized quite late in our process that BFF had two important limitations.
CHAPTER 4. RESULTS AND DISCUSSION

The first limitation was that BFF could not consistently transform a mesh into an arbitrary shape, even a relatively simple shape such as an oval. With oval, BFF would invariably fail to produce a viable mesh. We found that the only shape that it could consistently use as a target was a circle. Consequently, the circle became our standardized shape.

The second limitation was that if the input mesh was too irregular, BFF also had problems producing a viable output mesh. For example, segment region meshes that had appendages, such as a ‘T’ shaped boundary, would cause BFF to fail. Depending on the compactness parameter setting, \( m \), the SLIC segmentation algorithm would produce regions with appendages while segmenting certain images. Generally, these segments would better conform to feature boundaries, however, BFF would fail more often. To remedy this problem, we had to produce more compact, more regularly shaped convex regions. We had to make the trade-off and use SLIC0 (Section 2.2.1), the variant that consistently produced compact regions. Otherwise we could not rely on BFF to perform our transformations.

Impact of the circle standardized shape: Theoretically, when we transfer lines from the standardized shape back to the segment region, the transformation will distort the hatching lines based on the contours of the segment regions as well as the boundary of the standardized shape. We envisioned that our process could provide a library of standardized shapes, for example, a square, a rectangle, an oval, and a triangle. Each shape would provide its own distortion pattern to the hatching lines. Theoretically, we could even detect the primary shape of segment regions and select the appropriate standardized shape to generate specific effects. Due to the limitation in BFF, we could not realize this vision. If we could replace BFF with a different transformation tool that could handle any arbitrary shape boundary, we could potentially implement the standardized shape library.

Impact of SLIC0 In general, segmentation does not always produce regions that coincide with feature boundaries. Using SLIC0 further exacerbates the problem. The SLIC0 variant (Section 2.2.1) generally produces more compact regions. As a result, for features that are irregularly shaped or have appendages, SLIC0 will tend to produce multiple compact regions to cover the features instead of regions that more accurately conform to the feature boundaries. As a result, we have to produce hatching lines across multiple regions to cover the feature. Ideally, we could produce
hatching lines that span individual features, but this is not possible because of our reliance on SLIC0. If the BFF limitation were not present, we would have the option of running SLIC using various compactness parameters.
Chapter 5

Conclusion

5.1 Contribution

Our contribution is an automatic system that produces a new sub-style of line hatching drawings from a 2D input image. As far as we know, no other approach generates hatching line drawings that resemble our style. We generate line hatching images that somewhat resemble the Grendel image, which was our original inspiration. However, our approach can extend the style and produce output images that resemble mosaic glass tiles by creating closed cell structures. Our process can alter the line characteristics that vary the images from a soft curvy feel to a severe jagged appearance. We devised a workflow that uses existing processes in a novel combination. In particular, the transformation process from the segment region to a standardized shape, allows us to elegantly map strokes that conform to region contours. Furthermore, changing the standardized shape provides an easy method of producing additional distortions, which will dramatically impact the final hatching line contours. Our transformation process can be incorporated into a variety of other stroke based techniques.

Our overall process has five main phases with an optional line refinement phase. First we apply a segmentation algorithm to produce regions. We generate mesh over each region using Delaunay triangulation. We use Poisson Disc sampling to create the vertices for the Delaunay triangulation. The mesh for each region is transformed into the standardized shape. We draw straight lines on the standardized shape and then transfer them back to the region. The lines will conform to the region contours and be distorted by the standardized shape. We merge the regions back into an image. At this milestone in our process, we have produced an image, the core version. The resulting image has a novel style. We can further modify the image output by applying
additional line refinements. To express tonal information, we modify line thickness and line spacing. Darker regions will have thicker lines as well as more dense lines. The hatching line directions change in each segment to provide an additional visual cue of changes in segments. We introduced line perturbation to capture additional local variations in the image. To provide additional object context we augment the image by adding edge lines to capture silhouette and small features.

5.2 Discussion

In theory, each of our workflow phases are implementation-agnostic. However, we encountered limitations within the tools that restricted our implementation choices.

5.2.1 BFF limitations

We rely on BFF to perform conformal mappings, which is key to our workflow. The transformation process automatically introduces distortions that follow the segment region contours. This allows us to draw relatively simple lines on the standardized shape and then transfer the lines back to the segment region.

At the outset we wanted to use the original SLIC segmentation algorithm to generate irregular shapes. We would than generate hatching lines that spanned these irregular shapes. Image a region shaped like the capital letter ‘Y’. Now image hatching lines that span the main trunk of the region vertically. What happens to the line hatching lines as they travel upwards toward the junction; do they go left, do they go right, or do they split with half going left and half going right? Ideally our process would be able to generate the hatching lines for all three cases. During development, we encountered a problem with BFF where it could not successfully transform an irregularly shaped region into the standardized shape. This limitation forced us to use SLIC0 for segmentation, the SLIC variant that produces more regularly shaped convex regions. We could no longer generate irregularly shaped regions as a result.

We had also imagined the ability to transform any region shape to any other standardized shape, e.g., transforming the ‘Y’ shaped region into a triangle. This was the second limitation we encountered with BFF. Except for the circle shape, we could not get BFF to consistently transform to any other standardized shapes. As a result, we settled on the circle as the standardized shape in our process.
CHAPTER 5. CONCLUSION

We hope that future implementations of boundary-preserving conformal mapping tools overcome these limitations. Removing these limitations would allow our vision of mapping any input region shape to any target standardized shape to be realized and to produce hatching lines across irregularly shaped regions, such as the ‘Y’ shaped region.

5.2.2 Segmentation considerations

We were inspired by the Grendel image and wanted each region to have hatching lines with different directions. Also, we wanted similar regions to have hatching lines that were colinear. We introduced the reorienting lines operation to ensure that neighbouring regions with similar intensities did have colinear hatching lines. The result was that the neighbouring regions would appear as one larger region.

An alternative approach would be to merge the neighbouring regions that had similar intensities and handle them as a single region. We would then generate hatching lines for the merged region and the hatching lines would then be continuous across the original smaller regions, removing the need for reorienting lines. Although this is additional work, it would have the benefit of producing hatching lines that are longer and continuous across the original segment regions.

The limiting factor would be BFF. Merging the similar regions could produce an irregularly shaped region that BFF could not transform. Theoretically, we could perform a merge and test; merge two regions and test if BFF could complete the transform. If BFF is successful, we retain the merged region, otherwise, revert back to the original regions.

5.2.3 Drawing lines

When drawing lines on the standardized shape, we optimized the line representation by only storing the points where the line intersected with triangle vertices and triangle edges. Since we were only drawing straight lines, this optimization did not cause any issues. However, we cannot use the same strategy to draw other line shapes such as a circle. Instead, we need to determine the number of points required to represent the shape and store all the points, not just the points that intersect with the triangle edges.
5.3 Future Work

Expand standardized shape library: In our current implementation we have only implemented the circle as a standardized shape. One direction is to increase the number of Standardized shapes available in the Library, for example the square, the rectangle, the triangle, or the parallelogram. Each of the shapes could introduce a different transformation shape to the hatching lines. Of course, this would also require addressing the limitations of BFF or finding an alternative conformal mapping tool.

In the same vein, we currently draw straight lines on the standardized shape, but we could instead draw wavy lines, circles, squares, hexagon, or any other form that can be represented with discrete points. In combination with additional library shapes, we could produce many different distortions with little additional effort.

Use additional image properties: Currently our processes uses the segment region greyscale intensities to determine hatching line direction, line thickness, and line spacing. By only using the greyscale information, we are not using all information available to us. We could incorporate colour values in addition to intensity in all our property calculations.

Our process does not use any gradient information from the image. We could generate a vector field which can be used to inform the line direction for each region. Conceivably, we could use the vector information as generated by Kang et al. [29] to inform the orientation of our hatching lines.

Increase style options: Our process currently generates monochrome line hatching images. As implemented, our system does not generate painterly or false colour styles through parameter modification. Possible extensions to the system include: adding different strokes types and including colour in the strokes. This increases the number of styles available to the process. We could conceivably combine line hatching with painterly styles, for example.

In the current process, the lines are drawn per region. Although reorienting lines lengthens lines, the process does not create long continuous lines. We could look at line refinements that can merge lines based on certain criteria. This allows the system to control line length, for example, we could produce really long lines in brighter areas and produce shorter lines in darker areas.
**Processing video:** Our current implementation takes a single photo image and produces a corresponding line hatching image. To support video, we would need to modify the process to produce a sequence of cohesive output images. The process would need to maintain continuity between frames to avoid jarring transitions, e.g., sudden line directions discrepancies, disappearing and reappearing lines, and variable region shapes.

**Improve performance:** The current execution time does not allow this process to be used interactively or for real time systems. Improving the performance of the system would allow it to be used for real time video systems or as a filter in photo and video editing systems. The performance improvement would enable the creation of a user interface that provides users the ability to explore parameter modifications in near real time.
List of References


Appendix A

A.1 Additional isolation operation comparison grids
Figure A.1: Core Version and examples Line Refinements applied to Bathtub
Figure A.2: Core Version and examples Line Refinements applied to Elephant
Figure A.3: Core Version and examples Line Refinements applied to Baby
Appendix A.

(a) Core Version
(b) Line Density
(c) Line Width
(d) Line Extension
(e) Line Erosion
(f) Line Perturbation
(g) Reorienting Lines
(h) Edge Lines

Figure A.4: Core Version and examples Line Refinements applied to Stop sign
APPENDIX A.

(a) Core Version
(b) Line Density
(c) Line Width
(d) Line Extension
(e) Line Erosion
(f) Line Perturbation
(g) Reorienting Lines
(h) Edge Lines

Figure A.5: Core Version and examples Line Refinements applied to hands
Figure A.6: Core Version and examples Line Refinements applied to Athletes
Figure A.7: Core Version and examples Line Refinements applied to Skyscraper
A.2 Additional complex variant comparison grids
Figure A.8: Example complex base + Er10 + Ex + Pt combination applied to 7 input images.
Figure A.9: Example complex base + Er200 + Ex + Pt combination applied to 7 input images.
Figure A.10: Example complex base + Er10 + Ex + Pt + Ed combination applied to 7 input images.
A.3 Python Libraries

We implemented our system in Python. The system we implemented consists of readily available libraries as well as custom code. In this section we identify the libraries we used in our implementation. Where possible, we identify any parameters relevant to the library or code.

Alongside these libraries, we also implemented custom code and integration code that tied everything together.

<table>
<thead>
<tr>
<th>Library</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opencv2 v4.4.0</td>
<td>Common functions for reading image files, greyscale conversion, histogram equalization, and writing to image files.</td>
</tr>
<tr>
<td>SciPy v1.4.1</td>
<td>Library providing common scientific functions such as Delaunay triangulation and Canny edge detection.</td>
</tr>
<tr>
<td>Scikit-image v0.17.2</td>
<td>Image processing library for Python.</td>
</tr>
<tr>
<td>NumPy v1.18.5</td>
<td>General purpose fast array data structure support. Many of the other libraries, such as SciPy, use this data structure for data storage.</td>
</tr>
<tr>
<td>Matplotlib v3.3.1</td>
<td>Library providing plotting functionality. Used this library for drawing intermediate and final output image.</td>
</tr>
<tr>
<td>Boundary First Flattening v1.4</td>
<td>C++ utility provided freely by Sawhney and Crane. Used to reshape triangular mesh to standardized shape.</td>
</tr>
</tbody>
</table>

Table A.1: Libraries used in implementation.
<table>
<thead>
<tr>
<th>Implemented functionality</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barycentric coordinates</td>
<td>Implemented this functionality due to data structure being incompatible with existing libraries. We calculate the barycentric coordinates of the points relative to their respective triangles. We implement the logic to determine the barycentric coordinates and to determine the cartesian coordinates.</td>
</tr>
<tr>
<td>Create region mask</td>
<td>After segmentation (Phase 1), we want to create a triangle mesh over each segment region. We create a region mask that covers the segment region for further processing. In phase 2, we place points to cover the region mask. We can further apply additional transformations to the region mask, such as dilation and erosion.</td>
</tr>
<tr>
<td>Collision detection</td>
<td>When lengthening lines during the line extension operation we want to ensure that lines do not cross. As we extend lines, we add a new point to each end of the line. Before we add the new point, we check for collision against existing lines.</td>
</tr>
<tr>
<td>Draw lines</td>
<td>We draw straight lines on the standardized shape. We start an exterior triangle vertex and then walk in a pre-determined direction. When we encounter the edge of a triangle, we record the point of the intersection as the next point in the line. We repeat this process until we reach of the edge of the standardized shape.</td>
</tr>
<tr>
<td>Populate perimeter points</td>
<td>We pre-populate the region mask perimeter with points. Since we apply Poisson disc sampling to the interior of the region, the points on the interior of the region are at least $r$ distance apart. We calculate the number of points to place on the perimeter such that the distance between these points are no more than $r$ distance apart.</td>
</tr>
</tbody>
</table>

Table A.2: Custom functionality implemented.