THE UNIVERSITY OF CALGARY

Precise Ink Illustrations of Polygonal Models

by

Kevin Graham Foster

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

CALGARY, ALBERTA

MARCH, 2005

THE UNIVERSITY OF CALGARY

FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled “Precise Ink Illustrations of Polygonal Models” submitted by Kevin Graham Foster in partial fulfillment of the requirements for the degree of Master of Science.

Supervisor, Dr. Mario Costa Sousa, Department of Computer Science

---

Supervisor, Dr. Brian Wyvill, Department of Computer Science

Dr. Faramarz Samavati, Department of Computer Science

Dr. Joaquim Jorge, Departamento de Engenharia Informática, Instituto Superior Técnico

---

External Examiner, Dr. Gerald Hushlak, Department of Art

Date

Abstract

This research contributes two approaches for pen-and-ink stroke extraction and stylization from 3D polygonal meshes. The pen-and-ink medium is commonly used by artists to highlight shape and form of objects. The first approach presents a general solution to remove errors from silhouettes extracted from polygonal meshes in object space. This approach uses reverse-subdivision multiresolution to remove errors and perform automatic smoothing stylization on silhouette strokes. The second approach creates accurate, technical pen-and-ink styles using an edge-based approach. Ink is placed using edge-based shape measures generalized from geomorphology and controlled by a user-driven interface. The focus of this work is to provide a user with high-level interactive control of ink placement, while automatically generating individual strokes.

Acknowledgements

I would like to thank my supervisors, Dr. Brian Wyvill and Dr. Mario Costa Sousa for allowing me to pursue this interesting topic, for support and encouragement throughout my studies and for the great time in Granada. Also, thanks to Dr. Faramarz Samavati for improving my knowledge of multiresolution and geometric modeling and for extensive help with this thesis. Dr. Joaquim Jorge also provided a great deal of help with this write-up and my defence. Dr. Gerry Hushlak from the art department also provided a great deal of insightful feedback and help with this work.

My fellow graduate students have truly made the time spent completing my studies the best time of my life. Many thanks to my partner in NPR-crime, Meru Brunn, for extensive help over the past two years and for proofreading drafts of this document. Also, many thanks to Martin Fuhrer for his incredible plant meshes, to Ryan Schmidt for his impressive knowledge of compilers, optimization and processors, and to Tobias Isenberg for his impressive knowledge of NPR silhouette methods. Also, thanks to everyone else in the lab for making my experience memorable, especially Julia Taylor-Hell, Pauline Jepp, Anand Agarawala, Michael Boyle, Callum Galbraith and Gregor McEwan.

Thanks also to my friends at Busking For Smiles, Chinook Tae Kwon Do and Capoeira Aché Brasil Calgary. Furthermore, a special thanks to my roommates over the years, Matt Smith, Spencer Cox, Jennifer Savage, Crystal Gilhooly and Jeronimo “Momo” Delgado. Further thanks to Cameron May, Jamie Cullen, Joe Shea, Daniel Heffner, Julie Kunnathu and Meaghan Nolan for the music. May it never stop. Also, a massive super-special thanks to my friends that have been with me forever, Michael Flach, Mark Senicki and (well, it feels like forever) Karol Zakiewicz. Thanks also to my grandmothers, Eleanor and Fern,

for always being positive and supportive in all of my endeavors.

Finally, eternal gratitude, above everyone else on this list multiplied by at least ten, goes to my parents, Marjorie and Graham, who have always provided me with support, encouragement and patience. The proofreading and edits are very appreciated. Also, thanks for the food.

Table of Contents

Approval Page ii

Abstract iii

Acknowledgements iv

Table of Contents vi

1 Introduction 1
- 1.1 Pen-and-Ink Rendering . . . 3
  - 1.1.1 Contour (Silhouette) Illustration . . . 4
  - 1.1.2 Precise Pen-and-Ink Illustration . . . 5
- 1.2 Methodology . . . 7
- 1.3 Contributions . . . 11
- 1.4 Overview . . . 13

2 Background 15
- 2.1 Silhouette extraction . . . 15
  - 2.1.1 Definition of a Silhouette . . . 15
  - 2.1.2 Silhouette Extraction Algorithms . . . 17
  - 2.1.3 Silhouette Error Correction . . . 22
- 2.2 Precise ink-based illustration . . . 25

3 The Silhouette Stroke Extraction System 31
- 3.1 The Edge-Buffer . . . 32
  - 3.1.1 Initialization . . . 32
  - 3.1.2 Run Time Operation . . . 33
- 3.2 Multiresolution . . . 34
- 3.3 The Silhouette Stroke Extraction System . . . 35
  - 3.3.1 Edge-Buffer modifications . . . 36
  - 3.3.2 Local and Global Filters . . . 38
  - 3.3.3 MAR: Multiresolution Artifact Removal . . . 43
  - 3.3.4 Chain Size . . . 45
  - 3.3.5 Resolution-Independent Strokes: Smoothing/Coarsening . . . 46
- 3.4 Stylizing . . . 47
  - 3.4.1 Object Space HLR . . . 48
  - 3.4.2 Image Space HLR . . . 49

4 The Interior Stroke Extraction System **51**
4.1 System Overview . . . 52
4.2 Edge-Buffer Modifications . . . 52
4.3 Edge-Based Shape Measures . . . 54
4.3.1 Morphometric Variables with Triangular Meshes . . . 54
4.3.2 Using Morphometric Variables for Ink Placement . . . 56
4.4 Stylizing the Interior Strokes . . . 60
4.4.1 Thickness Adjustment . . . 60
4.4.2 Ink Marking Styles . . . 61
4.4.3 Ink Distribution Effects . . . 64
4.4.4 Rendering . . . 64

5 Results **68**
5.1 The Silhouette Stroke Extraction System . . . 68
5.1.1 Timing . . . 69
5.1.2 User Input . . . 74
5.1.3 Global Multiresolution VS. Local Multiresolution . . . 74
5.1.4 Cubic B-Spline Subdivision VS. Dyn-Levin Interpolation VS. Chaikin Subdivision . . . 78
5.1.5 Comparison to Previous Work . . . 82
5.2 Interior Stroke Extraction System . . . 84
5.2.1 Input Mesh . . . 85
5.2.2 Level-of-Detail . . . 87
5.2.3 Stroke Spacing and Direction . . . 87

6 Conclusions and Future Work **94**
6.1 The Silhouette Stroke Extraction System . . . 95
6.1.1 Limitations and Future Work . . . 97
6.2 The Interior Stroke Extraction System . . . 97
6.2.1 Limitations and Future Work . . . 98

Bibliography **101**

List of Tables

2.1 A classification of polygonal silhouette extraction methods. 29
2.2 A classification of polygonal silhouette error-correction methods. From left to right, the columns describe the following: **Approach** provides the reference for the approach; **Error-Removal Method** describes the combination of image-space and object-space techniques the method uses; **Correction Source** displays whether the method corrects polygon edges or creates new sub-polygon errors; **Precision** tells whether the method returns pixel (image-space) results or edges from the geometry of the polygonal mesh and **Silhouettes Corrected** describes whether all silhouettes or only visible silhouettes are corrected. 30

5.1 Running times for error-correction for the meshes illustrated in this research. These results are plotted in Figure 5.1. 69
5.2 Average times (in seconds) for pre-processing and rendering the meshes presented in this research. These results are plotted in Figure 5.10. 85

List of Figures

1.1 A crosswalk street sign. 2
1.2 *Left:* "Sea urchin mouthparts", by Emily S. Damstra. *Right:* a house in a photorealistic and a non-photorealistic style. 2
1.3 Several illustrations consisting completely of silhouette strokes. 6
1.4 Real precise drawings: Clockwise from top-left, portion of American Portrait by William Henson, two archaeological artifacts, a shell and a bone all by Emily S. Damastra. 8
1.5 Real precise drawing: Adele Fatima, by Humberto Costa Sousa. 9

2.1 A silhouette drawing (using the artistic definition of a silhouette). 17
2.2 A silhouette point and a silhouette edge. The red arrow denotes the view direction V. *Left:* a silhouette point for a smooth continuous surface is defined as any point P whose normal is 90 degrees from the view vector. *Middle:* a silhouette edge for a polygonal mesh is defined as any edge which shares a front-facing (FF) and a back-facing (BF) polygon. 17
2.3 The silhouette of an ape mesh with highlighted errors and artifacts. . . . 23
2.4 Four images showing various silhouettes and underlying mesh that generated them. The silhouettes are presented at a perturbed view to provide a better understanding of the cause of the errors. Shaded polygons were back-facing when the silhouettes were extracted. 24

3.1 A visual description of the Edge-Buffer: (*a*) a simple polygonal object, with vertex numbers listed; (*b*) an example of how edge adjacency information is stored in the Edge-Buffer [BS00b] for this polygonal object; (*c*) the data stored in each node in the linked lists. 33
3.2 The *Silhouette Stroke Extraction System* pipeline. 36
3.3 The two pass process used to chain silhouette edges. *Left:* An example polygon mesh with vertex and face numbers and its associated Edge-Buffer structure. *Top-right:* The first step in silhouette chaining. In this example, two chains are created, shown in red and blue on the polygonal mesh. These are extracted individually, following nodes of the Edge-Buffer until no further connections can be created. *Bottom-right:* The second step for chaining. The ends of each chain are examined for any matching vertex indices. Matches are joined with preference for creating looping chains. 37

3.4 The bands of the filters for the cubic B-Spline case (the A, B, P and Q diagrams represent all non-zero entities of a row for the A and B matrices and of a column for the P and Q matrices). The gray circles show the center entity. 40
3.5 The bands of the Chaikin filters. 41
3.6 The bands of the Dyn-Levin interpolation filters. 41
3.7 The MAR approach uses multiresolution filters to decompose and reconstruct silhouette chains without errors. Here is an example session of the MAR approach for an **ape** mesh with 7434 faces. Proceeding from left to right, the silhouette chains are first decomposed twice from level C^0 to C^{-2}. Then, the system reconstructs to level C^0 using scaled details (here, $e = 0.3$). The effect of this process is the removal of errors. Finally, the MAR approach reconstructs past the original level of detail to C^1 to provide a smoother more stylized silhouette. 44
3.8 Strokes generated by the MAR system for the input in Figure 2.4. 45
3.9 The effect of removing details. In this image, silhouettes from a **beaver** model are shown for two angles, head-on in the top row and from behind with mesh information on the bottom row. One level of decomposition and reconstruction is used here with global cubic B-Spline filters. From left to right, $e = 1.0$, $e = 0.66$, $e = 0.33$ and $e = 0.0$. 46
3.10 The rightmost two **ape** images from Figure 3.7. Here the system performs reconstruction on corrected strokes above the original level of detail. This results in a smoothing effect, making the geometry of the underlying mesh less apparent. 47
3.11 *Left:* raw silhouette edges, each with a unique colour. *Middle:* results of object-space HLR on processed strokes. *Right:* results of image-space HLR on the processed strokes. 50

4.1 Discrete diagram for a 9-point finite difference scheme for numerical differentiation (*right*), constructed from pair of mesh faces (F_1, F_2), shared by edge *ab* (*left*). 55
4.2 Shaded relief images of *Venus* model (5,584 faces) with (1) lighter yellow referring to greater slope steepness, and with (2) darker blue and red referring to regions of greater convexity and concavity, respectively. . . . 57
4.3 Feature edges displaying thresholded shape measures within (min,max) range. 58
4.4 **Stanford bunny** (built from laser scans), with slope steepness $GA(0.81, 1.0)$ and positive mean curvature $+H(0.57, 1.0)$, rendered with filled (*left*) and serrated (*right*) pen marks ($res = 15$, see Section 4.4.2). *Model source: Stanford University Computer Graphics Lab.* 62

4.5 *Top-Left*: serrated marks placed at mesh edges. *Top-Right*: A single mark *s*, defined by vertices(p,q), can vary in location and length within (t_1,t_2,t_1',t_2'). *Bottom*: four examples of possible serrated marks in the (t_1,t_2,t_1',t_2') area. 63
4.6 The number of (p,q) strokes in each (a,a',b',b) area is scaled based on the length of edge *ab*. 63
4.7 A visual effect of stroke thickness distribution: **Hand** with normal stroke thickness distribution (*left*) and with increased thickness of strokes closer to the viewer (*right*). 65
4.8 Another visual effect of stroke thickness distribution: **Mount St. Helens** with normal stroke thickness distribution (*left*) and with strokes farther from the viewer having their thickness scaled down (*right*). *Model provided by www.artcam.com*. 65
4.9 **Female head** (built from laser scans); Top-row, left*:* starting with the wire-frame mesh, filled strokes are placed at slope steepness $GA(0.7,1.0)$, and positive mean curvature $+H(0.47,1.0)$. Notice how various anatomic and hair features are revealed. Next, filled with serrated marks are used, resulting in a soft stippling effect. Bottom images*:* Side views of model rendered with filled strokes using the setup used for top-right image. *Model source: Cyberware.* 67

5.1 A plot of the data in Table 5.1. The solid line represents the average execution time for the local multiresolution approach. The dotted line represents the average execution time for the global approach. The expense of the global approach is always higher than the local approach, but appears to be highly variable with respect to the length and number of silhouettes from the mesh. 70
5.2 *Top*: Original silhouette from a model of the **Toutalis asteroid**. *Bottom*: Results after processing with the MAR system. In this session, two levels of global cubic B-Spline decomposition and reconstruction with $e = 0.1$ were used. Red shaded polygons are back-facing and blue polygons are front-facing. 71
5.3 *Top*: Original silhouettes from an **inner-ear** model. *Bottom*: the results after processing two levels of decomposition and reconstruction with local cubic B-Spline filters and $e = 0.0$. Stroke thickness varies in this image as a function of depth. 72

5.4 *Top-Left*: raw silhouette from a coarse **ox** mesh. *Top-Right*: processed strokes with global cubic B-Spline filters. *Bottom row*: alternate views of the silhouette with the original strokes (*left*), processed strokes with global cubic B-Spline filters (*middle*) and global Chaikin filters (*right*). In both cases, two levels of decomposition and reconstruction are used with $e = 0.35$. This is an example of the system producing poor accurate output: the corrected strokes do not adhere well to the original mesh due to the low resolution of the initial silhouettes. 73
5.5 A detailed **foot** mesh. Removing silhouette errors on large meshes is more important when zooming in on the mesh. Errors are circled for three enlarged areas. These images use two levels of decomposition and reconstruction, $e = 0.2$ and local cubic B-Spline filters. 75
5.6 *Top*: Raw silhouettes extracted from a **cat** mesh. *Bottom-left*: local B-Spline subdivision with two steps of decomposition and reconstruction 30% details included. *Bottom-right*: global B-Spline with the same settings. 76
5.7 *Left*: Silhouette from a **head** mesh with several large errors at the temples and six results of running, from left to right, B-Spline, Dyn-Levin and Chaikin filters on the silhouettes using one level of decomposition and reconstruction with 20% details included. The top row uses local filters and the bottom row uses global filters. 79
5.8 An analysis of the effects of various filters viewed from an angle alternate to that used to extract silhouettes from the **Kleopatra asteroid** model. *Top*: The raw silhouettes. Blue edges are front-facing and red-edges are back facing. *Left-column, from top to bottom*: Local B-Spline, Dyn-Levin and Chaikin approaches. *Right-column*: Global filters in the same order. In all images, $e = 0.0$ is used with two levels of decomposition and reconstruction. 80
5.9 Silhouettes extracted from a range scan mesh of a skull. *Left:* with Hidden Line Removal (HLR). *Middle:* without HLR. *Right:* a side view of the silhouette edges extracted. 83
5.10 A plot of the data in Table 5.2. The dotted line represents the average times for rendering. The solid line represents the average pre-processing time required. 86

5.11 Results of the *Interior Stroke Extraction System* for medical illustration of a **hip** model, built from laser scans: (*a*) first, silhouettes are rendered with varying thickness as a function of curvature; (*b*) concave formations are revealed by placing filled strokes in locations with negative mean curvature $-H(-0.459,-0.001)$; (*c*) creases are delineated with $D(155,170)$, and convex formations are revealed by placing filled strokes in locations with positive mean curvature $+H(0.56,1.0)$; (*d*) finally, the view-depth effect is applied to improve the depth perception. *Model source: Cyberware*. . 87
5.12 Results for an archeological illustration of primate model **Homunculus Patagonicus** built from laser scans; (*a*) evenly distributed filled strokes; (*b*) replacing filled with serrated marks and applying the view-depth effect. (*c*) placing filled marks to reveal slope steepness and serrated marks for slope aspect, keeping view-depth effect on. *Model source: Dr. A.L. Rosenberger, Smithsonian Institution BioVisualization Lab*. 88
5.13 A **mask** of artist Richard Jakopic (built from range images); *Left*: Covering the model with filled strokes on locations of negative mean curvature -H(-1.0, -0.01) and positive mean curvature +H(0.001, 1.0) and slope aspect A0(0.9, 1.0). Notice how facial features are revealed. *Model provided by Danijel Skocaj, University of Ljubljana Computer Vision Lab* [Sol00]. 89
5.14 The "**killeroo**" model with filled strokes placed by slope steepness, aspect and positive mean curvature. Model provided by *headus.com.au* 89
5.15 A **gargoyle** built from range images; *Left*: Stroke placed in areas of high steepness. *Right*: Placing strokes for all other shape measures, with emphasis on slope aspect. Notice how the directional variation of strokes reveal the curved shapes of the statue. *Model provided by Rich Pito, University of Pennsylvania GRASP Lab*. 90
5.16 An **Igea artifact** built from laser scans; *Left*: placing filled strokes with slope steepness GA(0.81, 1.0) and positive mean curvature H(0.57, 1.0). *Right*: placing serrated strokes with three marks per edge and with view depth effect, revealing shape measures of slope steepness GA(0.73, 1.0) and negative mean curvature -H(-0.5,-0.48). Notice that hair and anatomic features are clearly revealed. This has special significance for this model given that the original artifact has various degrees of erosion. *Model provided by Cyberware*. 91

5.17 **Archaeological objects** built from laser scans. *Top-left*: a large oval hammerstone with filled strokes revealing slope steepness and locations with negative mean curvature. *Top-right*: a broken preformed adze rendered with filled strokes for slope aspect; Notice the variations of stroke directions, revealing the irregularity of the shape structure. *Bottom-left*: another adze with a different stroke style. *Bottom-right*: a preformed adze with filled strokes for positive and negative mean curvature and the effect of increasing stroke thickness as they get closer to the viewer. Models provided by Dr. Jeff Clark, North Dakota State University Archaeology Technologies Lab. 92
5.18 The Interior Stroke Extraction System can produce poor results if the input mesh is coarse. In the left two images in this figure, individual edges are clearly visible from the ink producing a poor pen-and-ink effect. In the right image, input parameters have been carefully adjusted to minimize this effect. 93
5.19 The Interior Stroke Extraction System can produce unnatural effects if the input mesh consists of regularly-spaced triangles, such as the grid layout displayed here for the Mount St. Helens model. 93

6.1 Comparing real pen-and-ink silhouette rendering to the results of the *Silhouette Stroke Extraction System*. *Top-row:* a real illustration of a seagull and images of a similar level of detail generated with the system. *Bottom-row:* a real illustration of a plant (*left*), and images of several plants generated with the system. *Plant meshes courtesy Martin Fuhrer*. 96
6.2 Comparing real detailed pen-and-ink rendering to the results of the Interior Stroke Extraction System. *Top-row:* Real drawings of various subjects. *Bottom-row:* system results for similar subects. 100

Chapter 1

Introduction

From its inception over 35 years ago, research in computer graphics has been dominated by the development of tools, rendering methods and hardware to create realistic images. As demonstrated by many recent movies, physical simulations and video games, the state of the art has advanced so significantly that images indistinguishable from reality can now be rendered on a computer. Furthermore, hardware has advanced to a state where, for certain techniques, this rendering can be achieved in real-time.

Although a photorealistic style has many applications, it is not always the best method to represent information. Indeed, many situations call for a non-photorealistic interpretation of an object. To demonstrate this, consider the cross-walk sign in Figure 1.1. Pedestrians do not need information from the image of the person crossing such as the colour of that person's clothes or the time of day when the image was created as a photorealistic picture would convey. The only information that pedestrians need to know is where they are allowed to cross the street. The non-photorealistic interpretation of this successfully delivers this information by focussing the viewer's attention on the important details without confusing them with extraneous data. Other examples of the artist's focussing on particular information are visible in Figure 1.2. In the case of the sea-urchin, focus is placed on the structures that make up its mouth-parts using contour lines and colour. The house image provides both a photorealistic and a non-photorealistic interpretation. The non-photorealistic interpretation highlights and labels the different structures of the house in a drafting style that would be desirable for those constructing the house.

Figure 1.1: A crosswalk street sign.

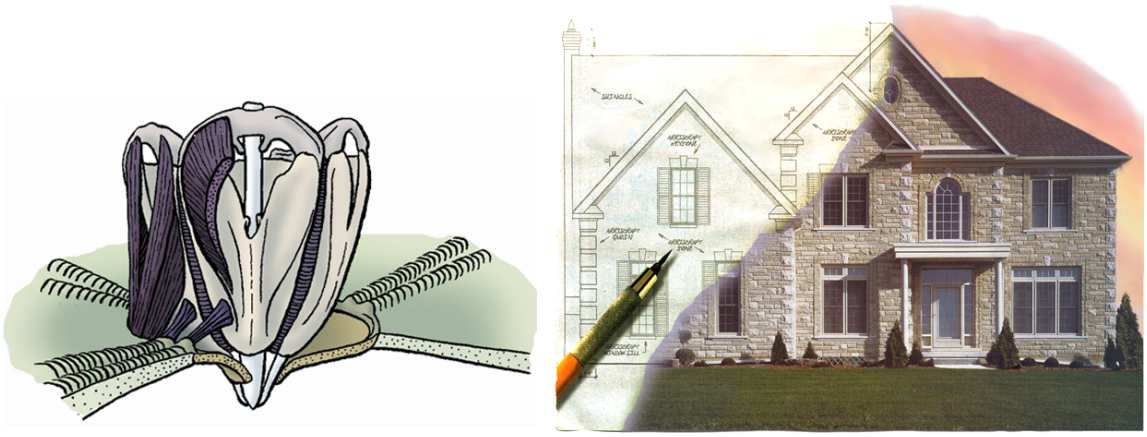


Figure 1.2: *Left:* "Sea urchin mouthparts", by Emily S. Damstra. *Right:* a house in a photorealistic and a non-photorealistic style.

Non-photorealistic images can be found in magazines, medical textbooks (Figure 1.2, left) and technical manuals (natural science and technical illustration), hospitals (medical imagery), building and electrical drafting (Figure 1.2, right), animation, comic books and many other locations. This widespread use can be explained for several major reasons, including: (1) non-photorealistic images can focus a viewer's attention on specific detail that the artist wishes to convey; (2) they can convey extra information which is impossible to include in photorealism and (3) they can add a special expression or feeling to the image.

Non-photorealistic rendering (commonly referred to as NPR) is a young research field in computer graphics that aims to provide techniques to help create images in expressive and interpretive styles for use in the situations described above. Although it is strange to define a research field by what it is not, the term non-photorealistic rendering has endured. Many techniques have already been presented that simulate mediums such as painting, pen-and-ink and pencil drawing from various types of data, such as 3D surfaces, point clouds, 3D volumes or 2D images. Some of these techniques are user-driven and some are automatic, using a physically-based system or an ad-hoc approximation. Green et al. [Gre99] provide a system to classify each NPR technique with regard to its objectives:

- **Natural media emulation:** intended to mimic one distinctive artistic medium.
- **Image enhancement:** aim to increase the visual qualities of the final image by application of effects.
- **Artistic expression:** aim to give the user the greatest degree of control over image creation.
- **Animation:** focus on producing animated imagery.

This research falls into two of these categories: natural media emulation and artistic expression. This is because it covers techniques that emulate pen-and-ink illustration for 3D surfaces and allows for a certain degree of artistic expression.

1.1 Pen-and-Ink Rendering

Pen-and-ink as a medium is used for both interpretive and expressive rendering [Sim92, Hod89]. Expressive rendering is characterized by loose or tight pen strokes to convey a

special feeling in an image. Interpretive rendering focusses on conveying structure and form of objects, highlighting certain detail and minimizing the possibility for misinterpretation. Whatever the style, these pen-and-ink illustrations consist solely of black ink marks on white paper. No tone or colour is used in these images. Despite this simplicity, pen-and-ink has several appealing properties. Pen strokes can represent virtually any shape if used properly [Sim92, Hod89]. This makes them ideal for printing and harmonizing with text due to their scale and use of the same ink [Sal97]. Also, pen-and-ink images handle duplication and degradation much better than images created with traditional halftoning processes. Pen-and-ink images can be found in many locations, including manuals, diagrams, animations, portraits, medical illustration and archaeological illustration.

There are two main challenges facing pen-and-ink illustrators. These are (1) representing tri-dimensionality in drawings, given that strokes are essentially 2D marks placed in a plane, and (2) simultaneously addressing tone and texture with only black ink strokes. To meet these challenges, pen-and-ink artists can use a vast array of styles, from loose, expressive illustrations to very precise pen-and-ink drawings. Furthermore, various pens (quill, flexible-nib and technical pens) and mark making techniques can be used to place ink. To compliment this style, images can incorporate various numbers of strokes, from very sparse applications to detailed applications incorporating thousands of marks.

This research deals with two styles, detailed precise pen-and-ink and contour (silhouette) pen-and-ink (Figures 1.3, 1.4, 1.5), each of which will now be introduced.

1.1.1 Contour (Silhouette) Illustration

A contour (silhouette) drawing only shows the outline of the subject, and does not use any interior strokes to reveal volume or mass (Figure 1.3). Artists place a great deal of attention

on illustrating contours and use them for many applications such as in cartoons, technical illustrations, architectural design and medical textbooks. A general principle in drawing states that a accurate set of contours highlighting an outline provides good perception of mass [Cra00]. This principal is supported by studies in perceptual rendering which reveal that a few simple lines defining the silhouette (or contour) of an object often suffice to determine its 3D surface [CKvD96]. Note that in NPR, contour illustrations are referred to as silhouette illustrations (Section 2.1.1). Silhouettes also convey important cues to distinguish between different objects and for object-to-ground recognition [IFH+03]. Artists illustrate silhouettes with two processes [Cra00, Hod89, Whi94]:

- by emphasizing the placement of the subject's outline outside the silhouette boundary of its form rather than within it.
- by carefully controlling the various characteristics or qualities of the line, in particular the suggestion of movement which is achieved by drawing long line segments with various degrees of linear weight and emphasis.

These processes can be seen in Figure 1.3. The weight and emphasis variation depends on the subject matter and on the information that the illustrator wants to present. For example, observe how the lines on the engine (Figure 1.3, bottom-right) vary with distance. In the seagull image (Figure 1.3, top-left) line width variations are also used to imply shadow.

1.1.2 Precise Pen-and-Ink Illustration

The primary purpose of precise pen-and-ink drawings is to make shape characteristics of the subject visible, to lessen the possibility of misinterpretation and to please the eye in terms of artistic handling of the subject (Figures 1.4 and 1.5). Traditionally, artists

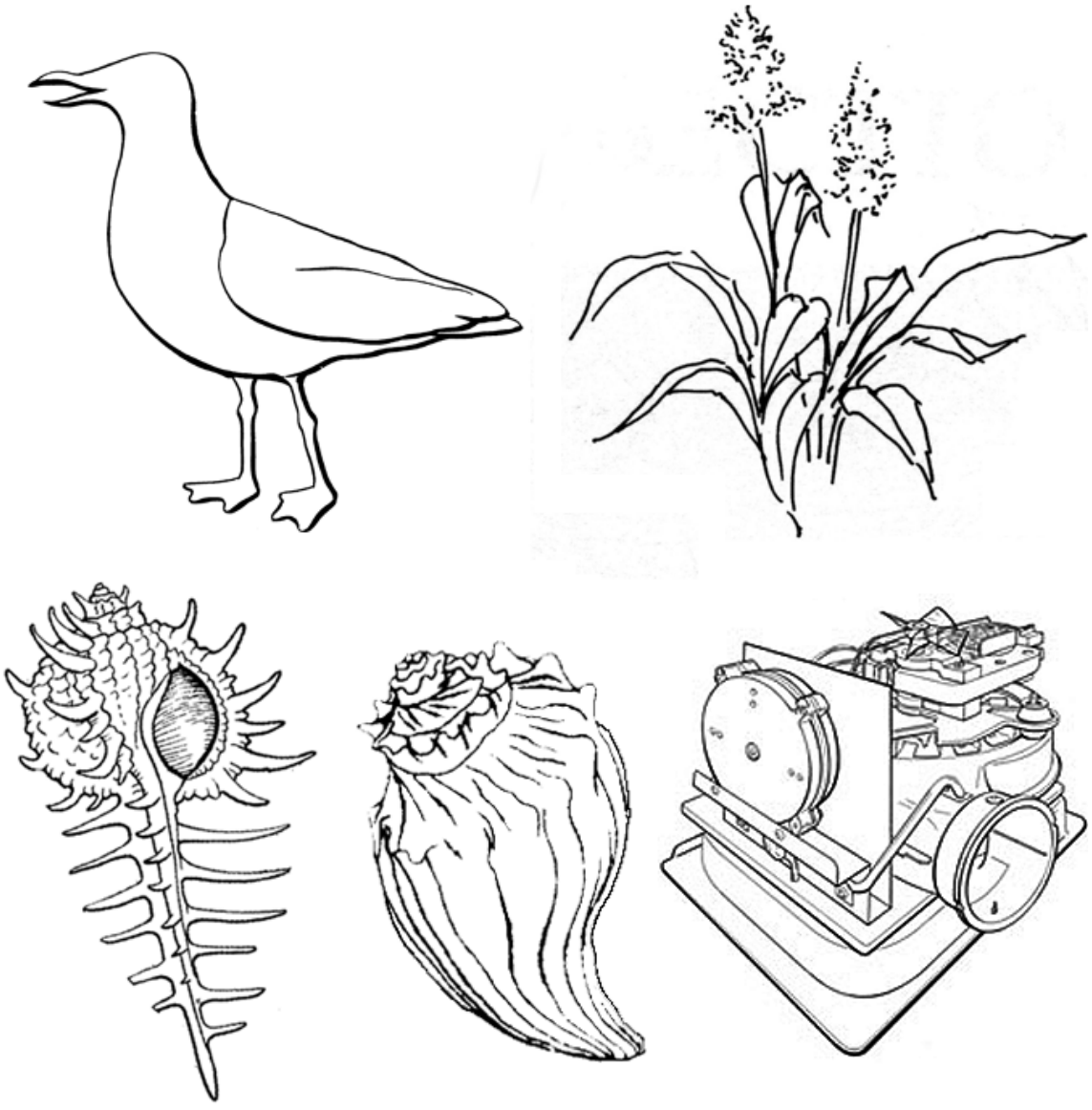


Figure 1.3: Several illustrations consisting completely of silhouette strokes.

follow three main steps to create these images: (1) shape characteristics are accurately identified and measured (i.e. folding regions, surface areas, volumes and curvatures); (2) regions related to the shape measures are lightly outlined using pencils and (3) these regions are filled with pen strokes, with a gesture that conveys a carefully constructed look [Sim92, Hod89, Raw87]. This process can often be quite time-consuming as precise pen-and-ink images can take hours or days for a highly-trained artist to complete.

As an individual primitive, individual interior strokes are either lines or dots, black, narrow, and consistent in thickness. In clusters, they create a cumulative effect resulting in tone values which help properly reveal the subject's shape characteristics. To control the cumulative effects of these strokes, artists:

- increase stroke width where surface curvature is high and at junctions and creases [Hod89, Raw87].
- carefully adjust stroke spacing and direction.

Observe the cumulative effects of grouping various types of short strokes in Figures 1.4 and 1.5. Many of these images incorporate groups of carefully placed points, a style called stippling. Also note short directional strokes in *American Portrait* (Figure 1.4, top-left) and in *Adele Fatima* (Figure 1.5).

1.2 Methodology

Two techniques are presented in this work—one specifically designed to create pen-and-ink silhouettes and the other to create precise pen-and-ink illustrations. The methodology in examining each technique will now be discussed.

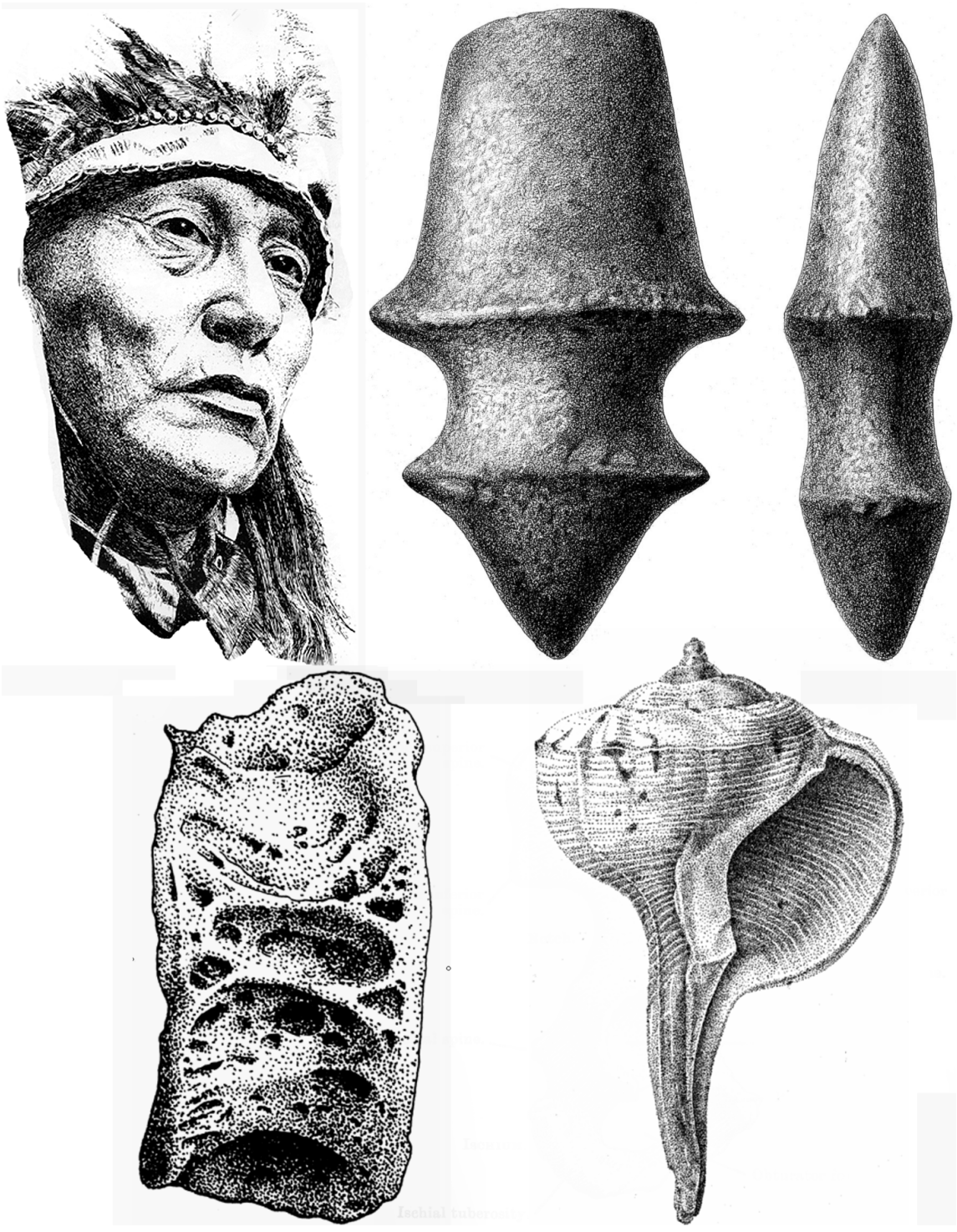


Figure 1.4: Real precise drawings: Clockwise from top-left, portion of American Portrait by William Henson, two archaeological artifacts, a shell and a bone all by Emily S. Damastra.

Figure 1.5: Real precise drawing: Adele Fatima, by Humberto Costa Sousa.

Various techniques have already been explored for silhouette extraction and stylization from polygonal meshes (Section 2.1). Despite this, a number of topics are still being developed and expanded upon. Specifically, current silhouette extraction methods that allow detailed stylization are error-prone (Section 2.1.3). Although some solutions to solve these errors have been explored, none of these approaches perform error-correction and allow for stroke stylization independent of the polygonal mesh. After analyzing various methods to accomplish this, an approach based on reversing subdivision was identified that might produce positive results. I implemented a system using this technique and found that it was successful at removing errors and providing a smoothing stylization for the stroke. Then, I added a stroke stylization module using traditional NPR techniques to add stroke qualities and to perform hidden line removal.

For precise pen-and-ink stroke extraction and stylization, an analysis of current approaches for polygonal meshes revealed that most methods use vertex-based methods to evaluate shape measures to extract long or short strokes across the surface. Some of these systems use stroke generation techniques based on illumination and surface reflectance to convey convincing pen-and-ink effects. No research has used an *edge-based* approach—an approach that evaluates specific measures for every edge in the input polygonal mesh and uses this data to create a stroke for each of these edges. To place these strokes, I selected shape measures that simulate techniques that artists use to apply ink. Furthermore, I stylize strokes with two styles, serrated and ink-filled silhouette, converted from a silhouette rendering system. I found that the local serrated edge and ink-filled styles globally produce realistic technical pen-and-ink styles involving stippling and short marks. With minor input, I created images fitting the accurate natural-science style described in Section 1.1.

For evaluation of both approaches, I checked execution times for efficiency, analyzed stroke quality and compared results generated with my approaches to images drawn by artists. Furthermore, I used a large variety of meshes created from various sources (modelling and range-scanning systems). Results from these tests are provided in Chapters 5 and 6.

1.3 Contributions

This research develops methods that create accurate silhouette and precise pen-and-ink images for 3D polygonal meshes. Primary goals this research include experimentation with shape measures to create surface revealing strokes and with multiresolution to stylize and remove errors from silhouettes. With this in mind, the approaches presented have been built for efficiency and stroke quality. The present work provides contributions in the following areas:

Concepts

- *To recognize multiresolution as a tool for polygonal silhouette error-removal and creation of resolution-independent strokes.* While multiresolution techniques have been successfully implemented for stroke stylization [FS94], no research has examined other possible applications to stroke-processing. My work demonstrates that multiresolution can remove silhouette errors and as an ideal tool to create resolution-independent strokes.

- *To recognize use of edges from a polygonal model as an efficient and accurate approach to directly create interior strokes and to recognize morphometric shape-*

measures as good tools to reveal these strokes. Most research for automatic interior strokes examines methods to procedurally generate strokes using vertex-based information. This research demonstrates that edge-based shape measures and stroke extraction using only edges is both efficient and can simulate different stroke styles.

Representations

- *An improved Edge-Buffer [BS00b] data-structure to extract silhouettes and shape measures efficiently and organize them in a simple manner.* This research improves on the Edge-Buffer, which uses a hash table representation of a polygonal mesh to extract silhouette edges. The improved data-structure can chain silhouettes and stores shape measures which are used to create precise pen-and-ink strokes and .

- *Resolution-independent silhouette chains*. These are strokes which can be set to any level-of-detail, including levels higher than the original data. Coarse strokes are useful to approximate detailed silhouettes from large polygonal meshes. In contrast, detailed strokes are desirable when an improvement of quality and smoothness is required, so that individual mesh edges are not visible in the silhouette. This is most useful for rendering strokes from low-resolution meshes and for rendering strokes from high-resolution meshes when strokes are viewed closely.

Algorithms

- *A new technique for removing silhouette errors and to provide resolution-independent silhouettes.* Object-space silhouette extraction is desirable because it allows for control of stroke stylization, extracts silhouettes at a geometric level instead of a pixel

level and can also extract hidden silhouettes from a surface. A drawback to object-space extraction is that it can suffer from errors and artifacts in the silhouette. Correcting these errors has been previously explored [IHS02, HZ00, NM00, CJTF98]. These solutions either correct the raw silhouettes on an error by error basis, or attempt to create better (sub-polygon) silhouettes that do not suffer from errors. This research presents a method that uses a general solution to remove silhouette errors without requiring a list of individual errors and the corresponding corrections for them. This method is closer to the sub-polygon approach because it modifies the edges to make them closer to the ideal silhouette. Furthermore, it is resolution-independent and offers user-controlled stylization independent of the input mesh.

- *A system to reveal shape features of polygonal meshes with procedurally generated interior pen strokes*. This uses an edge-based approach and modifications to the Edge-Buffer. Furthermore, ink is applied using measures from geomorphology on edges taken directly from the mesh instead of procedurally extracting new strokes from the surface. This method yields good rendering rates, provides an efficient scheme for shape measures calculation and delivers visual quality resembling certain styles of traditional precise pen stroke drawings.

1.4 Overview

Methods have been developed to create silhouette and interior strokes. Key goals include good rendering rates, efficient shape-measures calculation and visual quality resembling specific styles of pen-and-ink illustration. These methods are presented in two parts:

1. the *Silhouette Stroke Extraction System*: extracts silhouettes, then corrects errors in

them, performs hidden line removal and renders stroke qualities

2. the *Interior Stroke Extraction System*: extracts and stylizes precise pen-and-ink strokes on the interior of the surface

Both modules use a modified Edge-Buffer [BS00b] data structure. In the next chapter, relevant research to this project is reviewed. In Chapter 3, the *Silhouette Stroke Extraction System*, including error correction and stylization is presented. In Chapter 4, the *Interior Stroke Extraction System* is presented. Results for these systems are presented in Chapter 5. Finally Chapter 6 discusses these results, offering conclusions and future work.

Chapter 2

Background

In this chapter, an overview of the research related to this research is presented. First, research focussing on silhouette extraction, including an explanation of silhouette errors and error-correction techniques is reviewed (Section 2.1). Then, a summary of research focussing on *precise* pen-and-ink based illustration with interior mark extraction is presented (Section 2.2).

2.1 Silhouette extraction

There is a large body of work covering silhouette extraction and stylization. Efficient silhouette extraction is important because silhouettes are view-dependent and need to be reevaluated for each frame in an animation or after each viewing adjustment. These methods work in *object-space* (a 3D geometry-based approach), *image-space* (a 2D pixel-based approach) or use a combination of both. Before presenting a review of silhouette extraction methods, the definition of a silhouette as used by researchers in NPR is provided.

2.1.1 Definition of a Silhouette

The traditional, artistic definition of a silhouette is the outline of an object, or the boundary surrounding an object's shadow when the view direction is the same as the lighting direction (Figure 2.1). The NPR definition of a silhouette for a 3D surface differs from this slightly. In NPR, the silhouette is defined as the curve on a surface where the normal

direction[1] at every point on the curve is ninety degrees from the view direction (the direction from the eye to the point on the curve). This means that an object can have many silhouettes, that they can be inside the shadow boundary described above and that some of these can be occluded, depending on the dimensions and orientation of the object as displayed to the viewer.

Mathematically, the silhouette for a smooth surface is defined as follows: a point X on a surface with normal $\mathbf{N}_X$ is on the silhouette for an eye position E if the angle between $\mathbf{N}_X$ and $(\overline{X-E})$ is 90 degrees (Figure 2.2, left). This definition includes interior silhouettes as well as the object's outline. Unfortunately, this definition does not hold for polygonal meshes because surface normals for polygonal meshes are not defined at arbitrary points on the surface (they are usually only defined for each polygon or sometimes for each vertex). The silhouette set for polygonal meshes is defined as the set of edges which share a front-facing and a back-facing polygon (Figure 2.2, middle). The orientation of polygons is found as follows. X is set to the polygon's midpoint. Assuming the polygon's normal $\mathbf{N}_X$ is pointing away from the surface, if $\mathbf{N}_X \cdot (\overline{X-E}) > 0$, it is back-facing and if $\mathbf{N}_X \cdot (\overline{X-E}) < 0$, it is front-facing. The third case, $\mathbf{N}_X \cdot (\overline{X-E}) = 0$, means that the polygon is exactly edge-on to the view direction. In this case, all of the edges from the polygon are added to the silhouette set.

The simplest approach is to find the set of silhouette edges for a polygonal mesh is by brute-force. This method iterates through each edge in the polygonal model and finds silhouettes by determining the orientation for the polygons that the edge belongs to, as described in the previous paragraph. While this approach is easy to implement, there are many faster methods to find silhouette edges.

[1]The *normal* is the direction perpendicular to the surface for any point on the surface.

Figure 2.1: A silhouette drawing (using the artistic definition of a silhouette).

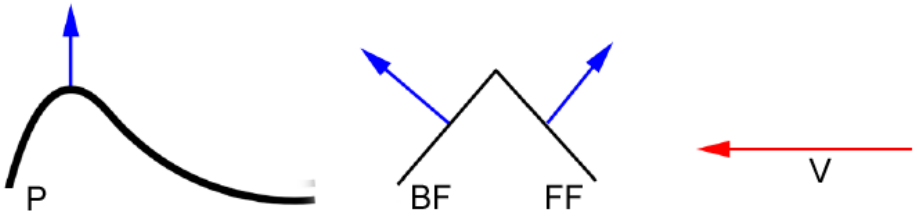


Figure 2.2: A silhouette point and a silhouette edge. The red arrow denotes the view direction V. *Left:* a silhouette point for a smooth continuous surface is defined as any point P whose normal is 90 degrees from the view vector. *Middle:* a silhouette edge for a polygonal mesh is defined as any edge which shares a front-facing (FF) and a back-facing (BF) polygon.

2.1.2 Silhouette Extraction Algorithms

There is a large body of work that explores silhouette extraction and stylization. These algorithms can be classified either as image-space or object-space (or both), whether they require visibility calculations or not, whether they extract silhouette edges or pixels and whether they allow animation or not [IFH+03] (Table 2.1). Since the silhouette system presented in this research is primarily concerned with addressing issues of object-space extraction (Chapter 3), this section focusses on object-space approaches. For complete-

ness, image-space and hybrid approaches receive brief attention below.

Image-space algorithms analyze discrete 2D image buffers created with data projected from the 3D scene and extract discontinuities to create silhouette pixels [ST90, Her99, DS00]. Saito and Takahashi [ST90] present the foundation for image-space silhouette extraction with several filters that estimate the first-order and second-order differentials of the image. Image-space methods based on this approach are generally the fastest approach to extract silhouettes. This is because they solve silhouette detection and visibility in a single step and extract discontinuities from image buffers efficiently [ST90, Her99]. Furthermore, these methods are often very simple to implement and entail little memory overhead. Unfortunately, image-space methods only extract visible silhouettes at the pixel-level; they suffer from aliasing artifacts and they do not lend themselves well to stylization. This is because individual pixels provide insufficient information to stylize a complete stroke and methods for applying stroke texture, realistically simulating width and stroke properties must manually be coded. Object-space approaches provide most of this functionality automatically.

Object-space algorithms are often used to extract silhouettes where stylization of the silhouette is required or when the actual 3D silhouette edges are required. Object-space algorithms extract geometric edges from the polygonal mesh (instead of pixels) and have access to surface information, such as the normal, for any point in the stroke. Furthermore, object-space approaches can extract all parts of the silhouette, not just those that are visible. They usually do this by comparing the view direction to surface normals as described in Section 2.1.1 and some methods use techniques to ignore edges that cannot be silhouettes. These properties make this approach much more suitable for stylization than image-space approaches. Unfortunately, object-space approaches are more computa-

tionally expensive than image-space approaches and often require a secondary process to determine silhouette visibility.

Hybrid algorithms attempt to maintain the fast extraction of image-space approaches and incorporate more stylization control using object-space. These approaches do not extract the actual silhouette. Instead, they use a special rendering pipeline which modifies the position of front-facing and back-facing polygons so that the silhouette is highlighted when rendered with the z-buffer [GSG+99, RvE92, Rus89, RC99]. Most of these methods require several rendering passes to function. For example, Raskar and Cohen's approach [RC99] uses a 2-pass rendering. During the first pass, all polygons are rendered in the background colour with depth-testing enabled. During the second pass the polygons are rendered again, except this time they are draw in the silhouette colour with front-face culling enabled. Silhouettes appear by employing the equal-to depth function during this pass. The primary advantage of hybrid approaches is that they provide more stylization options than image-space methods at a comparable speed. Unfortunately, they do not provide the stylization control of object space approaches, can suffer from z-buffer inaccuracy and only provide a pixel-level representation of the silhouettes.

Object-Space Silhouette Extraction Methods

Specific techniques for object-space silhouette extraction will now be discussed.

The "Edge-Buffer" technique [BS00b], used in this research (Section 3.1), extracts all silhouettes efficiently via a partial vertex adjacency graph. This graph contains a set of bits for each edge in the polygonal mesh. Every time the scene is rendered from a different angle or the model moves, these bits are modified on a per-polygon basis depending on if the polygon is front or back-facing. Then, individual silhouette edges are quickly extracted

using bit-wise logical operators. The advantages of this method are that it works with animated surfaces, it extracts every silhouette edge (and other types of edges specified by the user) and it does not require the expensive preprocesses required by other guaranteed techniques detailed shortly [GSG+99, HZ00, SGG+00].

Other research attempts to lower the number of silhouette tests stochastically by estimating which edges are most likely to be silhouettes. Markosian et al. [MKT+97] use probabilistic testing and chaining to find silhouettes. Their method tests edges that were silhouettes in previous rendering steps and random edges in the vicinity of these. When a silhouette edge is found, their method recursively follows the silhouette until it loops or degenerates. Unfortunately, this approach doesn't guarantee that untested edges are not silhouettes.

There are several other methods that lower the number of silhouette tests but guarantee that all silhouettes will be extracted [GSG+99, HZ00, SGG+00]. These methods use various types of space partitioning to determine quickly which faces contain silhouettes. Unfortunately, spacial partitioning requires complicated implementation and intensive preprocessing. This makes such approaches not suitable for animation, memory intensive and challenging to implement.

Gooch et al. [GSG+99] present a system for interactive technical illustration of polygonal meshes. This system includes a module which colours the interior of the surface, a module which creates silhouettes and two hybrid silhouette revealing techniques. They also present an object-space software method which uses a preprocessing step to allow a fast runtime extraction. This preprocess projects the vertex normals for each edge onto a sphere called a "Gauss Map" and saves the arcs created. At runtime, silhouettes for a certain viewing direction can be found by determining the arcs which intersect a plane

through the origin of the sphere. This method gains in efficiency by storing arcs in a hierarchy which allows for quick culling of regions that cannot contain a silhouette.

Hertzmann and Zorin [HZ00] present a system which calculates hatch marks and silhouettes. Their system also employs a method that quickly culls faces which cannot contain a silhouette based on geometric duality. Each vertex is mapped onto a hypercube in 4D space using its position and tangent plane. The problem of finding faces which intersect the silhouette is reduced to intersecting the triangles in the 4D space with the viewpoint's dual plane. Any edge that intersects in this test intersects the silhouette. The system's speed gain comes from an octtree space subdivision which subdivides the vertices on each side of the hypercube and can quickly determine which groups of edges contain silhouettes.

Sander et al. [SGG+00] find groups of faces that might contain the silhouette using a hierarchical tree that stores mesh edges and their associated faces and "anchored cones". The system binds two cones to each node in the tree. One cone represents the viewpoint for the entire group of faces in a node to be front-facing, and the other cone represents the position of the viewpoint for all of the faces to be back-facing [IFH+03]. By determining if the viewpoint lies inside any of the cones, their system can quickly cull groups of faces which cannot contain a silhouette.

The processing time required to set up the data-structures used for some of these systems [SGG+00, HZ00] can be very expensive for large polygonal meshes, making animation and morphing difficult or impossible with current hardware. Despite this, these methods are useful to accelerate silhouette extraction for static surfaces where they provide a large speed increase over the brute-force approach.

2.1.3 Silhouette Error Correction

Silhouettes extracted directly from 3D meshes may contain artifacts such as “zig-zags”, overlaps and loops (Figures 2.3, 2.4). The causes of these artifacts are:

1. Numerical instability: Methods that extract edges from the polygonal mesh compare the result of a dot product operation with zero. This can return incorrect results where polygons are nearly edge-on to the view direction, due to floating point precision problems.

2. Unsuitable edges from the mesh: Since the mesh is a discrete approximation of a surface, edges that make up this mesh will rarely conform exactly to the “actual” silhouette. Depending on the connectivity and orientation of the edges, completely unsuitable edges might be used in the silhouette.

These artifacts compromise the quality of the stroke stylization process and subsequent rendering results. In Figure 2.4, four frames illustrate different combinations of artifacts. Silhouettes for these images have been calculated with view direction α, but are displayed with view direction β, such that $\alpha \neq \beta$. Note the black line, which is the extracted silhouette, the unshaded front-facing polygons and the shaded back-facing polygons. As the silhouette crosses the mesh surface, it moves back and forth across an invisible threshold which is the “actual” silhouette. Edges taken directly from the mesh only provide an approximation to where the actual silhouette should appear.

Table 2.2 lists methods that provide polygonal mesh silhouette error correction. These either (1) correct errors from silhouette chains created from the actual mesh edges [NM00], (2) create more suitable, sub-polygon silhouette lines without using the edges in the mesh [CJTF98,

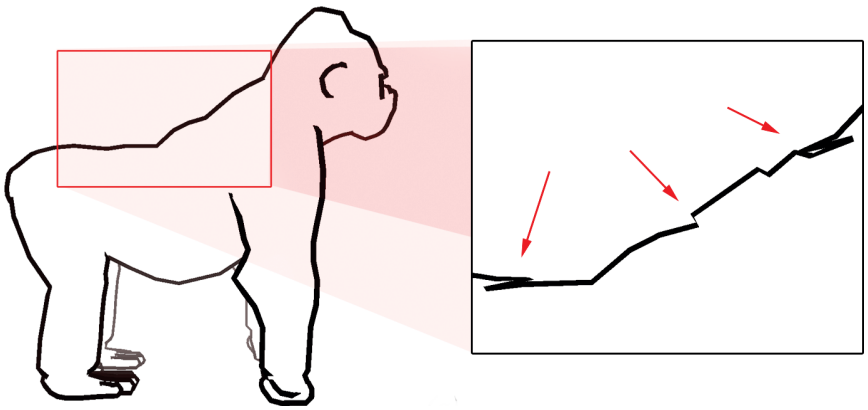


Figure 2.3: The silhouette of an ape mesh with highlighted errors and artifacts.

HZ00] or (3) do both [IHS02]. These methods use object-space approaches [HZ00, IHS02] or hybrid combinations of image-space and object-space [NM00, CJTF98].

Côrrea et al. [CJTF98] create continuous, smooth silhouettes on a 3D model. This system uses an image-space solution to generate new sub-polygon silhouette edges. Their system creates 2D *u,v-images* which are coloured based on the u,v coordinates of the mesh. Discontinuities in this image correspond to visible silhouettes and boundaries on the 3D model and are found by analyzing pixel-neighborhoods. For each silhouette pixel, a silhouette edge is mapped into object-space using the depth buffer. Curves are created by joining these newly created silhouette edges. Unfortunately, this system requires a large amount of user input to function.

Northrup and Markosian [NM00] present a system which extracts silhouettes in object-space and performs corrections in image-space. Silhouettes are extracted using the process described in [MKT+97] and are projected to image-space. Then their system checks for error-cases and applies the corresponding solutions. These include elimination of undesirable silhouettes, joining uneven endpoints and other operations to create smooth chains.

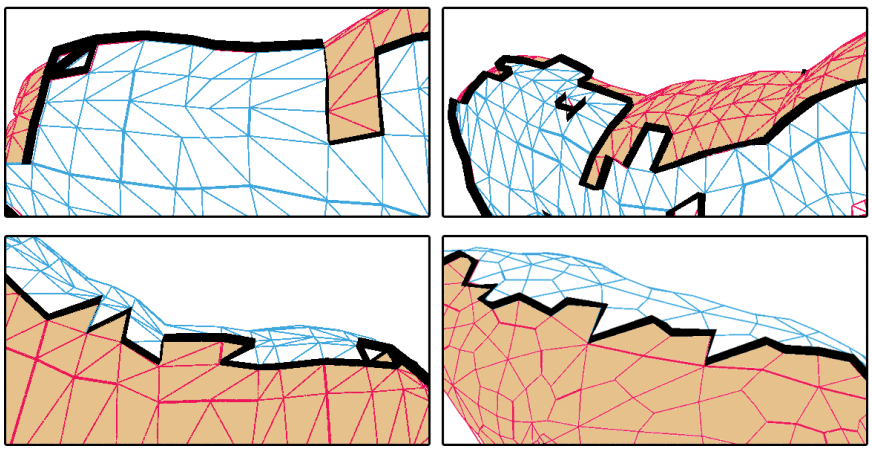


Figure 2.4: Four images showing various silhouettes and underlying mesh that generated them. The silhouettes are presented at a perturbed view to provide a better understanding of the cause of the errors. Shaded polygons were back-facing when the silhouettes were extracted.

To stylize, the system renders corrected chains in image-space using an “artistic-stroke” method to create a wide range of expressive strokes and styles.

Isenberg et al. [IHS02] also correct silhouette errors directly from the edges by checking for various error-cases and providing a solution for each case. In their system, however, stroke cleaning occurs in image-space and object-space. To accomplish this, their system uses a two-pass approach which first analyzes silhouettes based on the mesh and then checks their appearance. In the first pass, adjacent edges connecting at acute angles below a given threshold are replaced by a single edge, to prevent “zig-zag” patterns. Also, triangle clusters which create large artifacts on the silhouettes are identified and removed. At this stage, the silhouettes contain only edges from the mesh. The second pass removes silhouette “zig-zags” and short segments by analyzing vertices in image space merging them. Moreover, sharp angles in the silhouette are unlinked so that good output is created

during stylization. At this stage, since vertices have been modified, the silhouette edges will not follow the exact geometry of the surface.

Hertzmann and Zorin [HZ00] present an object-space approach that minimizes errors by creating new sub-polygon silhouette edges for smooth polygonal meshes converted from free-from surfaces. Their system relies on vertex normals instead of standard face normals. Silhouette edges are created by estimating the exact position where the silhouette crosses edges from the mesh by interpolating vertex-normals along the edge and joining adjacent pairs of these points to create edges. Their system also supports a hatching algorithm to create interior strokes along principal directions of curvature.

In contrast to these previous approaches, the MAR error-removal system presented in this research (Chapter 3) works without requiring classification of errors and evaluation of fixes. Furthermore, like Isenberg et al. [IHS02] and Northrup and Markosian [NM00], MAR removes errors from silhouette chains created from mesh edges instead of procedurally generating new edges. The MAR approach, however, modifies the silhouette edges using sub-polygon resolution to better approximate the silhouette. Furthermore, the silhouettes thus generated are resolution-independent which is useful to simulate realistic pen strokes. This also proves to be useful for examining silhouettes from detailed meshes closely and when extracting silhouettes for simple meshes. Finally, the system can also generate artistic expressive strokes.

2.2 Precise ink-based illustration

Various approaches for placing small pen-and-ink primitives over 3D models have been proposed. Winkenbach and Salesin [WS94, WS96] present convincing methods to render

pen-and-ink strokes on polygonal meshes and parametric surfaces. They introduce “stroke textures” for polygonal meshes [WS94] allowing for procedural accumulation of strokes to create tone and texture, and “controlled density hatching” for parametric surfaces [WS96]. This method distributes strokes on the parametric surface to create a selected tone or effect. These approaches can achieve effects such as silhouette generation, scale-dependant rendering and shadow casting.

Several systems build on this technology of stroke textures [WS94] including Praun et al. [PHWF01] who introduce “tonal art maps”. These organize pre-rendered strokes as a sequence of mip-mapped images. Strokes within the images are scaled to attain appropriate stroke size and density at various resolutions. This method, an early example of using hardware multi-texturing for NPR, uses lapped textures [PFH00] to place seamless patterns of strokes on the surface. Lapped texturing is a method of repeatedly placing small image patches on a surface in a way so that joins are not visible until the surface is entirely covered.

Elber [Elb98] provides techniques for creating short strokes across freeform implicit and parametric surfaces. Notably his system includes the ability to draw strokes in various directions, such as principal directions of curvature. It is also possible to group strokes in several configurations in several interesting stroke styles.

Deussen and Strothotte [DS00] present a system which generates pen-and-ink images of 3D trees. Their system uses a hybrid pixel/geometric approach to handle the geometry of the tree skeleton and leaf particles. The main strength of this system is the ability to generate pen-and-ink images at different levels of detail using various styles. This is important for illustration since level of detail is used by artists for depth perception, to focus viewers’ attention and to add style and feel to the image.

An important pen-and-ink style is stippling, a style that involves precise positioning of small dots such that their density produces tone and implies shape and texture. Recent investigation into this style has focused on the geometric relation between the stippled dots [DHvOS00] and on interactive direct volume illustration systems [LME+02]. The system from Deussen et al. [DHvOS00] provides an automatic method to place stipples based on halftoning images and a user-driven painting interface which generates convincing stipples extremely quickly. Using this interface, the user needs only to "paint" the regions on the surface where stipples exist. The system automatically places the individual stipples using a relaxation method based on Voronoi diagrams to evenly distribute the stipples, randomly jitter the stipples and to shape the stipples.

Secord [Sec02] presents two techniques, one real-time using pre-computed information and a slower high-quality iterative technique for generating stippled drawings using weighted Voronoi centroidal diagrams. This system places small arbitrarily-shaped primitives on grayscale images to achieve a convincing stippling effect and requires very little user input. More recently, Pastor et al. [PFS03] attached stipple particles to the surface of the model, using a point hierarchy to control the stipple shading density in a realtime system suitable for animation.

Interior mark extraction

An important component of precise ink-based illustration is the proper extraction of interior marks. These marks are used to illustrate depth, form, texture, curvature and other surface properties. Researchers have explored the use of lighting parameters/equations for measuring and rendering models in the context of technical and scientific illustra-

tion [GGSC98, GSG+99, HS99]. The approach used in this research is to extract and render shape features by calculating local shape measures directly at the 3D mesh, with no need for either illumination or surface reflectance information. For polygonal meshes, shape measures are usually estimated at every vertex of the model, taking into account some local properties of the adjacent triangles to each vertex, such as triangle angles, areas, and edge lengths [Boi95, YKM99, GIHL00]. In NPR, shape measure use has been focussed on finding principal directions of curvature to guide stroke placement [GIHL00, HZ00, RK00, PHWF01]. This work employs geomorphological shape measure calculation schemes to guide stroke placement [Eva72, MH93, SSM02, Woo96]. They provide a large and computationally stable collection of shape measures.

For the interior stroke creation method presented in this research, these methods are adapted to work directly with 3D triangle meshes. To do this, the Edge-Buffer data structure [BS00b] is modified so that various shape measures can be directly calculated at each edge of the model using only the information of its two adjacent faces. The benefit of this is that this system can use edges directly from the mesh to create strokes.

Approach	Method		Additional Operations	Precision		Misc.	
Reference	Image Sp.	Object Sp.	Requires Visibility Test	Pixel	Edge	Allows Animation	Intelligent Extraction
[ST90]	√			√		√	
[Her99]	√			√		√	
[DS00]	√			√		√	
[GSG+99]	√	√		√		√	
[RvE92]	√	√		√		√	
[Rus89]	√	√		√		√	
[RC99]	√	√		√		√	
[BS00b]		√	√		√	√	
[MKT+97]		√	√		√	√	
[GSG+99]		√	√		√		√
[HZ00]		√	√		√		√
[SGG+00]		√	√		√		√

Table 2.1: A classification of polygonal silhouette extraction methods.

Approach	Error-Removal Method		Correction Source		Precision		Silhouettes Corrected	
Reference	Image Sp.	Object Sp.	Polygon Edges	Sub-polygon Edges	Pixel	Geometry	Visible	All
Côrrea et al. [CJTF98]	√			√	√		√	
Hertzmann and Zorin [HZ00]		√		√		√		√
Northrup and Markosian [NM00]	√		√		√		√	
Isenberg et al. [IHS02]	√	√	√			√	√	

Table 2.2: A classification of polygonal silhouette error-correction methods. From left to right, the columns describe the following: **Approach** provides the reference for the approach; **Error-Removal Method** describes the combination of image-space and object-space techniques the method uses; **Correction Source** displays whether the method corrects polygon edges or creates new sub-polygon errors; **Precision** tells whether the method returns pixel (image-space) results or edges from the geometry of the polygonal mesh and **Silhouettes Corrected** describes whether all silhouettes or only visible silhouettes are corrected.

Chapter 3

The Silhouette Stroke Extraction System

The *Silhouette Stroke Extraction System* (1) links single edges from the Edge-Buffer [BS00b] into long strokes, (2) removes artifacts and errors from the silhouette using the MAR (Multiresolution Artifact Removal) approach and (3) stylizes the strokes.

The main contribution of this system is the ability to remove errors from the silhouette strokes. These errors occur because of numerical instability and unsuitable edges from the polygonal mesh (Section 2.1.3, Figures 2.3, 2.4). The quality of the stroke stylization process and subsequent rendering results are compromised due to these errors. Four approaches have already been presented to remove these errors [NM00, IHS02, CJTF98, HZ00]. They remove errors with a set of error-cases and corresponding solutions or they extract sub-polygon silhouettes that do not contain errors. The results of all of these methods are bound to the resolution and dimensions of the polygonal mesh.

The approach presented here (the *MAR system*) uses multiresolution based on reversing subdivision [BS00a, SB99] to remove errors from the discrete raw silhouette data (Section 3.3.2). This approach results in good approximations to traditional hand-drawn pen-and-ink silhouettes (Figure 1.3). This system is resolution-independent as the multiresolution error-removal filters can create coarse approximations of chains and can smooth the strokes to a higher level of detail than that of the original data. The primary advantage of this method is that error-corrected sub-polygon strokes can be generated from the raw silhouette data without the need to inspect individual error-cases. Furthermore, the strokes generated with this method include automatic stylization controllable by the user to create

strokes with various levels of accuracy.

This chapter first describes work related to the key concepts of our method. This includes (Section 3.1) the Edge-Buffer system and (Section 3.2) related work for multiresolution. Then this chapter provides details for (3.3) the *Silhouette Stroke Extraction System*. This includes (3.3.1) modifications to the Edge-Buffer, (3.3.2) a detailed explanation of the multiresolution filters, (3.3.3) how these filters are used to remove silhouette errors, (3.3.5) notes on resolution-independence and (3.4) the system's silhouette stylization method.

3.1 The Edge-Buffer

The Edge-Buffer, designed by Buchanan and Sousa [BS00b], efficiently extracts feature edges from open or closed polygonal meshes. This approach considers feature edges as silhouettes, boundaries (edges which are only connected to one polygon) and artist edges (edges specified by the user to always be drawn). To this end, the system uses a specialized data structure and a simple traversal algorithm to find all such edges in a single pass.

3.1.1 Initialization

To operate, the Edge-Buffer approach requires two data-structures: the Edge-Buffer itself and a data-structure for the polygonal mesh that stores indexed faces and vertices. Moreover, each face stores the indices of the vertices they contain. Once the polygonal mesh has been loaded and initialized in such a structure, the Edge-Buffer is initialized as an array V containing all m vertices in the mesh. For any $V[1 \cdots m]$, $V[i]$ is the linked list of all the vertices v_j adjacent to vertex v_i (Figure 3.1a,b). Note that this list is sorted in increasing order of vertex numbers j starting from i, such that each pair (v_i, v_j) forms an

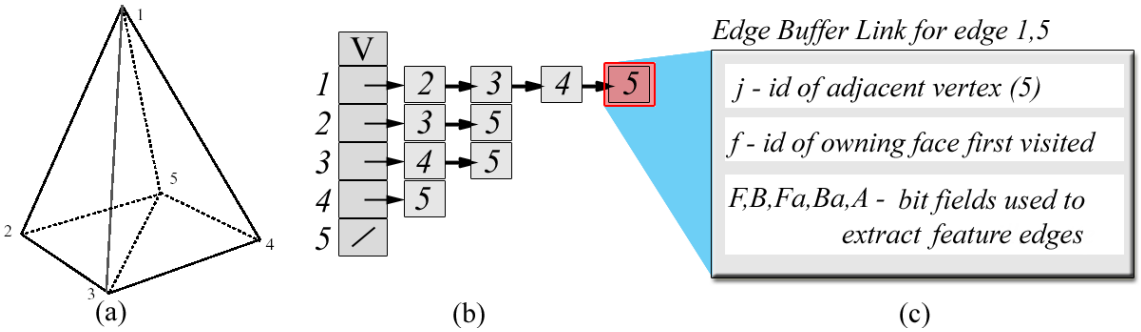


Figure 3.1: A visual description of the Edge-Buffer: (*a*) a simple polygonal object, with vertex numbers listed; (*b*) an example of how edge adjacency information is stored in the Edge-Buffer [BS00b] for this polygonal object; (*c*) the data stored in each node in the linked lists.

edge. Thus there is only one entry for each i, j combination, where $i < j$. Note that an effect of i, j ordering is that higher indexed vertices have fewer links. Each node in the linked list contains the following data (Figure 3.1c):

1. j, the identifier of vertex v_j adjacent to v_i. If all vertices are stored in an array, then j is the index to v_j in the array.

2. five bit fields (F, B, F_a, B_a, A). Each of these fields contains a boolean value used during run-time to extract the silhouette, boundary and artist edges quickly. Their use is explained in Section 3.1.2 below.

3.1.2 Run Time Operation

After initialization, feature edges are ready for extraction and display using five bit fields (F, B, F_a, B_a, A). F and B are used for silhouette and boundary extraction and F_a, B_a, and A are used to extract artist edges.

First, the bit fields FBF_aB_a are initialized to 0. Then, the system classifies every polygon in the mesh as either front or back-facing using the dot-product operation (Section

2.1.1). As each polygon is checked, the bit fields FB for each of the edges that make up the polygon are updated. This can be done efficiently because each polygon has its edges cached in the polygonal mesh's data-structure. The bits are updated in the following manner. The current value of F is inverted if the polygon is front-facing. Similarly the current value in B is inverted if the polygon is back-facing. Once all polygons are tested, non-boundary edges will have been visited twice and boundary edges will have been visited once. Silhouette edges are are extracted when $FB = 11$, and boundary edges are extracted when when $FB = 10$ or $FB = 01$.

Artist edges are handled as follows: the user first initializes all artist edges by setting their A bit. The F_a and B_a bit fields are used to determine which of these edges are front-facing or back-facing respectively. As each polygon is checked, the F_a bit for each edge in the polygon is set to 1 if it is front-facing. Likewise, the B_a bit is set to 1 when the polygon is back-facing. After all edges are processed, front-facing and back-facing artist edges are extracted when $F_aB_aA = 101$ or $F_aB_aA = 011$ respectively.

3.2 Multiresolution

The Silhouette Stroke Extraction System system uses a multiresolution approach to remove the silhouette errors (the MAR approach). Multiresolution (Section 3.3.2) is a technique whereby a set of data can be *decomposed* into a set of coarse data and details, each of which is usually half the size of the original data. Then, the original data can be completely *reconstructed* using only the coarse data and details.

Finkelstein and Salesin [FS94] demonstrate the first use of multiresolution in NPR with a curve-editing system based on wavelets. More recently, Kirsanov et al. [KSJ03]

use coarsening methods to simplify silhouettes from detailed polygonal meshes. More information on the wavelet multiresolution approach is found in Stollnitz et al. [SDS96].

To remove errors from polygonal silhouette chains, the MAR approach uses a different type of multiresolution that is based on reversing subdivision. This multiresolution, developed by Samavati and Bartels [BS00a, SB99], offers simple linear time operations [SB04] and several different filter sets to operate. Furthermore, Samavati and Bartels present two versions of this approach, *local* [BS00a] and *global* [SB99] filters. The local filters are obtained based on solving the best solution via a local least squares problem while the global filters are obtained based on a global least squares problem. These filters produce an optimal solution intrinsically without any extra work in implementation. In the case of the local filters, the implementation is very simple and the coarse data it generates is the best l^2 approximation for a set of data in a local neighborhood. In contrast, coarse data found with the global approach is the best l^2 approximation for the entire set of data. The global approach requires a more complicated implementation, however it still produces results in linear time.

3.3 The Silhouette Stroke Extraction System

To create silhouettes, the *Silhouette Stroke Extraction System* follows three steps as illustrated in Figure 3.2. First, silhouettes are extracted with the Edge-Buffer and the edges are connected into long stroke chains. Then, errors are removed using the multiresolution system. Finally, a stylization step is performed using traditional NPR stroke creation methods. With the exception of the Edge-Buffer, which is detailed previously (Section 3.1), all these steps are described in the given order in the following sections.

Figure 3.2: The *Silhouette Stroke Extraction System* pipeline.

3.3.1 Edge-Buffer modifications

The original Edge-Buffer system [BS00b] extracts individual edges as described in Section 3.1. The MAR approach (Section 3.3.3) requires complete silhouette chains, so the Edge-Buffer must be extended to create these chains from individual edges. Note that the *Silhouette Stroke Extraction System* treats boundary, silhouette and artist edges (Section 3.1) equally; all types of edges extracted by the Edge-Buffer are chained together.

A two-pass algorithm is used to chain individual silhouette edges extracted by the Edge-Buffer. First, the system links all extracted edges on the model by finding the connected components of the Edge-Buffer (Figure 3.3, *top-right*). The algorithm iterates through all edges (v_i, v_j) for each vertex v_i in the Edge-Buffer. Once this step finds a silhouette edge, it proceeds to vertex v_j and searches for another silhouette edge. The system continues in this fashion until it cannot find another silhouette edge. This step must know which edges are already part of a chain and which edges have not been used yet. This is accomplished by adding another bit-field, the used bit "*U*", to each Edge-Buffer node. Each time the view transformation changes (eg. when generating animated sequences, or when changing the view direction), U is initialized to 0. During the chaining process, any extracted edge that is used in a chain has its U bit set to 1. While creating future chains, any edge with its U bit set to 1 is excluded from chaining. Thus, each edge can only belong to one chain. Once all extracted edges have been used, this portion of the algorithm is complete. At this point the chains will only contain increasing vertex elements (Figure 3.3).

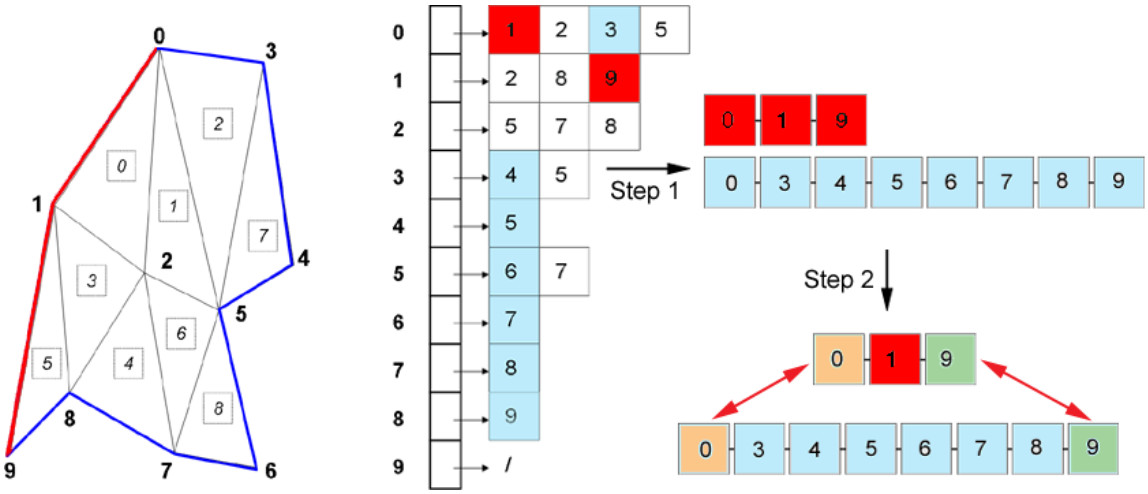


Figure 3.3: The two pass process used to chain silhouette edges. *Left:* An example polygon mesh with vertex and face numbers and its associated Edge-Buffer structure. *Top-right:* The first step in silhouette chaining. In this example, two chains are created, shown in red and blue on the polygonal mesh. These are extracted individually, following nodes of the Edge-Buffer until no further connections can be created. *Bottom-right:* The second step for chaining. The ends of each chain are examined for any matching vertex indices. Matches are joined with preference for creating looping chains.

This is due to the directed nature of the Edge-Buffer. Thus, a second step is needed to join chains that do not have increasing vertex numbers (Figure 3.3, *bottom-right*).

In this second pass, the system joins chains with matching vertex numbers on their bounds. If more than two chains can be linked, priority is given to chains that will create long loops when joined. Looping chains take precedence because the MAR pipeline (Section 3.3.2) handles looping and non-looping chains slightly differently (non-looping chains are interpolated at the start and end of the chain). Thus, it is important to identify looping chains to avoid creating small new artifacts at boundaries of the chain. The MAR pipeline also requires a minimum chain length to function properly (the lowest is a length of 4), depending on the type of filter used (Section 3.3.2).

This chaining method cannot guarantee the longest connected chains. Instead, it guar-

antees satisfactory long chains for use with the multiresolution filters.

3.3.2 Local and Global Filters

Once complete silhouette chains have been constructed from the Edge-Buffer, the *Silhouette Stroke Extraction System* employs the MAR error removal pipeline, described in Section 3.3.3. Before details are provided for this pipeline, an effective review of the multiresolution techniques [SB99, BS00a] that form its base is in order.

Multiresolution methods decompose a dataset C^{k+1} into a low-resolution approximation C^k and a set of high frequency details D^k. Note that k is used in this document to specify the level of detail in the data. The original data C^{k+1} can at any time be reconstructed from C^k and D^k. The process of transforming C^{k+1} to C^k and D^k is called *decomposition*, while and generating the original data C^{k+1} from C^k and D^k is called *reconstruction*. These can be applied to C^{k+1} more than once.

In the functional view, C^{k+1} is the coefficient vector of high resolution scaling functions, C^k is the coefficient vector of low resolution scaling functions and D^k is coefficient vector of Wavelet functions. If C^{k+1} is a silhouette chain, C^k shows the overall sweep of the silhouette and D^k shows silhouette errors (Section 2.1.3) because these are the high frequency portions of the chain.

The multiresolution operations can be specified in terms of the banded matrices A^k, B^k, P^k and Q^k. The matrix A^k transforms C^{k+1} to C^k:

$$C^k = A^k C^{k+1} \tag{3.1}$$

and B^k extracts details:

$$D^k = B^k C^{k+1} \tag{3.2}$$

P^k and Q^k act on C^k and D^k to reconstruct C^{k+1}:

$$C^{k+1} = P^k C^k + Q^k D^k \tag{3.3}$$

These matrices have a regular banded structure for every resolution for the looping case. In the non-looping case, the matrices are regular except at the bounds of the matrix, where data is interpolated. Examples (refer to [BS00a]) of the non-zero entries that can be used for the rows that make up the A^k and B^k matrices and the columns that make up the P^k and Q^k matrices are provided in Figures 3.4-3.6. These matrices can be viewed as filters that operate on C^{k+1}, C^k and D^k due to their regularity. Furthermore, the only implementation difference between A^k and A^{k-1} is their size. Consequently, the superscript of matrices can be removed.

In order to find these four matrices, most multiresolution research uses wavelet-based techniques [FS94, SDS96, KSJ03]. In the case of smooth curves, the resulting wavelets are often inadequate (see appendix of Finkelstein and Salesin [FS94] or page 94 of Stollnitz et al. [SDS96]). For the MAR approach, C^{k+1} is a discrete approximation of a silhouette curve and the only requirement is use of the appropriate appropriate *A*, *B*, *P* and *Q* filters. Therefore, a discrete approach of multiresolution systems that directly operates on discrete data is fitted here more effectively. Samavati and Bartels [BS00a, SB99] provide this kind of multiresolution based on reversing subdivision. They also demonstrate that their results are more effective for discrete data sets than conventional wavelets. In the *Silhouette Stroke Extraction System*, their multiresolution filters constructed based on reversing *cubic*

B-Spline subdivision, *Chaikin* subdivision and *Dyn-Levin interpolation* subdivision are used. The bands that make up the *A*, *B*, *P* and *Q* filters are provided for these systems in Figures 3.4, 3.5 and 3.6.

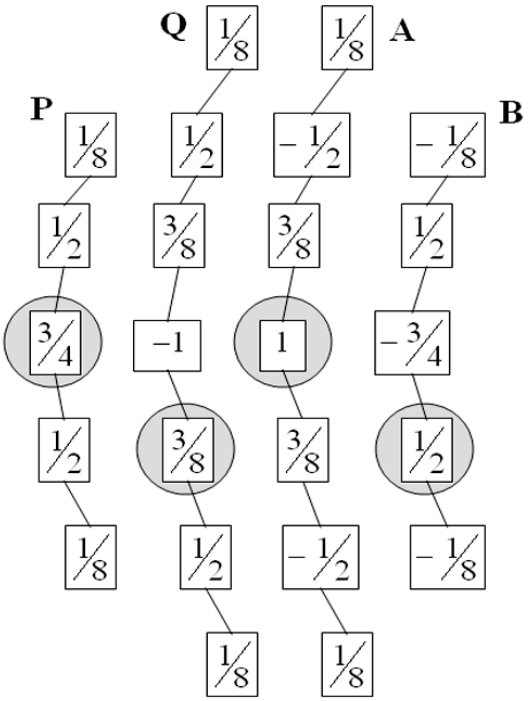


Figure 3.4: The bands of the filters for the cubic B-Spline case (the A, B, P and Q diagrams represent all non-zero entities of a row for the *A* and *B* matrices and of a column for the *P* and *Q* matrices). The gray circles show the center entity.

For implementation, *A* and *B* are applied to C^{k+1} to obtain C^k and D^k. Again by applying *P* and *Q* filters on C^k and D^k (or, as is done in the next section to remove errors, a modified version of D^k), C^{k+1} can be reconstructed. Recall that these filters produce an optimal solution intrinsically without any extra work in implementation, they usually require no extra space and they provide linear-time operations. For the local approach, *A*, *B*, *P* and *Q* are all regular banded matrices. In the global approach, *A* and *B* are full matrices. Nevertheless, they still have the regular structure. In order to achieve linear time operations,

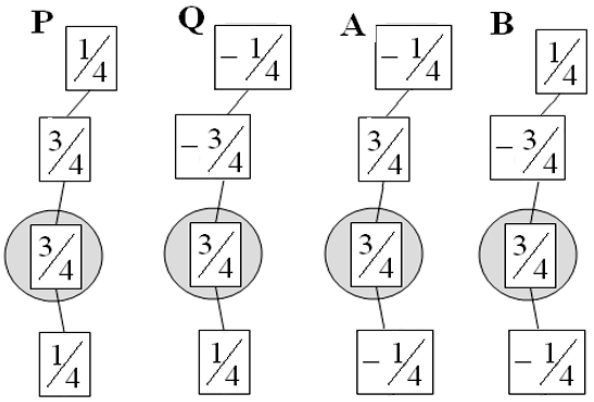


Figure 3.5: The bands of the Chaikin filters.

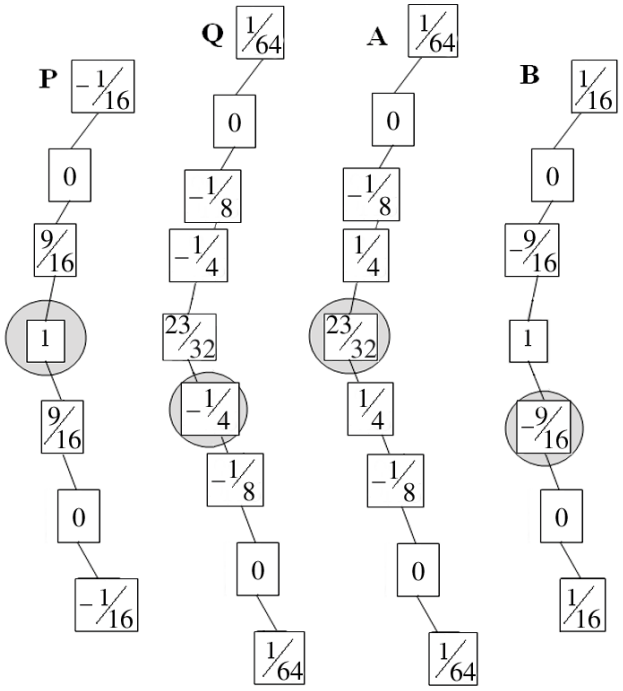


Figure 3.6: The bands of the Dyn-Levin interpolation filters.

the MAR approach solves the following banded system for global decomposition [SB99]:

$$(P^t P)C^k = P^t C^{k+1} \tag{3.4}$$

$$(Q^t Q)D^k = Q^t C^{k+1} \tag{3.5}$$

In a recent work, Bartels et al. [BGS04] demonstrate that the local approximation is a good estimate of the global approximation. For silhouette error removal, experiments comparing local and global multiresolution show that for low resolution meshes, global multiresolution is required to create accurate enough strokes (Section 5.1). The drawback of using global multiresolution is the need of solving the systems in Equations 3.4 and 3.5.

Other Multiresolution Approaches

The reverse subdivision multiresolution filters of Samavati and Bartels [SB99, BS00a] were chosen over that of Finkelstein and Salesin [FS94] and Stollnitz et al. [SDS96] for three primary reasons. First, reverse-subdivision multiresolution has been demonstrated to be more effective for discreet data than wavelet-based approaches [SB99, BS00a]. Second, Samavati and Bartels provide various types of filters with a “local” or a “global” scope while Finkelstein and Salesin and Stollnitz et al. only provide filters for a wavelet-based cubic B-Spline multiresolution. This provides variety, to properly remove errors and stylize silhouettes, and efficiency, as the approaches provided by Samavati and Bartels are much faster than the wavelet-based approach (see [BSS04]). The third reason that reverse subdivision multiresolution was chosen over the wavelet-based approach is because the masks it uses are simpler. The masks that make up wavelet-based filters have a large width which contain complicated rational numbers compared to a narrow width with sim-

ple fractions used by Samavati and Bartels. Simple fractions are preferred to minimize rounding errors. Also, the narrower mask width of the reverse-subdivision filters further increases computational efficiency.

3.3.3 MAR: Multiresolution Artifact Removal

In this section, details are provided on how the multiresolution techniques presented by Samavati and Bartels [SB99, BS00a] are used in the Multiresolution Artifact Removal (MAR) approach. MAR's default silhouette error-removal pipeline decomposes silhouettes in two steps and reconstructs to the original level of detail with a scaled-down version of the high-frequency details to remove errors (Figure 3.7). Note that the user can change the number of times that the silhouette is decomposed as large errors may require three steps of decomposition and small errors may only require one step. A discussion of this is provided in Section 5.1.

As shown in the previous section, normal reconstruction with coarse information and high-frequency details returns the coarse data to its exact original form. Scaling down the amount of details means that the high-frequency data will be lessened in the silhouette, resulting in removal of errors. The MAR system modifies Equation 3.3 so that it can lessen the amount of details included in reconstruction:

$$\bar{C}^{k+1} = PC^k + eQD^k \tag{3.6}$$

where e is a scalar between 0.0 and 1.0 that varies the percentage of the detail data added to coarse data. The higher the value of e, the closer the stroke gets to the original data extracted.

Raw silhouette chains from the Edge-Buffer can be thought of as a low frequency

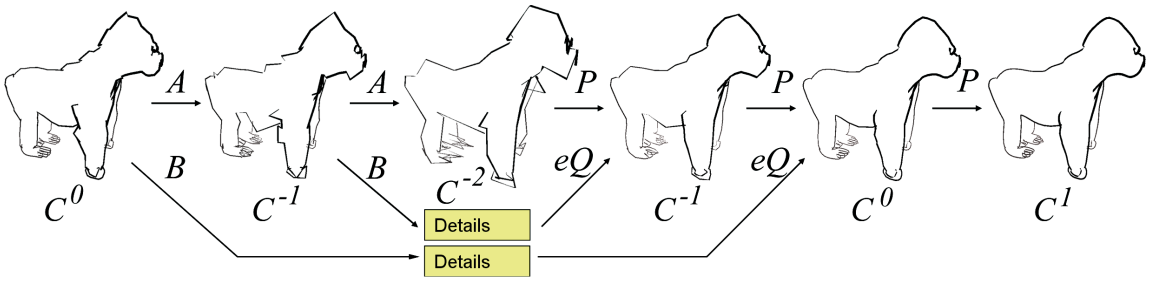


Figure 3.7: The MAR approach uses multiresolution filters to decompose and reconstruct silhouette chains without errors. Here is an example session of the MAR approach for an **ape** mesh with 7434 faces. Proceeding from left to right, the silhouette chains are first decomposed twice from level C^0 to C^{-2}. Then, the system reconstructs to level C^0 using scaled details (here, $e = 0.3$). The effect of this process is the removal of errors. Finally, the MAR approach reconstructs past the original level of detail to C^1 to provide a smoother more stylized silhouette.

"correct" path plus high frequency errors (Figure 2.4). Since the high frequency portion of the silhouette chain is extracted and stored in details, a lower value for e eliminates more errors as a lower percent of the high-frequency details are included in the reconstructed strokes. In Figure 3.8, the MAR approach has been applied to the silhouettes illustrated in Figure 2.4. Note that the high-frequency noise has been removed and that the accuracy of the stroke has been maintained.

The error-removal effect of the MAR approach is also illustrated in Figure 3.9. In this image, the original silhouettes have been decomposed and reconstructed once with global cubic B-Spline filters. In the leftmost image in this figure, 100% of the details are included in reconstruction ($e = 1.0$) resulting in the exact original silhouette being regenerated. Moving right in Figure 3.9, fewer and fewer details are included, until the rightmost image, where 0% of the details are included ($e = 0.0$). Note in the leftmost image that the jagged movement of the silhouette on the beaver's back has created some minor artifacts. As details are removed, these high frequency "zig-zags" are scaled down

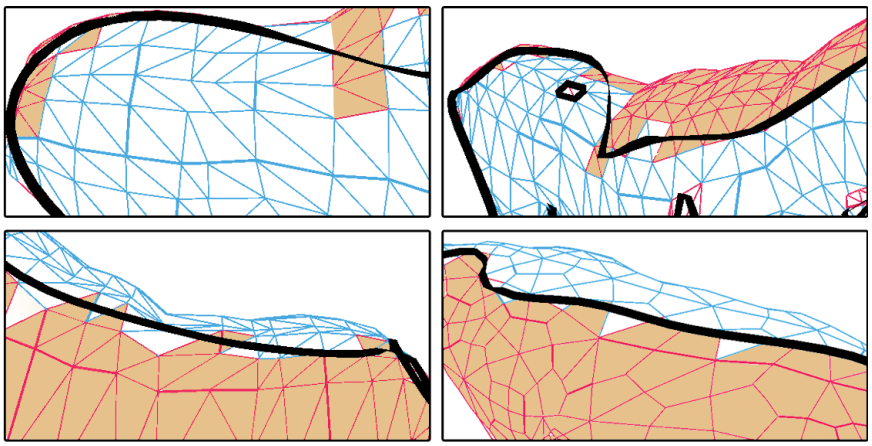


Figure 3.8: Strokes generated by the MAR system for the input in Figure 2.4.

while the low frequency, correct path of the silhouette is maintained.

The images provided in this work demonstrate that the output of this system is suitable for pen-and-ink silhouette illustration. Values from 0.0 to 0.4 are used for e, depending on the detail in the original mesh used. A discussion of the performance of this system is provided in Chapter 5.

3.3.4 Chain Size

The silhouette chains sometimes need to be modified before applying MR filters. Recall the decomposition process described in Section 3.3.2, where a dataset C^{k+1} is decomposed into a coarse approximation C^k and details D^k. If the length of C^{k+1} is n, then the length of C^k will be $n/2$ when the chain loops or $2+n/2$ otherwise. Since the length of the chain must be a whole number, C^{k+1} must have a length divisible by two for decomposition. The *Silhouette Stroke Extraction System* handles this by adding a single point to any chain of odd length before performing any level of decomposition. This point is added at the

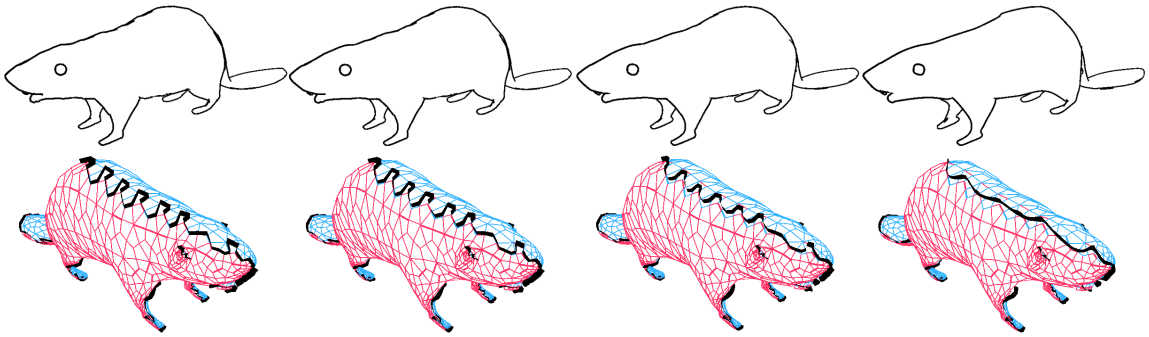


Figure 3.9: The effect of removing details. In this image, silhouettes from a **beaver** model are shown for two angles, head-on in the top row and from behind with mesh information on the bottom row. One level of decomposition and reconstruction is used here with global cubic B-Spline filters. From left to right, $e = 1.0$, $e = 0.66$, $e = 0.33$ and $e = 0.0$.

second last position in the chain, interpolated between its neighboring points. During reconstruction, the system removes these extra points once they are reconstructed.

3.3.5 Resolution-Independent Strokes: Smoothing/Coarsening

In the MAR approach, reconstruction can proceed to a higher level-of-detail than the original chain to smooth the silhouette strokes or can stay at a low level-of-detail to coarsen the strokes. For the higher levels of detail, since there are no details D^k associated with the strokes, this is accomplished simply by eliminating QD^k in Equation 3.6 to create:

$$C^{k+1} = PC^k \tag{3.7}$$

The P filter is actually a subdivision matrix. Thus, using P alone increases the smoothness of C^k. This is useful when the input mesh has a low number of triangles or when one wants to view the silhouettes from a larger mesh closely and rough edges are not desired in the final output. This is illustrated for a mesh with a lower number of triangles in Figure 3.10.

With the MAR process, the user has control over the number of times to decompose and reconstruct, the method to do this decomposition and reconstruction (Chaikin, cubic

B-Spline or Dyn-Levin), the scope of the method (local or global) and the amount of details to include in the reconstruction (the *e* value). Note that low-pass filters do not give this level of control. A discussion of the results from using these different approaches is provided in Section 5.1.

3.4 Stylizing

The *Silhouette Stroke Extraction System* uses two steps to create appealing strokes that approximate real hand-drawn pen-and-ink illustrations. First, silhouette chains that appear too coarse can automatically be smoothed in the MAR process using the resolution-independent chains (Section 3.3.5). The second step to stylize the strokes is to use the angled-bisector strip presented by Northrup and Markosian [NM00]. This method converts the silhouette chains into triangle strips which simulate pen strokes that vary in width (controlled by the distance from the eye to each point in the stroke). Points in the chain closer to the eye are stylized as wide portions of the stroke while points farther away produce narrower portions.

The *Silhouette Stroke Extraction System* provides two options for Hidden Line Re-

Figure 3.10: The rightmost two **ape** images from Figure 3.7. Here the system performs reconstruction on corrected strokes above the original level of detail. This results in a smoothing effect, making the geometry of the underlying mesh less apparent.

moval (HLR). The first method works in image-space and is very efficient, however it fails when used with silhouettes from coarse meshes. The second method uses an image-space technique to produce more accurate results. Both methods are described in the next two sections.

3.4.1 Object Space HLR

For object-space HLR, the method from the original Edge-Buffer system [BS00b] is used. This method first renders the polygonal mesh in white and then draws the silhouettes in black. Thus, any strokes on the back-facing side of the surface will be occluded by the white mesh. To ensure that the mesh does not partially occlude edges on the front side of the surface, the silhouette edges are displaced away from the mesh slightly.

While this approach is acceptable for the Edge-Buffer system, it does not always work with the strokes generated with the MAR system because these processed strokes do not adhere exactly to the mesh (see discussion in Section 5.1). Thus, strokes that should be visible may be moved slightly behind the mesh and thus be improperly occluded and strokes that should be invisible may be moved enough so that they are seen. This effect is negligible for dense meshes, but is increasingly noticeable for coarser meshes (Section 5.1). This is why portions of the silhouette are missing in Figures 3.11 (middle) and 5.4. Some strokes from the local filters in Figure 5.6 (*bottom-left*) are also improperly handled at the cat's paws and ear.

A simple solution for this problem is to displace the strokes slightly towards the eye. This approach has been used for all of the result images in Chapter 5. Unfortunately, depending on the geometry of the mesh, this approach might not work for all parts of the stroke (Figures 3.11(middle), 5.6, 5.4). Specifically, if multiple silhouettes exists at very

close depths, this method may incorrectly display occluded silhouettes. Furthermore, if the mesh has many sharp features, this might not reveal the complete stroke in certain places. In these situations, a stronger form of HLR is required, as is presented in the next section.

3.4.2 Image Space HLR

Hidden line removal can also be accomplished with an image-space approach. In this approach, the assentation is that if a raw, unprocessed edge[1] is visible, then its corresponding processed edge (or edges, if the processed silhouette chain is at a higher level of detail than the unprocessed chain) is visible. To determine if an unprocessed silhouette edge is visible, an *ID buffer* is used [NM00]. **Unprocessed** silhouette edges are drawn in unique colours in the ID buffer along with a white version of the mesh to perform occlusion (Figure 3.11, left). Unique colours are used so that a list of visible unprocessed edges can be created by analyzing each pixel of the ID buffer. Once the system has built this list, it draws the following processed edges generated by the MAR approach:

- processed edges whose corresponding unprocessed edge was found to be visible in the ID buffer
- r non-visible edges between two visible edges; where r is related to the size of the errors; In the Silhouette Stroke Extraction System, $r = 2$ is used.

The extra r edges are drawn because the MAR approach changes vertex positions in the chain; therefore, small groups of invisible edges before processing, usually those found at

[1] An unprocessed edge, is an unmodified silhouette edge extracted from the polygonal mesh before the MAR approach has been applied.

error positions, will become visible after MAR processing. These must be drawn so that small breaks do not appear in the stroke.

This method provides more accuracy than the object-space approach, however it is computationally more expensive. The extra computation time required is directly linked to the number of pixels that must be analyzed. This step takes on average an extra 90 milliseconds for an 800 by 800 pixel display.

Figure 3.11: *Left:* raw silhouette edges, each with a unique colour. *Middle:* results of object-space HLR on processed strokes. *Right:* results of image-space HLR on the processed strokes.

Chapter 4

The Interior Stroke Extraction System

The *Interior Stroke Extraction System* generates realistic short strokes in a technical pen-and-ink style on 3D meshes. The key goals of this system are good rendering rates, an efficient scheme for shape-measures calculation and visual quality resembling traditional precise pen stroke drawings (Figures 1.4, 1.5).

Typical NPR approaches to procedurally generate strokes as individual primitives and to place them directly on 3D meshes involve processes that can be costly. These include distributing strokes across the surface, evaluating hand-gestures functions (i.e. pressure, slanting, waviness), linking strokes into chains and fitting curves to stroke sequences, among others. The technique presented here does not follow any of these approaches and instead embodies the following three main strategies:

1. One stroke per mesh edge: Each stroke has the same length and location of its corresponding edge, and is modelled and rendered individually (i.e. no chaining). This strategy provides rendering at reasonable rates with temporal coherence, as the strokes are fixed to their edges on the model, and are not redistributed for each frame.

2. Edge-based shape measures: The method calculates shape measures at every mesh *edge*, using only information from its two adjacent faces. This is achieved by extending the Edge-Buffer data structure [BS00b] and by adapting shape measure calculation schemes from geomorphology [SSM02].

3. Pen stroke thickness and styles: The system automatically adjusts the thickness of each stroke as a function of surface curvature estimated at the edge; the user controls the

parameters of stroke style for placing different types of pen marks and for achieving ink distribution visual effects.

This chapter provides (4.1) a system overview, (4.2) modifications to the Edge-Buffer to aid in extracting interior strokes, (4.3) methods to extract interior marks using morphometric variables, (4.4) several methods to apply ink to the model based on the morphometric variables and other measures.

4.1 System Overview

Before the *Interior Stroke Extraction System* creates strokes, a modified Edge-Buffer data structure is constructed for a 3D mesh surface. This modified structure stores information on shape measures directly at each edge (Section 4.2). These measures are taken from adapted numerical techniques used in digital terrain analysis (geomorphology) (Section 4.3). At run-time, the Edge-Buffer is traversed, carrying user information on (1) which shape measures to display, (2) threshold values for the shape measures, and (3) parameters to adjust stroke style attributes. Each edge is then modelled and rendered as a single stroke, with a specific thickness and style (Section 4.4). Stroke thickness is automatically adjusted by the pre-computed surface curvature measure associated with the edge (Section 4.2). Stroke styles are provided by an interactive stroke model which reproduces traditional pen marks and visual effects of ink-distribution.

4.2 Edge-Buffer Modifications

A detailed review of the original Edge-Buffer system is presented in Section 3.1. The goal of *Interior Stroke Extraction System* is to use shape measures calculated at every edge to

create stylized strokes. The measures provided are dihedral angle (D), slope steepness (GA), slope aspect ($A0$), and mean curvature (H), which are described in Section 4.3. These are calculated using data from the faces adjacent to each edge. To compute these measures efficiently, they are stored in each node in the Edge-Buffer system (along with the contents of Figure 3.1c) and their calculation is performed as a pre-process. In this way, they can be read quickly during run-time directly from the Edge-Buffer without requiring any extra calculation. The *Insert-Edge* function, executed during Edge-Buffer initialization, is modified from the original Edge-Buffer approach to calculate the shape-measures data during initialization:

INSERT-EDGE(a, b, f)

1 $v \leftarrow V[a].SearchForVertex(b)$
2 **if** $v = NULL$
3 **then** *construct new v with* $(j, f_1) = (b, f)$
4 $V[a].Insert(v)$
5 **else** $b = v.j,\ F_1 = v.f_1,\ F_2 = f$
6 *call* $ShapeMeasures(a, b, F_1, F_2)$

In line 2 of $InsertEdge()$, the edge *ab* is visited for the first time, so the system stores the index of the face f. For the second visit (line 5) the system retrieves the face id stored at the first visit and, together with the current face id, computes the shape measures associated with edge *ab* (Section 4.3).

4.3 Edge-Based Shape Measures

This section describes the computations involved in function *ShapeMeasures()* (called from *InsertEdge()*, line 6), where shape measures are calculated directly at every edge *ab*, using only information from its two adjacent faces F_1 and F_2. One common measure in NPR is the **dihedral angle** D between the normals n of faces F_1 and F_2, defined as $D = \arccos(n_{F_1} \cdot n_{F_2})$. This measure depicts creases, which are edges whose dihedral angle is within threshold limits (min, max) specified by the user, assuming that $0 \leq min \leq D \leq max \leq 180$ (Figure 4.3(a)).

Unfortunately, creases alone do not distinguish 3D shape features. In addition to dihedral angle, a more general scheme to allow the calculation of a larger collection of shape measures is required to reveal more shape features. The approach used in this research is to investigate shape measure methods from geomorphology (the study of landforms and the processes that produce them), in particular techniques for characterizing form (geomorphometry) by means of morphometric variables [Eva72, MH93, SSM02, Woo96].

4.3.1 Morphometric Variables with Triangular Meshes

Surface geomorphometry is most commonly modelled as Digital Elevation Models (DEMs), which are defined as collections of elevation points sampled above some datum describing a terrain surface. Elevation coordinates are usually organized in a regular grid with points equally spaced in X and Y regardless of the shape of the terrain. Parameters characterizing the terrain surface, known as Morphometric Variables (MVs), are then produced from the DEMs. One common aspect of MVs is that they can be calculated using methods based on the approximation of differential operators by finite differences, expressed via first and

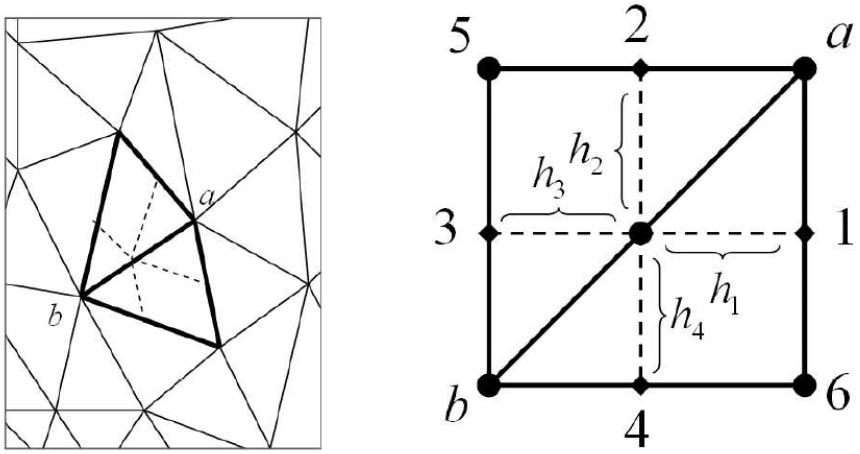


Figure 4.1: Discrete diagram for a 9-point finite difference scheme for numerical differentiation (*right*), constructed from pair of mesh faces (F_1, F_2), shared by edge *ab* (*left*).

second derivatives computed at every point of the DEM. Moving and evaluating a finite difference template along the regular DEM computes these derivatives. This template is derived from Taylor series expansion using a 3 by 3 grid [LP82].

For the system presented here, geomorphometrics that can be evaluated locally in a very small neighborhood are used, since they can be applied to arbitrary surfaces. The geomorphometry approach for calculating shape measures has been adapted using the local-coordinates (x, y, z) of the mesh models as points in DEMs, with the *z* coordinates of the vertices being the elevations and the (x, y) coordinates being the ground coordinates. This method is good for shape-measures calculation because it creates view-independent measures (since local-coordinates are used) and it provides several artistic effects explored in Section 4.3.2.

When the Edge-Buffer is initialized, each pair of mesh faces (F_1, F_2) defines four elevation points, from which a 9-point finite difference scheme for numerical differentiation is constructed. The discrete diagram for this situation is illustrated in Figure 4.1(right):

points $\{a,5,b,6\}$ correspond to the local-coordinates of F_1 and F_2 combined, and middle-points $\{0,1,2,3,4\}$ are computed along edges $\{ab,6a,a5,5b,b6\}$ respectively. Note that because polygonal meshes may have non-uniform edge lengths, the 3x3 template conforms to the general case where each spacing along x (h_1,h_3) and y (h_2,h_4) can be different [LP82]. The system can now estimate first and second partial derivatives of elevation z by plane coordinates x and y at point 0, the middle-point of the edge ab under consideration. The value of the partial derivatives f at location 0 is then formulated using central difference with variable steps. The notation $z_{\#}$ means "elevation z at point #" and $h_{\#}$ means "distance h from point 0 to #" (refer to Figure 4.1):

$$(f_{0,x}, f_{0,y}) = \left(\frac{\partial z}{\partial x}, \frac{\partial z}{\partial y}\right)_0 = \left(\frac{z_1 - z_3}{h_1 + h_3}, \frac{z_2 - z_4}{h_2 + h_4}\right) \tag{4.1}$$

$$f_{0,xx} = \left(\frac{\partial^2 z}{\partial x^2}\right)_0 = 2\left(\frac{h_1 z_3 - (h_3 + h_1) z_0 + h_3 z_1}{h_3 h_1 (h_3 + h_1)}\right) \tag{4.2}$$

$$f_{0,yy} = \left(\frac{\partial^2 z}{\partial y^2}\right)_0 = 2\left(\frac{h_2 z_4 - (h_2 + h_4) z_0 + h_4 z_2}{h_2 h_4 (h_2 + h_4)}\right) \tag{4.3}$$

$$f_{0,xy} = \left(\frac{\partial^2 z}{\partial x \partial y}\right)_0 = \frac{(z_a + z_b) - (z_5 + z_6)}{(h_3 + h_1)(h_2 + h_4)} \tag{4.4}$$

4.3.2 Using Morphometric Variables for Ink Placement

The next step is to decide which MVs best represent shape measures mostly used in the production of precise drawings. In geomorphology, various measures are employed to highlight certain features of the terrain surface [SSM02]. Some of these measures provide

Figure 4.2: Shaded relief images of *Venus* model (5,584 faces) with (1) lighter yellow referring to greater slope steepness, and with (2) darker blue and red referring to regions of greater convexity and concavity, respectively.

local shape measures such as slope, aspect, profile curvature, plane curvature, tangential curvature, flow path length, among others. This step considers the fundamental shape measures used by illustrators for highlighting structures of the subject, which include slant variations (in length and direction) and convex/concave formations across the shape of the subject [Cra00, Raw87], which correspond to elliptical surfaces. Mathematically, surface geometry can be classified into six categories defined by the values of the principal curvatures: convex and concave elliptical (*convex* when both curvatures are positive or *concave* when both curvatures are negative), hyperbolic (convex in one direction and concave in the other), flat (zero curvature in both directions), and convex and concave cylindrical (with zero curvature in one direction, *convex* if other curvature is positive or *concave* if other curvature is negative).

Illustrators appear to be most interested in revealing elliptical surface patches [Whi94,

(a) Dihedral Angle D(min, max)=(20,140) *(b) Slope Steepness GA(0.48,1.0)* *(c) Slope Aspect A0(0.63,1.0)* *(d) Mean Curvature (+) +H(0.54,1.0)* *(e) Mean Curvature (-) -H(-0.5,-0.01)*

Figure 4.3: Feature edges displaying thresholded shape measures within (min,max) range.

Cor56, Hod89, Sim92]. After studying complete systems of MVs [SSM02], I selected several measures that can be used to simulate fundamental methods artists use in precise illustration. *Slant variations* can be adequately represented by MVs of slope steepness and aspect. *Convex and concave regions* can be properly depicted with mean curvature measures. These MVs are calculated at the middle point 0 of the edge *ab* (Figure 4.1) as described next.

Slope Steepness (GA) is a measure of the rate of change of elevation. Geometrically, it is an angle ($0 \leq GA \leq \frac{\Pi}{2}$) between a horizontal plane and a tangential to land surface plane at the same point. This angle can be represented through first partial derivatives using the formula [Krc73]:

$$GA = \arctan(f_{0,x}^2 + f_{0,y}^2)^{0.5} \tag{4.5}$$

At large scales, slope steepness can be used as an isotropic variant of *shaded relief* map image, which is used to highlight structure within a DEM [MH93, SSM02]. Illustrators also use this measure for highlighting steep variations in the subject (Figures 1.4, 1.5).

Figures 4.2 (in yellow) and 4.3(b) illustrate this effect as computed by the *Interior Stroke Extraction System*.

Slope Aspect (A0) is a measure of the direction that a slope faces. It identifies the steepest downslope direction at a location on a surface. Geometrically, it is the angular distance (counted clockwise) between the directions of a fixed point and the slope. This angle ($-\frac{\Pi}{2} \leq A0 \leq \frac{\Pi}{2}$) can be represented through first partial derivatives using the formula [MH93]:

$$A0 = \arctan(f_{0,y}/f_{0,x}) \tag{4.6}$$

Artistically, slope aspect is useful for indicating hatching marks as clusters of parallel lines following a particular drawing direction (Figure 4.3(c)).

Mean Curvature (H) The mean curvature of a surface at a point is one half the sum of the principal curvatures at that point. Geomorphologically, negative H values describe mean-concave land forms, while positive H values refer to mean-convex ones. Technically, it can be represented through first, second partial derivatives using the formula [MH93, Gra97, SSM02]:

$$H = -\frac{(1+f_{0,y}^2)\,f_{0,xx} - 2\,f_{0,x}\,f_{0,y}\,f_{0,xy} + (1+f_{0,x}^2)\,f_{0,yy}}{2(1+f_{0,x}^2+f_{0,y}^2)^{3/2}} \tag{4.7}$$

This formula provides good discrete approximations of mean curvature, allowing proper detection of convex and concave formations across the mesh (Figures 4.2 and 4.3, *d and e*).

4.4 Stylizing the Interior Strokes

Figure 4.3(a-e) illustrates the case in which thresholded feature edges are rendered as single thickness lines, revealing specific shape measures. In traditional production of precise drawings, this corresponds to the stage at which the illustrator has accurately measured and lightly delineated the shape characteristics across the surface of the subject (Chapter 1). The 3D perception of the shape measures can be improved by adjusting the thickness of the strokes and by rendering them with different marking styles.

4.4.1 Thickness Adjustment

The system creates a quad for each edge to be displayed by extruding its vertices (a,b) in the direction of their normals (n_a,n_b). This results in a new pair of vertices $(a',b') = (a+n_a\rho, b+n_b\rho)$ with the amount of extrusion given by ρ. Intuitively, this amount of extrusion corresponds to the amount of ink placed at the stroke (Figure 4.4 left). In traditional illustration, ink flow is precisely controlled to make very heavy to very fine lines in order to depict regions of high and low curvature, respectively [Cra00, Raw87]. The system therefore defines $\rho = H_{ab}\phi_{ab}$, where H_{ab} is the mean curvature estimated at edge ab (Equation 4.7), and ϕ_{ab} is a user-defined positive scaling factor, which gives further control over the stroke thickness adjustment process. Note that for values of $H_{ab} < 0$ (concave regions), $\rho_{ab} = H_{ab}(-1.0)\phi_{ab}$ to guarantee that the extrusion from (a,b) moves in the outward direction of the normals (n_a,n_b).

4.4.2 Ink Marking Styles

Two pen-and-ink styles are provided in the *Interior Stroke Extraction System*: filled and serrated. These are used to simulate ink-application by artists for precise pen-and-ink illustration. The filled style is useful to create short directional marks on the surface. They are implemented by simply rendering the stroke as the quad $\{a, b, b', a'\}$ in black (Figure 4.4, left). This quad is defined by the thickness adjustment described in the previous section. The serrated style is useful to simulate stippling and fur styles. It is modelled by distributing marks with different directions and lengths within the ribbon $\{a, b, a', b'\}$ (Figure 4.4, right). As shown in Figure 4.5 and in the algorithm below, a stroke s with vertices (p, q) is defined at each parametric distance t along edge ab. The algorithm for generating serrated strokes is as follows:

SERRATED-STROKE$(a, b, a', b', res, l_1, l_2, d_1, d_2)$

1 $res \leftarrow res\ (|ab|/l_{max})$
2 **for** $t \leftarrow 0$ *to* $1,\ step \leftarrow 1/res$
3 **do** $(\delta_1, \delta_2) \leftarrow (l_1/2res, l_2/2res)$
4 $(t_1, t_2) = (t - \delta_1,\ t + \delta_1)$
5 $(t_1', t_2') = (t - \delta_2,\ t + \delta_2)$
6 $v = a + unif(t_1, t_2)(b - a)$
7 $v' = a' + unif(t_1', t_2')(b' - a')$
8 $p = v + d_1(v' - v)$
9 $q = p + d_2(v' - v)$
10 $DrawLine(p, q)$

In *SerratedStroke()*, the variable *res* corresponds to the user-defined number of marks to be placed along edge *ab*. Line 1 adjusts *res* considering the length of edge *ab* and the maximum edge length l_{max} in the mesh (both lengths related to the current view). Thus,

the number of serrated marks that a short edge receives is less than that of a longer edge (Figure 4.6). Lines 3-5 compute how much stroke s (Figure 4.5) can vary to the left and right of t, with $(l_1, l_2) \in [0,1]$. Lines 6-7 compute the stroke coordinates (v, v'). The function $unif(i,j) = i + (j-i)rand()$ returns a uniformly distributed real number between i and j, $(i < j)$, and $rand()$ returns a pseudo-random real number uniformly distributed between 0 and 1. Note that v and v' can be anywhere along (t_1, t_2) and (t'_1, t'_2), respectively, thus resulting in orthogonal or diagonal strokes along ab. Finally, lines 8-9 adjusts the length of s, with $0 \leq d_1 < d_2 \leq 1$.

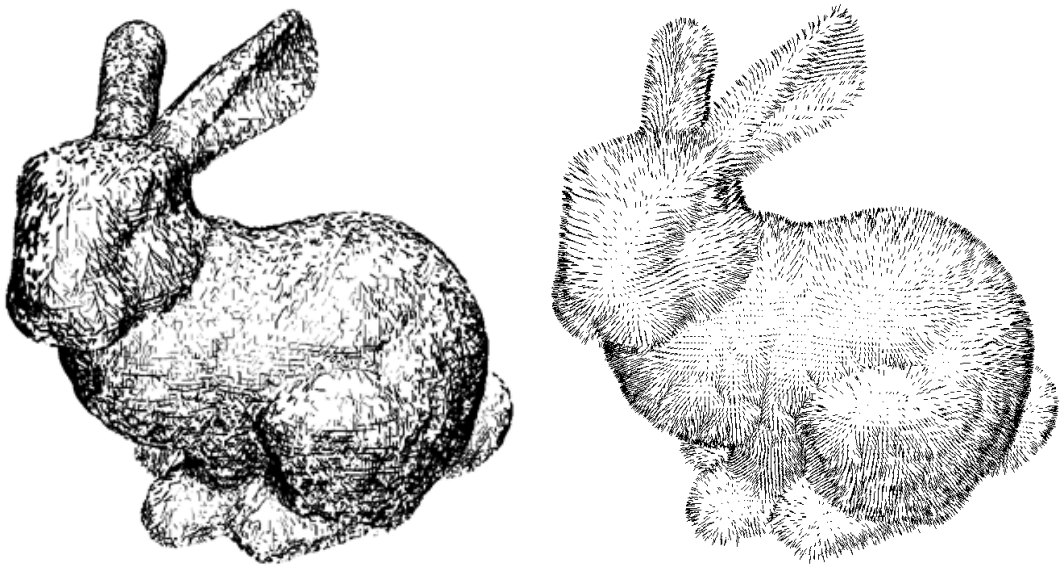


Figure 4.4: **Stanford bunny** (built from laser scans), with slope steepness $GA(0.81, 1.0)$ and positive mean curvature $+H(0.57, 1.0)$, rendered with filled (*left*) and serrated (*right*) pen marks ($res = 15$, see Section 4.4.2). *Model source: Stanford University Computer Graphics Lab.*

Figure 4.4(right), shows an example of serrated marks. Notice that the edges $a'b'$ and ab are not displayed, only the serrated marks s along them. Also notice the fur-like effect on the bunny due to a reduction on the amount of marks (variable *res* in the above

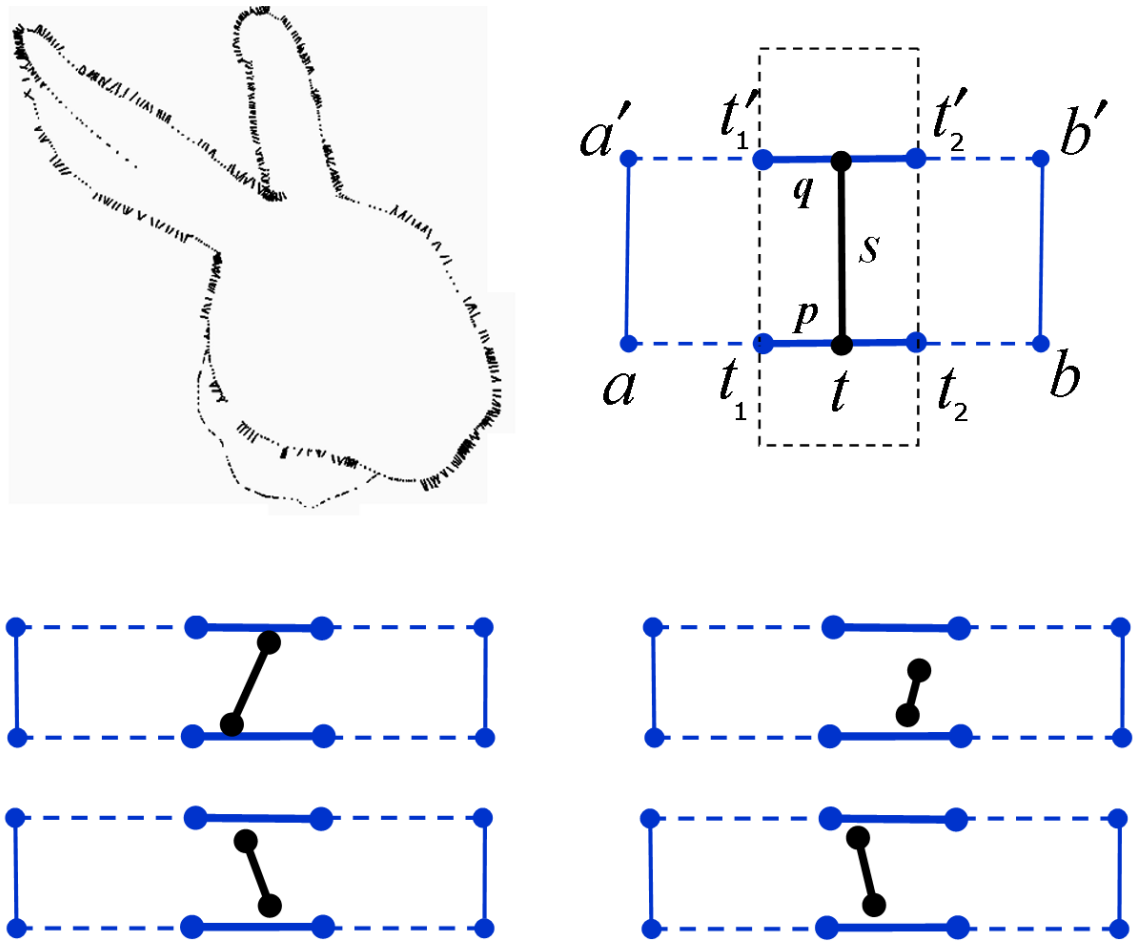


Figure 4.5: *Top-Left*: serrated marks placed at mesh edges. *Top-Right*: A single mark *s*, defined by vertices(p,q), can vary in location and length within (t_1,t_2,t_1',t_2'). *Bottom*: four examples of possible serrated marks in the (t_1,t_2,t_1',t_2') area.

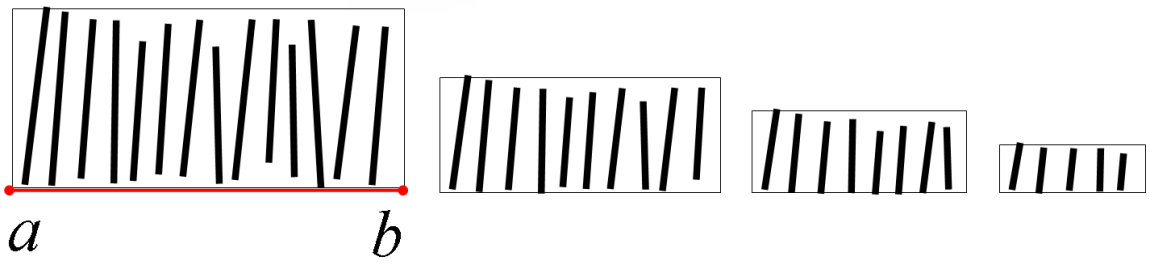


Figure 4.6: The number of (p,q) strokes in each (a,a',b',b) area is scaled based on the length of edge *ab*.

algorithm) and to an increase in the thickness of the strokes by adjusting variable ϕ_{ab} (Section 4.4.1).

4.4.3 Ink Distribution Effects

Ink distribution is an intuitive way of referring to stroke thickness distribution across the mesh. As described in Section 4.4, the thickness of every front-facing interior edge *ab* is determined by extruding *ab* in world-space by some ρ amount in (n_a, n_b). This results in larger perceived stroke thickness as (n_a, n_b) becomes more orthogonal to the view vector. Different visual effects can be achieved by simply adjusting the amount of extrusion ρ. The *Interior Stroke Extraction System* implements two effects dependent on the parametric view depth distance d_{ab} of each edge *ab*, given by:

$$d_{ab} = 0.5(\ (|\vec{v}| + |\vec{w}| - 2z_{min})\ /\ (z_{max} - z_{min})\ ) \tag{4.8}$$

where $(\vec{v}, \vec{w}) = (a - eye, b - eye)$, and (z_{min}, z_{max}) are the minimum and maximum view depth distances, respectively. The **first effect** is given by adjusting $\rho_{ab} = \rho_{ab}(1.0 - d_{ab})$, which slightly increases the thickness of strokes as they get closer to the viewer. This results in a more balanced distribution of stroke thickness across the mesh (Figure 4.7). The **second effect** is given by $\rho_{ab} = \rho_{ab}(2.0 - d_{ab})$, where strokes farther from the viewer will have their thickness scaled down, resulting in better depth cues (Figure 4.8).

4.4.4 Rendering

The *Interior Stroke Extraction System* renders strokes using OpenGL calls. For thickness adjustment (Section 4.4.1) notice that (a', b') is defined in 3D, and then projected by

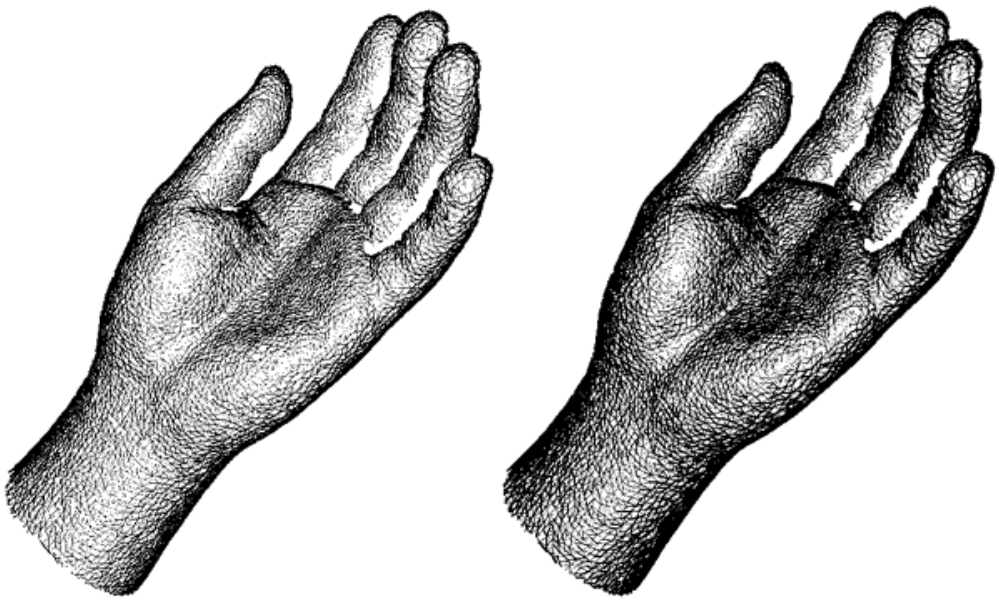


Figure 4.7: A visual effect of stroke thickness distribution: **Hand** with normal stroke thickness distribution (*left*) and with increased thickness of strokes closer to the viewer (*right*).

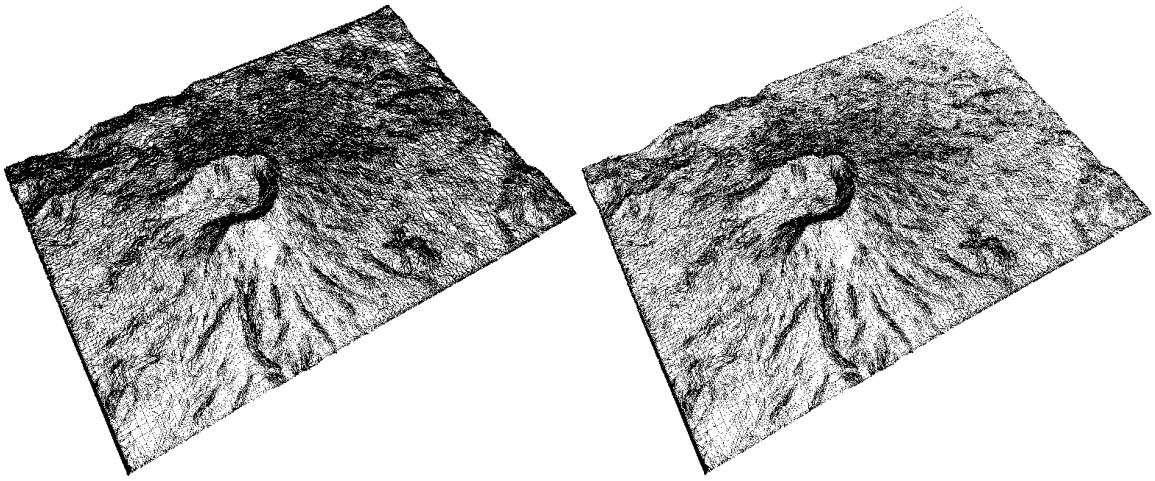


Figure 4.8: Another visual effect of stroke thickness distribution: **Mount St. Helens** with normal stroke thickness distribution (*left*) and with strokes farther from the viewer having their thickness scaled down (*right*). *Model provided by www.artcam.com*.

OpenGL. This means that thickness of a line in the image-space is defined by the projection of the vertex normal on the picture plane. To perform hidden line removal (HLR), the Z-buffer is used as described in Section 3.4.1: the polygonal mesh is first rendered in the background colour (white) and stroke ribbons are then rendered in black. Since Z-buffer computations are imprecise, some of the ink rendered from edges *ab* may get clipped or hidden by the mesh. The system solves this problem by translating the edges *ab* by a set small amount along the direction of the normals (n_a, n_b) prior to adjusting stroke thickness. Recall from Section 3.4.1 that, for the *Silhouette Stroke Extraction System*, this approach requires user-input and does not produce perfect results. This approach is suitable for the *Interior Stroke Extraction System* because the system accurately places ink directly at edges from the mesh. Thus, the mesh is an accurate gage of visibility for the strokes.

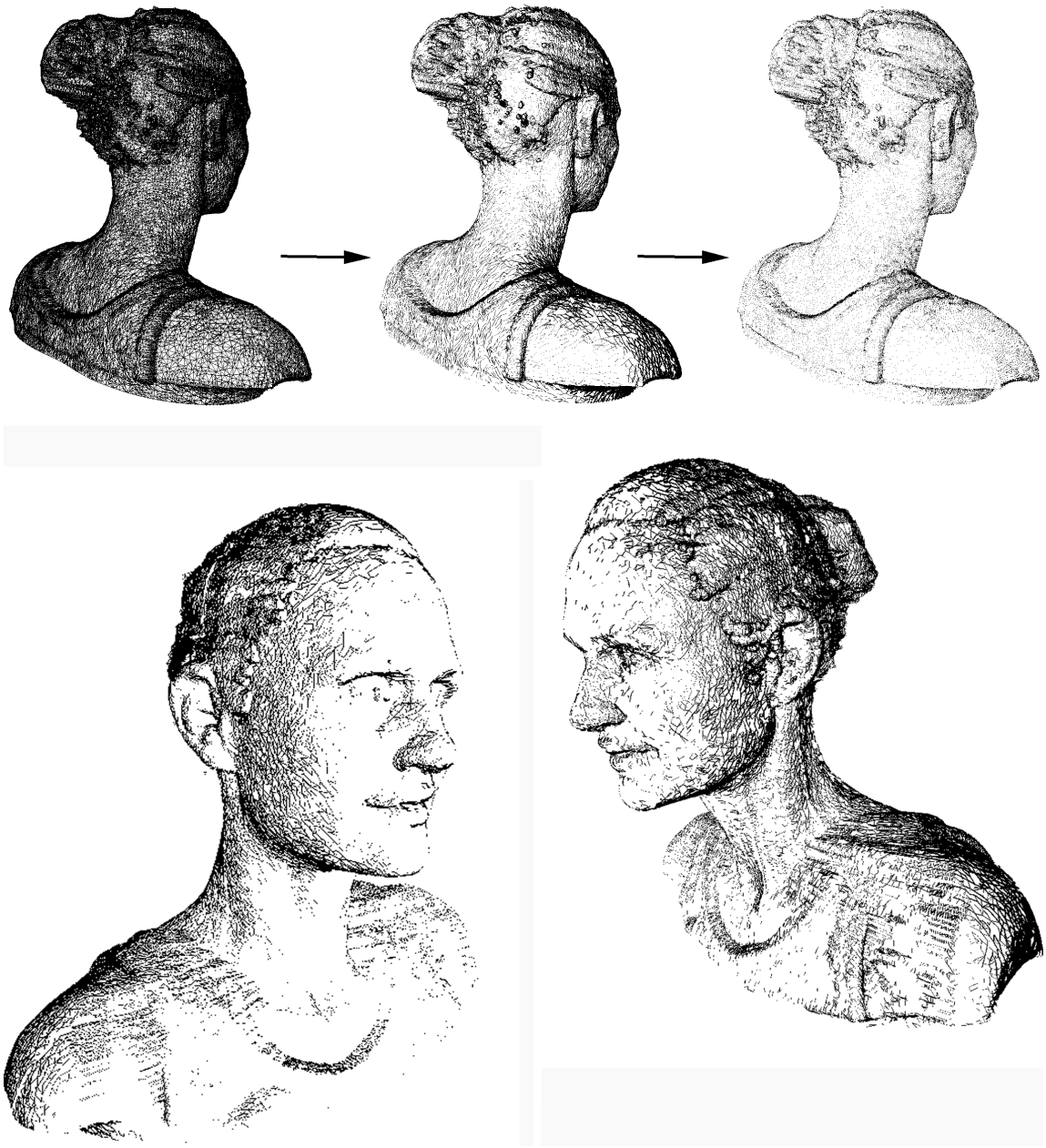


Figure 4.9: **Female head** (built from laser scans); Top-row, left*:* starting with the wire-frame mesh, filled strokes are placed at slope steepness $GA(0.7, 1.0)$, and positive mean curvature $+H(0.47, 1.0)$. Notice how various anatomic and hair features are revealed. Next, filled with serrated marks are used, resulting in a soft stippling effect. Bottom images*:* Side views of model rendered with filled strokes using the setup used for top-right image. *Model source: Cyberware.*

Chapter 5

Results

In this chapter, results and discussions are presented for the (5.1) *Silhouette Stroke Extraction System* and the (5.2) *Interior Stroke Extraction System*. The provided computation times were gathered using a 2.65 GHz Pentium 4 with OpenGL/ATI Radeon 9700 graphics and 1 gigabyte of RAM running Windows XP.

5.1 The Silhouette Stroke Extraction System

The Silhouette Stroke Extraction System's MAR process works effectively for most meshes and can generate error free strokes with minor user input. Results generated by the system are illustrated in Figures 5.2 to 5.8. The MAR process achieves fast computation rates including preprocessing (building the Edge-Buffer) and rendering (chaining, multiresolution filtering, and stroke stylization). Furthermore, the multiresolution methods employed [BS00a, SB99] operate quickly and can produce resolution-independent silhouettes.

MAR is more suitable for finer, denser meshes as it might remove important detail from meshes with a low polygon count. For these coarse meshes, MAR presents a tradeoff between feature-preservation and quality of filtering (directly controlled by the value e). It can sometimes be impossible to remove errors from silhouettes of simple meshes without losing stroke accuracy (Figure 5.4). A solution to this is to subdivide these meshes using a method such as Catmull-Clark or Doo-Sabin subdivision before extracting and correcting

Model	Figure	Polygons	Number Chains	Chain Length	Local Time (ms)	Global Time (ms)
Ox	5.4	652	11.4	16.3	0.414	1.55
Ape	3.8	1490	35.0	24.8	0.742	7.17
Beaver	3.9	2286	13.8	39.1	0.613	3.669
Face	5.7	2940	24.5	23.8	0.922	7.01
Kleopatra	5.8	4092	8.6	38.9	0.437	3.207
Cat	5.6	7819	41.7	27.9	2.241	39.84
Toutalis	5.2	12796	30.4	20.1	1.065	13.671
Inner ear	5.3	32702	99.4	27.1	9.589	155.73
Foot	5.5	46045	180.6	29	60.007	77.195

Table 5.1: Running times for error-correction for the meshes illustrated in this research. These results are plotted in Figure 5.1.

silhouettes.

Running times are provided in Table 5.1 for various polygonal meshes for the local and global multiresolution approaches, following with a discussion of the quality of the results with notes on mesh size, user input, the global and local approaches and the different filter types. Finally, the MAR approach is compared to other error removal approaches [CJTF98, HZ00, NM00, IHS02].

5.1.1 Timing

Table 5.1 provides running times for the system, when two levels of decomposition and reconstruction for local and global cubic B-Spline filters are used.

These results are averaged from 256 tests with silhouette chains extracted at random viewing directions and are plotted in Figure 5.1. These results illustrate that the MAR approach is efficient; meshes less than about 20000 faces usually run in real time and larger meshes, such as the foot (Figure. 5.5), run at interactive speeds. The speed of the multiresolution

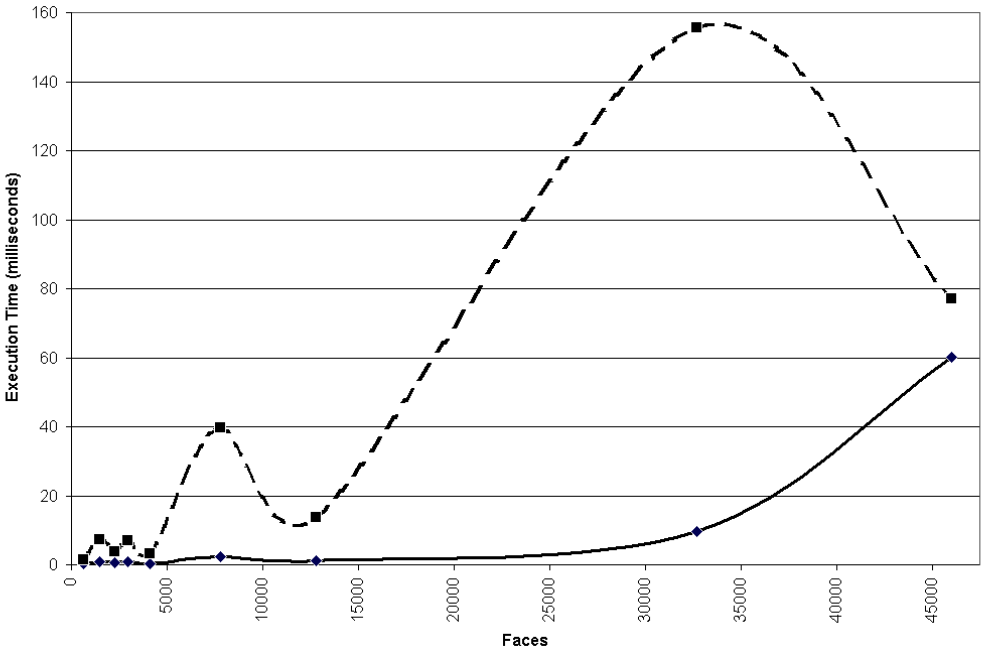


Figure 5.1: A plot of the data in Table 5.1. The solid line represents the average execution time for the local multiresolution approach. The dotted line represents the average execution time for the global approach. The expense of the global approach is always higher than the local approach, but appears to be highly variable with respect to the length and number of silhouettes from the mesh.

filters is determined by the number and size of the silhouettes extracted.

These results also reveal that the global approach takes more time to operate than the local approach. This is because global multiresolution requires solutions to Equations 3.4 and 3.5. Fortunately, the results for the global approach are still realtime or interactive for the small to medium-sized meshes of sizes up to 30,000 triangles displayed in Table 5.1. The added accuracy of global methods is not required for high resolution meshes (of size greater than about 10,000 faces) because the strokes from large meshes adhere well to model. This is due to the higher resolution and smaller average error size in the silhouette chains extracted from these meshes.

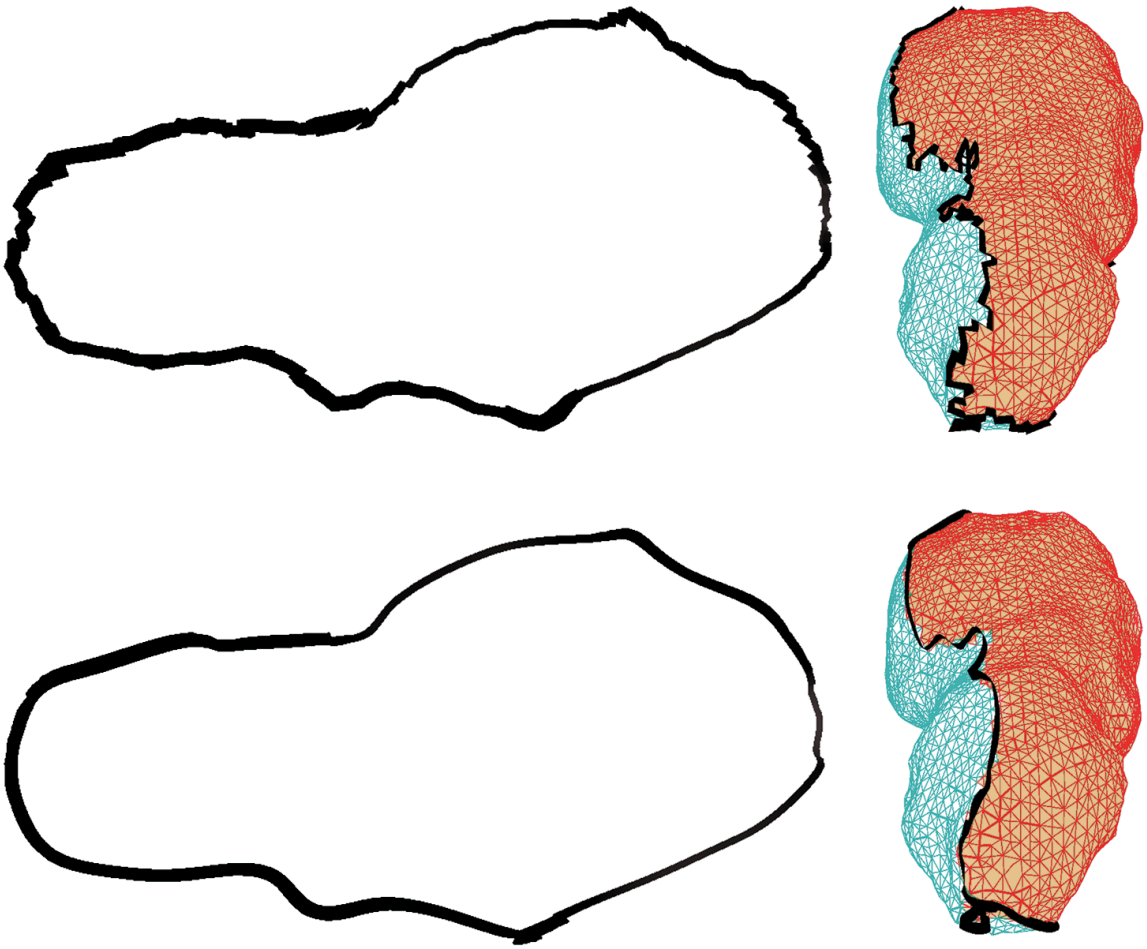


Figure 5.2: *Top*: Original silhouette from a model of the **Toutalis asteroid**. *Bottom*: Results after processing with the MAR system. In this session, two levels of global cubic B-Spline decomposition and reconstruction with $e = 0.1$ were used. Red shaded polygons are back-facing and blue polygons are front-facing.

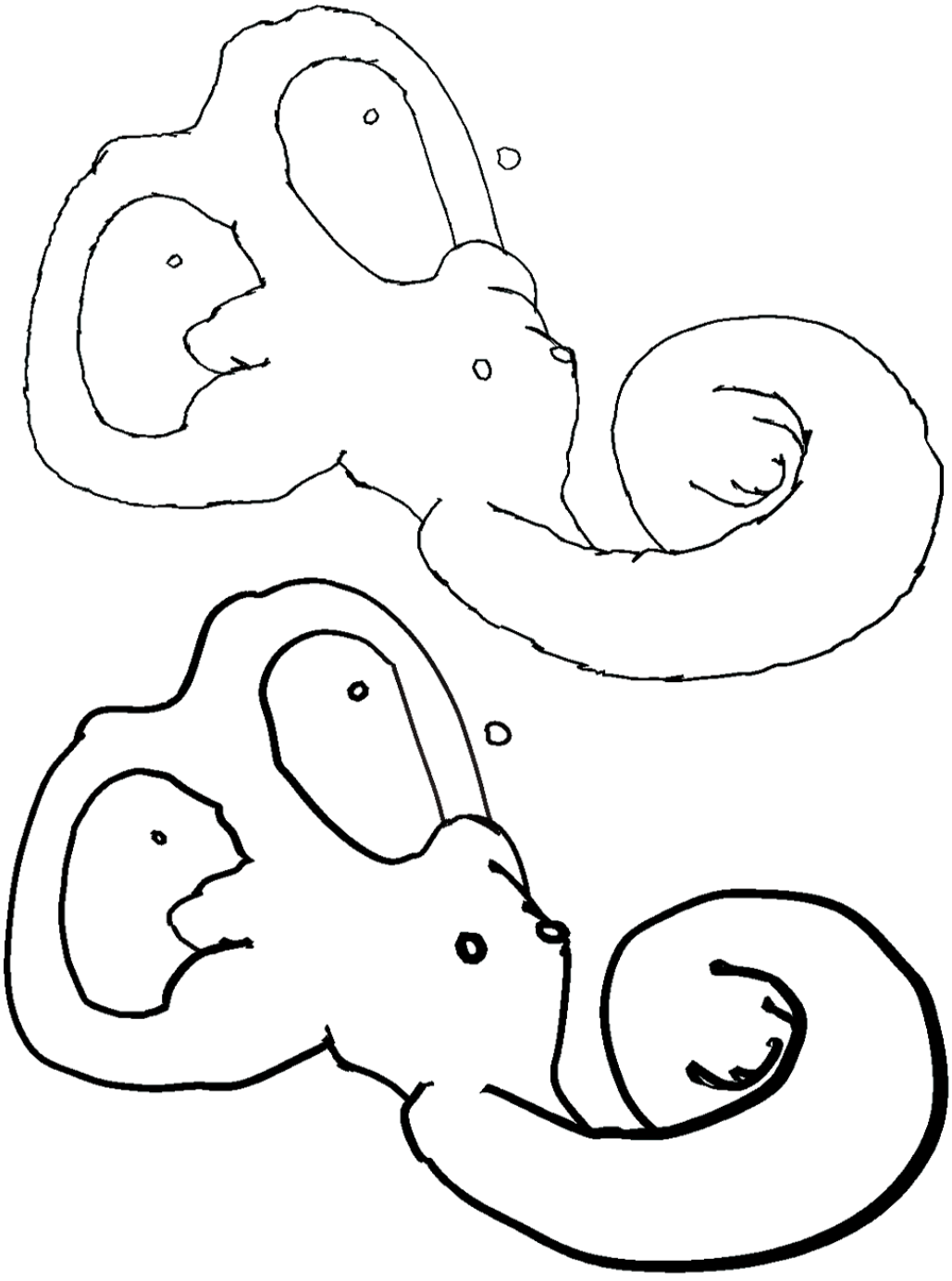


Figure 5.3: *Top*: Original silhouettes from an **inner-ear** model. *Bottom*: the results after processing two levels of decomposition and reconstruction with local cubic B-Spline filters and $e = 0.0$. Stroke thickness varies in this image as a function of depth.

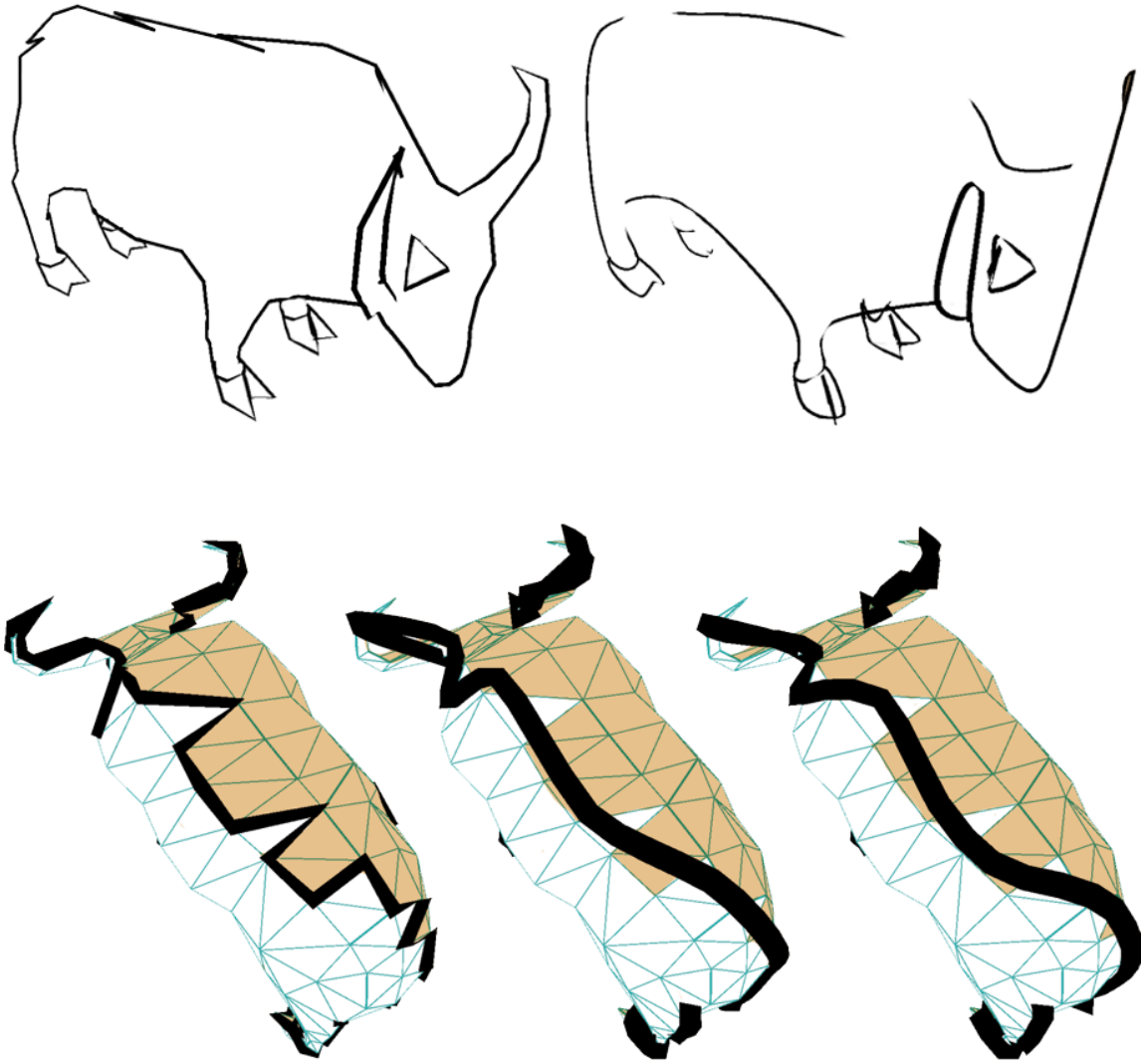


Figure 5.4: *Top-Left*: raw silhouette from a coarse **ox** mesh. *Top-Right*: processed strokes with global cubic B-Spline filters. *Bottom row*: alternate views of the silhouette with the original strokes (*left*), processed strokes with global cubic B-Spline filters (*middle*) and global Chaikin filters (*right*). In both cases, two levels of decomposition and reconstruction are used with $e = 0.35$. This is an example of the system producing poor accurate output: the corrected strokes do not adhere well to the original mesh due to the low resolution of the initial silhouettes.

5.1.2 User Input

Meshes with more than about 10000 faces require little or no user-input (Figures 5.2, 5.3 and 5.5). For these meshes, error free strokes with no accuracy loss can often be generated with local multiresolution using two levels of decomposition and reconstruction and some small *e* value for details. The more detailed the mesh, the smaller *e* can be while still maintaining accurate strokes. For results in this research, $e <= 0.1$ was employed for meshes larger than 10000 faces. To generate accurate strokes for smaller meshes (Figures 3.7, 5.4) or to accommodate small sharp features on larger meshes (those only defined by several triangles), the global multiresolution approach must be used (see next section for a detailed discussion) with precise input for the amount of details and the number of decomposition and reconstruction steps. It is in these situations that varying *e* can result in a noticeable tradeoff between error-removal and feature preservation. Accurate strokes can be generated for smaller meshes if the mesh is subdivided before extracting silhouettes; however, this requires a subdivision preprocess and will not generate an accurate error-free silhouette for the original mesh—the silhouette of the subdivided mesh will be corrected.

5.1.3 Global Multiresolution VS. Local Multiresolution

The cubic B-Spline, Dyn-Levin and Chaikin filter matrices (Figures 3.4-3.6) have been tested with local and global multiresolution methods.

Visually, the global method produces more accurate results, an effect most noticeable for meshes with a smaller number of faces and edges. Global results are directly compared to local results in Figures 5.6 and 5.8. Note in Figure 5.6 that although the local method (Figure 5.6 *bottom-left*) appears to solve most of the errors highlighted in the raw image (Figure 5.6 *top*), it loses accuracy in several areas, notable especially around the cat's

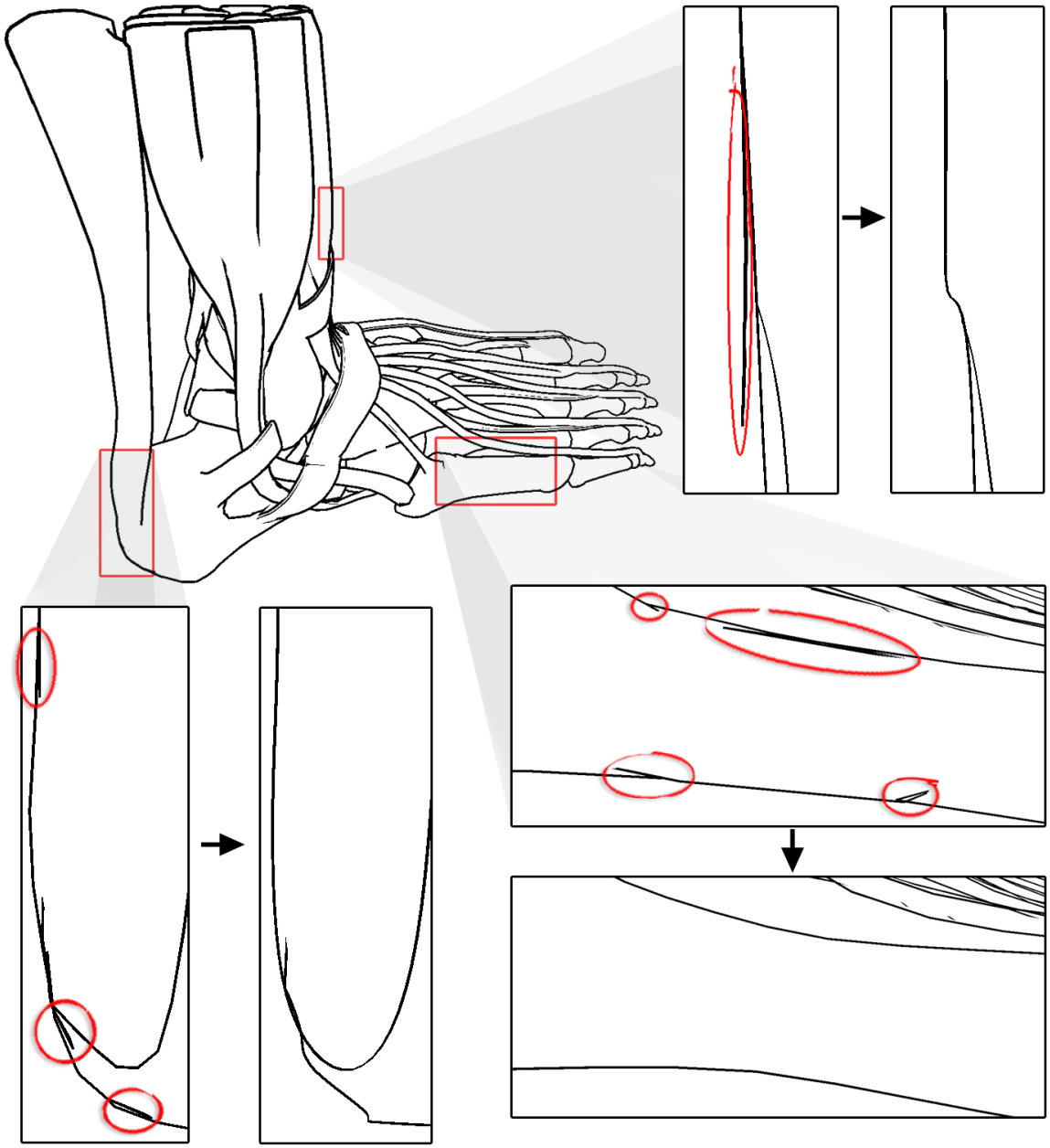


Figure 5.5: A detailed **foot** mesh. Removing silhouette errors on large meshes is more important when zooming in on the mesh. Errors are circled for three enlarged areas. These images use two levels of decomposition and reconstruction, $e = 0.2$ and local cubic B-Spline filters.

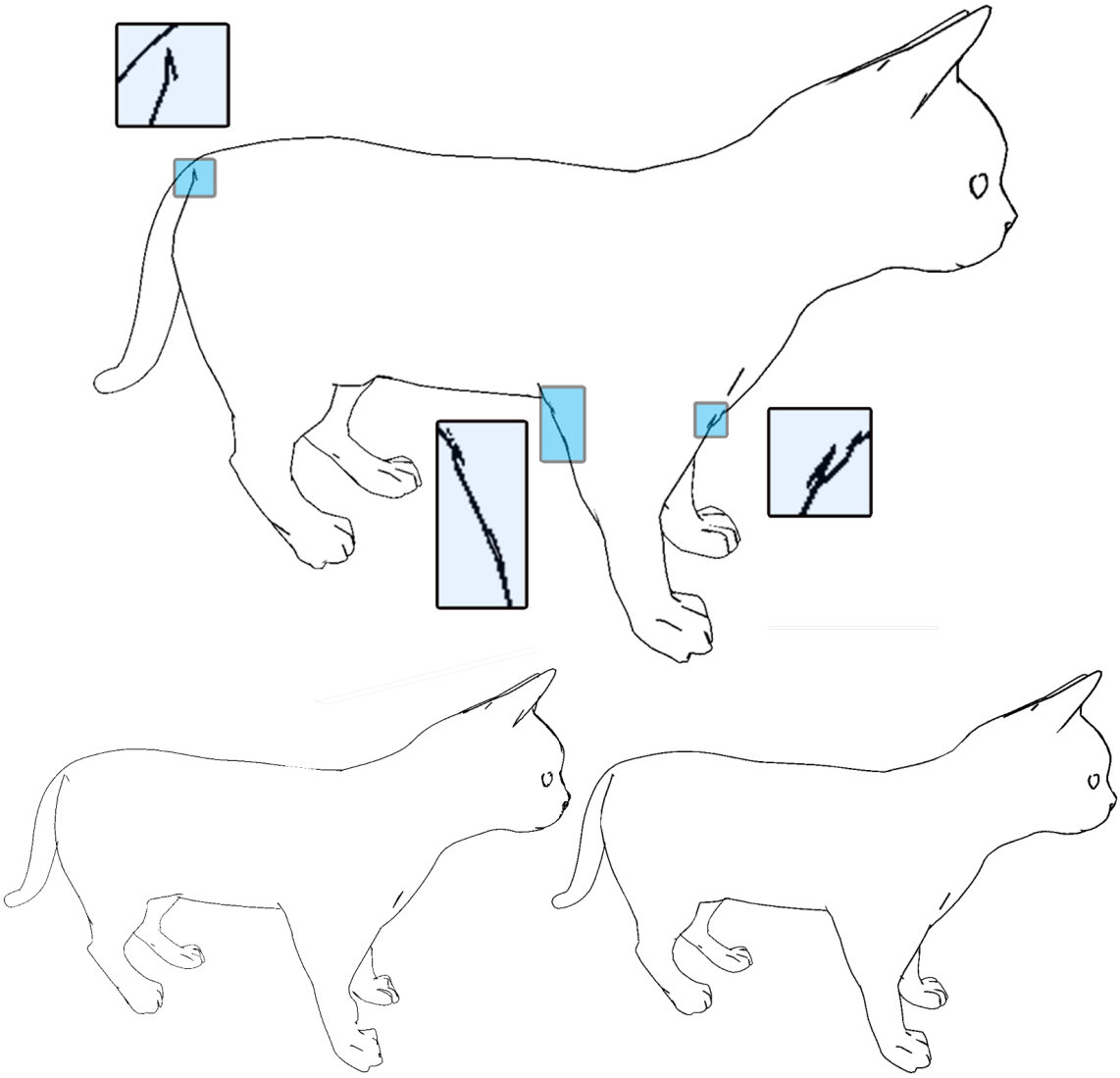


Figure 5.6: *Top*: Raw silhouettes extracted from a **cat** mesh. *Bottom-left*: local B-Spline subdivision with two steps of decomposition and reconstruction 30% details included. *Bottom-right*: global B-Spline with the same settings.

front paws and ear. This image was rendered using the object-space HLR approach and the loss of accuracy is the reason that some strokes improperly hidden and revealed. The global approach (Figure 5.6 *bottom-right*) maintains accuracy, removes all identified errors and is accurate enough so that the object-space HLR approach handles all parts of the silhouette curves properly. In Figure 5.8, local and global silhouettes for all three filters are shown at an alternate angle. In these images, the underlying mesh is also rendered with shaded back-facing polygons to reveal how accurate the processed silhouettes are. Also, Figure 5.8 displays local results (left-column) and global results (right-column). In the cubic B-Spline local example (top-left), the processed stroke loses the mesh significantly (most noticeable at the middle part of the silhouette). The cubic B-Spline global approach does not suffer from this loss of accuracy.

As presented in Section 5.1.1, the expense of global methods increases with the size and number of silhouettes. Fortunately, the visual accuracy improvement is usually only useful for smaller sized meshes (Figures 3.7, 3.8, 3.9, 5.4, 5.6) where the solution to the systems used in the global approach can be found quickly.

For the results illustrated in this research, local methods have been employed for Figures 3.8 (*left column*), 5.2, 5.3, 5.5, and 5.6 (*bottom-left*). Global methods have been employed for Figures 3.7, 3.8 (*right column*), 3.9, 5.4 and 5.6 (*bottom-right*). Figures 5.7 and 5.8 illustrate the results of executing both the local and global multiresolution methods for each filter implemented in the system.

5.1.4 Cubic B-Spline Subdivision VS. Dyn-Levin Interpolation VS. Chaikin Subdivision

There are three different sets of multiresolution filters [BS00a, SB99] implemented in the MAR process (providing the *A*,*B*,*P*,*Q* matrices, Section 3.3.2): cubic B-Spline subdivision, Dyn-Levin interpolation and Chaikin subdivision. Each of the filters produce slightly different results when used to remove errors from the silhouette curves. For large meshes with dense details, the different effects of the filters are not particularly apparent because the starting silhouette is very detailed and the errors are not large compared to the correct detail in the chain. Thus, it is difficult to see regions where the filters produce different results for detailed meshes, unless about three or more levels of decomposition are used. It is also difficult to classify the results of the filters for coarse meshes because, various input data can produce drastically different results. Despite this, it is important to understand the general differences between the filters to choose which filter to try first when correcting errors in silhouettes from coarse meshes and for large meshes when viewed closely.

The cubic B-Spline filters are based on a subdivision scheme and the points generated are C^2 continuous curves. This is the highest continuity of the three methods and makes appealing smoothed strokes. This continuity comes from the B-Spline filters' larger mask width. This larger width means that errors are removed quickly. However, this also means that the cubic B-Spline approach is more prone to inaccuracy over the other filters because it is a subdivision approach and the mask uses more influence from neighboring points (observe this in the local case for cubic B-Spline in Figure 5.8). Thus slightly higher values for *e* might be required with these filters. Figure 5.7 is provided to illustrate this effect. In this image, strokes generated with the cubic B-Spline filters (left column) give

the best impression of the face. The other two methods generate less appealing bumpy results, most noticeable in the error-filled temple areas.

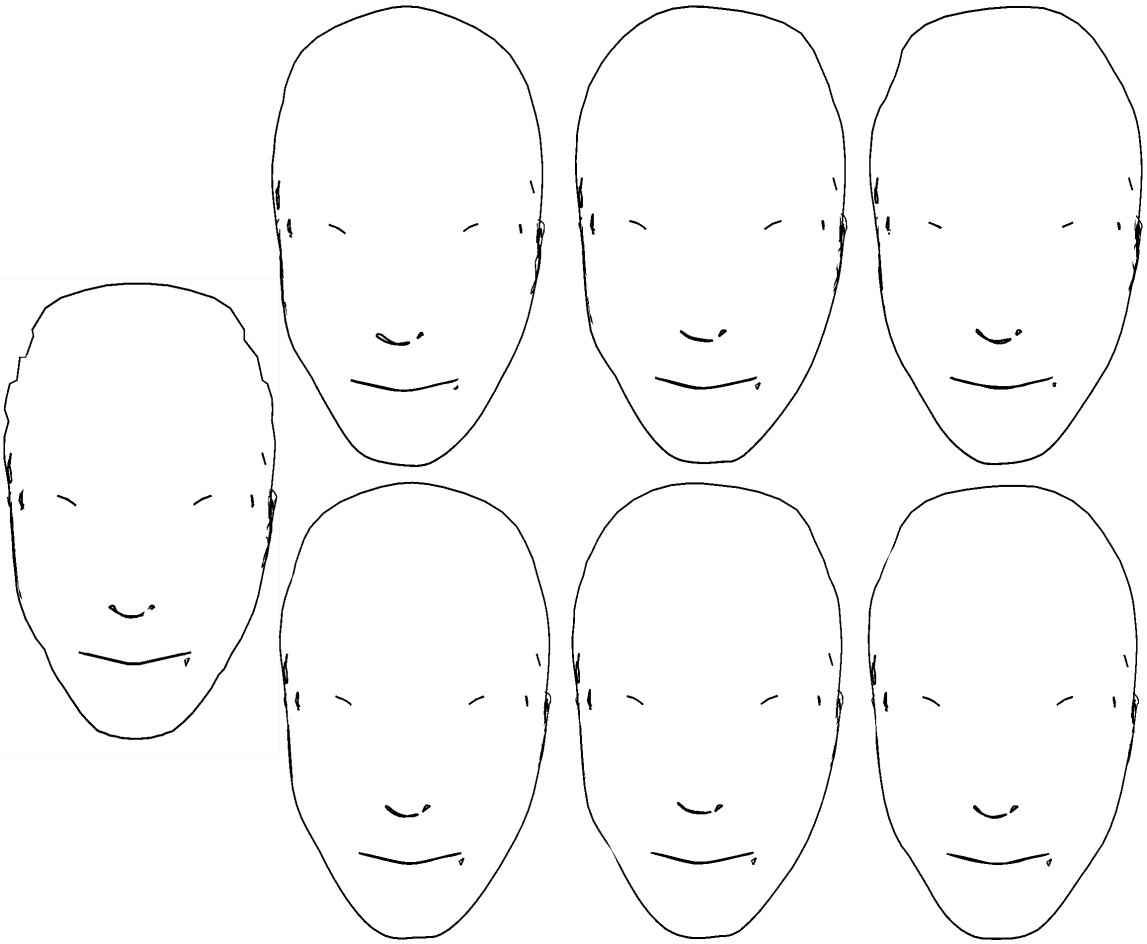


Figure 5.7: *Left*: Silhouette from a **head** mesh with several large errors at the temples and six results of running, from left to right, B-Spline, Dyn-Levin and Chaikin filters on the silhouettes using one level of decomposition and reconstruction with 20% details included. The top row uses local filters and the bottom row uses global filters.

The Dyn-Levin filters are created from a different subdivision method which is based on interpolation. Thus, strokes processed with this method adhere better to the original mesh than the B-Spline based methods. Although a larger mask width (see Samavati and Bartels [BS00a]) allows the Dyn-Levin approach to remove errors with about the same

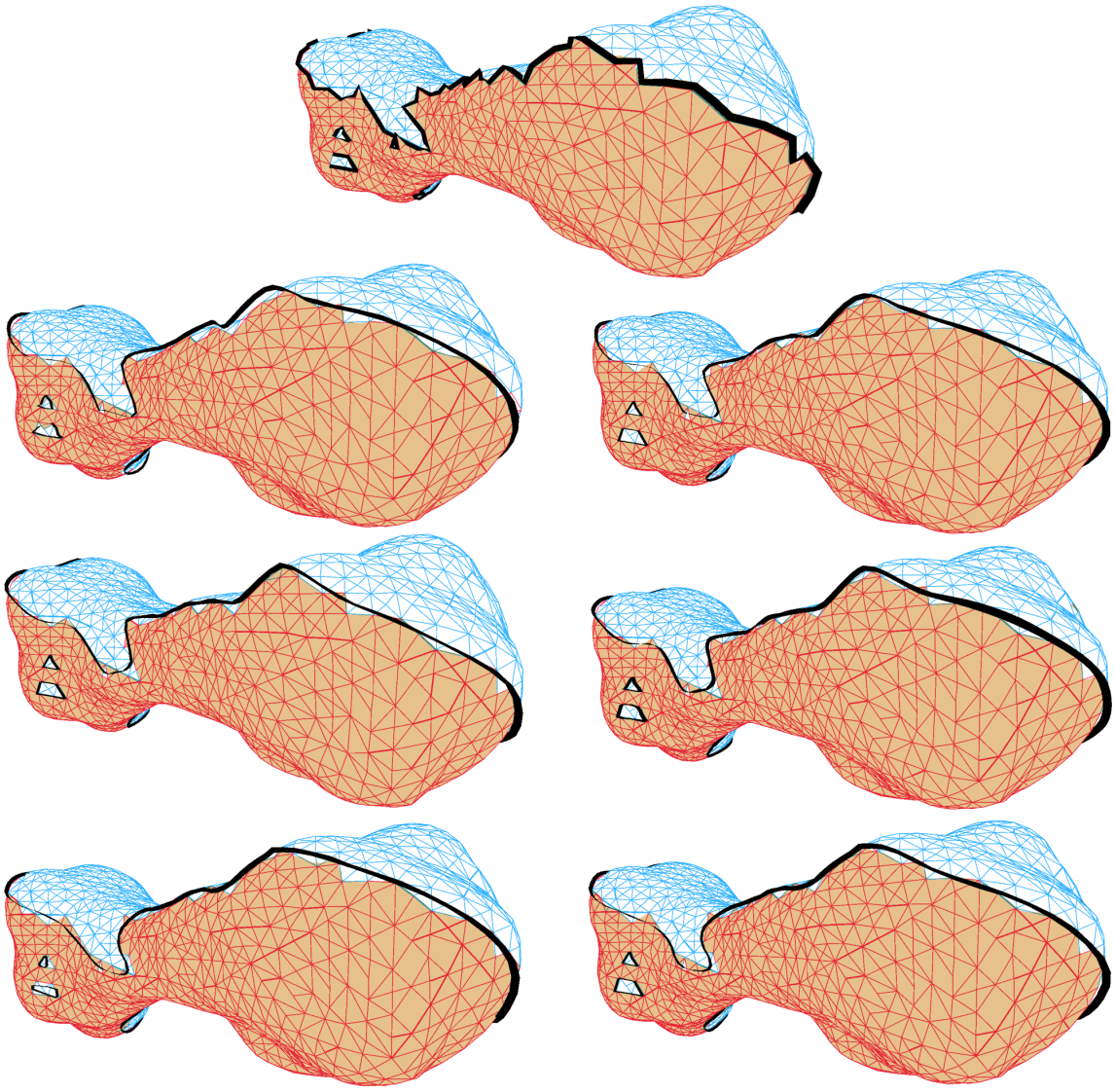


Figure 5.8: An analysis of the effects of various filters viewed from an angle alternate to that used to extract silhouettes from the **Kleopatra asteroid** model. *Top*: The raw silhouettes. Blue edges are front-facing and red-edges are back facing. *Left-column, from top to bottom*: Local B-Spline, Dyn-Levin and Chaikin approaches. *Right-column*: Global filters in the same order. In all images, $e = 0.0$ is used with two levels of decomposition and reconstruction.

settings as the cubic B-Spline approach, this method can exaggerate some features as a side effect of the interpolation. Observe this exaggeration in the middle column in Figure 5.7, where in the local case, the filter has pointed the head slightly in one direction and in the global case, the filter has distorted the right temple slightly. These filters should be used when accuracy of strokes is desired over quality of the final image.

The Chaikin subdivision filters are the fastest to execute due to the fact that their masks are the narrowest [BS00a]. The Chaikin filters provide a quadratic B-Spline subdivision which offers C^1 continuity. The result of using these filters on silhouette chains is somewhere between use of cubic B-Spline subdivision and Dyn-Levin. The processed strokes adhere better to the mesh than with cubic B-Spline, but not as accurately with as Dyn-Levin. This is visible in Figure 5.8 (left-column). It is important to note that in order to remove errors with the Chaikin filters, fewer details and sometimes an additional step of decomposition and reconstruction must be used in the MAR error-removal pipeline. Thus, after removing errors, strokes processed with Chaikin filters usually adhere *less* accurately to the mesh than with cubic B-Spline. These problems are due to the limited scope of the Chaikin mask. In Figure 5.7, all strokes are processed with one level of decomposition and $e = 0.2$. The strokes processed using the Chaikin filters (right-column) curve inwards and outwards strangely, still following the implying some of the "zig-zags" in the silhouette (this is most noticeable in the temples). Due to these problems, these filters are not recommended for use with this system when accuracy becomes important for low quality meshes.

For smaller meshes where accuracy becomes an issue, the cubic B-Spline filters should be used for the best looking strokes and the Dyn-Levin filters should be used for the most accurate strokes. When larger meshes are used, cubic B-Spline is recommended, unless

fast results are required. In this case, the Chaikin filters should be used because its smaller mask width lowers execution times.

5.1.5 Comparison to Previous Work

As detailed in Section 2.1.3, four methods have already been presented for silhouette error correction. Of these methods, there are two main approaches. Northrup and Markosian [NM00] and Isenberg et al. [IHS02] use the approach of correcting raw silhouette edges extracted from the polygonal mesh. Corrêa et al. [CJTF98] and Hertzmann and Zorin [HZ00] ignore raw silhouette edges from the mesh and to generate new, better edges to approximate the silhouette.

The MAR approach, like Isenberg et al. [IHS02] and Northrup and Markosian [NM00], corrects silhouette edges directly from the mesh. It offers an improvement over their approaches because it provides a general solution. In other words, since the method is evaluated evenly over the complete chain, no errors are missed. Isenberg et al. [IHS02] and Northrup and Markosian [NM00] require a series of error-cases and corresponding solutions which occasionally miss errors. Another important distinction between the systems is that the MAR system corrects errors in all strokes (Figure 5.9, middle), while Northrup and Markosian's [NM00] method and Isenberg et al. [IHS02] only correct visible strokes. This is useful when some sort of transparency stylization is desired, however it can mean processing many extra silhouettes for noisy meshes, such as those produced by range-scans (Figure 5.9). A drawback of the MAR approach to Isenberg et al. [IHS02] and Northrup and Markosian [NM00] is that it cannot generate accurate strokes for coarse meshes (Section 5.1.2, Figures 5.6, 5.4), while their approaches work better in this case.

Hertzmann and Zorin [HZ00] and Corrêa et al. [CJTF98] provide techniques that gen-

Figure 5.9: Silhouettes extracted from a range scan mesh of a skull. *Left:* with Hidden Line Removal (HLR). *Middle:* without HLR. *Right:* a side view of the silhouette edges extracted.

erate new, more suitable edges for silhouettes. Hertzmann and Zorin [HZ00] provide a more general version of these two methods which will now be discussed. In their approach, silhouettes are generated by estimating the exact position on the polygonal mesh that the silhouette would intersect if the polygonal mesh were a smooth surface. Both the MAR approach and Hertzmann and Zorin's approach generate sub-polygon silhouettes close to the "actual" silhouette for the polygonal mesh (if it was a smooth surface) and do not miss individual errors. Furthermore, both approaches have problems with coarse meshes. Hertzmann and Zorin's approach experiences problems performing hidden line removal for these meshes because their interpolated strokes might go over a back face and become invisible due to z-buffer occlusion. In the case of the MAR system, issues of stroke accuracy arise for coarse meshes (Section 5.1.2, Figures 5.6, 5.4). The primary difference between these two approaches is that Hertzmann and Zorin's approach generates edges exactly on the mesh while the MAR approach smoothes edges to various levels of accuracy controlled by the user.

The following scheme should be used to choose between the approaches. If completely accurate corrected silhouettes are desired, Hertzmann and Zorin's [HZ00] approach should be used because it is guaranteed to produce accurate error-corrected chains. If resolution independent strokes smoothed independent of mesh geometry are desired, the MAR approach should be used. The only case where this does not hold is for very coarse meshes where the approaches presented by Isenberg et al. [IHS02] or Northrup and Markosian [NM00] will produce the best results.

5.2 Interior Stroke Extraction System

The *Interior Stroke Extraction System* achieves fast computation rates including pre-processing (building the Edge-Buffer and calculating shape measures) and rendering (automatic stroke thickness adjustment and interactive pen marking). Figures 4.4 to 4.9 and 5.11 to 5.17 show results using the system. For many of these figures, the different shape measures selection, thresholds, and pen-marking styles are provided. For the user-defined extra thickness scale ϕ_{ab} (Section 4.4.3), values between 0.005 and 0.01 are used for all the results presented here.

The measured frame rate, shown in Table 5.2, provides the user with an acceptable level of interactivity for exploring and illustrating various shape measures on meshes. Preprocessing and rendering times increase linearly with mesh size with the exception of the preprocess for the Hammerstone which is very expensive, due to the operating system swapping data from virtual memory. For this reason, it is to be expected that meshes the size of the Hammerstone and larger will all experience this slowdown.

In the *Interior Stroke Extraction System,* the user is able to quickly select and thresh-

Model	△s	*edges*	*preproc.*	*render*
Hand	26,373	13,339	2	0.25
Mt St Helens	65,932	33,822	4	1
Bunny	69,451	36,742	4	1
Killeroo	92,092	46,351	5	1
Rihard Jakopič	108,507	56,610	6	1
Broken Adze	118,676	59,339	6	1
Female Head	154,650	78,560	8	1
Gargoyle	206,982	103,740	11	2
Hip	265,084	136,099	14	2
Igea Artifact	268,686	136,483	14	2
Fossil Skull	284,458	146,123	15	4
Preformed Adze	401,060	200,529	22	4
Hammerstone	725,828	362,915	235	7

Table 5.2: Average times (in seconds) for pre-processing and rendering the meshes presented in this research. These results are plotted in Figure 5.10.

old clusters of feature edges related to certain shape measures to adjust the parameters of stroke styles and thickness distribution effects. The system has effective temporal rendering coherence with only some temporal aliasing occurring at the silhouette edges as new strokes are added based on the silhouette extraction and automatic thickness adjustment. Three important points must be made about this system, concerning the input mesh, level-of-detail and stroke spacing and direction.

5.2.1 Input Mesh

Any mesh can serve as input for this system, provided that any non-triangle faces are converted to triangles before system execution. Unfortunately, two important points about mesh-type and the quality of the results must be made. First, the system only guarantees good strokes with dense meshes. Use of less detailed meshes can result in poorly placed

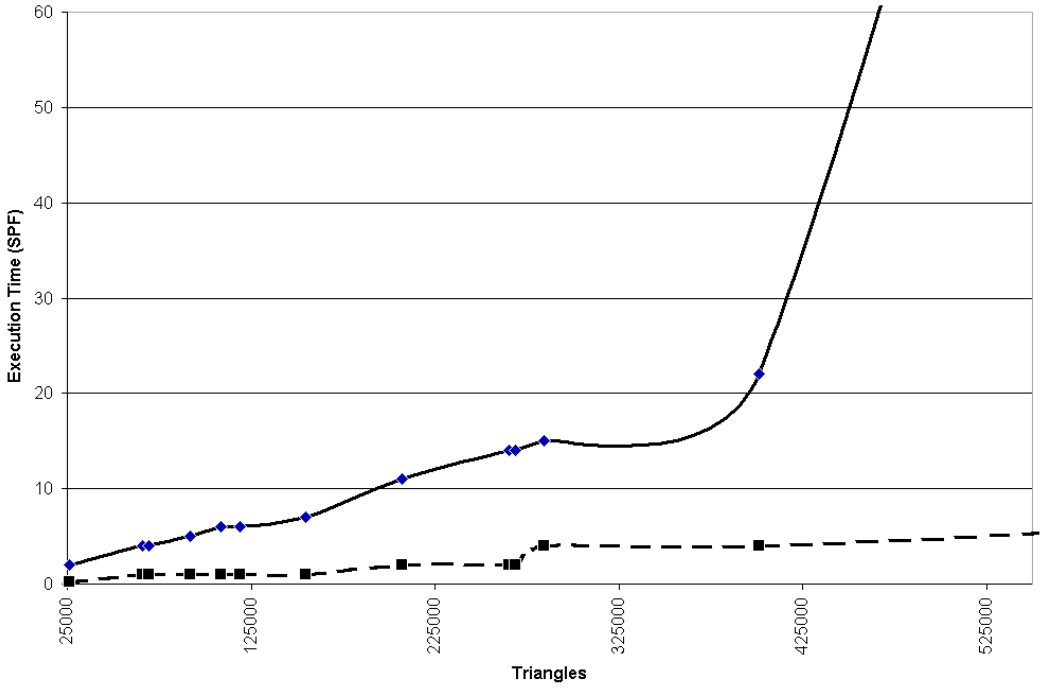
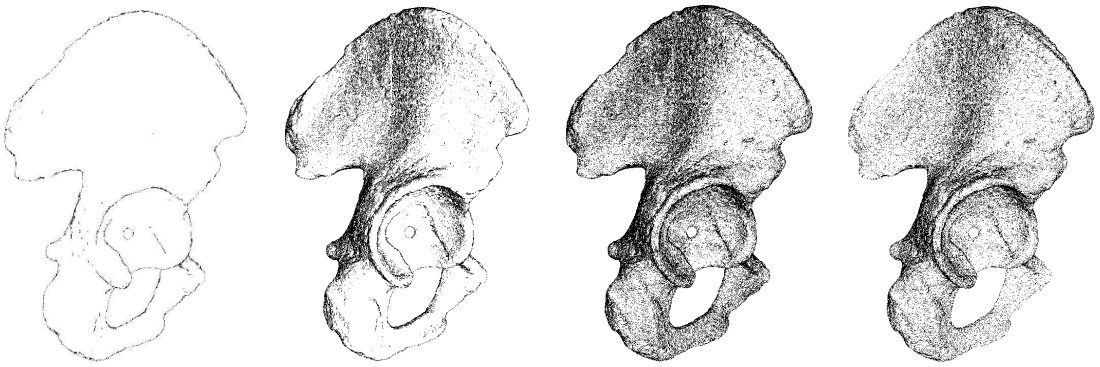


Figure 5.10: A plot of the data in Table 5.2. The dotted line represents the average times for rendering. The solid line represents the average pre-processing time required.

strokes that are distracting and do not properly convey the desired artistic effect. Observe this effect for the Venus model in Figure 5.18. Second, certain types of meshes containing many flat-surfaces (i.e. modern buildings), would either be under-tessellated or be composed of mostly regular 'quads', either of which could produce unwanted artifacts when placing the strokes. For instance, in Figure 5.19, Mount St. Helens includes many quads which are visible in the final rendering. Whereas these quads do not necessarily negatively affect the shape-revealing nature of ink for terrain features, they introduce artifacts. Applying a subdivision method to the surface before running the system can partly solve both of these problems. However, since subdivision methods tend to smooth the surface, the morphometric shape measures produce less pronounced differences. This results in less



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

Figure 5.11: Results of the *Interior Stroke Extraction System* for medical illustration of a **hip** model, built from laser scans: (*a*) first, silhouettes are rendered with varying thickness as a function of curvature; (*b*) concave formations are revealed by placing filled strokes in locations with negative mean curvature $-H(-0.459, -0.001)$; (*c*) creases are delineated with $D(155, 170)$, and convex formations are revealed by placing filled strokes in locations with positive mean curvature $+H(0.56, 1.0)$; (*d*) finally, the view-depth effect is applied to improve the depth perception. *Model source: Cyberware*.

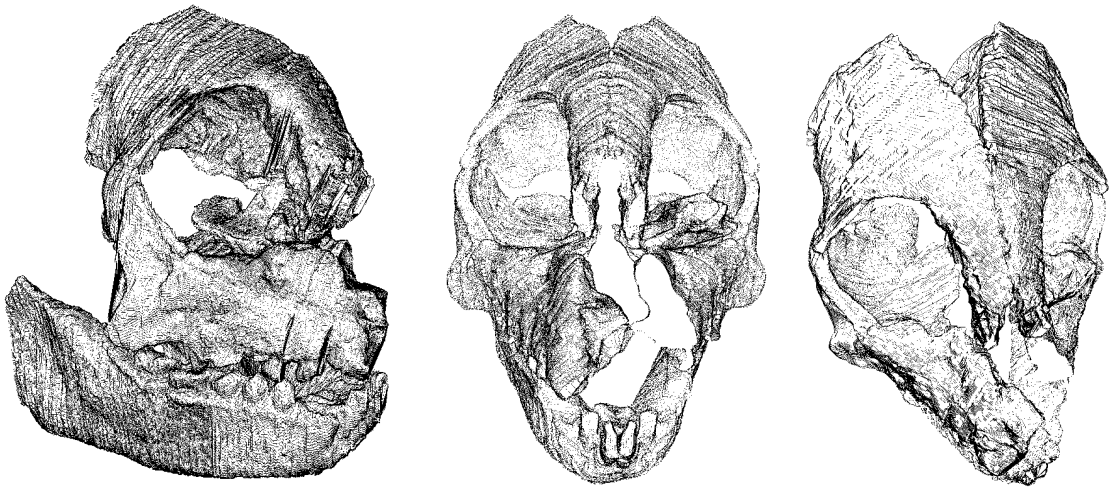
shape-revealing variation when applying ink.

5.2.2 Level-of-Detail

This method provides level of control when the object is viewed from far away and close-up. The user can adjust style parameters for pen marks (Section 4.4.2) and also select depth-dependent ink distribution effects (Section 4.4.3). It is important to consider other level-of-detail controls, including measures for omitting or adding edges while preserving local shape measures and rendering effects.

5.2.3 Stroke Spacing and Direction

A goal in producing this system has been to experiment with morphometric variables, with line direction given by slope aspect, which results in good shape revealing. Observe,



(a) (b) (c)

Figure 5.12: Results for an archeological illustration of primate model **Homunculus Patagonicus** built from laser scans; (*a*) evenly distributed filled strokes; (*b*) replacing filled with serrated marks and applying the view-depth effect. (*c*) placing filled marks to reveal slope steepness and serrated marks for slope aspect, keeping view-depth effect on. *Model source: Dr. A.L. Rosenberger, Smithsonian Institution BioVisualization Lab.*

for instance, that Figure 5.13 (the Artist's mask) has edges aligned along the direction of the nose. The "killeroo" (Figure 5.14), has lines following curvature on its back that reveal shape. The broken preformed adze of Figure 5.17 also has short directional strokes revealing the irregularity of rock formations. Despite these positive results, a limitation of this system is that it cannot create strokes in any direction at arbitrary position on the surface. A more controllable set of line directions schemes would improve visual results even further, including principle line directions and other types of line directions possibly found in geomorphology.

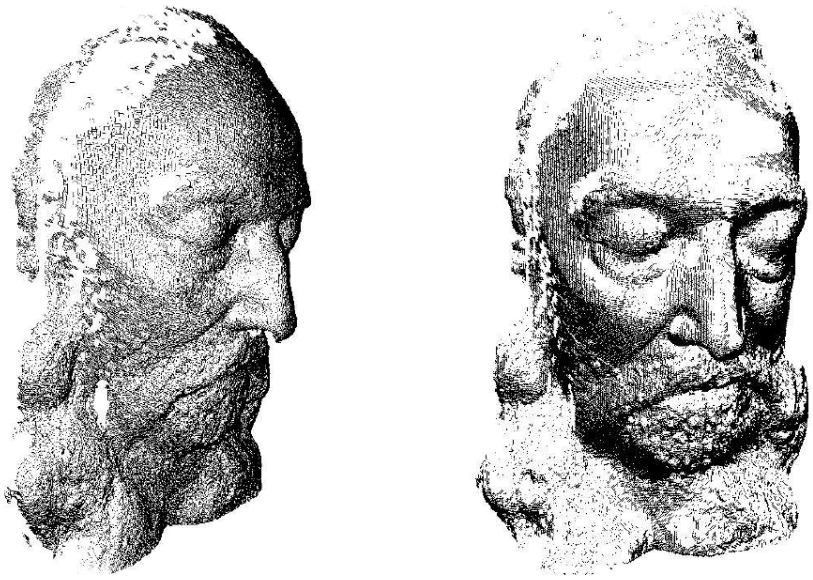


Figure 5.13: A **mask** of artist Richard Jakopic (built from range images); *Left*: Covering the model with filled strokes on locations of negative mean curvature -H(-1.0, -0.01) and positive mean curvature +H(0.001, 1.0) and slope aspect A0(0.9, 1.0). Notice how facial features are revealed. *Model provided by Danijel Skocaj, University of Ljubljana Computer Vision Lab* [Sol00].

Figure 5.14: The "**killeroo**" model with filled strokes placed by slope steepness, aspect and positive mean curvature. Model provided by *headus.com.au*

Figure 5.15: A **gargoyle** built from range images; *Left*: Stroke placed in areas of high steepness. *Right*: Placing strokes for all other shape measures, with emphasis on slope aspect. Notice how the directional variation of strokes reveal the curved shapes of the statue. *Model provided by Rich Pito, University of Pennsylvania GRASP Lab.*

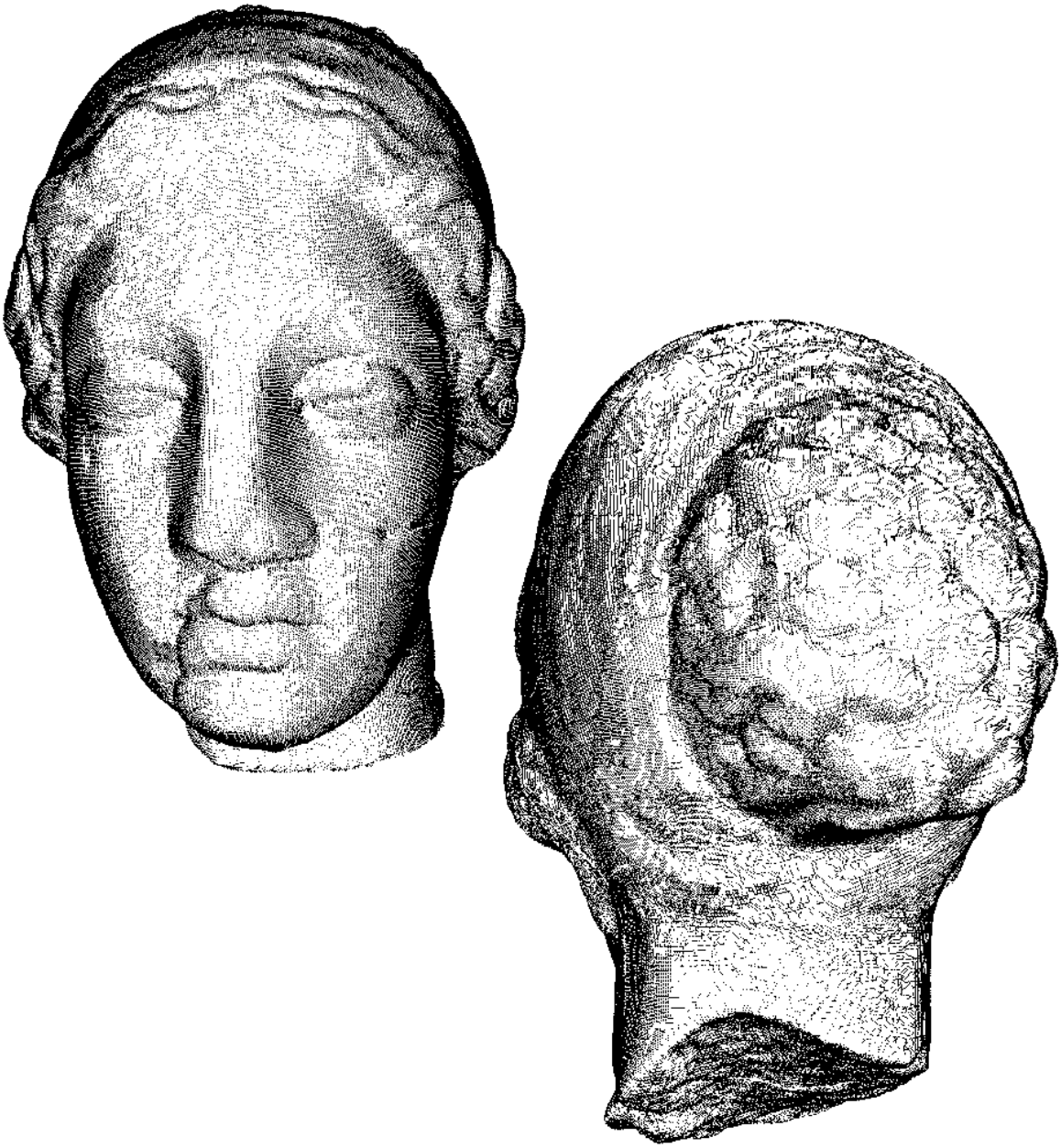


Figure 5.16: An **Igea artifact** built from laser scans; *Left*: placing filled strokes with slope steepness GA(0.81, 1.0) and positive mean curvature H(0.57, 1.0). *Right*: placing serrated strokes with three marks per edge and with view depth effect, revealing shape measures of slope steepness GA(0.73, 1.0) and negative mean curvature -H(-0.5,-0.48). Notice that hair and anatomic features are clearly revealed. This has special significance for this model given that the original artifact has various degrees of erosion. *Model provided by Cyberware.*

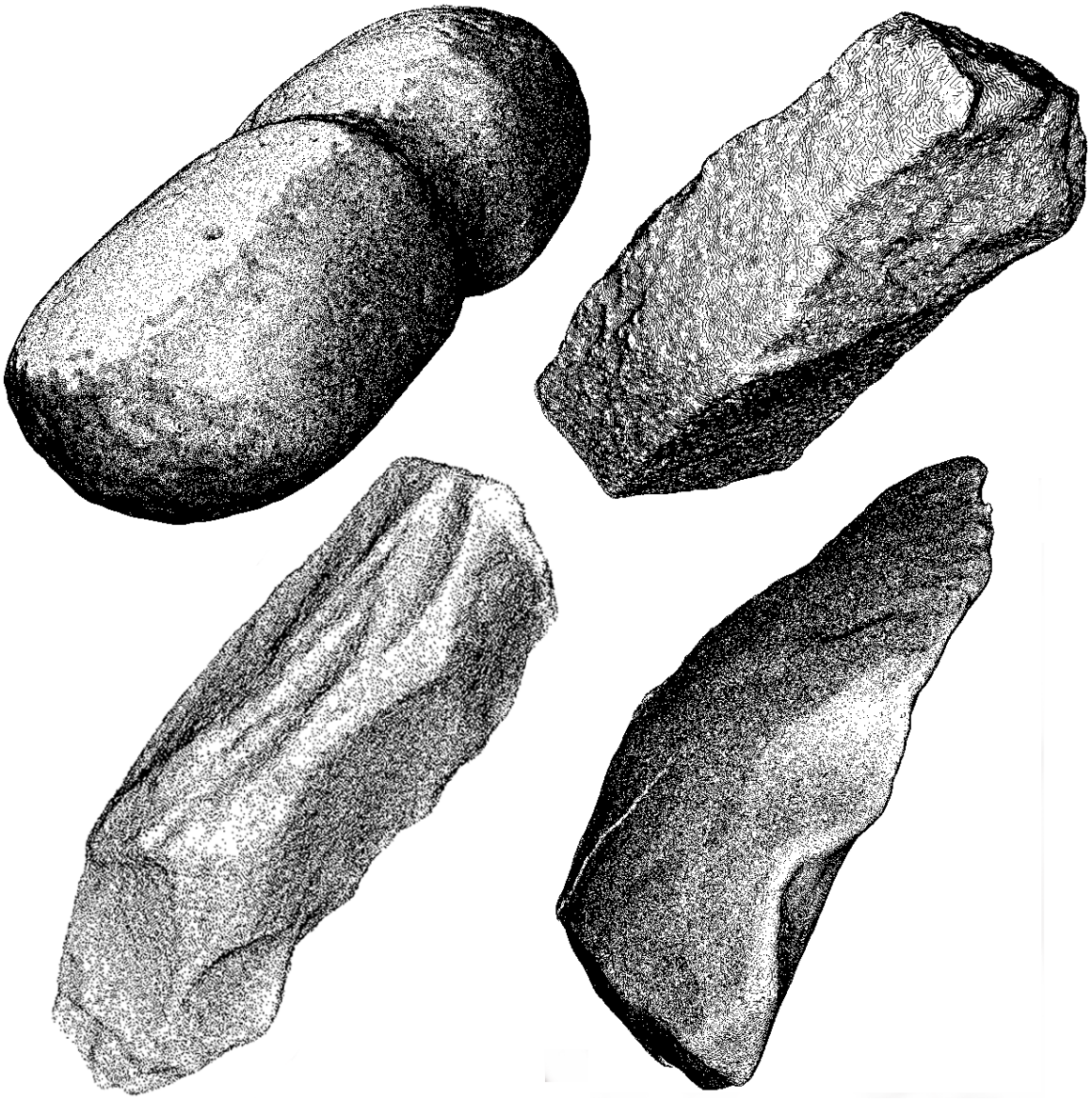


Figure 5.17: **Archaeological objects** built from laser scans. *Top-left*: a large oval hammerstone with filled strokes revealing slope steepness and locations with negative mean curvature. *Top-right*: a broken preformed adze rendered with filled strokes for slope aspect; Notice the variations of stroke directions, revealing the irregularity of the shape structure. *Bottom-left*: another adze with a different stroke style. *Bottom-right*: a preformed adze with filled strokes for positive and negative mean curvature and the effect of increasing stroke thickness as they get closer to the viewer. Models provided by Dr. Jeff Clark, North Dakota State University Archaeology Technologies Lab.

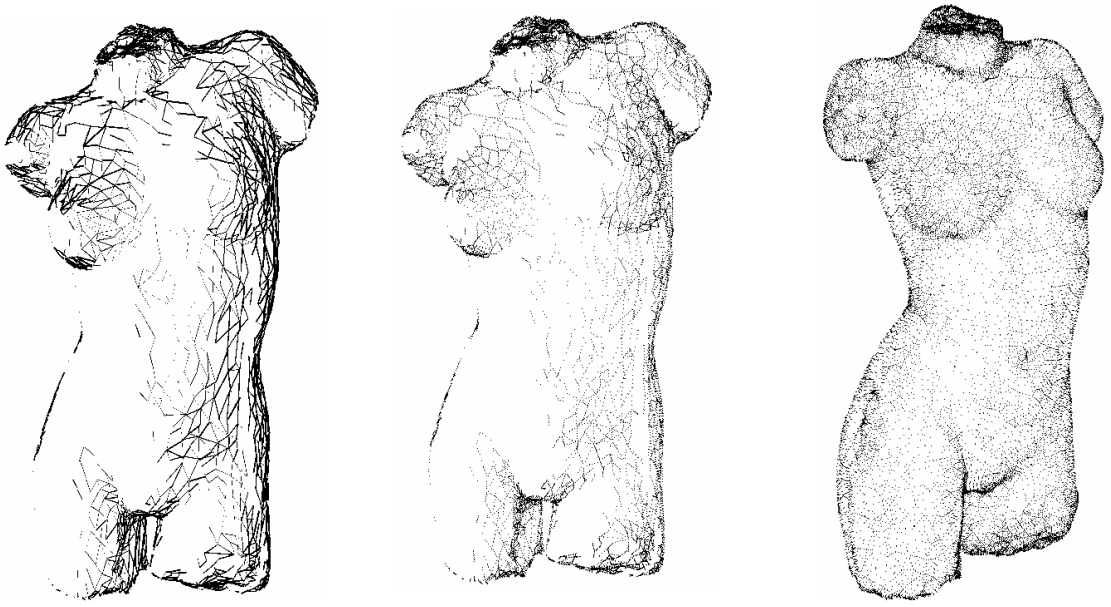


Figure 5.18: The Interior Stroke Extraction System can produce poor results if the input mesh is coarse. In the left two images in this figure, individual edges are clearly visible from the ink producing a poor pen-and-ink effect. In the right image, input parameters have been carefully adjusted to minimize this effect.

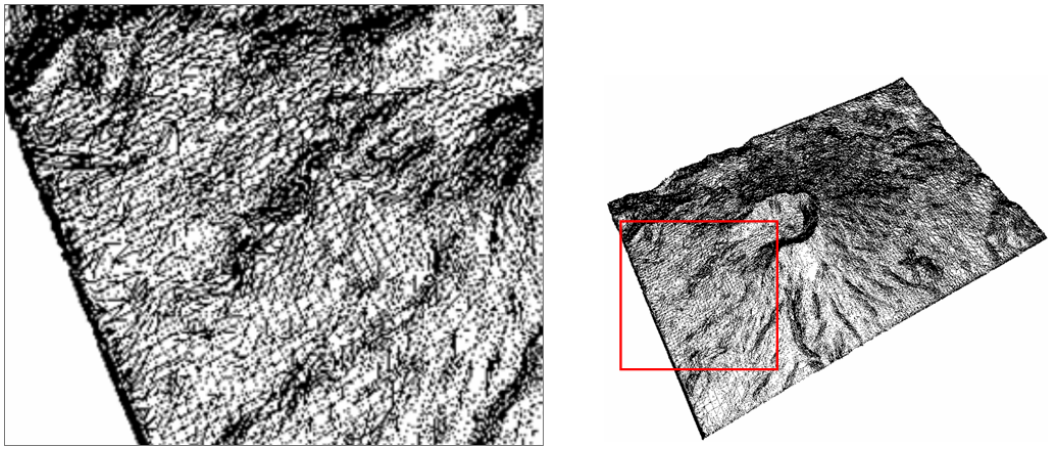


Figure 5.19: The Interior Stroke Extraction System can produce unnatural effects if the input mesh consists of regularly-spaced triangles, such as the grid layout displayed here for the Mount St. Helens model.

Chapter 6

Conclusions and Future Work

In this work, two methods to create accurate pen-and-ink images from 3D polygonal meshes have been presented. These methods place strokes automatically using a variety of styles and mark-making techniques. Furthermore, they are efficient as even extremely large meshes can be rendered interactively (Tables 5.1 and 5.2).

Users can apply these methods to create pen-and-ink images of 3D meshes. These approaches offer several degrees of freedom as their parameter setting allow users to easily find a good orientation and frame to illustrate objects and instantly change the application and style of strokes. Alternately, these methods can automatically generate images with use a predefined parameters to create certain styles.

A formal evaluation has not yet been performed for either of these systems. Results have been evaluated

1. in terms of performance, by comparing to previous methods and evaluating execution times (Chapter 5)

2. visually, by comparing them directly to images generated by pen-and-ink (Figures 6.1, 6.2)

A detailed evaluation of these techniques requires expertise in the human-computer interaction area and is considered beyond the scope of this thesis. The individual techniques presented in this research will now be discussed and directions for future work will be described.

6.1 The Silhouette Stroke Extraction System

The *Silhouette Stroke Extraction System* system efficiently computes 3D stylized, smooth, error-free silhouettes in a pen-and-ink style. Users can provide input to the system to determine the type of silhouette strokes generated. A comparison between real artwork in this style and the results of the system are displayed in Figure 6.1. The main contribution of this system is the Multiresolution Artifact Removal (MAR) approach which successfully eliminates errors in polygonal silhouettes and contributes to the stroke-stylization step using multiresolution filters. This approach represents an improvement over previous works because:

- it provides resolution-independent silhouettes not bound to the geometry of the mesh. This means the silhouettes can be smoothed or coarsened to a different resolution from the raw silhouette and provide a more realistic pen-and-ink style. Smoothing is important to create natural looking silhouettes and is critical for a realistic effect when one views silhouettes from detailed meshes closely, or when one views silhouettes of simple meshes. Coarsening is useful to create simpler strokes from very complicated meshes, benefiting systems such as Kirsanov et al. [KSJ03];

- it does not require specialized error/solution cases to remove errors and thus provides a more general solution than previous techniques.

- it generates sub-polygon strokes (closer to the real location of the silhouette) for arbitrary meshes efficiently.

Figure 6.1: Comparing real pen-and-ink silhouette rendering to the results of the *Silhouette Stroke Extraction System*. *Top-row:* a real illustration of a seagull and images of a similar level of detail generated with the system. *Bottom-row:* a real illustration of a plant (*left*), and images of several plants generated with the system. *Plant meshes courtesy Martin Fuhrer*.

6.1.1 Limitations and Future Work

The *Silhouette Stroke Extraction System* has several limitations which present opportunities for future research. First, stroke accuracy can be lost with coarse meshes. In these cases, only artistic smooth strokes can be generated. Processes to maintain accuracy should be explored. Perhaps an approach that varies the amount of detail included along the chain during reconstruction could be applied to this problem. Although subdividing the mesh before extracting and correcting silhouettes solves this problem, this solution only provides a modified smooth silhouette (not the exact silhouette for the original mesh). Another limitation is the lack of a robust and efficient Hidden Line Removal approach. New methods for HLR must be investigated for the MAR approach. Finally, a limitation of all object-space methods is that they do not provide a way to eliminate duplicate silhouette chains that occur in noisy meshes, such as those produced by range-scans (Figure 5.9).

Another area of future research is to use a MAR-like multiresolution pipeline to stylize interior strokes as an improvement to traditional B-spline or low-pass filtering methods [DFRS03, SP03]. The MAR approach could also be combined with the approach presented by Kalnins et al. [KDMF03] for coherent silhouettes.

6.2 The Interior Stroke Extraction System

The *Interior Stroke Extraction System* reproduces the traditional technique of precise ink drawings, where short pen marks on the interior of the silhouette are used to depict the geometric forms that give 3D objects their characteristic shape. The system offers an efficient approach to use polygonal edges as measures for strokes, instead of using a vertex-based approach [Boi95, YKM99, GIHL00]. The system also presents a technique to model and

render pen strokes based on morphometric measures [SSM02] with automatic thickness adjustment and interactive control of pen marking styles.

The system demonstrates that precise drawing effectively illustrates complex mesh models in a simple, informative manner that is valuable, especially for illustrating regions of interest while maintaining shape perception. Images drawn by illustrators and images generated by the *Interior Stroke Extraction System* are compared in Figure 6.2. Initial feedback from illustrators is very positive. They are enthusiastic about the usefulness of the system for generating images, in particular for natural science subjects.

6.2.1 Limitations and Future Work

Several limitations to the *Interior Stroke Extraction System* offer opportunities for future work. Since this approach only generates strokes coincident with the edges of the polygon mesh, the system produces poor results when:

- the mesh contains few triangles. In this case, strokes from individual edges do not provide adequate coverage of the surface to create strokes. (Figure 5.18)
- the mesh is made up entirely of triangles following a certain pattern, creating a grid effect (Figure 5.19).

The course to solve this could involve improving the input polygon mesh or modifying the approach to render ink so that it can be displaced from the edge. A related limitation is that this approach cannot create strokes in arbitrary directions or in arbitrary groups at any position on the surface, although it can select existing edges pointing in certain directions for display. This problem is inherent to the system since it relies on unmodified mesh edges to place ink.

Several extensions can be explored with this work. Other elevation interpretations, to convert the DEM measures to work with 3D meshes, should be investigated. While the method provided in Section 4.3.1 (using local z-coordinates of the polygonal mesh as the elevation) generates good results and experiences many positive qualities, such as view-independence allowing evaluation of shape-measures as a preprocess, more accurate interpretations of mesh elevations and the resulting shape measures should be explored. Other possible extensions to the system include investigating the value of other morphometric shape-measures for ink-placement, using these measures to trace long strokes across the surface and to compare the performance of the *Interior Stroke Extraction System* to other NPR stroke-based renderers to assess the effectiveness of using a precise drawing system.

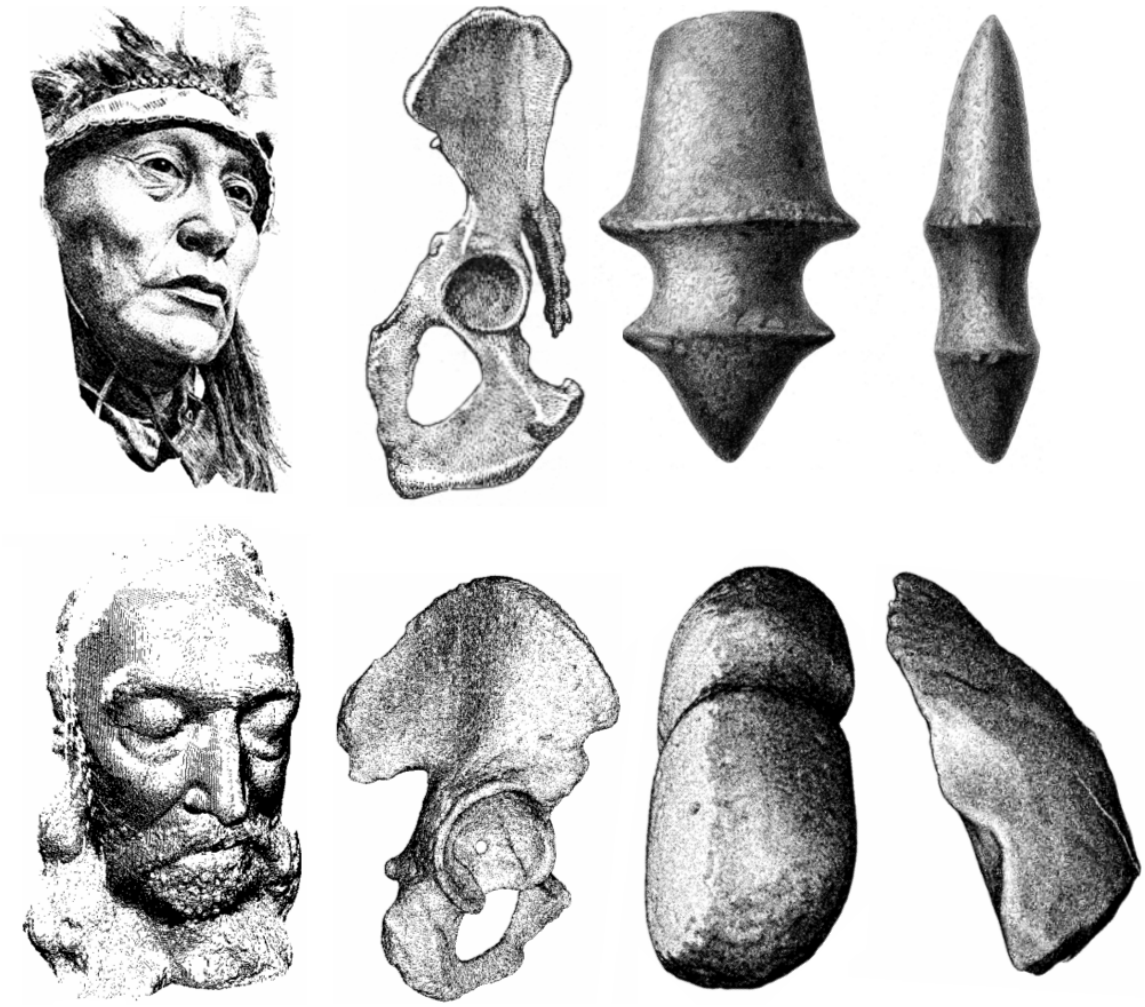


Figure 6.2: Comparing real detailed pen-and-ink rendering to the results of the Interior Stroke Extraction System. *Top-row:* Real drawings of various subjects. *Bottom-row:* system results for similar subects.

Bibliography

[BGS04] R. H. Bartels, G. H. Golub, and F. F. Samavati. Some observations on local least squares. In *Stanford University, Tech. Rep. SCCM-04-09*, 2004.

[Boi95] E. Boix. *Approximation lineaire des surfaces de $\Re^3$ et applications*. PhD thesis, Ecole Polytechnique, France, 1995.

[BS00a] R.H. Bartels and F.F. Samavati. Reversing subdivision rules: Local linear conditions and observations on inner products. In *Journal of Computational and Applied Mathematics, vol. 119, Issue 1-20*, pages 29–67, 2000.

[BS00b] J.W. Buchanan and M.C. Sousa. The edge-buffer: A data structure for easy silhouette rendering. In *Proc. of NPAR 2000*, pages 39–42, 2000.

[BSS04] M. Brunn, M.C. Sousa, and F. Samavati. Capturing and re-using artistic styles with reverse subdivision-based multiresolution methods. In *Department of Computer Science Tech. Report, University of Calgary*, 2004.

[CJTF98] W.T. Corrêa, R.J. Jensen, C.E. Thayer, and A. Finkelstein. Texture mapping for cell animation. In *Proc. of SIGGRAPH '98*, pages 435–446, 1998.

[CKvD96] C. Christou, J. J. Koenderink, and A. J. van Doorn. Surface gradients, contours and the perception of surface attitude in images of complex scenes. pages 701–713. Perception, 1996.

[Cor56] I.W. Cornwall. *Bones for the Archeologist*. Phoenix House, London, 1956.

[Cra00] W. Crane. *Line Form*. George Bell Sons, London, 1900.

[DFRS03] D. DeCarlo, A. Finkelstein, Szymon Rusinkiewicz, and A. Santella. Suggestive contours for conveying shape. In *Proc. SIGGRAPH 2003*, pages 848–855, 2003.

[DHvOS00] O. Deussen, S. Hiller, C. van Overveld, and T. Strothotte. Floating points: A method for computing stipple drawings. In *Proc. of Eurographics 2000*, pages 40–51, 2000.

[DS00] O. Deussen and T. Strothotte. Computer-generated pen-and-ink illustration of trees. In *Proc. of SIGGRAPH 2000*, pages 13–18, 2000.

[Elb98] G. Elber. Line art illustrations of parametric and implicit forms. In *IEEE Transactions on Visualization and Computer Graphics, Vol. 4, No. 1*, pages 71–81, 1998.

[Eva72] I.S. Evans. General geomorphometry, derivatives of altitude and descriptive statistics. *Spatial analysis in geomorphology*, pages 17–90, 1972.

[FS94] A. Finkelstein and D. H. Salesin. Multiresolution curves. In *Proc. of SIGGRAPH'94*, pages 261–268. ACM Press, 1994.

[GGSC98] A. Gooch, B. Gooch, P. Shirley, and E. Cohen. A non-photorealistic lighting model for automatic technical illustration. In *Proc. of SIGGRAPH '98*, pages 447–452, 1998.

[GIHL00] A. Girshick, V. Interrante, S. Haker, and T. Lemoine. Line direction matters: an argument for the use of principal directions in 3d line drawings, 2000.

[Gra97] A. Gray. The gaussian and mean curvatures. *§16.5 in Modern Differential Geometry of Curves and Surfaces with Mathematica, 2nd ed., Boca Raton, FL, CRC Press*, pages 373–380, 1997.

[Gre99] Stuart Green. Non photorealistic rendering. In *Siggraph 1999 Course Notes*, 1999.

[GSG+99] B. Gooch, P.J. Sloan, A. Gooch, P. Shirley, and R. Riesenfeld. Interactive technical illustration. In *1999 ACM Symposium on Interactive 3D Graphics*, pages 31–38, 1999.

[Her99] A. Hertzmann. Introduction to 3d non-photorealistic rendering: Silhouettes and outlines. In *Non-Photorealistic Rendering (SIGGRAPH '99 Course Notes)*, 1999.

[Hod89] E.R.S. Hodges. *The Guild Handbook of Scientific Illustration*. Van Nostrand Reinhold, 1989.

[HS99] J. Hamel and T. Strothotte. Capturing and re-using rendition styles for non-photorealistic rendering. In *Proc. of Eurographics '99*, pages 173–182, 1999.

[HZ00] A. Hertzmann and D. Zorin. Illustrating smooth surfaces. In Kurt Akeley, editor, *Proc. of SIGGRAPH 2000*, pages 517–526. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[IFH+03] T. Isenberg, B. Freudenberg, N. Halper, S. Schlechtweg, and T. Strothotte. A developer's guide to silhouette algorithms for polygonal models. In *IEEE*

Computer Graphics and Applications, pages 28–37, 2003.

[IHS02] Tobias Isenberg, Nick Halper, and Thomas Strothotte. Stylizing Silhouettes at Interactive Rates: From Silhouette Edges to Silhouette Strokes. *Proc. of Eurographics 2002*, 21(3):249–258, September 2002.

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

[Krc73] J. Krcho. Morphometric analysis of relief on the basis of geometric aspect of field theory. *Geographica Universitatis Comenianae, Geographico-Physica*, (1):7–233, 1973.

[KSJ03] D. Kirsanov, P. V. Sander, and S. J.Gortler. Simple silhouettes over complex surfaces. In *Proc. of First Symposium on Geometry Processing 2003*, pages 102–106, 2003.

[LME+02] A. Lu, C. Morris, D. Ebert, P. Rheingans, and C. Hansen. Non-photorealistic volume rendering using stippling techniques. In *Proc. of IEEE Visualization 2002*, pages 211–218, 2002.

[LP82] L. Lapidus and G.F. Pinder. *Numerical Solution of Partial Differential Equations in Science and Engineering*. John Wiley & Sons, 1982.

[MH93] H. Mitasova and J. Hofierka. Interpolation by regularized spline with tension: II. application to terrain modeling and surface geometry analysis. *Mathematical Geology*, 25:657–669, 1993.

[MKT+97] L. Markosian, M.A. Kowalski, S.J. Trychin, L.D. Bourdev, D.Goldstein, and J.F. Hughes. Real-time non-photorealistic rendering. In *Proc. of SIGGRAPH '97*, pages 415–420, 1997.

[NM00] J. Northrup and L. Markosian. Artistic silhouettes: A hybrid approach, 2000.

[PFH00] E. Praun, A. Finkelstein, and H. Hoppe. Lapped textures. In Kurt Akeley, editor, *Proc. of SIGGRAPH 2000*, pages 465–470. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[PFS03] O.E.M. Pastor, B. Freundenberg, and T. Strothotte. Animated, real-time stippling. In *IEEE Computer Graphics and Applications (Special Issue on Non-Photorealistic Rendering*, pages 62–68, 2003.

[PHWF01] E. Praun, H. Hoppe, M. Webb, and A. Finkelstein. Real-time hatching. In Eugene Fiume, editor, *Proc. of SIGGRAPH 2001*, pages 579–584. ACM Press, 2001.

[Raw87] P. Rawson. *Drawing*. University of Pennsylvania Press, 1987.

[RC99] R. Raskar and M. Cohen. Image precisiton silhouette edges. In S.N. Spencer, editor, *Proc. 1999 ACM Symp. Interactive 3D graphics*, pages 135–140. ACM Press, 1999.

[RK00] C. Rössl and L. Kobbelt. Line-art rendering of 3d models, 2000.

[Rus89] P. Rustagi. Silhouette line display from shaded models. In *Iris Universe*, pages 42–44, 1989.

[RvE92] J.R. Rossignac and M. van Emmerik. Hidden contours on a frame-buffer. In *Proc. 7th Eurographics Workshop Computer Graphics Hardware*, pages 108–204, 1992.

[Sal97] M.P. Salisbury. Ink-based pen-and-ink illustration. In *PhD Thesis*, 1997.

[SB99] F.F. Samavati and R.H. Bartels. Multiresolution curve and surface representation by reversing subdivision rules. In *Computer Graphics Forum, Vol. 18, No. 2*, pages 97–120, 1999.

[SB04] F.F. Samavati and R. H. Bartels. Local b-spline wavelets. In *Biometric 2004, University of Calgary*, 2004.

[SDS96] E.J. Stollnitz, T.D. Derose, and D.H. Salesin. Wavelets for computer graphics: Theory and applications. In *Morgan Kaufmann*, 1996.

[Sec02] A. Secord. Random marks on paper: Non-photorealistic rendering with small primitives. Master's thesis, Department of Computer Science, University of British Columbia, October 2002.

[SGG+00] Pedro V. Sander, Xianfeng Gu, Steven J. Gortler, Hugues Hoppe, and John Snyder. Silhouette clipping. In Kurt Akeley, editor, *Proc. of Siggraph 2000*, pages 327–334. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[Sim92] G. Simmons. *The Technical Pen*. Watson-Guptill Publications, 1992.

[Sol00] F. Solina. Internet based art installations. In *Informatica, 24(4)*, pages 459–466, 2000.

[SP03] M.C. Sousa and P. Prusinkiewicz. A few good lines: Suggestive drawing of 3d models. *Proc. of Eurographics 2003*, 22(3):327 – 340, 2003.

[SSM02] P.A. Shary, L.S. Sharaya, and A.V. Mitusov. Fundamental quantitative methods of land surface analysis. *Geoderma*, 107(1-2):1–32, 2002.

[ST90] T. Saito and T. Takahasi. Comprehensible rendering of 3d shapes. In *Proc. of SIGGRAPH'90*, pages 197–206, 1990.

[Whi94] S. Whitaker. *The Encyclopedia of Cartooning Techniques*. Running Press, Philadelphia, 1994.

[Woo96] J. Wood. *The Geomorphological Characterisation of Digital Elevation Models*. PhD thesis, University of Leicester, UK, 1996.

[WS94] Georges Winkenbach and David Salesin. Computer-generated pen-and-ink illustration. In *SIGGRAPH*, pages 91–100, 1994.

[WS96] Georges Winkenbach and David Salesin. Rendering parametric surfaces in pen and ink. In *SIGGRAPH*, pages 469–476, 1996.

[YKM99] P. Yuen, N. Khalili, and F. Mokhtarian. Curvature estimation on smoothed 3-d meshes. In *Proc. of British Machine Vision Conference '99*, pages 133–142, 1999.