Image Stylization using Depth Information for Hatching and Engraving effects

by

Carlos Alberto Aviles Galarza

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
May, 2017

The undersigned hereby recommends to the
Faculty of Graduate and Postdoctoral Affairs
acceptance of the thesis

Image Stylization using Depth Information for Hatching and Engraving effects

submitted by **Carlos Alberto Aviles Galarza**

in partial fulfillment of the requirements for the degree of

Master of Computer Science

Professor David Mould, Thesis Supervisor

Professor Robert Biddle, School of Computer Science

Professor WonSook Lee,
School of Electrical Engineering and Computer Science

Professor Tony White, Chair,
School of Computer Science

Ottawa-Carleton Institute for Computer Science
The School of Computer Science
Carleton University
May, 2017

Abstract

In this thesis inspired by artists such as Gustave Doré and copperplate engravers, we present a new approach for producing a sketch-type hatching effect as well as engraving art seen in the US dollar bills using depth maps as input. The approach uses the acquired depth to generate pen-and-ink type of strokes that exaggerate surface characteristics of faces and objects. The background on this area is presented with an emphasis on line drawing, pen-and-ink illustrations, engraving art and other artistic styles. The patterns generated are composed of parallel lines that change their directionality smoothly while avoiding intersections between them. The style communicates tone and surface characteristics using curve deformation, thickness, density, and spacing while at the same time adding crosshatching effects. This is achieved with our algorithmic approach which uses the data to perform operations that deforms patterns. The components of our methodology and algorithms are detailed, as well as the equations that govern these effects. This work also presents an array of different results by varying parameters or by rendering variations.

This thesis is dedicated to my parents Juan Carlos and Olivia Margarita as well as my sister Maria Olivia, for their love and immense support through my life...

Acknowledgments

I would like to express my very great appreciation to my supervisor Dr. David Mould, your guidance and knowledge was without of doubt vital during the development of this thesis. I think I was very lucky to have you as a guide, mentor and friend. Thank you for your patience and advise during this process.

I wish to thank my parents for their immense support and encouragement throughout my studies.

I would also like to thank the thesis committee for analyzing this work and providing important valuable suggestions. My special thanks are extended to my friends in the GIGL lab as well as The School of Computer Science.

Table of Contents

List of Figures

Chapter 1

Introduction

Image stylization is the process of taking an image, extracting important information, and generating a result with a specific style that mimics an artistic impression. The following thesis describes a set of approaches to automatic Line Drawing, specifically on the pen-and-ink illustration styles to produce engraving effects. Line drawing has been for centuries one of the simplest artistic styles. Using vector-based lines gives us benefits: Lines are computationally inexpensive to handle, easy to store, and resolution independent. Lines can communicate contrast by their density, shape, thickness and curvature as well as portraying bright and dark regions on a sketch. The thesis will describe a method for converting input depth images into line drawings.

1.1 Goal

Our main objective is to create an image with a pen-and ink style, specifically a hatching effect. We were inspired by the illustrations of the artist Gustave Doré [16] and by coppersmith engravers that produce artistic pieces such as the ones in US dollar bank notes. Our work uses a single algorithmic approach for both of these hatching and engraving stylizations. These effects have very important characteristics such as:

- Parallel lines are drawn across objects and are deformed in such a way that they portray surface information.
- Each curve varies slightly from one another depending on the geometry of objects in the image.
- Curves avoid intersections.
- Curves have smooth changes of directionality.
- Crosshatching is used to differentiate between bright and dark regions.

The difference between these two effects consists of how the curves are rendered. For a hatching result, lines will be used to depict these characteristics. For the engraving effect, interruptions of the continuity of the curves is an additional aspect which complements the use of crosshatching for expressing bright and dark regions. We plan to use depth data for a pen-and-ink stylization. These strokes will be drawn in a way that suggests surface geometry and reflect the information of volume from objects in the scene. We want to exaggerate these characteristics in order to generate a powerful response for the viewer as usually achieved by artists in this style of art. Illustrating images with lines is easy to comprehend and interpret, and is usually more efficient than photographs as a visual communication means [21]. Lines can be compressed without difficulty and rendered in vector form for a resolution independent result.

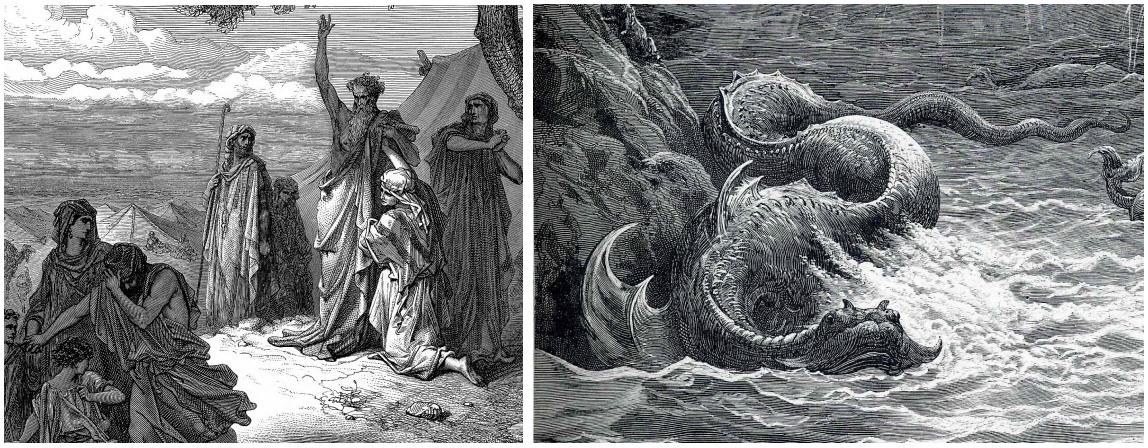

Figure 1.1: Artistic illustrations by Gustave Doré. [16]

Figure 1.1 shows two artistic pieces by Gustave Doré which present a well defined hatching effect. This technique highlights tone to communicate bright and dark areas that can be defined by the drawn strokes. Artists such as Doré control the thickness, spacing and length of strokes to differentiate regions. Greater thickness, density, and crosshatching are characteristics that control tone. There are a few types of hatching effects such as linear hatching, cross-hatching and contour hatching. We will explore each of these in the following sections.

Besides presenting our method to stylize a hatching effect, we also implemented an approach to render engraving effects such as Figure 1.2, inspired by the digital facial engraving method that Ostromoukhov presented in 1999 [40]. This engraving

art, famous in the 17th and 18th centuries, was done manually by artists.

Both of these pen-and-ink effects have the advantage of being simple and aesthetically pleasing. Winkenbach and Salesin [51] considered the style as ideal for printed publications because of their simplicity which "provides an appealing crispness and directness". The engraving style itself can be associated with wealth due to its noticeable presence in banknotes, while its original purpose can be attributed to counterfeit deterrence.

Figure 1.2: Engraving effects.

Generating these effects with computational tools provides us an opportunity to present a new and simple approach to these illustration styles. Our results have a big similarity with both the Engraving and Hatching effects, and are contrasted with artistic impressions as well as previous NPR techniques.

1.2 Contributions and Organization

Our main contribution is the use of depth maps as input images, and using them to render and exaggerate the surface geometry of objects or faces. Since our artistic inspirations such as Doré and copperplate engravers highlights these features, we want to make use of depth to ensure that we can seamlessly emulate the style, considering the important characteristic of conveying surface geometry from objects in the scene. A depth map is an image with distance information from the corresponding object to the camera. Approaches using this have been previously used for stylization purposes [8] [35].

Some contributions from this work are the following:

• We present an approach that uses depth maps to generate strokes that will communicate and exaggerate surface geometry. This is done by the displacement of parallel lines, which are calculated by the depth. The resulting curves will portray a stylization of hatching and engraving effects.

• We present operations on our rendering method to create variations of our desired style. These operations will offer similar hatching and engraving effects, but generating a different form of art.

Three-dimensional content has become one of many new functionalities that are more accessible to consumers. This inspired us to make use of good quality depth maps in order to generate our desired effects. Previous attempts have mostly used meshes to generate this style, which gave them a great advantage in terms of generating shaders for creating hatching stylization.

Nowadays, off-the-shelf cameras are becoming more and more sophisticated. Sensors such as Kinect can provide 3D data, depth maps, and stereo-pairs just to name a few related data types. We work specifically with depth information that contains geometric data from the scene. Each of the pixels on this map will have a depth value which we acquire, for generating the effects that we try to accomplish. Stereo-pairs provide 3D data from the scene with a couple of photographs taken side by side. The two vantage points can be used for the calculation of depth. The use of these images are susceptible to noise and require a technique for disparity detection in order to generate the map. By using just a depth map instead, we handle the data from a single image and no further methods are required. Even though our approach uses 3D data, an image is used as input which contains this depth information and generates vector-based primitives that will be resolution independent. The use of photographs will not provide any real data from the object's surface characteristics without some complex calculations.

This document is organized as follows, Chapter 2 reviews the relevant literature review concerning line drawing, engraving art, and segmentation approaches. Chapter 3 explains our algorithms in detail. A pipeline presents our flow of important methods which is how this chapter is structured, explaining each important component. Chapter 4 compares our results and shows a few different rendering styles obtained by varying some parameters. Our last chapter is the conclusion to our work which highlights important points of this thesis and analyzes future research paths and alternative routes that would be a logical next step.

Chapter 2

Background and Previous Work

This chapter explains previous work on fields of relevance such as Non-Photorealistic Rendering, Line drawing and Pen-and-ink illustration. There has been a lot of interest in these fields during the last few decades. This chapter provides background to our work, and discusses results previously obtained and how our approach can be compared. Section 2.1 describes Non-Photorealistic Rendering, some of its impact on the computer graphics community, its history, and how it has expanded over the last couple of decades. Relevant Object-based rendering work and Image stylization techniques are reviewed and presented in this section; these methods and approaches are divided into these two categories to explain this field in a more coherent way. Section 2.2 presents some techniques for Line drawing which is of extreme importance in our work. The Pen-and-ink Illustration background goes deeper into our work in Section 2.2.1, and later 2.2.2 for Engraving which defines both of the effects that we implemented. Our final section analyzes two important segmentation methods, a manual and an automatic algorithm; these are essential because our methodology is a region-based scheme.

2.1 Non-Photorealistic Rendering (NPR)

Non-Photorealistic Rendering is one of many fields in computer graphics. Unlike photorealistic rendering that makes an effort to render objects realistically, NPR solely focuses on presenting techniques that are artificial and stylized, usually inspired by art. NPR uses information either from objects or images for creating abstract or stylized renders that reproduce illustrations done by artists [45].

Aaron Hertzmann argues that NPR research "plays a key role in the scientific

understanding of visual art and illustration" because it will eventually answer two very important questions [20]:

- How do artists create imagery?
- How do observers respond to this artistic imagery?

These questions have yet to be answered, but through years of research on NPR, several techniques, approaches and methods have been presented, mostly trying to emulate famous artistic pieces or styles of art. The area has been divided into many fields of research such as abstraction, impressionistic stylization, painterly rendering, stippling, watercolor, black and white, cartoon stylization, and line drawing to name a few. The important characteristic of this area is the power to communicate information in several forms of expressive rendering styles [38].

While we named several styles that have been developed over the years, NPR has to be divided in two categories which are Image stylization and Object-based rendering techniques. This separation defines the type of data that is being used as input in order to process and provide rendered stylized work. Image Stylization techniques receive images as input and uses a set of tools such as filters, edge enhancement techniques, to use all the information available in order to achieve the proper effects. Object-based rendering deals with 3D models or meshes in order to render objects from geometry; most of these approaches try to show objects as a two-dimensional result.

There has been a vast amount of approaches and techniques that have been proposed over the years, but most of them are based on image post-processing methods. Haeberli is most likely the first to actually work on NPR presenting "Paint by Numbers" [19]. In the same year Saito presented algorithms for dealing with edges, discontinuities, contour lines and curved hatching [43]. The key in NPR is that it can communicate expressive styles to the viewer in a more effective way. Styles such as Line Drawing can be easy to store and easy for a viewer to comprehend and interpret.

The following section will focus on explaining relevant NPR methods. Our work focuses specifically on the area of Line Drawing in which we present a new approach for hatching and engraving styles. The background on these areas will be explored on section 2.2 in more detail, while the methodologies themselves will be detailed on Chapter 3.

2.1.1 Object-based rendering

Rendering from geometry approaches depend solely on 3D models or point-cloud data. This has been more thoroughly explored than its image-based counterpart.

One of the most relevant work on Object-based techniques is Illustrating Smooth surfaces by Hertzmann and Zorin [21]. They presented a set of algorithms which automatically rendered a hatching effect in real-time using a multiple silhouette detection approach that drew hatch marks while highlighting the model's volumetric information. This work generates the exact hatching effect that this thesis is trying to achieve.

The use of 3D models is a great asset to their work , mainly because there are several 3D operations that allow these models to extract silhouettes, or edges, that will certainly provide advantages on the rendering of the intended effects. The proposed algorithms are based on computing silhouette drawings, defined by "boundaries, creases, silhouette lines, and self-intersection lines". Direction fields are then calculated to use directionality on the rendered hatches. Their results are impressive and desirable for our work. See Figure 2.1.

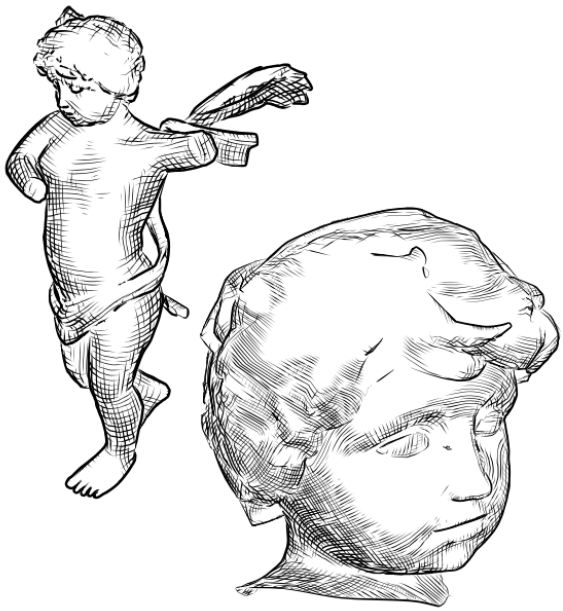

Figure 2.1: Hertzmann and Zorin's Illustrating Smooth Surfaces hatching effects [21].

In 2003, DeCarlo et al. proposed an NPR system that acquired suggestive contours from objects to convey shape, which meant using algorithms that could further highlight significant outlines in order to portray the image just by contours [9]. Compared to the normal contours seen in the left image of 2.2, the right image highlights how their work renders the suggestive contour adding more relevant information to the result. This is done by an algorithm that finds the zero crossings of the radial curvature, which are regions where the "surface bends away from the viewer" and where suggestive curves occur. This is very valuable for complementing line drawing methods.

Kalnins et al. 2003, Raskar 2001 also presented similar systems that relied on 3D models in order to generate line drawing effects [25] [42]. These techniques were not intended for generating stylized results.

Figure 2.2: De Carlo et al. shows an example of contours only on the left and suggestive contours on the right. [9]

2.1.2 Image Stylization

Image stylization approaches, as the name suggests, works exclusively with images as input, in which relevant data is processed and extracted in order to create artificial and artistic content [33]. Image Stylization has been usually divided into filtering or stroke-based approaches [28].

Our work focuses specifically on an image-based line drawing technique, we explore hatching, and digital engraving approaches and related work for comparison purposes in later sections. Hertzmann proposed the division of these two rendering fields, in order to better categorize the work on NPR [21].

There have been several years of research on this specific field. Our focus involves three very specific areas: Line Drawing, Pen-and-ink illustration, and also Digital Engraving. Even though these effects might seem aesthetically similar, they have been approached and treated very differently.

Early in the 90's, work such as Haeberli's [19] was among the first to start working artistically with images, trying to achieve pleasing abstract results. His work presented a few stroke based techniques with the goal of creating abstract impressionistic images from photographs.

An approach that calculated a normal map from cartoons in order to make the 2D representation smooth-shaded and "more like CG"was presented by Johnston in 2002. Johnston presented Lumo, a method that could mimic the environmental illumination on a 2D scene [24]. This made cartoons look like three-dimensional smooth-shaded surfaces. Even though this was applied to cartoons, the idea of taking a flat image and inferring volume from boundaries is of great importance for making an artificial depth map using interpolation or normal map calculation using edges. The approximation of a normal map was used to calculate the illumination from the scene which made cartoons look like a rendering from a 3D model. Similar and more versatile effects were achieved by Orzan et al. using difussion curves [39]. This technique involved a Poisson equation to generate smooth shaded images using diffusion curves [4]. The users had to manually draw diffusion curves so that they can be used as input in order to apply their technique. Their results were vector-based primitives which made them resolution independent.

Textureshop, software created by Fang et al. [13], uses a Lambertian reflectance model in order to calculate the surface normals from the image, using shape from shading techniques. Shape from shading uses the image information and infers gradient and height fields with the purpose of synthesizing texture [57] [56]. This approach while complex resembles our idea when using texture distortion on the surface geometry of objects. It generates several patches in which texture gets mapped locally and then produce a 3D texture mapping effects just using an image as input. This patch-based approach connects every region, where it disguises inconsistencies on its

borders using graph cut blending. This approach could very well be a competitive solution for our work, but at the same time is a more complex option that focuses more on texture mapping than stroke generation.

Relevant work for Line drawing and Pen-and-ink illustrations techniques will be more broadly analyzed on the following sections.

2.2 Line Drawing

Line drawing is one of the most ancient techniques in terms of visual communication. Its use can be traced back millennia. It is a very simplistic approach to reproduce forms, objects and figures while focusing attention and implying shape [10,21,27,49]. It has been considered to outperform photo-realistic imagery in terms of "efficiency" and "precision" [21], and it can be easy to manage, store, and be resolution independent when in vector form. Moreover, it is widely used for medical illustrations, cartoons, textbooks, technical manuals, among other forms of media.

There has been considerable research in the past few decades. Pure line drawing techniques often involve edge and outline rendering, excluding any texture, tone and shadow information. A Pen-and-ink illustration is a type of line drawing that tries to target tone and shading with strokes.

Winkenbach and Salesin [51] proposed a texture generation system for creating stroke-based textures, also describing the principles of pen-and-ink illustration techniques based on Arthur Guptill's classic text "Rendering in pen-and-ink" [18]. While it was presented as a computer graphics approach, it laid crucial groundwork for the field of pen-and-ink illustration techniques. Subsequently, Winkenbach and Salesin [51] presented a new set of algorithms for these illustrations, a "Controlled-density hatching" technique.

A fundamental method that has become a standard in the line drawing field is Canny's edge detector [5]. Salisbury et al. introduced a pen and ink interactive system based on Canny's detector for rendering a contour hatching effect with outline strokes [44]. This was a user-based system that required a direction field reference in order to generate the curved strokes. Afterwards, Litwinowicz made a follow up using a similar technique [34], for paint brush strokes in order to highlight feature lines. Using the Canny edge detector and following Meer and Comaniciu's mean-shift segmentation algorithm [7], DeCarlo and Santella presented work on image abstraction

of photographs [11].

Markosian et al. [36] highlighted the importance of economic line drawing in order to convey shape with a minimal amount of strokes. Their work focused mostly on identifying silhouettes and minimal shading using 3D models.

In 1998 Curtis proposed a "loose and sketchy" filter for animations [8]. His approach used a depth map from a 3D model as input. An edge map was generated using the gradient of the image. Tracking of particles was performed to create a sketch-like effect using a physically-based particle system and ruled by a force field acquired from the depth map. The vector field arranged particles along the silhouette while perpendicular to the depth maps gradient. These results were very similar to pen-and-ink illustrations.

Gooch et al. in 2004 [17] worked on a facial illustration system. They implemented a Difference of Gaussians (DoG) filter with the purpose of creating black and white illustrations. Kang et al. would later use the DoG as the starting point for their Flow Difference of Gaussian (FDoG) method.

Kang et al. proposed Coherent Line Drawing [27], an automatic method for line-drawing from photographs. Their work relied on calculating a flow-like smooth direction field from the image so that salient features from the image are prominent and unaffected. Their Edge tangent flow (ETF) technique generated a smooth vector field using the gradient of the image. This was done by kernel-based nonlinear smoothing of the vector field that maintains the most salient features of the image to acquire directionality. This flow-like vector field was then integrated with the DoG filter to introduce a Flow Difference of Gaussians (FDoG), which offers a nice outline drawing effect [26]. See Figure 2.3. Their method for calculating the ETF is used in our work.

Winemöller [52] recently expanded DoG implementation, presenting several improvements with his xDoG approach in 2011. Figure2.4 shows DoG, XDoG and a hatching variation approach. Winemöller would also present stylizations using colored images and videos [53].

Another group of techniques started to appear using Particle Tracing systems, these relied on the paths that particles followed. Xu and Mould [55] presented a curvature-controlled aesthetic curves using magnetic fields. They used a particle tracing method ruled by a magnetic field equation. While this algorithmic approach

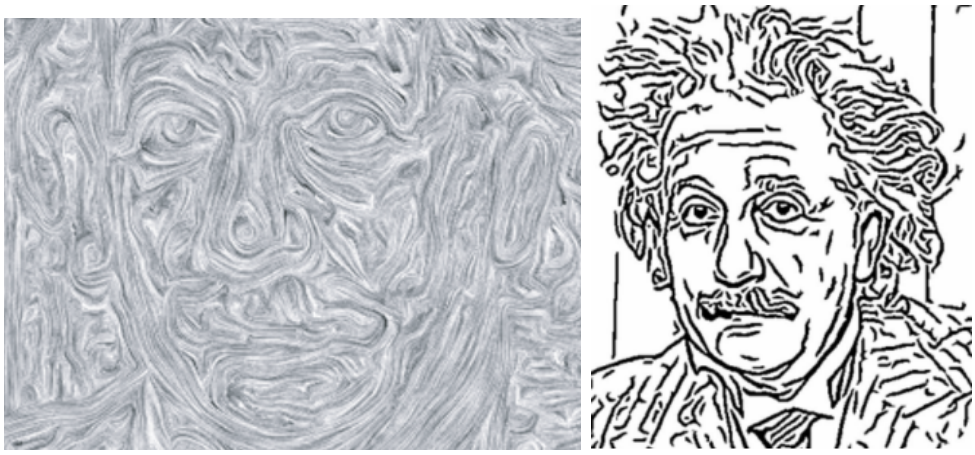

Figure 2.3: Left: Edge Tangent Flow (ETF). Right: Flow-based Difference of Gaussian (FDoG). [26]

Figure 2.4: Left: DoG. Center: xDoG. Right: xDoG hatching. [52]

generates curve-like abstract stylization, mainly to depict fire, hair, and trees. It is highly relevant in line and curve generation, generating aesthetically pleasing effects.

Figure 2.5: Wei and Mould's work with particle tracing to produce pen-and-ink illustrations. [49, 50]

Following Li and Mould's tessellation technique by growing curves [30], Wei and Mould presented a coordinated particle tracing system [49, 50] that created a pen-and-ink effect as seen in Figure 2.5. This work was also inspired by Gustave Doré's hatching effects and Ostromoukhov's engraving results. Overall, this particle tracing followed some of our principles such as strokes possessing individual directions, never crossing each other and being almost parallel with each other. By using these laws governing the system, particle's paths are traced to draw curves, and a birth and death of particles methodology was used to fill empty spaces. Furthermore, they used the stylization work for generating Optical Art by Inglis and Kaplan to test some of their results [22, 23]. This system provided aesthetically pleasing results that come very close to various pen-and-ink artists. Their results however differ from Doré's original illustrations, lacking one of the most important hatching characteristics which is conveying shape. We thought it was ideal to introduce a depth-aware system that will display this characteristic.

While not necessarily in the Line drawing area, Inglis and Kaplan introduced an approach for creating Op Art. This technique relies on the use of parallel lines, in

order to generate patterns that creates "illusory contours by the line bends" [22]. A requirement for this system is the input of 2-colour input maps or 3-colour input maps used as a reference, so that their algorithm renders strokes at different angles. Their system avoids generating artifacts such as line breaks. A 4-colour algorithm is explored, although the authors warn of the inevitability of artifacts in these cases.

Aesthetically similar areas include Half-toning and Stippling. These images share some characteristics with Line Drawing in terms of storage and easy comprehension. Half-toning has a great importance for printing purposes. An Error Diffusion technique was presented by Li and Mould in 2010 [29]. Relevant Stippling methods that show nice stylizations have been proposed by Secord which introduced a method that used Centroidal Voronoi Diagrams [46]. A more recent approach was presented by Li and Mould in 2011 [31]. These image stylization techniques, while not related to line drawing, share the same objective of being an efficient visual means of communication. Additionally, some of our variant results become similar in style.

2.2.1 Pen-and-ink Illustration stylization

Pen-and-ink stylization uses hatching to depict tone and shading. Pen-and-ink illustrations can be divided into the drawing of outlines and hatching. Outline drawing is constrained to external boundaries and edges which in line drawing is crucial to convey the simplistic visual appearance of objects [54] while lacking the shading part and surface properties. Hatching in illustration can communicate object's tone which defines dark region, and geometry which provides a sense of surface details depending on the quality of the strokes. This quality will display well defined sharp strokes, or very smooth curves.

Outlines

Work on outlines can be separated into pure line drawing techniques such as Kang et al.'s Coherent Line Drawing technique. Our system does not explore integrating an outline approach. While our work could benefit from this, it focuses entirely on the hatching approach and also presenting engraving art results. Therefore, outline integration is a problem that can complement our results in future work.

Hatching

Hatching is a very old and traditional effect. Several artists have portrayed this effect through centuries, considering it probably originated in the middle ages where it was later complemented with the use of crosshatching in the 15th century [51]. It is mainly characterized by communicating tone, forms, and suggesting object's geometry. One distinctive characteristic from other line drawing techniques is that artists surround the objects in the scene with these patterns of lines, due to its composition of several line strokes. How many strokes, their spacing, thickness and deformation will highlight and communicate a powerful image in terms of tone and illusion of geometry [48]. There are three types of hatching:

• linear hatching: These sketches are composed of parallel lines only with changes in directionality to convey surface geometry.

• crosshatching: Complementing linear hatching with an additional layer at a perpendicular angle will produce this style. Its use is ideal to depict dark regions.

• contour-hatching: Use of linear hatching around the contours of the drawings usually to portray illumination and darkness.

We target the use of linear hatching and crosshatching. We do attempt to get some results related to contour-hatching but we do not go into further detail, mainly due to the need of also using contour and edge detection enhancement for outline rendering. Outline rendering is not addressed in our work.

2.2.2 Digital Engraving

Engraving art is a very old effect believed to be originally used as an illustration style for book printing [15] and later evolved to copperplate engravings. While similar to hatching, it is a separate traditional illustration technique. People who possessed the skills to perform this were called engravers. Master E. S. and Marting Schongauer where famous artists devoted to this art [47].

This art meant making cuts or grooves on a flat metallic surface to convey shapes and present illustrations. These grooves followed most of the same laws that govern hatching effects. They were parallel to one another, following individual directions while conveying shape and surface characteristics. The grooves were dotted in sections to communicate brightness, while crosshatching was used to portray darkness. Once an engraving was finished on a flat copperplate surface, black ink is applied on the

metallic plate which fills the grooves, so that pressing the plate onto a cloth or paper would efficiently transfer the engravings. This art was perfected around the 17th and 18th centuries [47].

Our work is inspired by copperplate engraving styles as well as the US dollar-bill banknotes. Work on this specific effect has been scarce on NPR. Ostromoukhov

Figure 2.6: Digital Facial Engraving approach proposed by Ostromoukhov. [40]

[40] proposed a technique for facial engraving in 1999. He presented a method that works by superimposition of several layers. A transformation map was calculated for each layer in order to render the stroke patterns onto the images. This system required a significant level of user input in order to segment and separate layers. The transformation map used to curve the patterns, never took in consideration any data from the input image. Therefore, the patterns were very smooth and didn't provide any real information from the surface. This system also generated crosshatching styles, while exaggerating the surface geometry and texture. Our engraving effect can be considered a follow up to this approach for this is to our knowledge the only

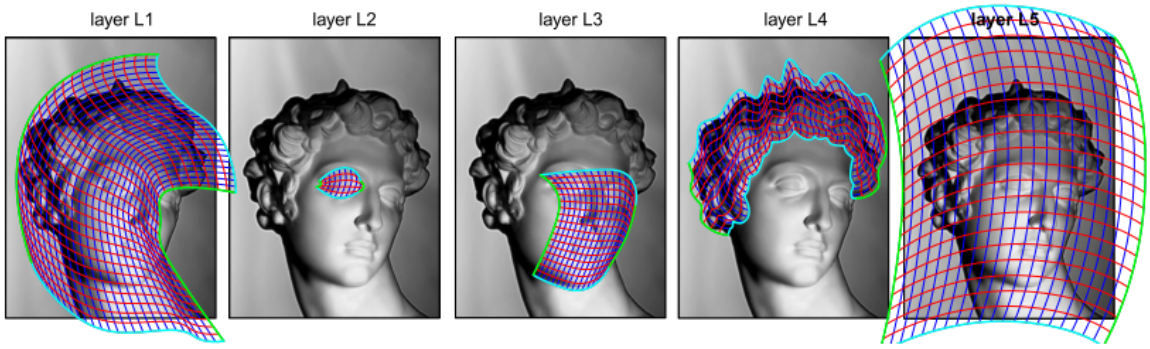

Figure 2.7: Ostromoukhov used a parametric grid to deform each layer individually. [40]

technique ever presented.

2.3 Segmentation Methods

Segmentation is an extremely important tool for image processing. Over the years several methodologies have been proposed. The role of segmentation is to divide the image into sections, a group of pixels or super-pixels. Acquiring these super-pixels provides image processing techniques the possibility of applying their methodologies on a region-based approach, a powerful way to work on elements from the scene individually. Segmentation is also very important in areas such as computer vision, where the algorithm's speed is a huge factor since most vision applications need to be real-time.

In our work two methods have been implemented, a manual and an automatic approach. Our desire to implement a segmentation component is the need of separating objects in our images. Allowing us to divide sections in the scene and applying a mask-based approach for pen-and-ink rendering method with different tilts since we do not want the same orientation of lines in the entire image.

Watershed algorithm

We have selected a marker-based watershed algorithm as our manual approach which requires the user to draw scribbles over the image. This algorithm takes these scribbles to "flood" the surface and obtaining boundaries which can be later mapped [3] [2].

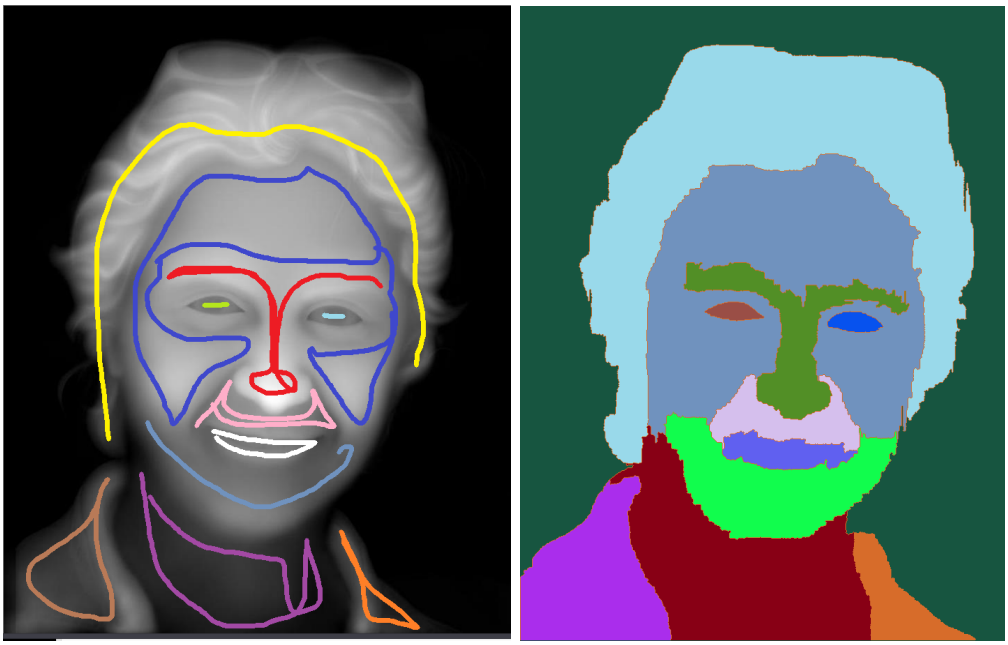

Figure 2.8: Left: Scribbles are drawn into the image in order to apply watershed algorithm that will segment the image according to a similarity of color function. Right: The manual watershed algorithm provides the segmentation of the input image.

These maps can then be later used as masks. Figure 2.8 shows how scribbles are drawn by a user and the resulting segmentation at the right. The drawing of scribbles must be performed by the user which inspired the inclusion of an automatic segmentation approach. Overall, the manual input will only involve the drawing of scribbles over the image, so that the user can decide which are the most important elements to be segmented.

Simple Linear Iterative Clustering

Achanta et al. introduced Simple Linear Iterative Clustering (SLIC) [1], a sophisticated segmentation method which generates uniform, compact, detailed super-pixels. An over-segmentation as seen in Figure 2.9 shows how many detailed regions are computed, and how well the boundaries partition the image along edges from the image. SLIC does a local k-means clustering approach on pixels, in a computationally efficient way. This has become very helpful for mosaic effects [12]. Doyle and Mould recently used this method and applied some modifications to get stained glass stylizations.

Since SLIC is an automatic approach, it became inspiring to integrate it with our system since we also wanted to present a version with minimal user input.

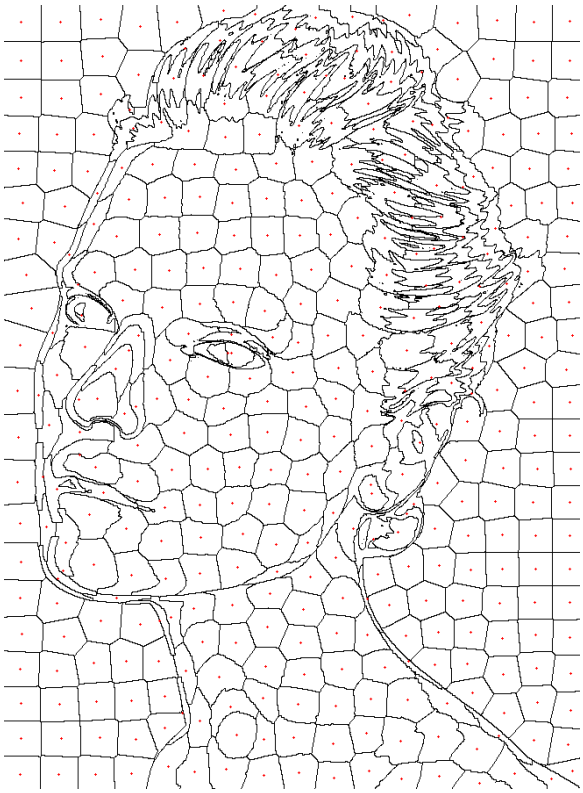

Figure 2.9: An over-segmentation using SLIC approach.

Chapter 3

Algorithms

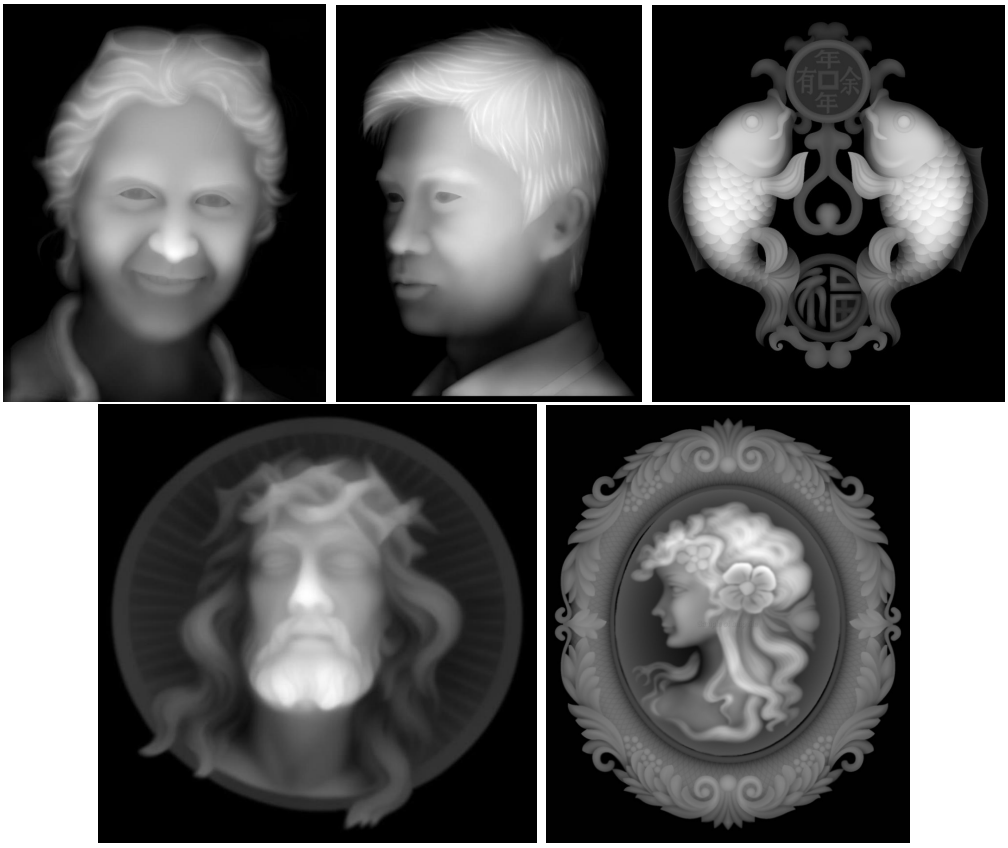

Figure 3.1: The input images used for our approach.

3.1 Overview

Our background chapter went through previous work on NPR, Line drawing, Pen-and-ink illustration techniques, and Hatching and Engraving effects. Our general pipeline is displayed in Figure 3.2. It shows how our approach goes through each of our important components. This section will discuss each of these, while the following sections will analyze and describe in detail every component individually.

We begin with our input depth map. Figure 3.1 displays examples of the depth-maps used as input. This image will have an intensity value I for each pixel that will depend on the depth from objects in the scene. These pixels are depicted as intensity values ranging from 0 to 255 since we use a 8-bit gray-scale image. A zero-intensity value will indicate total blackness in the image (i.e. the furthest level in terms of depth), while a 255 value is the closest value in the image. It is important to note that using this 8 bits allows only a narrow amount of depth values.

Our methodology has two scenarios. Either we select a manual segmentation method which is our watershed algorithm as explained on Section 2.3 or we use an automatic approach with our implementation of the Simple Linear Iterative Clustering algorithm. These two paths occur in our segmentation approach, and are displayed in our Segmentation and Mask generation pipeline as seen in Figure 3.4.

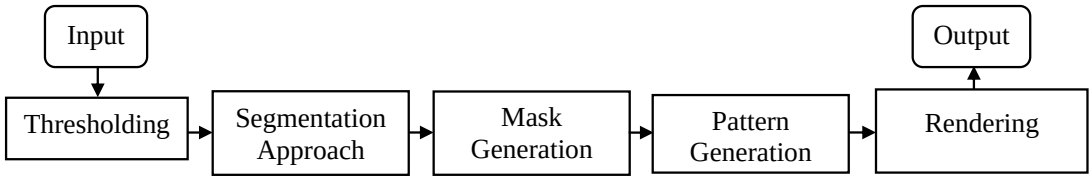

Figure 3.2: Our pipeline.

Our image is going to be composed of line strokes or hatches. These primitives will provide shape, shading and contrast to the image. Our system uses the provided depth to curve these primitives and create our artistic styles.

Finally, the Rendering component draws the strokes as splines or as line segments with a varying thickness. It also applies operations to the obtained curves in order to finalize the effects, separating hatching and engraving effects. Furthermore, this component also produces several other variations of results by the use of different operations and parameters.

Our main pipeline defines each component. The following sections explain selected parts in more detail. These are Thresholding, Segmentation, Mask Generation, Pattern Generation, and a Rendering module.

3.2 Thresholding

We use a depth image as input. Complementary to this we use a normal photograph from the scene, or a calculation of illumination is done on the depth map to locate shadows. Additionally, we remove the background by not considering the furthest depth level (i.e. intensity value of zero). We apply thresholding to this photograph to define shadows or dark regions. This is useful to detect which regions should be depicted with a higher density of lines, we used them as regions for crosshatching. The user can select the threshold τ in order to produce a high or low contrast in the final render. After this step we go to segmentation.

3.3 Segmentation

The pipeline for the transition between the segmentation and mask generation component is detailed in Figure 3.4. The blue section displays the segmentation part of the process. We have two alternate approaches to segmentation. In our manual approach, the user draws scribbles on important sections from the image, as previously seen in Figure 2.8. The algorithm will then segment the image and provide a map of regions with defined boundaries, each with a different ID. SLIC, on the other hand, will provide an automatic over segmentation.

Afterward, we follow our manual or automatic segmentation method. We segment our photograph or depth-map with any of the two routes. If we decide to use our watershed option, this step will jump directly to generate masks. Alternatively, with our SLIC option, we get an over-segmentation from the image that we try to benefit from. We apply the Edge tangent Flow methodology from Kang et. al. [27] to get a vector field. We then acquire the tilt from the vectors for each of this segmented regions. We can later get patterns that will follow a Flow-like directionality. After our segmentation method, an array of masks is stored, which will be used by our Pattern Generation module for a region-based rendering of the curves.

3.3.1 Edge Tangent Flow

For our automatic scenario, we acquired an oversegmentation from SLIC. This allowed us to manage a significant amount of segments.

The manual approach used a random angle for tilting each region. For this scenario, we integrated Kang et al's ETF method which provides a vector field. This field provides us with a sense of flow and directionality which is influenced by the detected edges. For each region, the vector is averaged and stored. Therefore, we save each of the mask and a vector for each boundary. In figure 3.3, we show each of the regions with the vector positioned on the centroid of each super-pixel.

It is important to highlight that some vectors have a very small magnitude and are not as reliable. We filter these and exclude them from calculations, which are recalculated averaging neighboring vectors to fill these empty values.

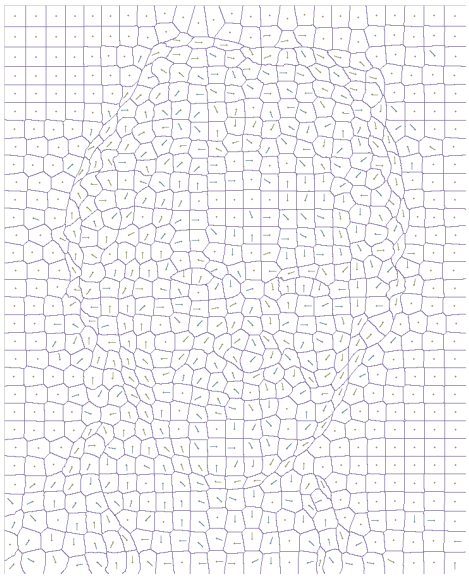

Figure 3.3: With our SLIC boundaries, we visualize the vectors provided by the Edge tangent flow method at the centroid.

The following section will go into further detail into the mask generation module.

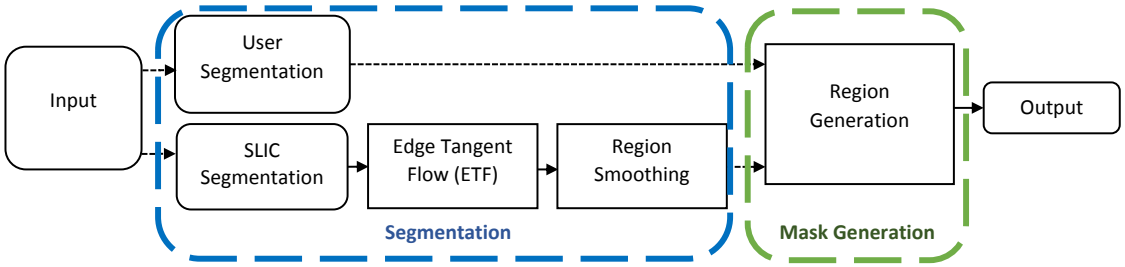

Figure 3.4: Pipeline that depicts the transition between the Segmentation and Mask Generation component referenced on our main pipeline.

3.4 Mask generations

In the Mask Generation component, there are two scenarios to be considered: manual and automatic. Both provide our desired hatching and engraving effects.

We will explain our manual methodology first. Our input image is the depth image for both cases: a gray scale depth map is provided, for which we extract its information and save every pixel into a matrix with values between 0 and 255. The size of the matrix is the same as the resolution of the input image. For our manual approach we use our watershed implementation in order to segment our image. The user draws scribbles on the image in order to separate important features from the scene, such as faces, eyes, clothing, hair, objects, or scenery. We take this partition as seen in Figure 3.5 and we store it as an array of binary maps which will serve as masks in our following steps. The use of masks provides us with the advantage to work on individual objects and assigns a direction vector for controlling the tilt of the curves. Objects will have different hatching directionalities and will stand out from each other.

This simple approach is repeated in our automatic case with one big difference. In our manual case we use a random direction for each mask in order to draw patterns in a certain direction. For our automatic case we integrated it with Kang's ETF approach. Using the acquired over-segmentation from the image, we get a lot more regions and therefore masks. It can be exhausting for a user to get this level of detail by scribbling markers on the image.

Figure 3.5: Segmentation provided by a manual segmentation approach.

3.5 Pattern Generation

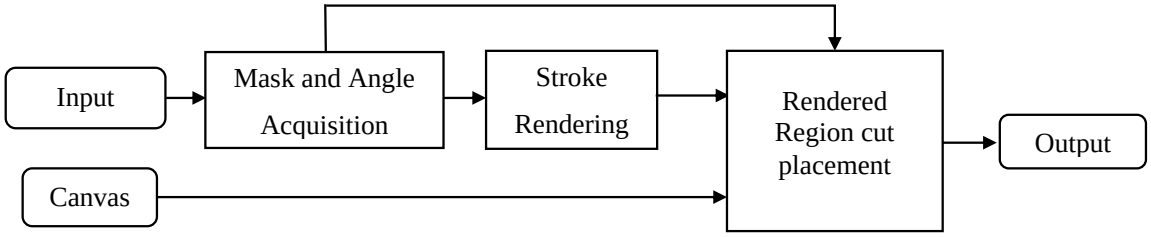

Figure 3.6: Pipeline for our Pattern Generation component.

The implementation for our pattern generation module is the most important component of this work. Figure 3.6 details a pipeline for this process. We are provided with individual masks and a direction vector associated with them which allows us to acquire an angle for tilting the curves. We use the mask dimensions to constraint our rendering to a Region of Interest (ROI). Subsequently, using our depth image, ROI and direction, we apply the stroke rendering module. This module consists of using our parametrized approach for curve generation. Once a patterned region has been rendered it is then placed on the canvas, then cut making use of the mask. For the remaining regions, this component is recalculated using a different angle each time,

and using the current canvas that will start to be completed piece by piece. As a result, every segmented object will not have the same tilt for its curves.

3.5.1 Simplifying our approach

In order to explain this approach simply, suppose that the direction angle is 0. In this case we will be dealing with a pattern of horizontal lines. For each Region we create a ROI with dimensions big enough to inscribe our mask inside it.

Our algorithm will work on a stroke by stroke basis. An acquired ROI narrows the rendering to be performed on the required region, instead of plotting all the curves on the entire image. Every curve will be separated by a specific spacing δ_y parameter. The parameter defines how many lines will fit vertically and therefore how many will be drawn.

Working on a curve, the parameter μ sets how many samples are taken on each line. These samples will be evenly spaced, a distance d_x from one another. Since we know the position of x and y on the plane, this sample will allow us to acquire the intensity value $I(x, y)$ from our depth map. In the left image of Figure 3.7 two horizontal dotted lines and the samples taken are illustrated. This example shows a depth map of a sphere, and the calculation of two curves. Our first curve is portrayed as a green line; it illustrates the array of samples taken on the image. We are given a depth value for each of the samples. The role of our parametrized approach is to calculate the displacement of each of the samples(vertices) on the dotted strokes, thus obtaining the deformed curves.

Parametrized approach

We are measuring the displacement using depth from the input depth map. The greater depth from the object's geometry will result in a more deformed curve, providing us with a sense of surface information from objects which is what artists seek to communicate. In Figure 3.7 the green and blue curves illustrate how the curvature changes drastically by the given depth map. Due to the shape of the sphere, the blue stroke will have a greater deformation than the green one.

Our algorithm receives parameters such as the amount of samples μ on the line, amount of strokes υ and thickness γ.

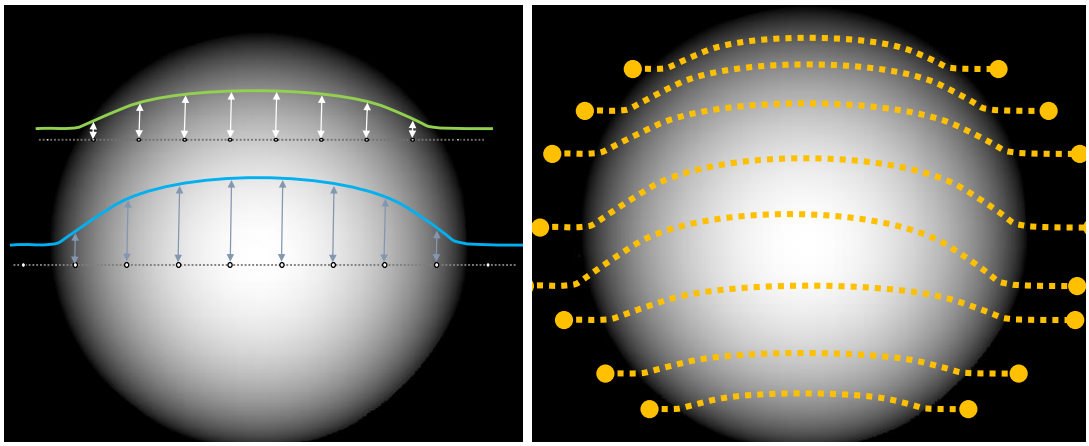

Figure 3.7: Left: Two horizontal dotted lines display the samples taken and their depth value. This depth will make an impact on the curvature of the rendered green and blue strokes.

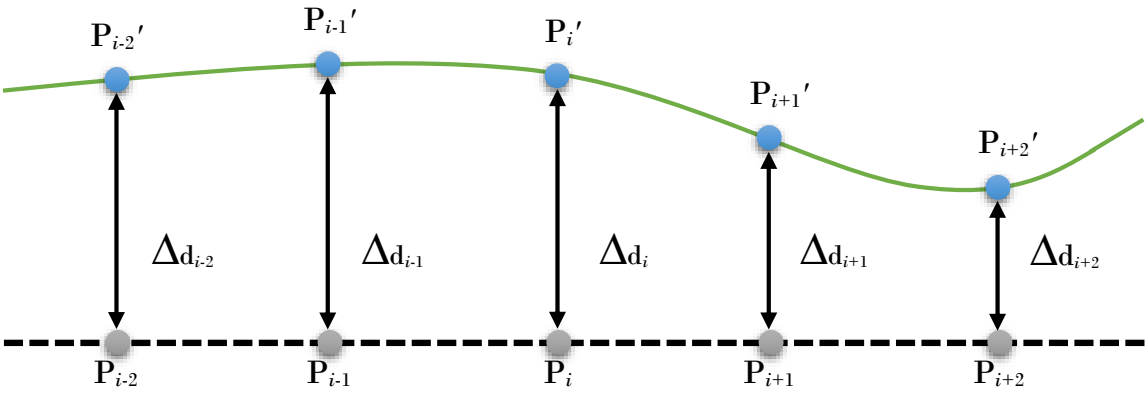

Figure 3.8: Every P_i sample is parsed through the line, the goal is to find the Δ_{d_i} displacements in order to find the displaced vertices $P_i\prime$.

The amount of curves l_j defines every curve to be rendered. On each curve, the amount of samples P_i ranges from $1 \leq i \leq \mu$. Parameters of μ and υ can not be used to compare results because they will depend on the resolution of the input images (e.g. having 100 strokes on a 500x500 image will look denser than having the same amount on a 1000x1000 image). Therefore, we use d_x as the spacing between samples and d_y as the spacing between curves.

Our goal is to find the displaced point denoted by $P_i\prime$. In the simple case the displacement will only occur vertically, so x_i will not be affected.

$$x_i\prime = x_i \tag{3.1}$$

Every position of every sample is defined by Equations 3.2 and 3.3. Each vertex P_i is located in (x_i, y_i). Our equation uses the depth value from the vertex position to directly affect the displacement of the sample. This depth value ranges from $0 \leq I(x, y) \leq 255$ which portrays the 256 levels of depth.

$$x_i = \delta_x i \quad ; \quad i = 1, 2, ..., \mu \tag{3.2}$$

$$y_i = \delta_y j \quad ; \quad j = 1, 2, ..., \upsilon \tag{3.3}$$

In equation 3.4, we divide the measured depth $I(x, y)$ by the I_{max} value which is 255. We acquire a factor between 0 and 1. This factor multiplied by a parameter sets the maximum amount of deformation the curves can have. We define φ as the exaggeration of displacement parameter, which controls the maximum amplitude of the displacement.

$$0 \leq \frac{I(x_i, y_j)}{I_{max}} \leq 1 \tag{3.4}$$

Having defined the amount of deformation according to the level of depth, we define $y_i\prime$ as our initial y_i position plus the displacement Δd_i from Equation 3.5. Therefore, we define our displacement as Equation 3.6, with the displacement ranging from 0 to φ. This is clearly displayed in Figure 3.8 where a $P_i\prime$ is calculated by adding the displacement value Δ_{d_i}.

$$\Delta d_i = \varphi \left[\frac{I(x, y)}{I_{max}} \right] \tag{3.5}$$

$$y_i\prime = y_i + \Delta d_i \quad ; \quad 0 \leq \Delta d_i \leq \varphi \tag{3.6}$$

These equations are used for every sample in the line. The obtained vertices will

be used for rendering the deformed curve. Consequently, this approach is used for every stroke. The number of samples will cause a change in the quality of the line. Moreover, for rendering purposes, having a small quantity of splines can also generate a smooth result due to the use of a spline rendering tool as we will explain further in our rendering section. See Figure 3.9. It is important to highlight that if the parameter φ is too big, it can cause strokes crossing each other, which is undesirable.

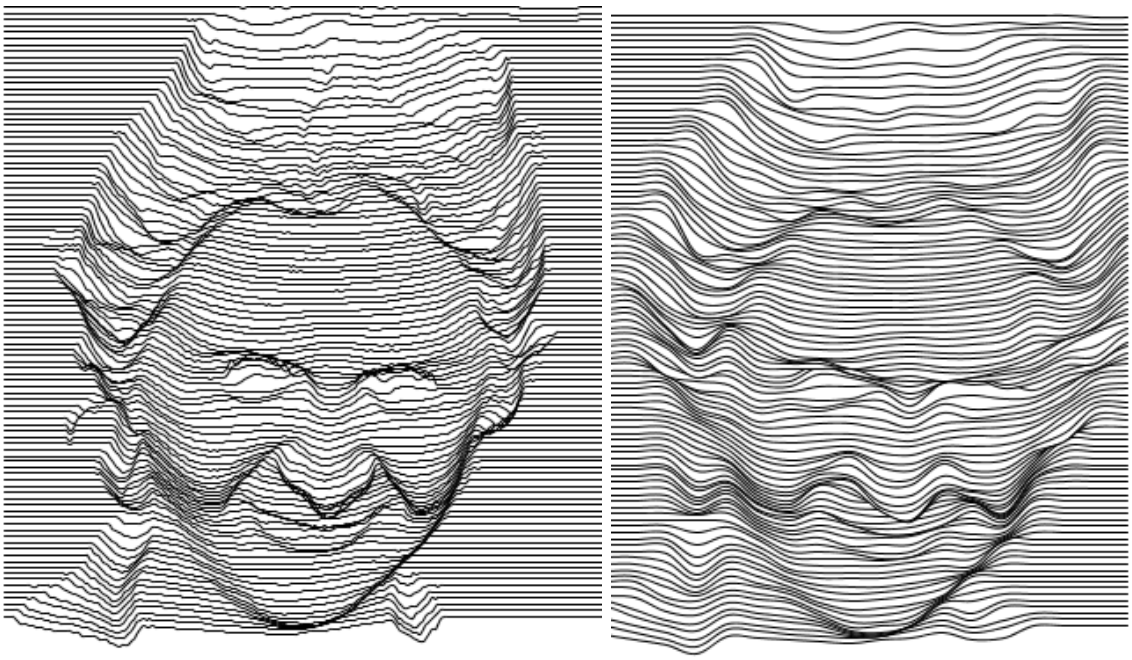

Figure 3.9: Left: Horizontal case. Right: Horizontal case with very few samples.

Having explained the curvature for one line in a horizontal case, we continue to deform every line in our ROI. Once this is completed we use our mask for this region to cut undesired parts of these lines in order to get a rendered region. Once a region is completed, our approach will go to the next region with another pattern direction angle, until every region is rendered and our final image is created.

In a sense, our approach works as a puzzle, working on each of the regions (i.e. masks) and rendering patterns to fill the entire image. Besides using our input parameters and mask, we also have our canvas image that stores each of these pieces once rendered. Since we work with our segmented masks, lines that extend beyond the mask will be cut, and lines from a region will never cross with lines from another region.

Figure 3.10: The generated patterns hatching effect.

This concludes our horizontal scenario, in which we have the same zero degree angle for directionality of our pattern. Figure 3.10 displays the same scenario but taking into consideration the same tilted angle for the entire image. The following section will detail how this is achieved.

3.6 General Stroke Generation

For our general approach, we are required to change our system's coordinates to be able to use our pattern generation approach. The angle Θ defines the direction of our patterns. We apply rotation on the plane using this angle, as seen in Figure 3.11. Different pattern orientations are displayed on different parts of the image.

Using rotation equations, we rotate the plane around the z axis, which will change our coordinate system and then allow us to apply our pattern generation approach. This will grant the ability to draw the strokes on any angle required. Using a rotation matrix displayed in Equation 3.7, we change the tilt of the coordinates used in the

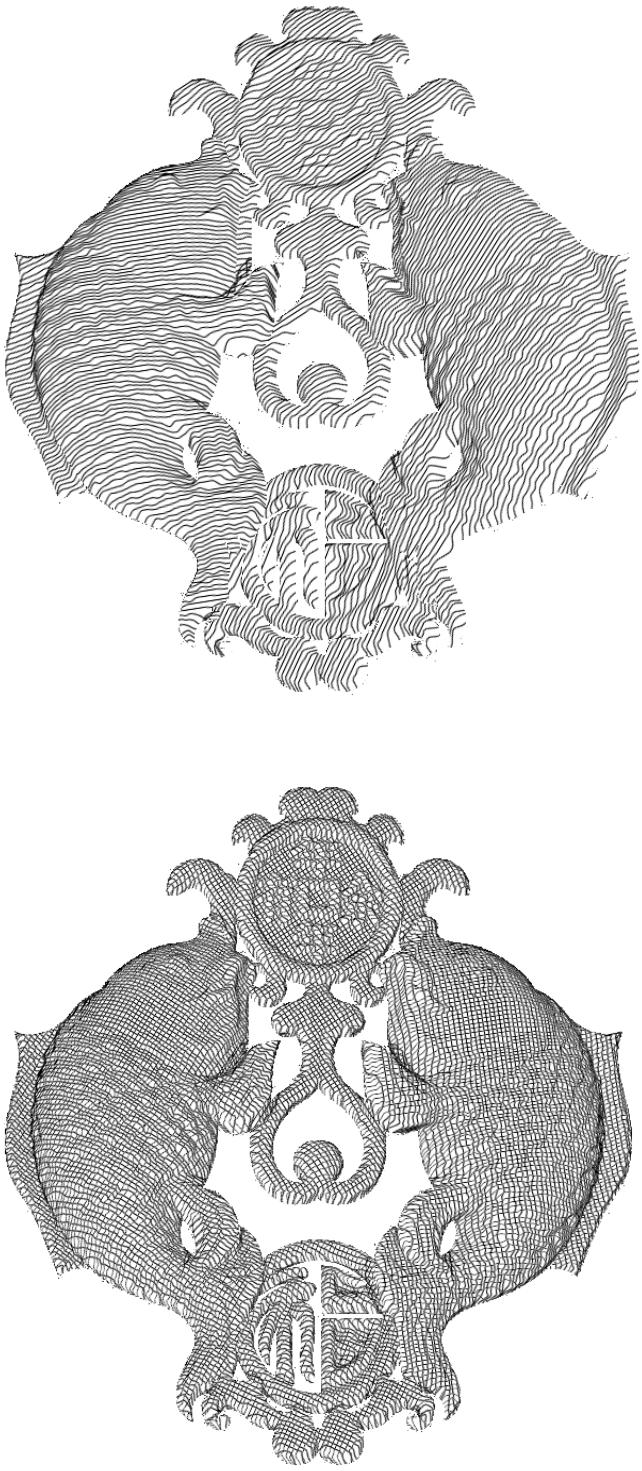

Figure 3.11: Our hatching effect is displayed on our first image. The curvature has enough detail that texture information from the image is also perceived. Applying the crosshatching effect to the entire image creates an aesthetically interesting result.

pattern generation approach with Equations 3.8, 3.9, and 3.10.

$$R_z(\Theta) = \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 & 0 \\ \sin\Theta & \cos\Theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x\prime \\ y\prime \\ z\prime \\ 1 \end{bmatrix} \tag{3.7}$$

$$x\prime = x\cos\Theta - y\sin\Theta \tag{3.8}$$

$$y\prime = x\sin\Theta + y\cos\Theta \tag{3.9}$$

$$z\prime = z \tag{3.10}$$

Finally, having described our approach for generating patterns, we should note that our image is not only composed of linear hatching, but our method also produces crosshatching effects as seen in Figure 3.12, in order to do this we use the thresholded data from the image that defines bright and dark areas in the image using a τ parameter, which is perfect to recognize dark areas (i.e. regions where artists will use crosshatching or have more density of lines). We apply our same approach for this region but instead of using our Θ angle, we use a perpendicular angle Φ, which provides our desired effect using the thresholded map as a mask. See Figure 3.12.

Our overall algorithmic approach is displayed in Figure 3.13 which details every step for generating the effect on the entire image. While the algorithm specified in Figure 3.14 specifies the approach for calculating displacements per region.

3.7 Rendering

We render our curved patterns on a white background. Overall, our sketch-like results look very close to our intended effects, although our preliminary implementation did not have the desired smooth touch we expected, lacking an anti-aliasing method while also drawing curves with very sharp turns.

We exported the data in a file that contained information for each vertex by

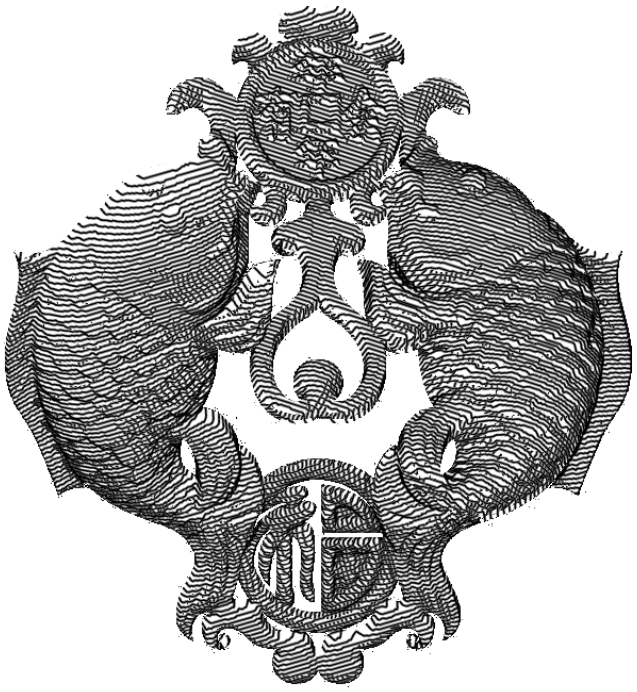

Figure 3.12: Our hatching effect with the use of crosshatching using our thresholded map.

Algorithm 1 Pseudocode for pattern generation approach

Input: Masks M[ ]
Output: sketchlike canvas
1. **Threshold image**(τ)
2. if(automatic approach)
    2A. etf ←**ETF calculation**
3. R*i*←**Segmentation approach();**
4. **For each region R*i***
4A. Generate ROI
4B. directionAngle(Θ)←**Get angle()**; // random or from vectors in ETF
4C. crosshatchingAngle(Φ) ←directionAngle+α; α=angle between normal and crosshatching strokes (usually 90°)
4D. PatternStrokes ← **createPattern**(Θ,M[i])
4E. CrosshatchingStrokes←**createPattern**(Φ,M[i])
4F. result← **Add**(PatternStrokes, CrosshatchingStrokes)
5. **endforEach**
6. canvas← **Add**(canvas,result)

Figure 3.13: Our main Algorithmic approach.

Algorithm 2 Pseudocode for creating patterns (createPattern)

Input: depth-map D, Mask M, angle θ, int υ, int μ, int f_w
Output:
1. $\delta_y \leftarrow$ D.getHeight / υ
2. $\delta_x \leftarrow$ D.getWidth / μ
3. **For each** $\boldsymbol{l_j}$**(#curves)**
   3A. **For each** $\boldsymbol{P_i}$ **(#samples)**
   3B. $\boldsymbol{P_i}'$ (displaced Vertex) $\leftarrow$ **CalculateDisplacement**($\boldsymbol{P_i}$, δ_x, δ_y) (See Equation 3.6)
   3C. Store $\boldsymbol{P_i}'$ into arranged by each $\boldsymbol{l_j}$
   3D. if($\boldsymbol{P_{i-1}}'$ exists){ $\boldsymbol{P_{i-1}}' \leftarrow$ **getPreviousDisplacedVertex**() }
   3E.$\gamma_{l_j} = [I(\boldsymbol{P_i})/I_{max}]\, f_w$
   3F. renderedImage $\leftarrow$ **Render_Primitive**($\boldsymbol{P_{i-1}}'$, $\boldsymbol{P_i}'$, γ_{l_j})(See Equations 3.12)
   3G. **endforEach**
4. **endforEach**
5. **Cut**(renderedImage,M)
6. **return** renderedImage;

Figure 3.14: Algorithm for pattern creation.

stroke. This meant we would organize the amount of strokes, and how many vertices are contained by each one individually. This file is then read by software that could render our stroke-based technique and provide very smooth results such as those seen on Figure 3.18. We then used Processing to render our results. Basically, an implementation to read and generate these lines was performed with a function that parsed through the files gathering data to categorize each stroke and its vertices. The curves were rendered as Catmull-Rom splines [14] which produces smooth curves as seen in Figure 3.15.

This component is not crucial to our pipeline for obtaining results. We can get our hatching and crosshatching effect as seen in Figure 3.17. But this component allows us to render curves in a more aesthetically pleasing way, since strokes have a more smooth and natural feel. More importantly, aliasing is no longer an issue.

$$l = \bigcup_{i=1}^{\mu} \overline{P_{i-1}\prime P_i\prime} \tag{3.11}$$

Additionally, in this component we can render each stroke with Catmull-Rom splines

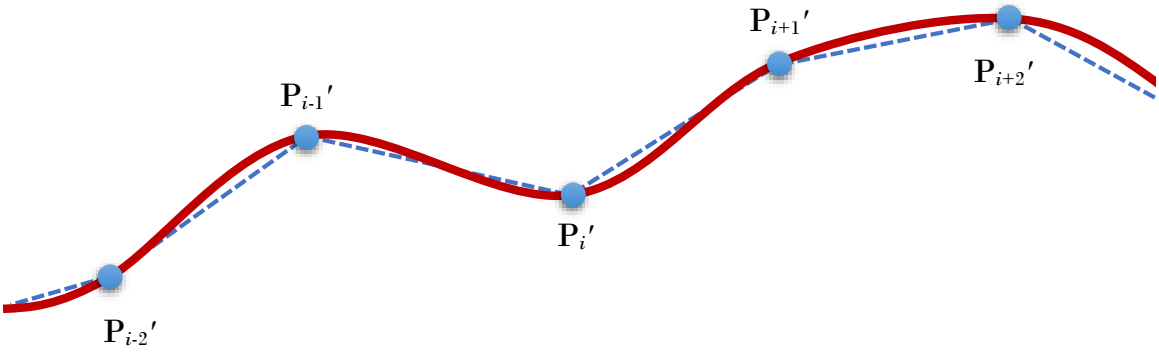

Figure 3.15: Using Processing's Catmull-Rom splines create a smooth stroke through the displaced samples.

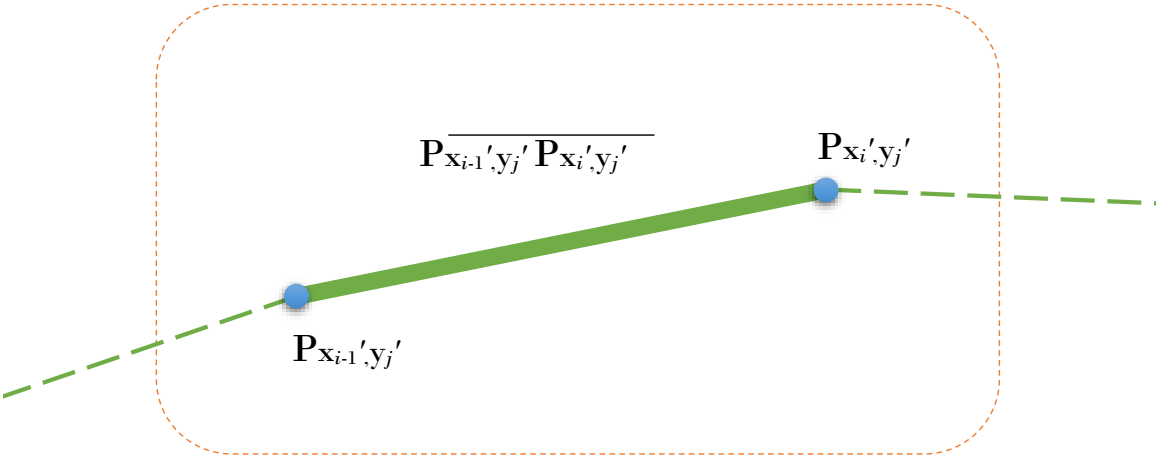

Figure 3.16: A segment of a curve is denoted as $\overline{P_{i-1}'P_i'}$.

or by drawing segments of lines individually. This allows us to vary the thickness along the curves to create a more faithful effect of engraving art. In equation 3.11, we define every stroke on the image as the union of all these segments of line per curve.

In order to define thickness denoted by γ, we calculate the intensity value from the original image, or by performing an illumination calculation using the depth map. We divide this intensity value by I_{max}, which will be multiplied by f_w. This will control the maximum thickness as denoted by Equation 3.12. Our thickness parameter will therefore affect the segment of line $\overline{P_{i-1}\prime P_i\prime}$ which is seen in Figure 3.16.

$$\gamma = \left[\frac{I(x_i\prime, y_j\prime)}{I_{max}}\right] f_w \tag{3.12}$$

Finally, this component can also blend some of the lines by extending them into other regions. This is done by morphological operators on the masks. We generate an extension of the masks by applying the morphological operator of Dilation. Using the extended masks, we render strokes that extend into other regions. Some engravers like to extend layers of strokes into others which is why we complemented our work with this functionality.

Figure 3.17: Our hatching effect preliminary results.

Our approach, while simple, does generate curved lines in the desired way as expected. The behavior of the lines provides a sense of surface geometry, like artist's sketches. This is controlled by our equations, which allows us to exaggerate our curves with our established φ parameter. Parameters can provide us with several variations of the style, which will be addressed on our next chapter.

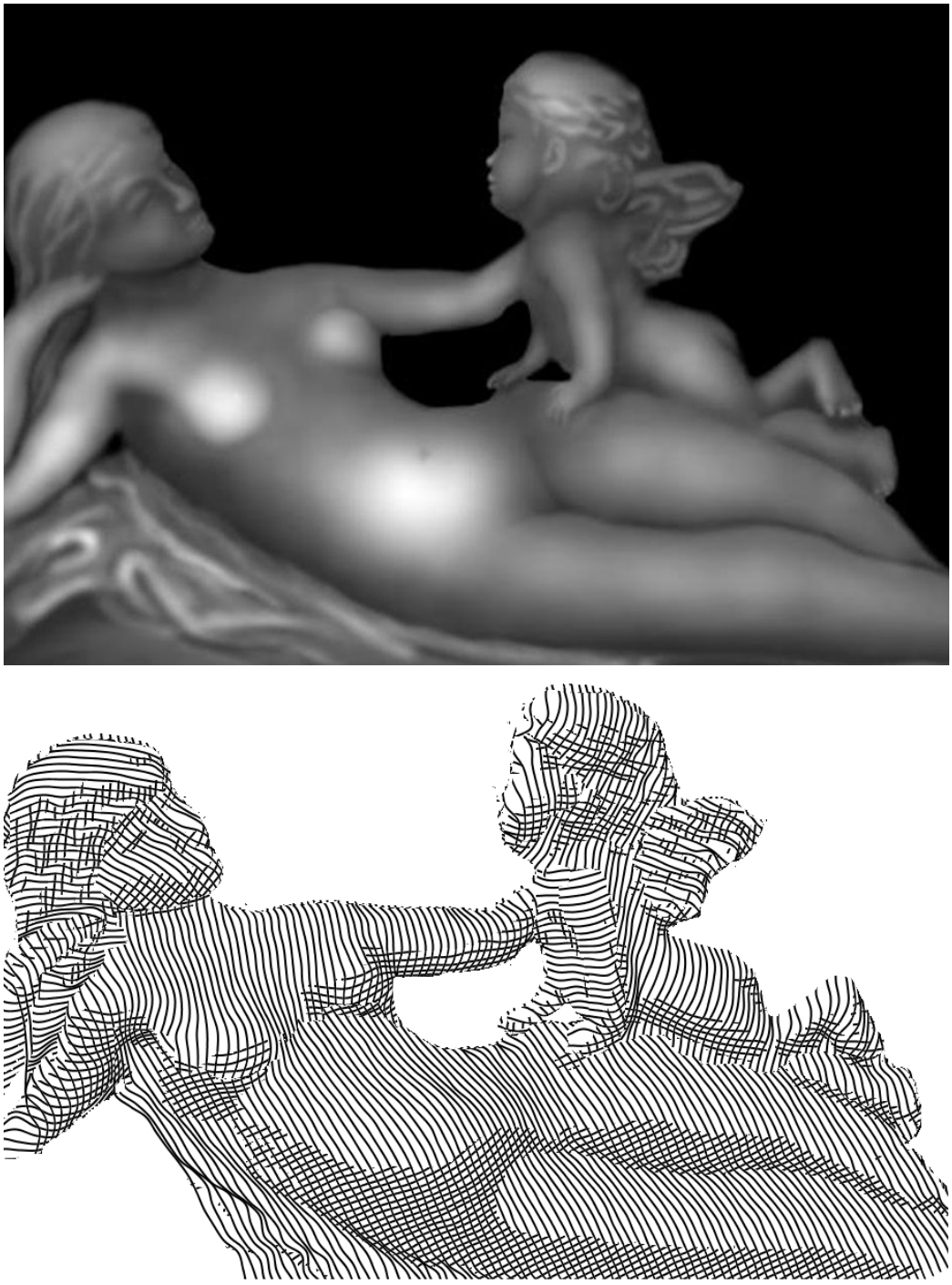

Figure 3.18: Results rendered with splines.

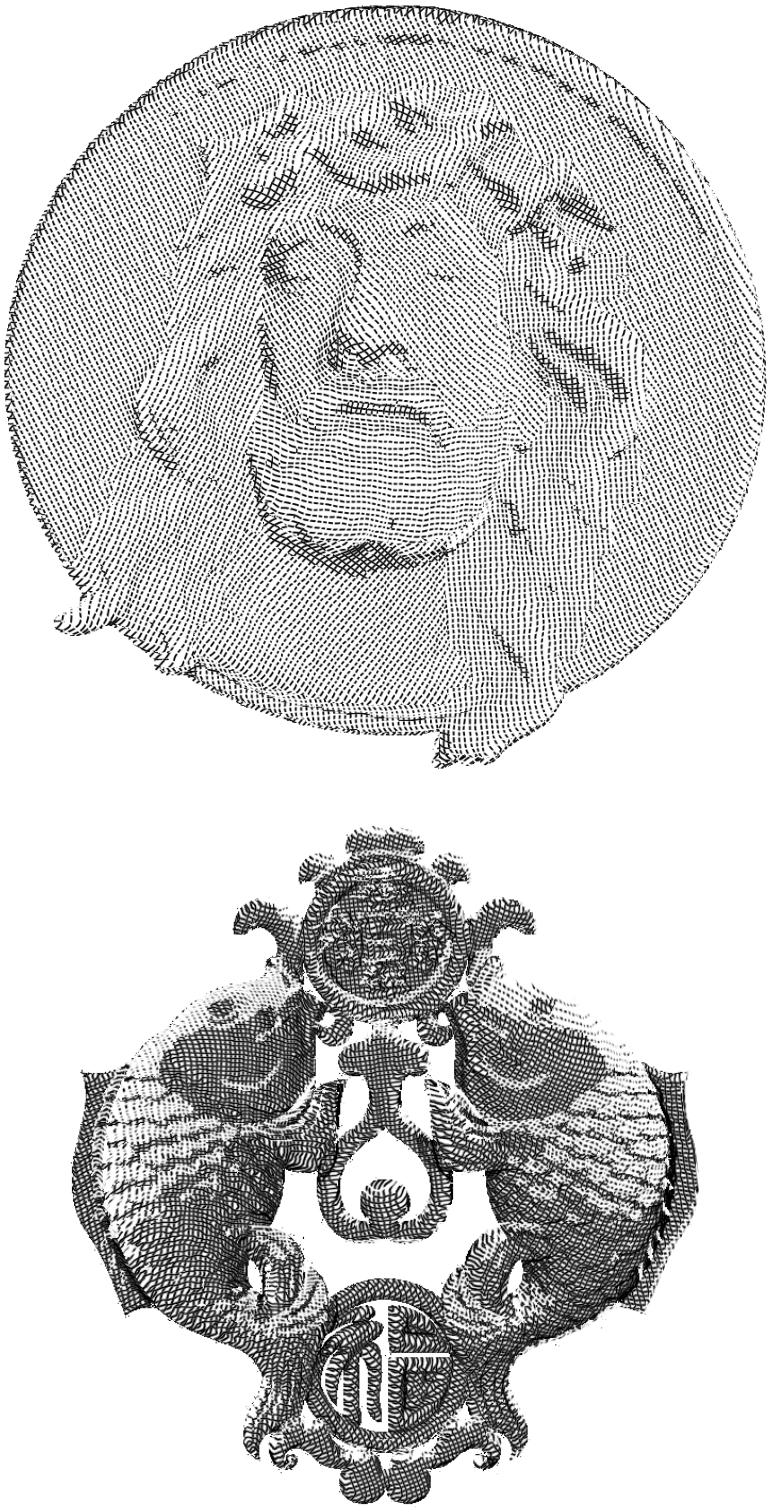

Figure 3.19: A few of our engraving stylization results.

Chapter 4

Results and Discussion

Having explained our methodology, this chapter will discuss the obtained results, show how parameters affect our effects, analyze and compare our approach with previous relevant work, and present a few different variations of our results.

4.1 Stages: Step by step

Our first step is to create a binary map to separate dark from bright regions. This map indicates which areas require crosshatching. We use the original photograph from the scene or we apply an illumination method using the depth map. We use this for applying this thresholding step, where we get a binary map as seen in the top left image in Figure 4.1, which will later be used in the rendering process.

Having acquired this black and white representation we proceed to our segmentation approach. Both of the segmentation methods provide an array of masks, SLIC providing more regions than the user approach. The top right image in Figure 4.1 shows the detected masks displayed with different colors. Such information allows us to store an array of masks with an individual ID.

Once we have obtained every mask, the angle Θ is assigned to define each pattern's tilt. The pattern generation step will go through every region using an individual angle for each one. These angles will also define their perpendicular Φ angle for the tilt in crosshatching areas.

The pattern generation component draws the curves considering parameters, such as the sampling distance δ_x, stroke spacing denoted by δ_y and the user-defined curve exaggeration factor φ. Every rendered region is then added to the canvas individually.

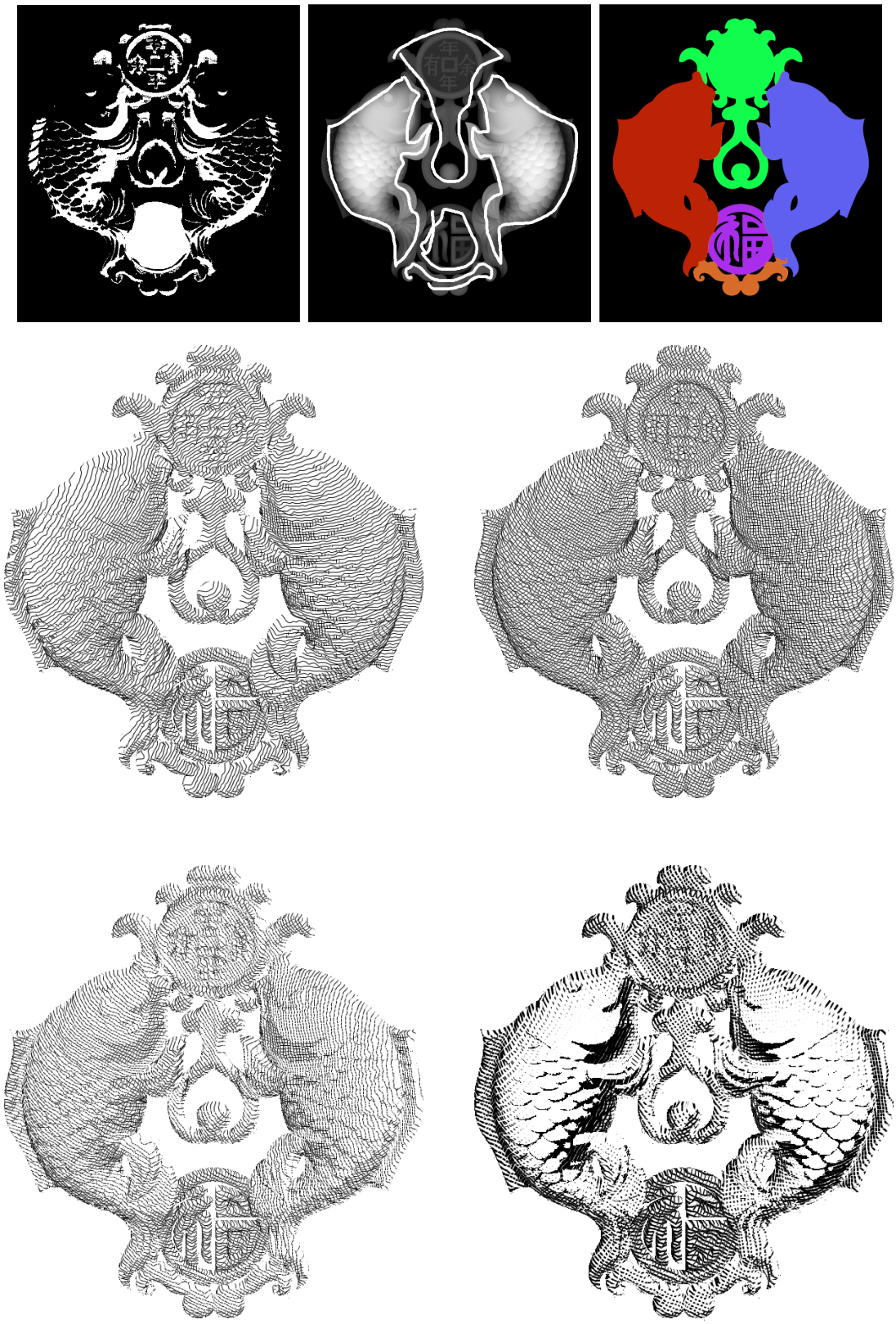

Figure 4.1: Our overall process for generating our hatching and engraving effect.

At the same time, for the thresholded areas that portray darkness, a separate pattern generation instance is performed, using Φ. When finished, our rendering process will join the hatching and crosshatching areas using our thresholded map as a mask. The complete hatching result can be observed on the second row left image on Figure 4.1. Note how the details of the surface texture can be transmitted to the viewer. In this case the scales are portrayed by the small variations of the splines.

Additionally, we create engraving art styles by subtracting a complete crosshatching map from the hatching result. This cross-hatching map can be seen in the second row right image from Figure 4.1. We then complement it with the cross-hatching areas using the thresholded map which will result in our desired engraving effect as displayed in the bottom left image from the same figure.

To finalize our effect while plotting lines individually and trying to achieve our engraving effect, we apply a different thickness γ for every segment of the strokes, taking into account the intensity value of the original or illuminated image. The user can define a maximum value of stroke thickness with a f_w factor as previously explained. The result in the bottom right image in Figure 4.1 is very reminiscent of a contour-hatching effect.

Use of SLIC

To complement our methodology we opted to include an automatic approach as explained in previous chapters. The use of SLIC allowed us to generate more regions than the ones provided by a user input approach. This meant having to work with more masks. We used Kang et al.'s ETF method, where we are provided with a vector field. We use each vector to define each region's tilt as seen in Figure 4.2. The top right image is very inspiring for future work; the effect looks very similar to a maze. Pedersen and Singh presented a method for generating mazes in 2006 [41]. Another possible direction is the integration of Inglis and Kaplan's Op art styles [22,23]. Using masks as input will most likely generate some interesting effects.

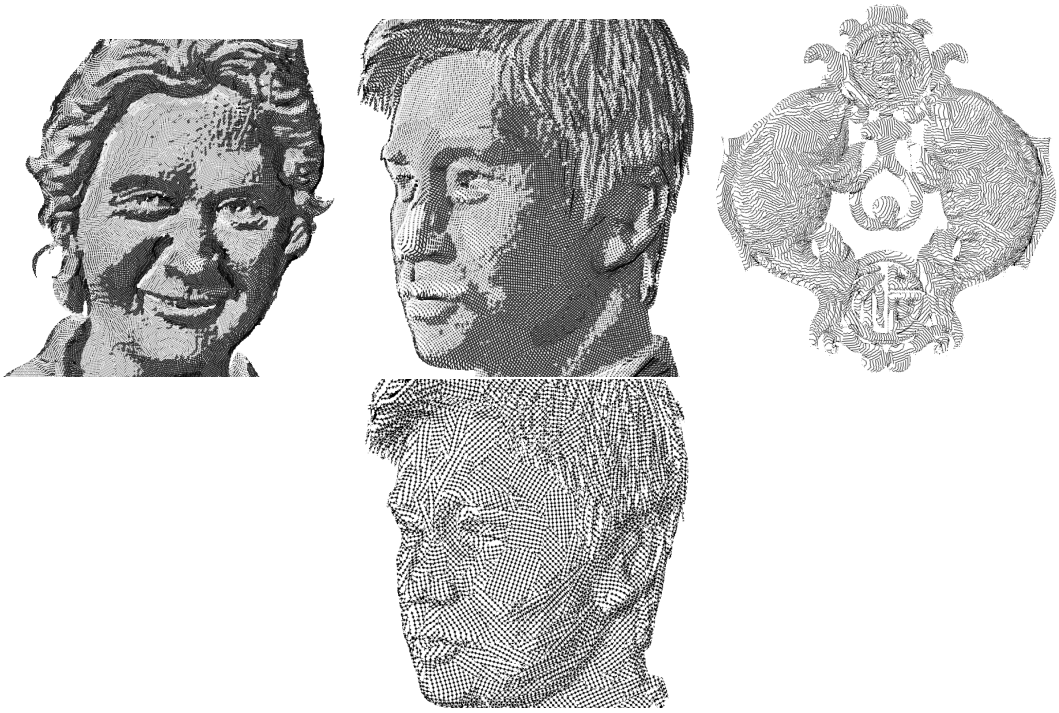

Figure 4.2: Using SLIC generates a lot more masks which are assisted by ETF to define the tilt on each patch. In the bottom image, by using lines and circles, an interesting effect is generated.

4.2 Parameters

This section describes how parameters affect the results. The following subsections will display how variating these variables will change our final effect. The parameters are the following:

- Spacing between samples (δ_x)
- Spacing between strokes (δ_y)
- Exaggeration of displacement (φ)
- Thickness (γ)
- Threshold amount(τ)

4.2.1 Spacing between strokes and varying sampling

Parameters δ_x and δ_y define the distance between samples and strokes, respectively. These parameters will change our results radically depending on their values. The variable δ_x will control how many samples will be taken on a line, whereas δ_y, the amount of lines per region, is calculated using the height instead. Since μ and υ will vary the results depending on the resolution of the image, we calculate them using the defined spacing of δ_x and δ_y, which are general parameters. We present some results with δ_x and δ_y in Figure 4.3.

Overall, the manipulation of these parameters will change the quality of the stroke and the density of curves in the images. It is important to highlight that even in cases where few samples are taken, the use of Catmull-Rom splines will provide very smooth results. On the other hand, rendering each section of line individually using very few samples will result in strokes with very sharp turns, which is undesirable. The top image in Figure 4.4 displays the use of very few samples, a μ of 30. This result serves as a motivation for a texture generation method of such as hair. Generating a fake depth-map can be used at creating some interesting textures while resolution independent.

4.2.2 Displacement Exaggeration

When we defined our parametrized function, our calculation of Δd provided us with the distance we needed to displace each point depending on the depth. The factor

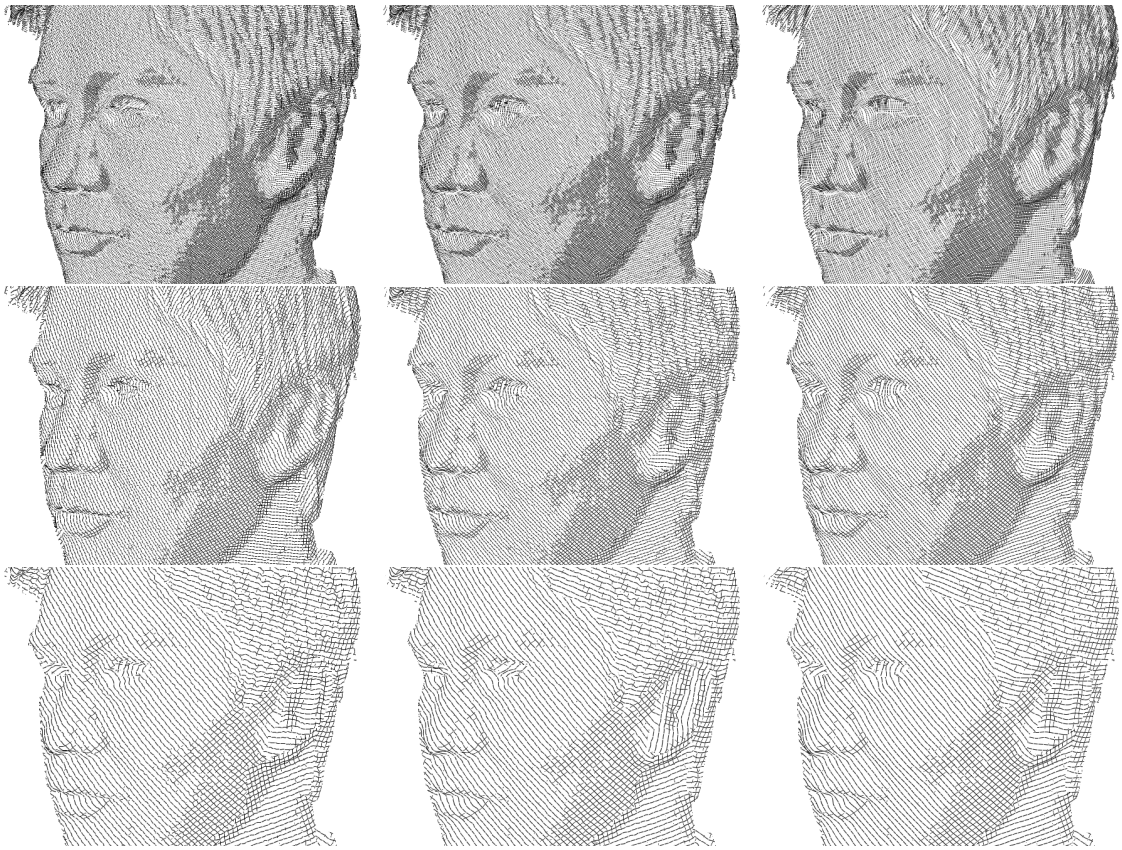

Figure 4.3: Changing δ_x and δ_y. We test this with $\delta_x = 4,7,13$ displayed on each of the columns and variating $\delta_y = 3,5,7$ on each row. Therefore, the top left image has a sample separation $\delta_x = 4$, and a stroke spacing of $\delta_y = 3$, whereas the bottom right corresponds to $\delta_x = 13$ and $\delta_y = 7$.

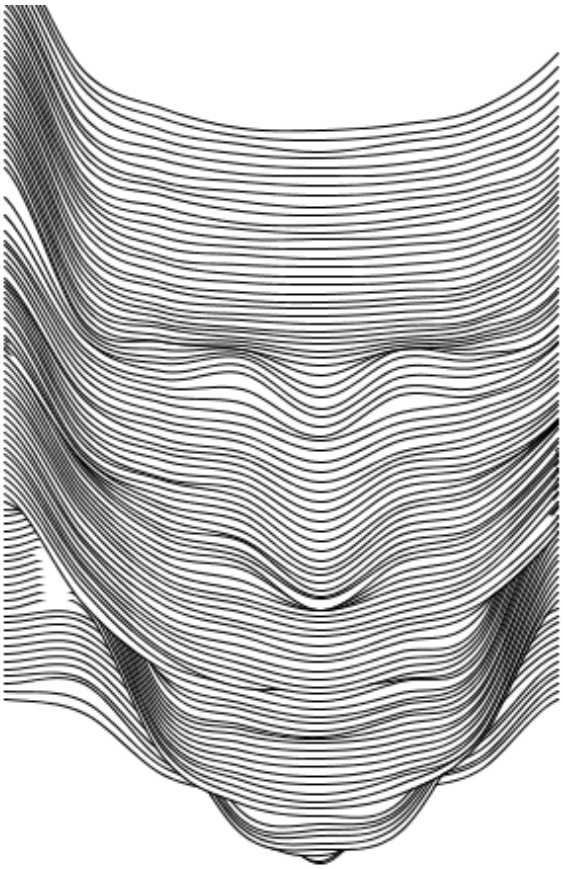

Figure 4.4: An interesting stylization obtained while getting preliminary results.

φ controls the amplitude of displacements. This parameter is critical for our desire to exaggerate surface characteristics. As seen in Figure 4.5, φ= 10 and φ=20 do not exaggerate the patterns very much. The first portrays no surface characteristics, while the latter only beginning to imply geometry in a subtle way. For generating our results in this thesis, we usually selected exaggeration values between φ=30 and φ=40. See Figure 4.6.

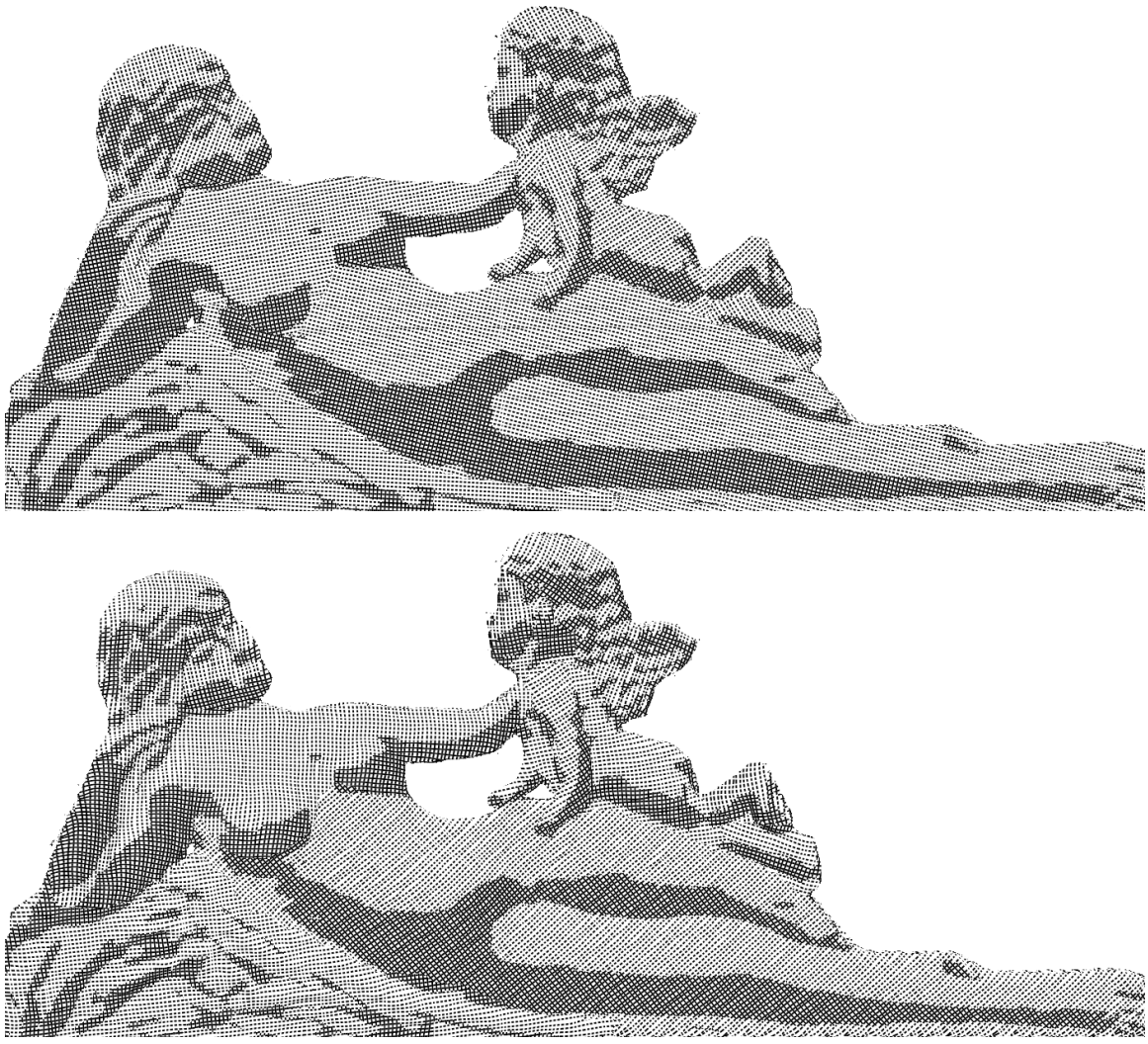

Figure 4.5: Varying φ will change the amplitude of the curves. The first image has a very small amplitude, which barely portrays surface information ($\varphi = 10$).

Using very high values of φ will exaggerate the curves greatly, making intersections between strokes inevitable. Our system tries to avoid these intersections, which can

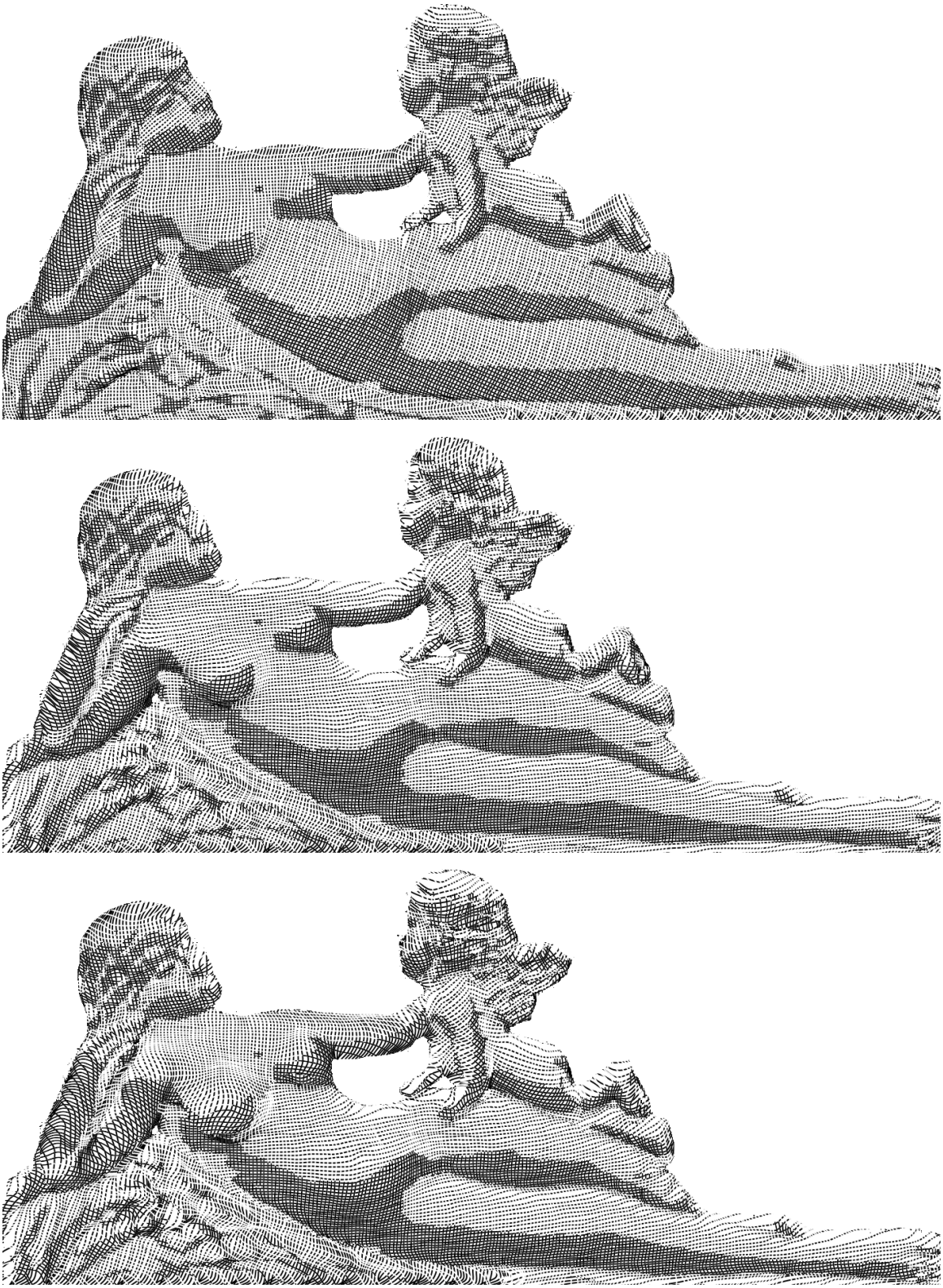

Figure 4.6: Results(top to bottom) obtained for $\varphi = 20$,30, and 40; they exaggerate the data in a more faithful way.

be unavoidable in certain cases due to surface geometry. This is a limitation that does not present itself often mainly because the objects are segmented and they are usually faces or objects with smooth surfaces. In the case where objects are complex in terms of shape (e.g. a complex tree), if not segmented properly our method will produce some undesirable results such as line intersection or curves with very sharp turns and being cut by the masks to name a few.

4.2.3 Varying Thickness

An important feature in engraving art is varying the width or thickness of the curves. We base line thickness on the intensity value $I(x\prime, y\prime)$ on the illuminated map. Having defined the maximum value of thickness with f_w, the thickness γ will vary on each segment of the spline l using the intensity values as explained on Equation 3.12. The impact of the changing these values are displayed with our engraving approach which provides some pleasing results in Figure 4.7. The use of this parameter drastically changes the tone of the image while also transmitting a sense of depth. Segments with a high level of thickness are susceptible to have an intersection with a neighboring line. This was expected because the closeness of lines is what portrays darkness. Nevertheless, in a scenario where the δ_y is very small and f_w is a number beyond 2, the width of the stroke can expand beyond neighboring lines which can cause intersection. A constraint that is defined depending on δ_y and f_w, can assign a maximum value of thickness that will allow them to have contact, but no overlap.

4.2.4 Thresholding

For providing greater contrast between bright and dark regions using the tone from the original or illuminated image, we use the thresholded map that separates bright from dark areas. The threshold factor τ is used in our system so that the user can judge the amount of threshold he wants to define so shadows and dark areas from the photograph are more prominent. By variating this threshold value the tone of the sketch changes drastically. In Figure 4.8, we use γ=15,25,50,90 and 200, respectively. If our thresholded map defines a dark section in the whole image, a cross-hatching effect will be drawn in the entire sketch. This generates a net-like style as seen in the last image on the second row of the figure 4.8.

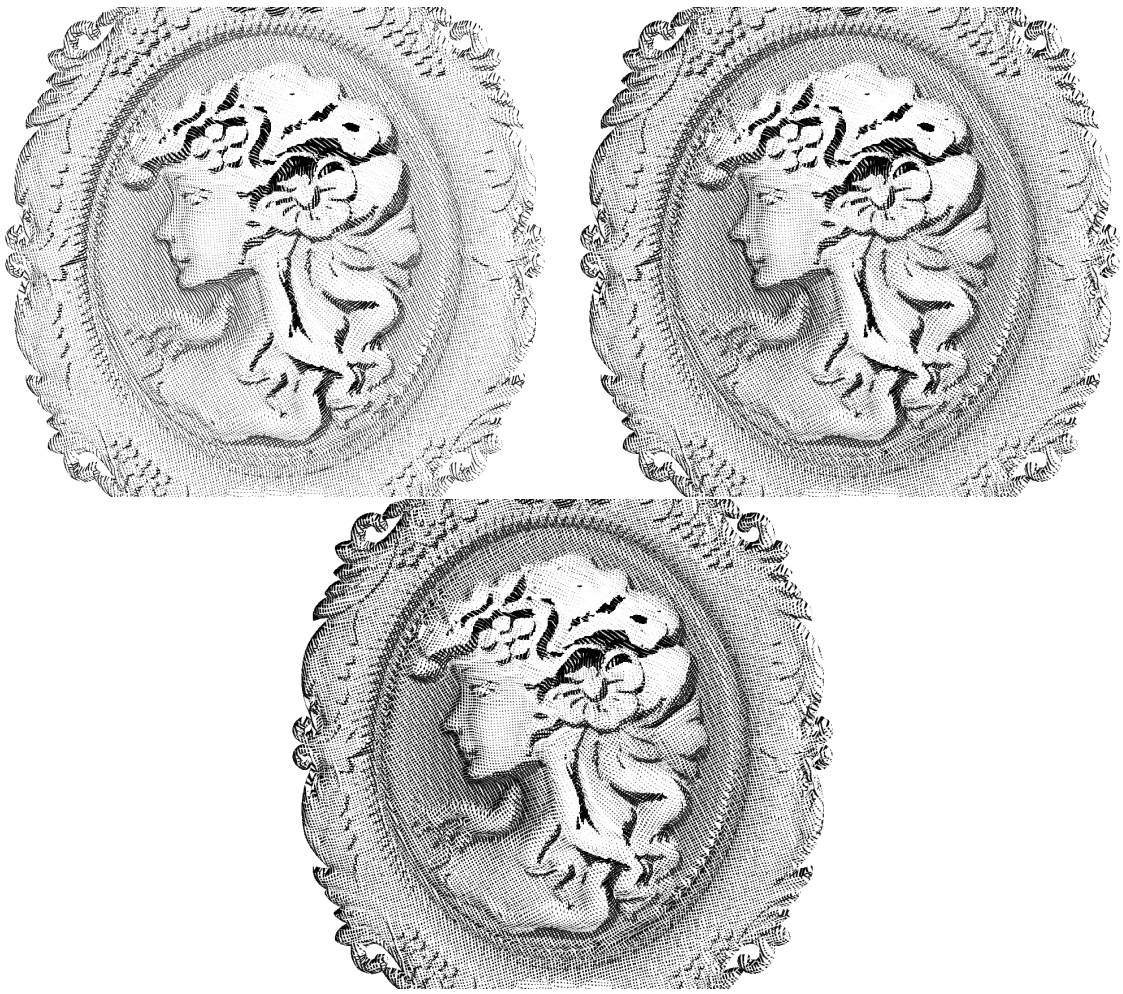

Figure 4.7: The user-defined f_w value, changes the thickness of strokes using the depth information. The parameters used where $f_w = 2$,$f_w = 3$, and $f_w = 4$, respectively.

Figure 4.8: When we vary the thresholding parameter τ, we display dark regions using crosshatching in those regions.

4.3 Comparisons and Discussion

4.3.1 Illustrating Smooth Surfaces

One of the most relevant works concerning hatching effects is Illustrating Smooth Surfaces by Hertzmann and Zorin [21], in which they present a method that uses 3D models in order to render their effect. Their algorithms work mostly with silhouette detection to generate several cuts as seen in the top row of Figure 4.9. They rely in 3D models to generate their smooth result. In our case, we use a depth image. For comparison purposes we rendered our hatching effect with a heavy use of crosshatching (i.e. a high value of thresholding), varying thickness using the illumination map, and thresholding our line rendering using the intensity values of the illuminated image to mimic their results. A comparison can be seen in center row of Figure 4.9. One of our limitations is that we have not integrated our system with a outline rendering method that could make our results almost identical to theirs. Their use of 3D models gives them a great advantage in terms of calculating the direction fields, nevertheless our curve deformations do portray geometry without this. While we still depend on depth maps, shape from shading techniques can be applied in the future to calculate a depth map from a photograph.

4.3.2 Coordinated Particle Tracing

One of the most relevant works in pen-and-ink is Wei and Mould's [49,50] coordinated particle tracing system. As previously mentioned, they used a particle system to generate the exact hatching effect that our implementation tries to achieve. A major strength is the use of photographs for achieving their hatching and engraving results. They use particles that follow most of our same rules for drawing splines. While our method uses sampling along lines to curve these primitives, their system uses particle tracing by the use of forces that displace particles through the input image. Communication between these particles is essential to avoid intersecting strokes. As seen in Figure 4.10, their result provides a very good result in terms of producing the desired effect. Our work improves the hatching and engraving renderings by the portrayal of 3D surface characteristics through the deformation of splines. As seen in the right image, our result can convey shape and depth while using few strokes

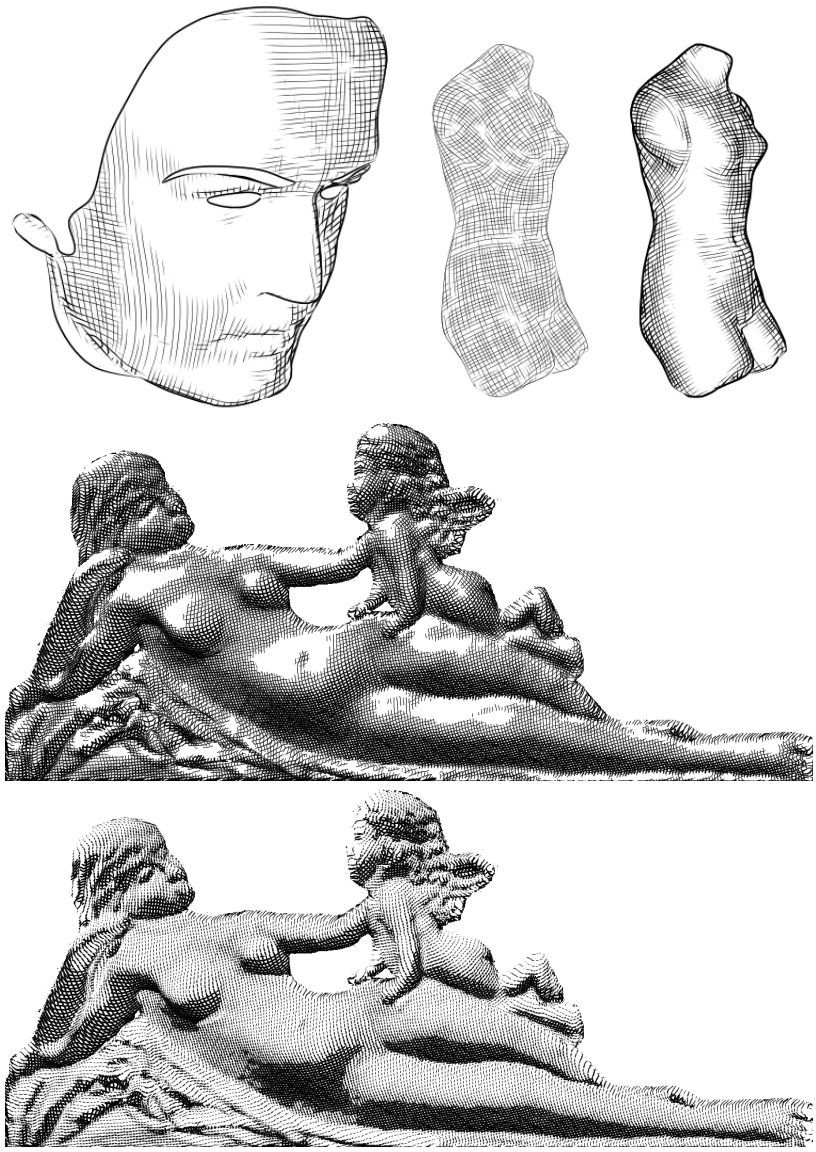

Figure 4.9: Top: Hertzman and Zorin's results. [21] Center: Using the described variant approach, we generate a similar illustration result. Bottom: Our engraving art result.

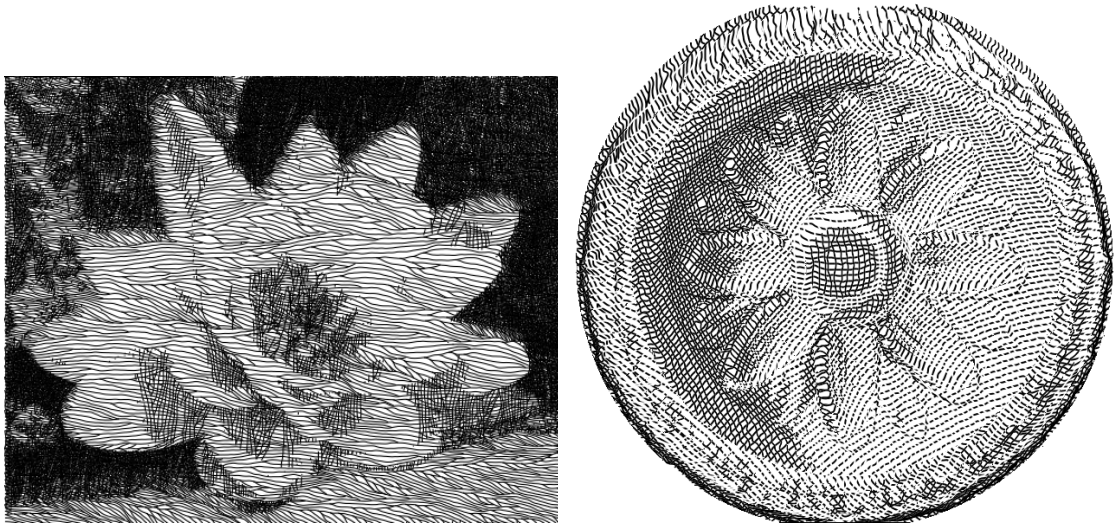

Figure 4.10: Left: Wei and Mould's coordinated particles system result. They match tone and display hatching and crosshatching but no information from the object's surface is conveyed [49]. Right: In our engraving result we use crosshatching and communicate volume. We used a $d_x = 6$ and a $d_y = 5$.

while still distinguishing bright from dark areas. Moreover, illuminating the depth map provides us with the ability of thresholding line rendering, resembling contour-hatching.

4.3.3 Digital Facial Engraving

Ostromoukhov's approach for digital facial engraving is the closest approach for creating engraving art [40]. Their method uses a similar approach concerning the use of a layer based method. Instead of using the object's characteristics and features to create deform patterns, they use a parametric grid in order to merge these patterns into a final rendering as seen in Figure 4.11. Overall, they do not possess or calculate any 3D information to portray surface characteristics.

Our method uses our mask-based approach to generate regions separately, parsing through the samples and measuring their depth data to render stroke by stroke. Thus, generating deformation from surface topography. This not only communicates the geometry, but when using a high number of samples, we can convey the small surface details as well. Our obtained patterns which are produced in a vector-based result communicates geometry from objects in the scene mimicking engraving art. His

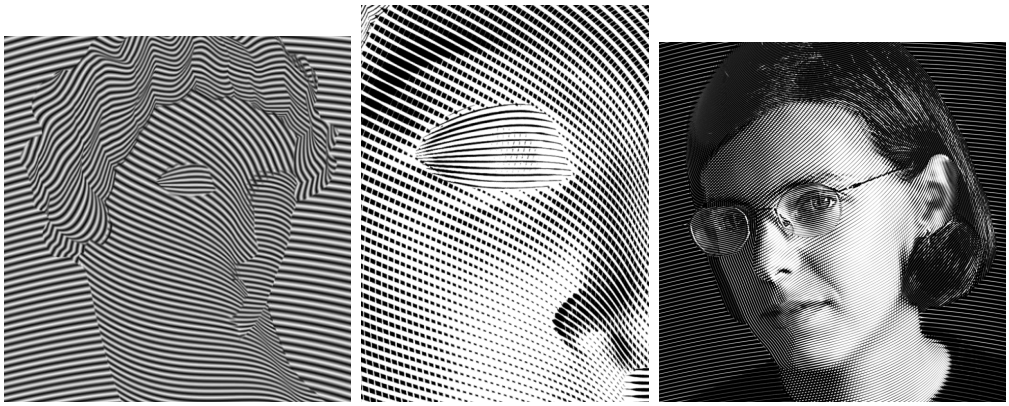

Figure 4.11: Left: Ostromoukhov's parametric grid using layers generates patterns. Middle: His engraving effect. Right: His final result matches the tone of the image. [40]

approach depends greatly on the tone matching method. Without it, the deformed patterns will provide fake surface characteristics that do not represent the geometry of the objects.

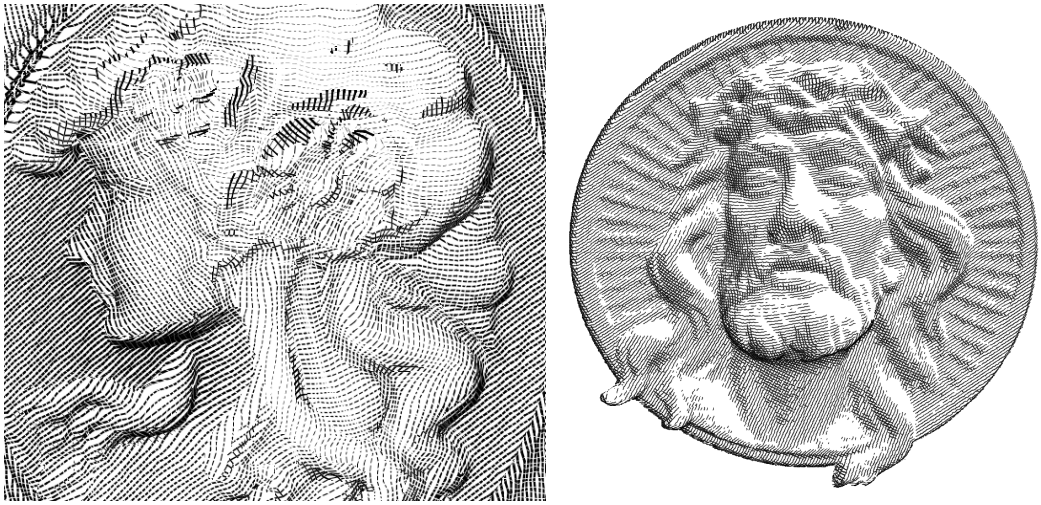

Figure 4.12: Engraving results

4.4 Rendering Variants

In this section we present a few variations of our renderings. Some were inspired by artists, while others were obtained while we were testing the rendering component and changing some parameters. Our goal was to generate hatching and engraving effects such as these results in Figure 4.13. Along the process of the testing of our approach, we obtained several interesting results that resemble stippling and different pen-and-ink results.

Different results were achieved by changing the rendering approach; Figure 4.14 displays three different renderings. The top left image shows the complete crosshatched result generated by using a a completely thresholded illumination image. The achieved texture resembles reptile skin in certain regions. The top right image from Figure 4.14 shows our engraving approach, omitting the crosshatch drawing, but using a crosshatching map as the one on the top left image to erase the patterns at their intersections. The resulting image is reminiscent of stippling rendering. Inspired by some engraving artists we used the same approach as the recently explained stipple-like rendering, but white lines drawn over a black background. The result can bee seen in the bottom image from Figure 4.14.

Apart from varying the operations for rendering our results, we employed the use of other primitives, which is the case of the top image in Figure 4.15. The same engraving approach was used. However, instead of using just lines, we complemented them with large stipples on the positions of the samples. Subtracting a crosshatching map using these primitives resulted in the displayed stylization. This was not trying to mimic a specific effect but rather to shows the ability to generate new styles by modifying the primitives.

The center left image in Figure 4.15, shows our rendering approach that varies the thickness of the strokes. No crosshatching was added to this rendering. The center right image portrays an exaggerated thresholding with few samples, which provides a highly contrasted smooth result. The bottom image emulates the US bank notes in a very pleasing way. The crosshatching obtained on the lady's right arm is one of our most desired effects. The varying of the thickness seamlessly creates the same effect portrayed in US bank notes. Using the same approach, we attempted to render a contour-hatching effect which produced the result seen in Figure 4.16. This was

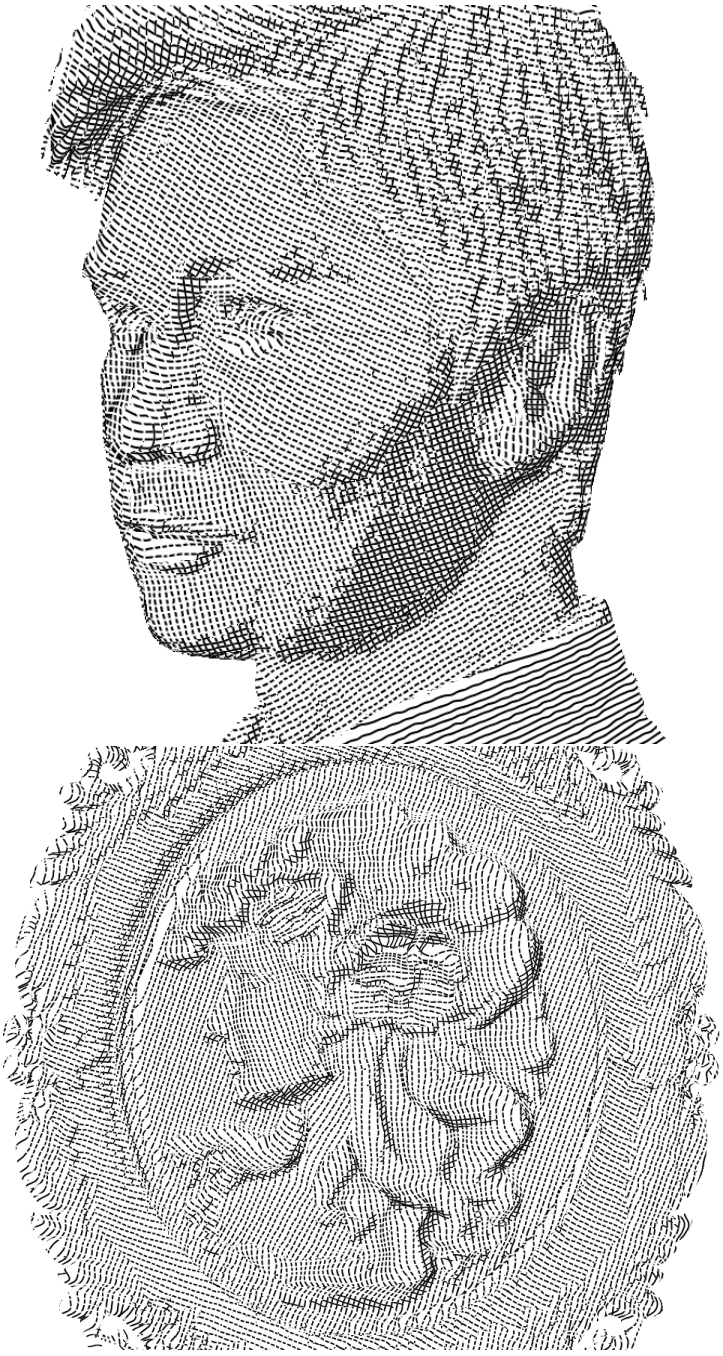

Figure 4.13: Our engraving stylization results.

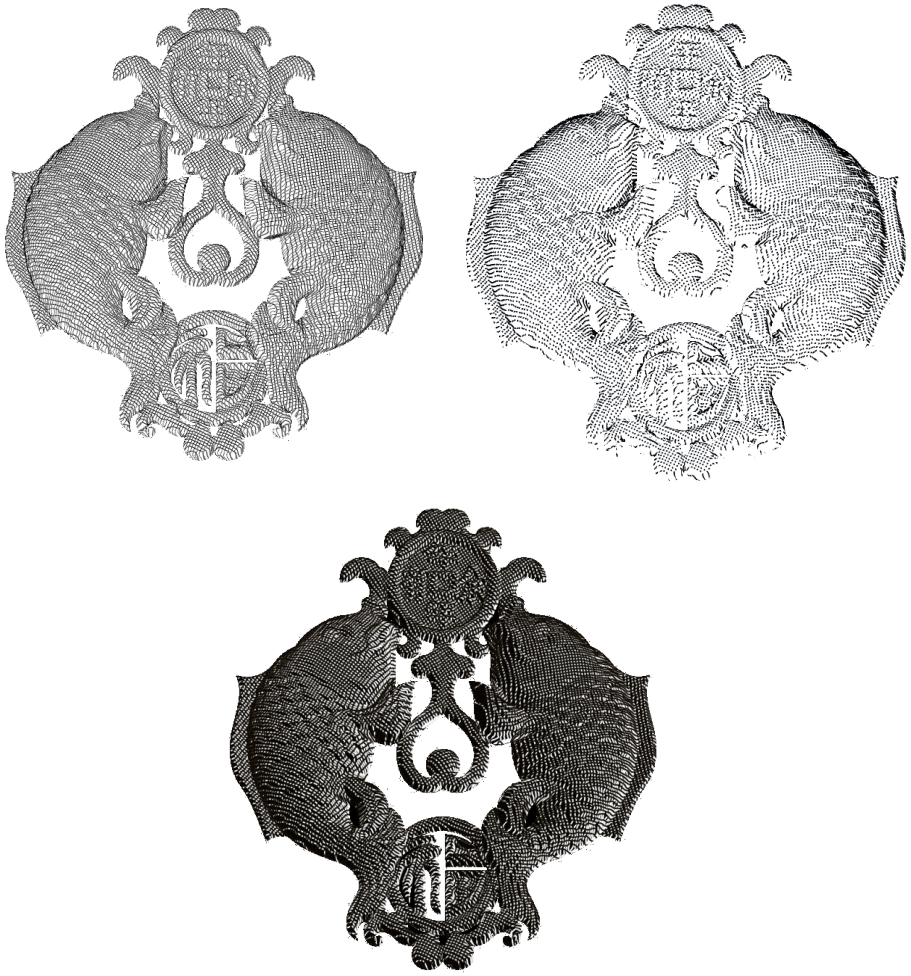

Figure 4.14: Top Left: Using crosshatched patterns on the entire image. Top Right: Our engraving approach that subtracts the crosshatched patterns using very thick splines from the hatched image. Bottom: This rendering uses the same approach as the middle image, but it works on a black canvas and works with white splines.

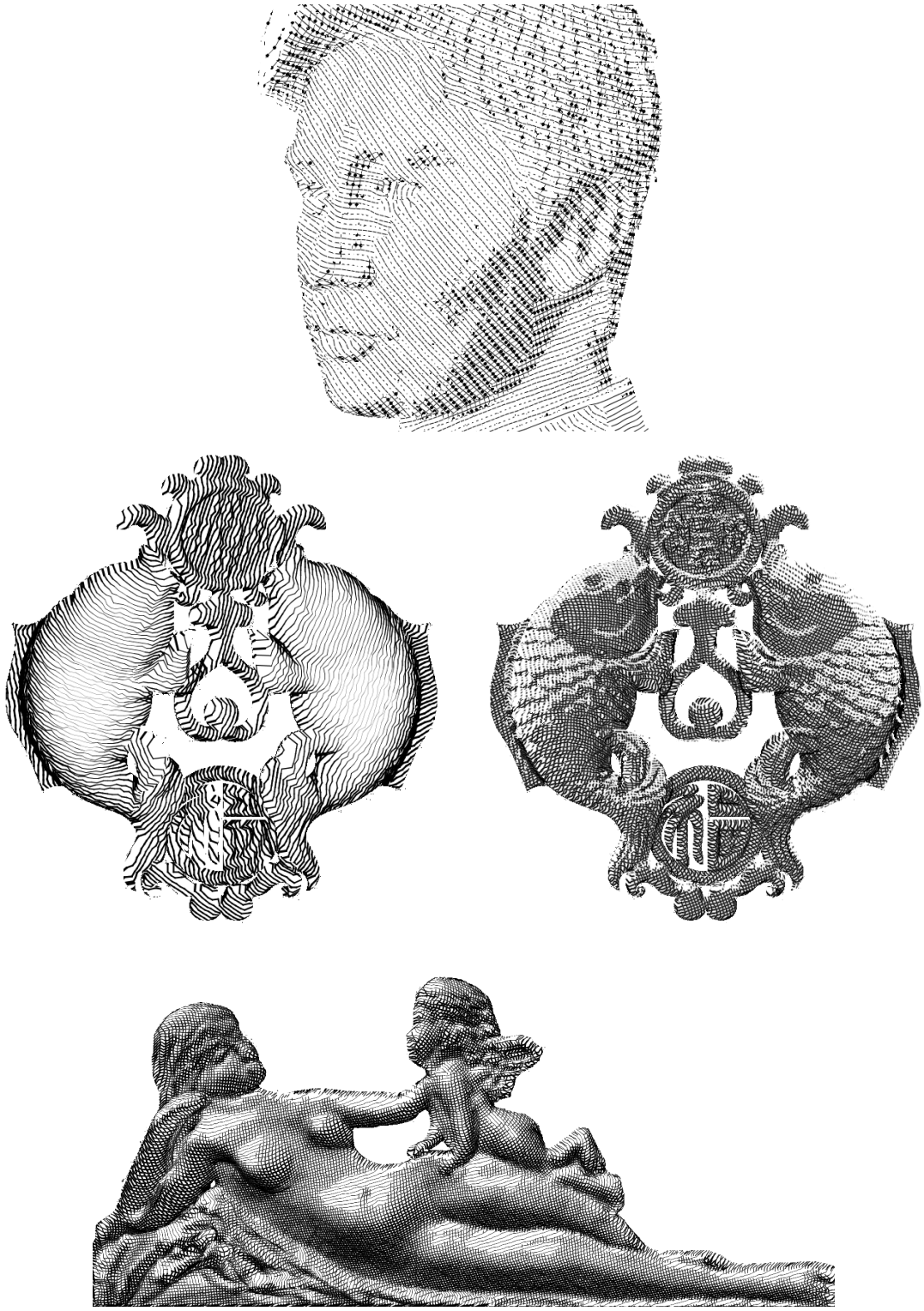

Figure 4.15: Several variations of parameters and rendering operations can generate very different styles.

achieved by thresholding the curve rendering in areas of high intensity. We do this for both the hatching and crosshatching areas with a slightly bigger threshold value for the latter. The result displays the hatching extending the crosshatching in a subtle way.

Figure 4.16: A contour-hatching result.

Finally, we attempted to use our system with photographs as input, using a gray scale version as a depth map. Even though there is no depth data, the intensity values on the image manages to deform the patterns while suggesting geometry from the face. The results in Figure 4.17 seem promising for future work involving the generation of fake depth maps from photographs. The upper images in Figure 4.17 show hatching results with different thickness and by not rendering segments of lines in areas of high intensity. It is important to highlight that this picture was taken with a flash, which made closer portions of the face brighter than the rest, therefore emulating a depth map. The bottom left image in Figure 4.17 displays the same thresholding of high intensity values when rendering lines for a hatching result, with a higher level of threshold. The bottom right image in Figure 4.17 shows our engraving approach applied to a photograph.

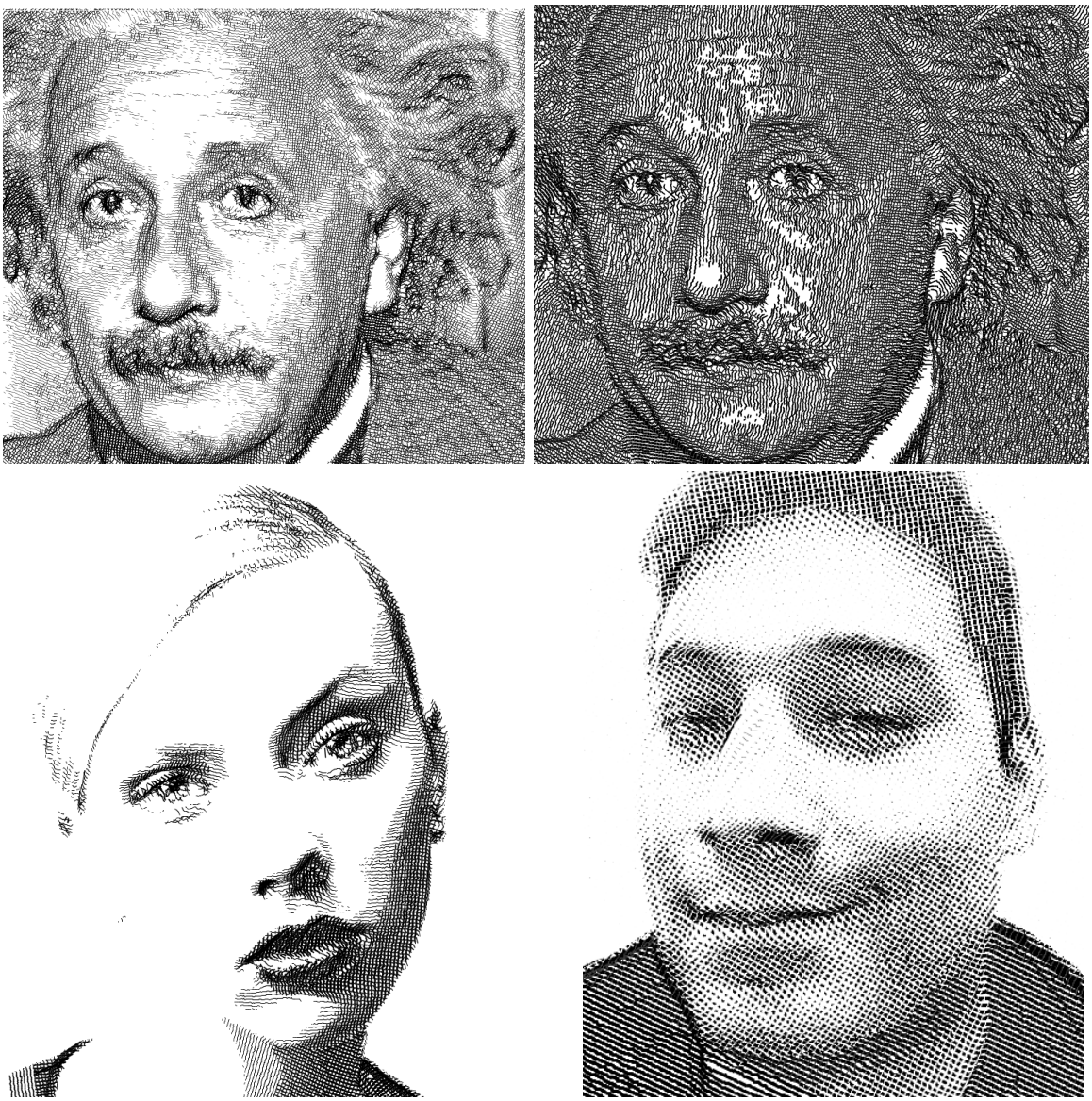

Figure 4.17: Applying our approach to ordinary photographs.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this thesis we present an approach for generating stylizations using depth information for hatching and engraving effects. These are characterized by the use of parallel lines that exaggerate surfaces of faces and objects. These styles have been greatly inspired by hatching artists such as Gustave Doré and Copperplate engraver artists.

We presented background on NPR, specifically on the areas of Line drawing, Pen-and-ink Illustrations styles and Engraving art, which are areas in which we believe we are making a relevant contribution with our work. Past techniques have produced very good results in generating these effects, although some of them possessing the advantage of 3D models for faithfully mimicking the effects.

Image-based stylizations of these effects have provided results with an absence of suggesting geometry, which is one important characteristic from these artistic pieces. To achieve our goal we assume our input to be a depth map, introducing a sampling approach that parses through our image to render smooth strokes. The taken samples are displaced by a parametrized approach which deforms curves based on the geometry from the scene. These mathematical methods calculate the displacement of the samples and work in a region-based method provided by two segmentation scenarios.

Our system generates results comparable if not an improvement to previous attempts to emulate hatching and engraving effects as we compare them with relevant techniques. Several variations of our results are also displayed, and new and interesting styles are discussed. These were done by varying parameters or changing rendering strategies when creating results. Additional variants were achieved by the use of depth to change thickness, thresholding and adding other primitives such as

stipples in the rendering process.

A few limitations on our system include the need of using a good quality depth map. Otherwise, noise could become present in the curves. Furthermore, when complex surfaces have to be depicted, it increases the chance of lines intersecting with each other which is an undesirable outcome. This can be prevented by the good use of segmentation for separating objects, thus avoiding big changes in depth between neighboring objects. We also tested the use of photographs as fake depth images provide promising results. Our work while constrained by the need of images abundant in 3D data, opens the possibilities for future projects. The ability to calculate artificial depth maps from photographs is the next logical step to make this approach applicable for arbitrary input images.

5.2 Future Work

Our methodology focused entirely on the use of depth maps to generate strokes that exaggerate object's surfaces, in order to provide stylization effects such as famous artists have done in the past. Our method provides results that do resemble hatching and engraving effects. We also wanted to test our approach by creating an artificial depth map, by taking away our depth map assumption. Just using the information provided by a photograph as input image, we want to create our own fake data. To do this we created our artificial depth map using our manual segmentation watershed algorithm. We manually separate important objects, afterwards, we apply a distance transform algorithm to each of these segmented regions, and we combine them into a integrated gray scale image. The resulting processed image uses the distance from edges and suggests a sense of volume. Once we get this artificial depth map, we apply our approach. To achieve a more faithful depth image, a shape from shading technique could potentially be very helpful.

Inspired by Lumo [24] and diffusion curves [39], we think that edge information can be used to interpolate between regions to make a 3D looking result. While this work seems to work better on cartoons as seen in Figure 5.2, the concept of generating an artificial depth map looks promising. Therefore, the implementation of a more robust method for creating depth maps using segmentation and distance transformation, complementing it with Poisson solver for more smoothed results could be an interesting direction to follow in the future.

Making a more automatic approach is the end goal for this stylization. Some previous approaches could very well be integrated to provide more up to date methods. In our thresholding component, instead of just applying a simple thresholding tool, the use of black and white approaches like Li and Mould's can significantly improve our results [32] [37]. Integrating this work with other methods such as Pedersen and Singh's maze generation method [41], Inglis and Kaplan's Op art [22,23] for creating similar stylizations using our mask-based approach. Introducing color in our curve rendering can also be a fine addition to these effects; Ostromoukhov explored this idea in his Facial Engraving work. Color is also a powerful way to communicate art and transmit emotion to viewers [6].

Finally, integrating our work with Kang et al's FDoG [26] can significantly improve the artistic value of our results, by the rendering of outlines which are also present on pen-and-ink illustrations.

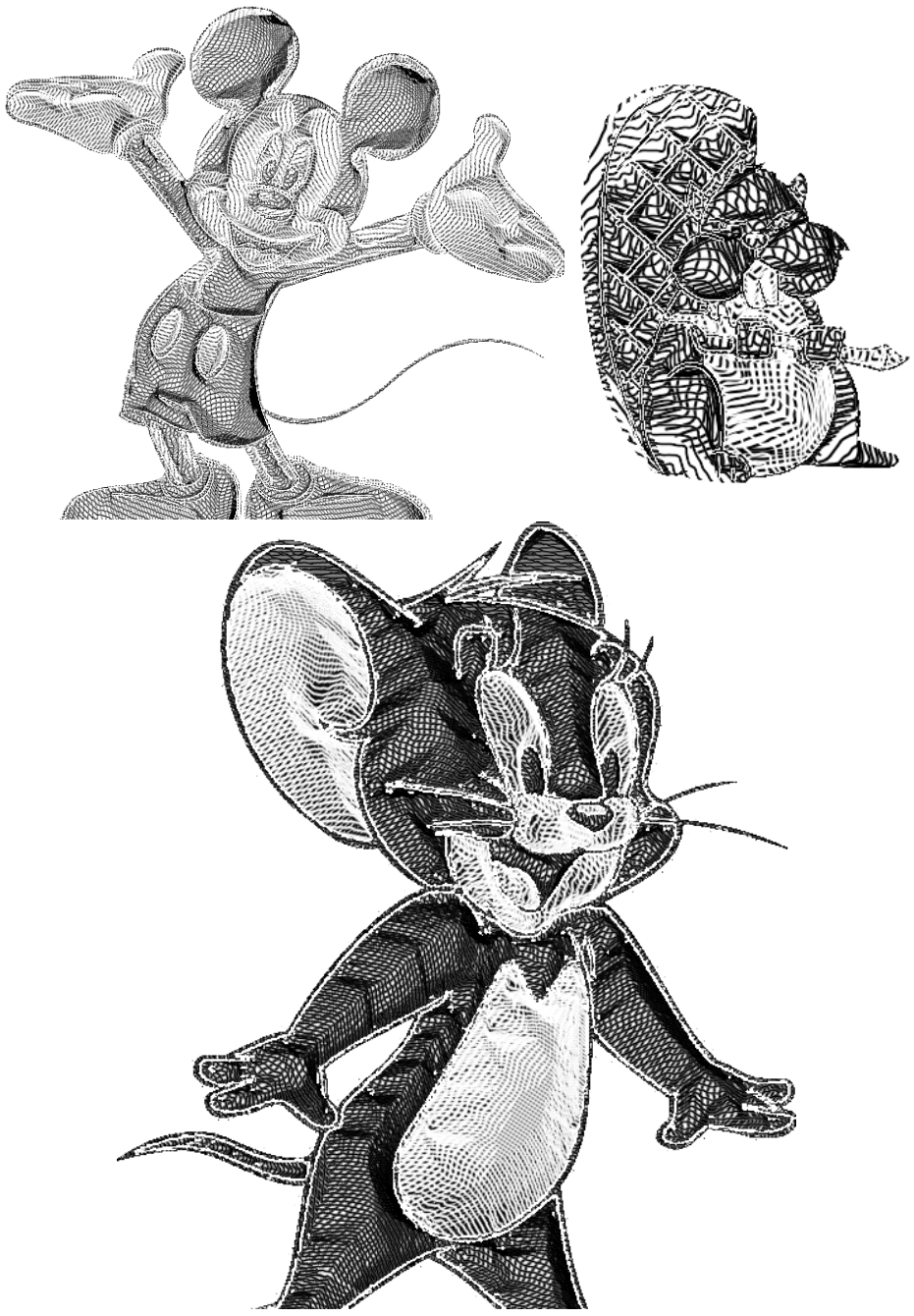

Figure 5.1: The interpolation between boundaries generates a sense of volume. The integration of a Poisson solver can improve the results. The distance transform calculation must be recalculated to generate smooth results.

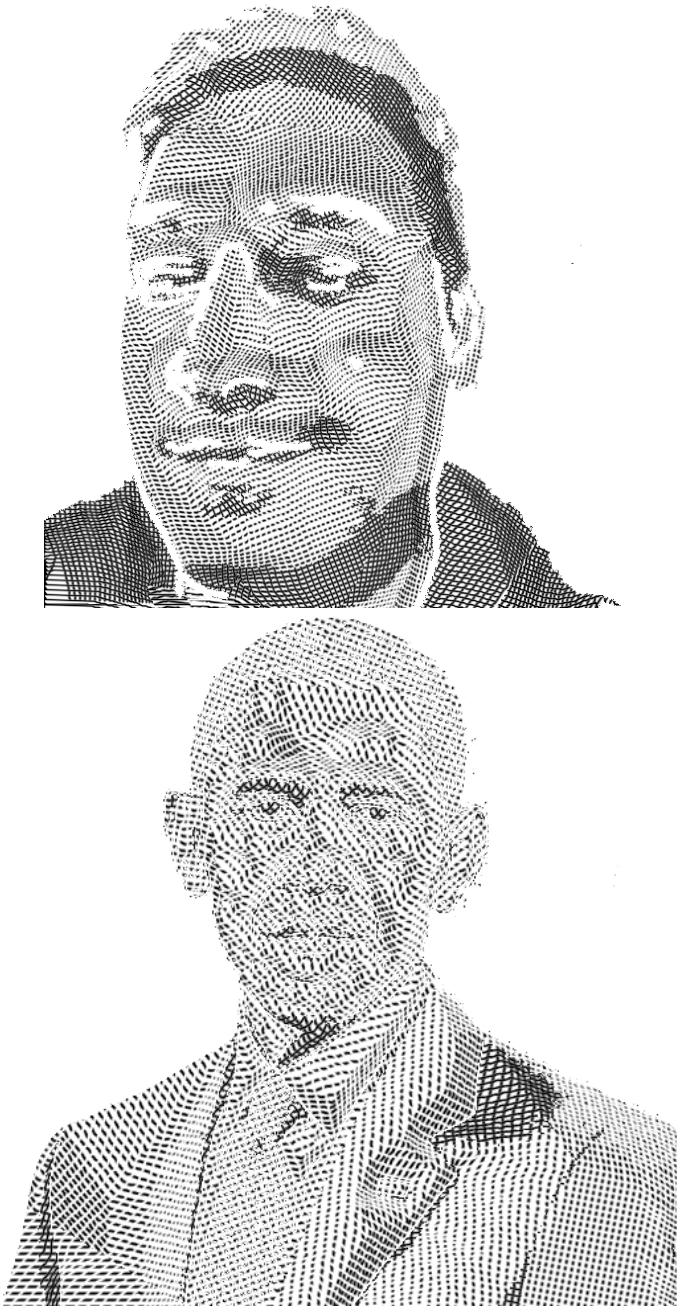

Figure 5.2: A good next step is to apply our approach to photographs.

List of References

[1] R Achanta, A Shaji, K Smith, A Lucchi, P Fua, and Sabine Süsstrunk. SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(11):2274–2282, 2011.

[2] Lamia Jaafar Belaid and Walid Mourou. Image segmentation: A watershed transformation algorithm. *Image Analysis and Stereology*, 28(2):93–102, 2009.

[3] S. Beucher and C. Lantuejoul. Use of Watersheds in Contour Detection, 1979.

[4] Hedlena Bezerra, Bruno Feijó, and Luiz Velho. An image-based shading pipeline for 2D animation. *Brazilian Symposium of Computer Graphic and Image Processing*, 2005:307–314, 2005.

[5] John Canny. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986.

[6] Daniel Cohen-Or, Olga Sorkine, Ran Gal, Tommer Leyvand, and Ying-Qing Xu. Color harmonization. *ACM Trans. Graph.*, 25(3):624–630, July 2006.

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

[8] Cassidy J. Curtis. Loose and sketchy animation. In *ACM SIGGRAPH 98 Electronic Art and Animation Catalog*, SIGGRAPH '98, pages 145–, New York, NY, USA, 1998. ACM.

[9] Doug DeCarlo, Adam Finkelstein, Szymon Rusinkiewicz, and Anthony Santella. Suggestive contours for conveying shape. *ACM Transactions on Graphics*, 22(3):848, 2003.

[10] Doug DeCarlo and Szymon Rusinkiewicz. Highlight lines for conveying shape. *Proceedings of the 5th international symposium on Non-photorealistic animation and rendering - NPAR '07*, 1(212):63, 2007.

[11] Doug DeCarlo and Anthony Santella. Stylization and abstraction of photographs. *ACM Transactions on Graphics*, 21(3):1–8, 2002.

[12] Lars Doyle and David Mould. Painted stained glass. In *Proceedings of the Joint Symposium on Computational Aesthetics and Sketch Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering*, Expresive '16, pages 1–10, Aire-la-Ville, Switzerland, Switzerland, 2016. Eurographics Association.

[13] Hui Fang and John C Hart. Textureshop: Texture Synthesis as a Photograph Editing Tool. *ACM Transactions on Graphics*, 23(3):354–359, 2004.

[14] Processing Foundation. Processing: curveVertex(). `https://processing.org/reference/curveVertex_.html`.

[15] Weidong Geng. *The Algorithms and Principles of Non-photorealistic Graphics [electronic resource] : Artistic Rendering and Cartoon Animation / by Weidong Geng.* 2010.

[16] George Davidson. *The Drawings of Gustave Dore: Illustrations to the Great Classics.* Metro Books, 2008.

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

[18] A L Guptill and S E Meyer. Rendering Parametric Surfaces in Pen and Ink. *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques SIGGRAPH 96*, 30(Annual Conference Series):469–476, 1997.

[19] Paul Haeberli. Paint by numbers: abstract image representations. *ACM SIGGRAPH Computer Graphics*, 24(4):207–214, 1990.

[20] Aaron Hertzmann. Non-Photorealistic Rendering and the science of art. *Proceedings of NPAR*, 1(212):147, 2010.

[21] Aaron Hertzmann and D Zorin. Illustrating Smooth Surfaces. *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, (Section 5):517 – 526, 2000.

[22] Tiffany C. Inglis, Stephen Inglis, and Craig S. Kaplan. Op Art rendering with lines and curves. *Computers and Graphics (Pergamon)*, 36(6):607–621, 2012.

[23] Tiffany C. Inglis and Craig S. Kaplan. Generating op art lines. *Proc. CAe*, 1:25–32, 2011.

[24] Scott F Johnston. Lumo: Illumination for Cel Animation. *Proceedings of the Second International Symposium on Non-Photorealistic Animation and Rendering (NPAR 2002, Annecy, France, June 3–5, 2002)*, pages 45–52, 2002.

[25] Robert D. Kalnins, Philip L. Davidson, Lee Markosian, and Adam Finkelstein. Coherent stylized silhouettes. *ACM Transactions on Graphics*, 22(3):856, 2003.

[26] H. Kang, Seungyong Lee, and C. K. Chui. Flow-based image abstraction. *IEEE Transactions on Visualization and Computer Graphics*, 15(1):62–76, 2009.

[27] Henry Kang, Seungyong Lee, and Charles K Chui. Coherent line drawing. *Proc. NPAR*, 1(212):43–50, 2007.

[28] Jan Eric Kyprianidis, John Collomosse, Tinghuai Wang, and Tobias Isenberg. State of the 'Art: A taxonomy of artistic stylization techniques for images and video. *IEEE Transactions on Visualization and Computer Graphics*, 19(5):866–885, 2013.

[29] Hua Li and David Mould. Contrast-aware halftoning. *Comput. Graph. Forum*, 29(2):273–280, 2010.

[30] Hua Li and David Mould. Artistic tessellations by growing curves. *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Non-Photorealistic Animation and Rendering - NPAR '11*, 1(212):125, 2011.

[31] Hua Li and David Mould. Structure-preserving stippling by priority-based error diffusion. *Proc. GI*, pages 127–134, 2011.

[32] Hua Li and David Mould. Contrast-Enhanced Black and White Images. *Computer Graphics Forum*, 34(7):319–328, 2015.

[33] Jiayu Li. Image Warping for a Painterly Effect. Master's thesis, Carleton University, 2015.

[34] Peter Litwinowicz. Processing Images and Video for an Impressionist Effect. *Proc. SIGGRAPH*, pages 407–414, 1997.

[35] Jorge Lopez-Moreno, Jorge Jimenez, Sunil Hadap, Erick Reinhard, Ken Anjyo, and Diego Gutierrez. Stylized depiction of images based on depth perception. *NPAR '10 Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*, 1(212):109–118, 2010.

[36] Lee Markosian, Michael a Kowalski, Daniel Goldstein, Samuel J Trychin, John F Hughes, and Lubomir D Bourdev. Real-time nonphotorealistic rendering. *Proceedings of the 24th annual conference on Computer graphics and interactive techniques SIGGRAPH 97*, 31(i):415–420, 1997.

[37] David Mould and Kevin Grant. Stylized black and white images from photographs. *Proc. NPAR*, 1(212):49–58, 2008.

[38] David Mould, Regan L. Mandryk, and Hua Li. Emotional response and visual attention to non-photorealistic images. *Computers & Graphics*, 36(6):658–672, 2012.

[39] Alexandrina Orzan, Adrien Bousseau, Holger Winnemöller, Pascal Barla, Joëlle Thollot, and David Salesin. Diffusion curves. *ACM Transactions on Graphics*, 27(3):1, 2008.

[40] Victor Ostromoukhov. Digital facial engraving. *Proc. SIGGRAPH*, pages 417–424, 1999.

[41] Hans Pedersen and Karan Singh. Organic labyrinths and mazes. *Non-Photorealistic Animation and Rendering: Proceedings of the 4th international symposium on Non-photorealistic animation and rendering; 05-07 June 2006*, pages 79–86, 2006.

[42] Ramesh Raskar, Kar-han Tan, Rogerio Feris, Jingyi Yu, and Matthew Turk. Non-photorealistic Camera : Depth Edge Detection and Stylized Rendering using Multi-Flash Imaging. *ACM Transactions on Graphics*, 23(3):679–688, 2006.

[43] Takafumi Saito and Tokiichiro Takahashi. Comprehensible rendering of 3-D shapes. *ACM SIGGRAPH Computer Graphics*, 24(4):197–206, 1990.

[44] Michael P Salisbury, Michael T Wong, John F Hughes, and David H Salesin. Orientable Textures for Image-Based Pen-and-Ink Illustration. *Proceedings of ACM SIGGRAPH 97 (Los Angeles, CA, August 3–8, 1997)*, pages 401–406, 1997.

[45] Rezwan Sayeed and Toby Howard. State of the Art Non-Photorealistic Rendering (NPR) Techniques. *Proceedings of Theory and Practice of Computer Graphics 2006*, pages 89–98, 2006.

[46] Adrian Secord. Weighted Voronoi stippling. *Proceedings of the second international symposium on Non-photorealistic animation and rendering - NPAR '02*, 1:37, 2002.

[47] Helen South. *The everything drawing book: from basic shapes to people and animals, step-by-step instructions to get you started*. Adams Media, Avon, MA, 2005.

[48] Lszl Szcsi, Marcell Szirnyi, and gota Kacs. Tonal Art Maps with Image Space Strokes. In Giovanni Pintore and Filippo Stanco, editors, *Smart Tools and Apps for Graphics - Eurographics Italian Chapter Conference*. The Eurographics Association, 2016.

[49] Chujia Wei. Coordinated Particle Tracing. Master's thesis, Carleton University, 2013.

[50] Chujia Wei and David Mould. Coordinated particle systems for image stylization. In *Graphics Interface 2014*, pages 225–233, Montreal, Quebec, Canada, May 2014. Canadian Human-Computer Communications Society.

[51] Georges Winkenbach and David H. Salesin. Computer-generated pen-and-ink illustration. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, pages 91–100, New York, NY, USA, 1994. ACM.

[52] Holger Winnemöller. XDoG. *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Non-Photorealistic Animation and Rendering - NPAR '11*, page 147, 2011.

[53] Holger Winnemöller, Sven C. Olsen, and Bruce Gooch. Real-time video abstraction. *ACM Trans. Graph.*, 25(3):1221–1226, July 2006.

[54] Jun Xie, Aaron Hertzmann, Wilmot Li, and Holger Winnemöller. PortraitSketch: Face Sketching Assistance for Novices. *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, pages 407–417, 2014.

[55] Ling Xu and David Mould. Magnetic Curves: Curvature-Controlled Aesthetic Curves Using Magnetic Fields. *Computational Aesthetics 2009: Eurographics Workshop on Computational Aesthetics in Graphics, Visualization and Imaging (Victoria, British Columbia, Canada, May 28–30, 2009)*, pages 1–8, 2009.

[56] Ruo Zhang, Ping-sing Tsai, James Edwin Cryer, and Mubarak Shah. Shape from Shading : A Survey 1 Introduction. *Review Literature And Arts Of The Americas*, 21(8):1–41, 1999.

[57] Qihui Zhu. Shape from Shading: Recognize the Mountains through a Global View. *Science*, 2:1839–1846, 2005.