Line Drawings of 3D Models

Forrester H. Cole

A Dissertation
Presented to the Faculty
of Princeton University
in Candidacy for the Degree
of Doctor of Philosophy

Recommended for Acceptance
by the Department of
Computer Science
Adviser: Adam Finkelstein

June 2009

Abstract

Line drawings are commonly used for sketches, animations, and technical illustrations because they are familiar, simple, and easy to draw, yet wide-ranging and expressive. However, current tools for computer generation of line drawings do not match the range and expressiveness available to a practiced human artist. In part, these failings are due to technical limitations of current algorithms. However, there is also a lack of formal understanding of how artists make drawings, and where these drawings are effective.

This thesis describes recent work on line drawings, including two studies aimed at a quantitative understanding of how people create and perceive line drawings of 3D shapes. In the first study, we asked artists to create drawings under controlled conditions, and in the second, we asked people to view the drawings and record their shape impressions using a series of gauge figures. We conclude that most of the lines artists draw can be explained by currently known definitions, and that line drawings based on these definitions can successfully depict shape, though they usually fall short of shaded depictions.

The final section of the thesis describes a system for drawing stylized lines, with two novel and important features: it includes an effect called *stylized focus* that can help direct the viewer's gaze to important parts of the drawing, and it is fast enough and has sufficient frame-to-frame coherence that it is suitable for rendering complex models interactively.

The presented results can be applied across computer graphics, with benefits to computer animation, visualization, design, and games.

Acknowledgements

I would like to thank my advisor, Adam Finkelstein, for ceaseless encouragement and aid with just about every aspect of this work. It is a huge help to have someone who is always excited about your work, especially when you are not.

Also invaluable were Tom Funkhouser, Szymon Rusinkiewicz, Doug DeCarlo, and Manish Singh, for critical advice and contributions, and for pushing me to keep improving the work.

Thanks to Alex Golovinskiy, Michael Burns, Alex Limpaecher, Kenrick Kin, Keith Morley, and Anthony Santella, for helping make these projects happen.

Special thanks to Heather Stoddart Barros and her students at Art Collaborations, as well as the artists at Princeton Univerisity, for providing the artwork that we used in two of these studies. There would have been no thesis without their help.

Aim@Shape, VAKHUN, Cyberware, and Google provided the 3D models used in this work.

This work was sponsored in part by NSF grants CNS-0406415, IIS-0612231, CCF-0702672, CCF-0702672, IIS-0511965, and CCF-0347427, and by Google.

Contents

Chapter 1

Introduction

This thesis is about line drawings of 3D shapes. Line drawings are commonly used for sketches, animations, and technical illustrations because they are familiar, simple, and easy to draw, yet wide-ranging and expressive. However, current tools for computer generation of line drawings do not match the range and expressiveness available to a practiced human artist. Furthermore, current algorithms suffer from technical problems that reduce their usefulness in interactive applications, the one area where computer tools are indispensible.

This thesis aims to improve computer drawing algorithms by investigation in two directions. The first direction is to develop solutions to the technical challenges inherent in generating and animating drawings. The second direction is to conduct empirical studies to more formally understand how people create and perceive line drawings. Developments in the first direction yield better algorithms, while developments in the second help inform computer graphics researchers and also provide insights into human shape perception.

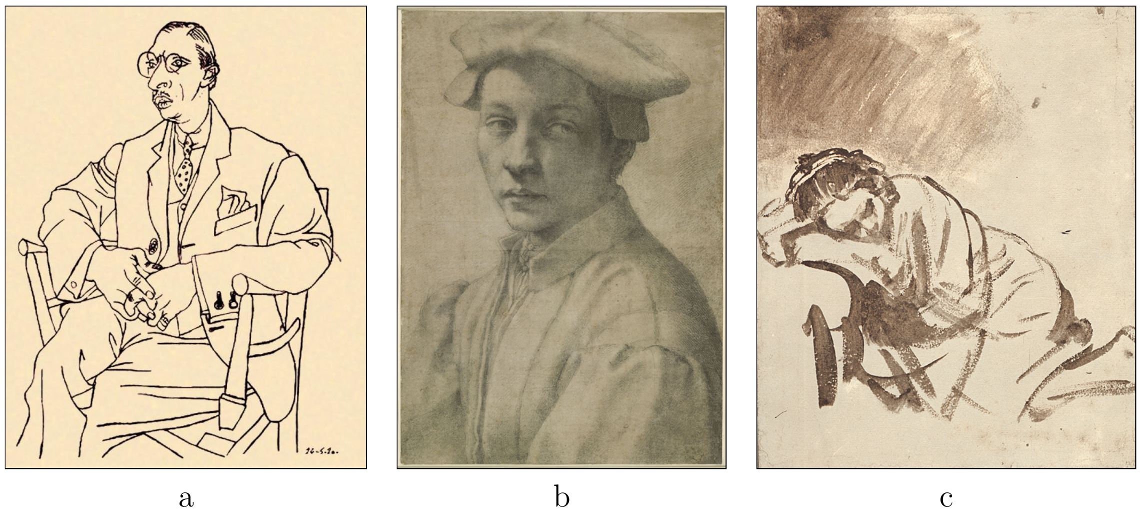


Figure 1.1: *Examples of drawing types.* (a) pure line drawing by Picasso, (b) mass and shading oriented drawing by Michelangelo, (c) drawing that uses elements of both line and mass drawing by Rembrandt.

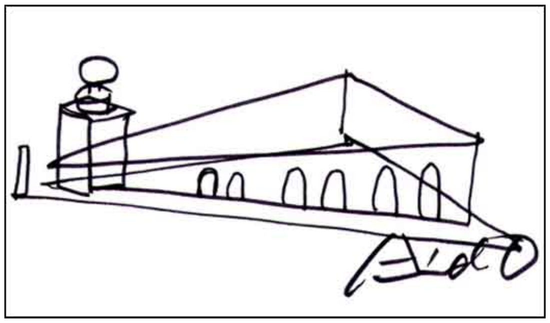
1.1 Why Line Drawings?

The term *line drawing* refers to drawings that are generally concerned with the outlines of shapes (Figure 1.1a). Line drawing can be contrasted to *mass drawing* (Figure 1.1b), which is concerned with the shading and bulk of a shape [69]. Most work by artists combines some aspects of these two techniques (Figure 1.1c), so we will sometimes refer to “line drawings” that actually include a mix of line and mass drawing. For clarity, the term *pure line drawing* will be used to mean a drawing without any shading at all.

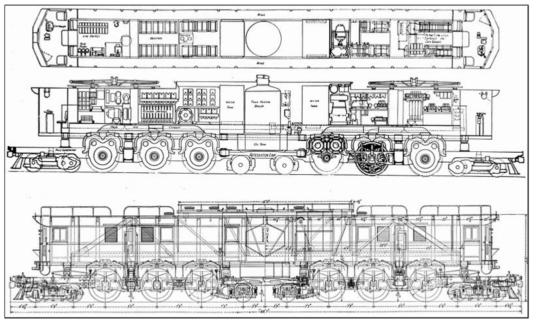
Artists often learn to make line drawings first, and then advance to shading and color. One important reason is that line drawings are easy and straightforward to create. Most common drawing media (pens, pencils, with the exception perhaps of watercolor) make lines in their most basic mode of use. Line drawings also require much less ink than shaded pictures, and are therefore quicker for an artist to draw (for example, it is much faster to draw four lines for a box than to carefully shade in the same rectangle). For similar reasons, computer graphics began with line drawings as well – the first graphics systems used monochrome, vector-based displays (e.g., Sketchpad [72]). Vector displays could draw lines easily but had difficulty shading, for the exact same reason artists have difficulty shading with a pen or pencil: they must build up the shading with many carefully placed strokes. For a while, creating good line drawings using vector displays was an active area of research. A particularly active area was the creation of drawings of 3D models with occluded lines removed ("hidden line drawing"), a subject we will revisit in Chapter 5. The invention of raster graphics and the z-buffer, however, made rendering shaded images fast and easy, and sidelined line drawing in computer graphics. Recent work in shaded graphics has enjoyed huge success, achieving near photorealism even in interactive applications.

Of course, there are other reasons to be interested in line drawings besides the simplicity of drawing them. One of the most interesting properties of line drawings is their power for abstraction, which can be loosely described as the ability to depict only the essential details of a scene or object. A specific example of abstraction is a pure line drawing's ability to convey shape without conveying much, or perhaps any, information about lighting or material. This is something that is, almost by definition, impossible for a photorealistic rendering to do; a photograph of an object is precisely a representation of the shading of an object. Photographers use an arsenal of tricks such as fill lights, gobos, and light boxes to reduce the distractions generated by unwanted shading effects [31], but can never eliminate shading information entirely. In terms of the effect on the viewer, a pure line drawing can depict a "box," while a photograph can at best depict a "gray box." Architects and designers exploit the power of abstraction to call attention to the shape and organization of a project, without focusing on details of its materials or construction (Figure 1.2). How artists construct these drawings, and the success with which humans interpret the results, are the subjects of the studies described in Chapters 3 and 4. Both studies explore the pure line drawing style.

Beyond abstraction, line drawings have a range of practical benefits that make



a



b

Figure 1.2: *Simple and complex design drawings.* (b) architectural sketch by Tadao Ando, (b) technical drawing of a Milwaukee Road EP-3 "Quill" electric locomotive.

them useful as a supplement to, or replacement for, shaded imagery in specific cases. For example, lines are commonly used to aid comprehension in diagrams or illustrations because of their ability to make shape features "pop" out of the background. Lines can help visualize overlapping or hidden shapes without causing excessive visual clutter [7]. Line drawings are also a good option when printing or reproduction requirements limit artwork to few colors.

However, perhaps the best reason to study line drawings is because of their incredible expressiveness in the hands of a master artist. The range of drawing styles available to an artist is explored in Section 2.1.

1.2 Research Contributions

This thesis makes both pure research and applied algorithmic contributions to the understanding and generation of line drawings.

Chapter 3 describes a study designed to learn about how people depict shape with line drawings. As part of the study, we asked a group of artists to make drawings of a set of 3D models. The artists made free-hand drawings, but also registered their drawings to computer renderings of the shapes, allowing us to compare artists' drawings to each other, and to computer-generated drawings of the same models.

Around 80% of the lines drawn by artists can be explained by prominent geometric or image features, while the remaining lines are apparently the result of higher-level choices. This work was previously published in [13], and has already helped prompt at least one new line drawing algorithm [83].

Chapter 4 describes a follow-up to our drawing collection project, in which we conducted a perceptual study of how well people perceive shape in the artists' drawings. The study methodology followed the gauge figure approach proposed by Koenderink [40], though we gathered a very large amount of noisy data rather than a small amount of very careful data. We included both hand-drawn and computer-generated drawings. Among other conclusions, we found that the computer-generated drawings often performed close to or as well as the drawings made by human artists. This work will be published later this year in [14].

Finally, Chapter 5 describes technical details of an interactive system for drawing stylized lines, using new techniques for computing visibility and for eliding lines. The system also supports rendering effects meant to direct the viewer's gaze. The effect of this stylized focus is measured with an eye-tracking experiment. The visibility algorithm was published in [12], and the stylized rendering effects were published in [10].

Chapter 2

Background

This thesis touches on several areas of computer science, art, and cognitive science. This chapter explains the background for the basic themes that are common to all the later chapters: drawings by artists, drawings by computers, and evaluation studies of computer-generated imagery. Specific background information appears in the later chapters where appropriate, but a knowledge of the subjects covered in this chapter will be assumed.

2.1 Drawings by Artists

Artwork by human artists is the inspiration for most work in NPR, and line drawing algorithms are no exception. The line work of a master is still far beyond the capabilities of current algorithms, but understanding the range and expressiveness of a skilled human artist can help computer scientists learn what is possible with the medium, and what an ideal line drawing system might be capable of. The purpose of this section is to give some impression of the breadth of techniques available to a human artist, and some context for the results that follow.

The range of line drawings as an art form can be nicely captured by comparing a sketch by Michelangelo (Figure 2.1a) with a technical illustration of a power tool

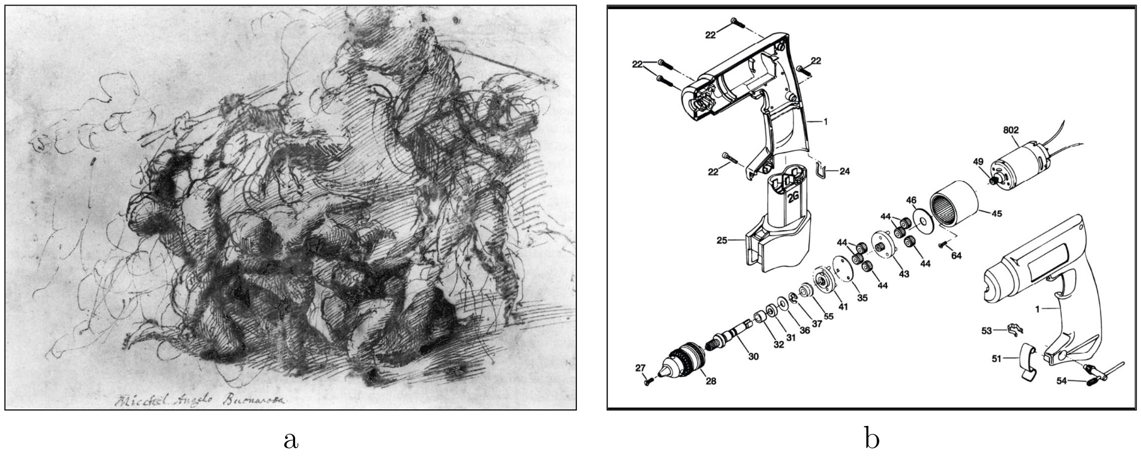


Figure 2.1: *Emotional and explanatory line drawings.* (a) sketch by Michelangelo, (b) Bosch Tools schematic diagram (from Li et al. [46]).

(Figure 2.1b). Both drawings create effects in the mind of the viewer that would be very difficult to imitate with a photorealistic style. However, the drawings affect the viewer in different ways – you can almost feel your brain switching gears as you look from one to the other.

In the Michelangelo sketch, the faces and sometimes limbs of the people are simply omitted, leaving the viewer to concentrate on the composition and motion of the drawing. The drawing has mostly emotional content, and asks the viewer to fill in the exact details with their imagination. The technical drawing, by contrast, is completely explanatory. The shapes are depicted thoroughly and precisely, and are arranged so as to explain their positions in the final assembled piece. The drawing is constructed precisely so that the viewer *does not* have to use their imagination to fill in the geometry of the scene. The drawing invites the viewer to construct a mental image of the arrangement of parts, and does not distract the viewer with any extraneous details. Both drawings are effective examples of abstraction.

The exact techniques by which the respective artists achieved these effects are the subject of centuries of study, and books on art instruction often do a good job of teaching them (e.g., [26, 51, 63]). Some examples that are particularly salient, however, include loose and controlled drawing, sparse and developed drawing, and abstract and veridical drawing.

When placing each line in a drawing, the artist can choose whether to make the line hew closely to the exact shape of the subject, or make the line a loose approximation. This decision is not based just on the time or effort required, though those are important considerations. A loose, sketchy style conveys a certain kind of abstraction: it indicates that there is more to the subject than is depicted on the page (Figure 2.2a). A precise, controlled style, by contrast, tends to convey the idea that what is on the page is a full representation of the subject, and does not leave anything out (Figure 2.2b).

Similarly, the artist can choose how many lines to use to depict the subject. It is almost always possible to add more detail to any part of a drawing, but such detail will usually change the overall effect. A sparse style, like a loose style (they are often combined) indicates that the artist is providing the gestalt impression of a scene, rather than the details (Figure 2.2c). However, the details can make all the difference: for example, the chains and gibbets in Figure 2.2d. Control over the level of detail in a drawing is the aim of line elision techniques such as described in Section 5.2.

Finally, an artist can create a careful line drawing that gives an abstract impression conveying something beyond the literal shape of the subject (Figure 2.2e), or a veridical, life-like impression (Figure 2.2f). Of all the techniques mentioned so far, this type of abstraction is perhaps the most challenging for a computer to mimic, because it requires a deep knowledge of the subject and an artistic flair that only a few people possess.

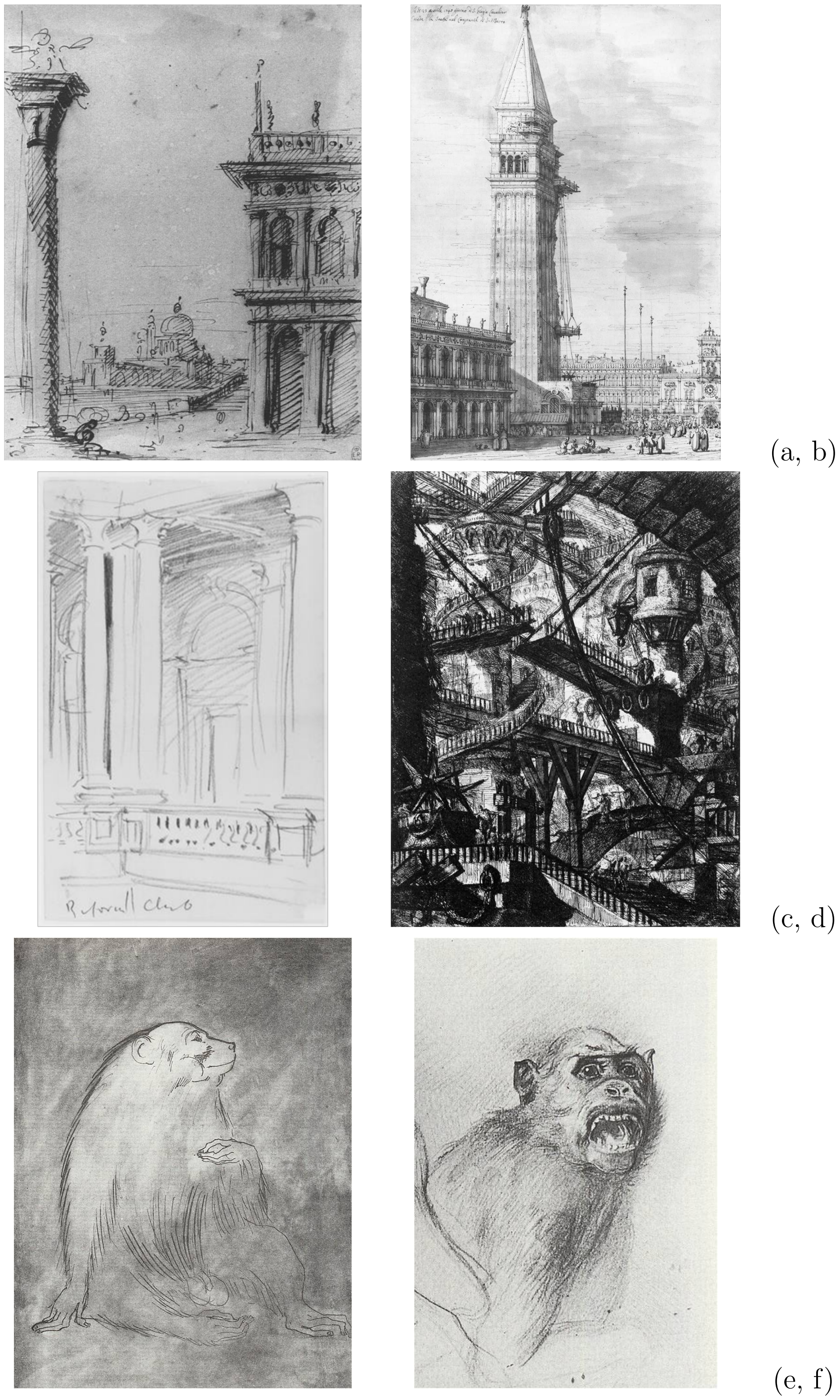


Figure 2.2: *Example drawings.* loose and controlled (a and b, Canaletto), sparse and developed (c, John Singer Sargent, and d, Piranesi), and abstract and veridical (e, Picasso, and f, Gustave Moreau).

2.2 Drawings by Computers

The ability of line drawings to convey ideas that are difficult or impossible for a photograph to capture has led to a resurgence in their use in computer graphics, as part of the field of *non-photorealistic rendering* (NPR). Common applications of NPR line drawing algorithms are to aid comprehension and to evoke the feeling of a drawing made by hand.

There are a variety of technical challenges involved in creating a system for making line drawings, including mathematical definitions for lines, hidden line removal, line rasterization and anti-aliasing, and level of detail control. In this section we briefly review current mathematical definitions for drawing lines on 3D shapes, since these definitions will appear in all the following chapters. The other technical areas will be covered in Chapter 5.

When lines are used to directly convey shape, such as in a pure line drawing, the lines must be given a mathematical definition in terms of the geometry of the surface and position of the viewer.

The two fundamental geometric lines for views of 3D shapes are discontinuities in depth—occluding contours [41, 50, 29], and discontinuities in surface orientation—sharp creases [50, 64]. Both are classical elements in line drawings [77], and are commonplace in systems for the non-photorealistic rendering of shape.

However, organic forms such as the human body are generally too smooth to contain significant discontinuities in surface orientation, and discontinuities in depth (*occluding contours*, Figure 2.3a) are generally insufficient to depict the shape effectively (while this point is intuitive for many shapes, this thesis includes empirical evidence that occluding contours alone cannot in general depict shape effectively — see Chapter 4). In order to depict geometric features on smooth surfaces without using shading, new mathematical definitions for lines are required. In the last few years, a large number of such definitions have been proposed.

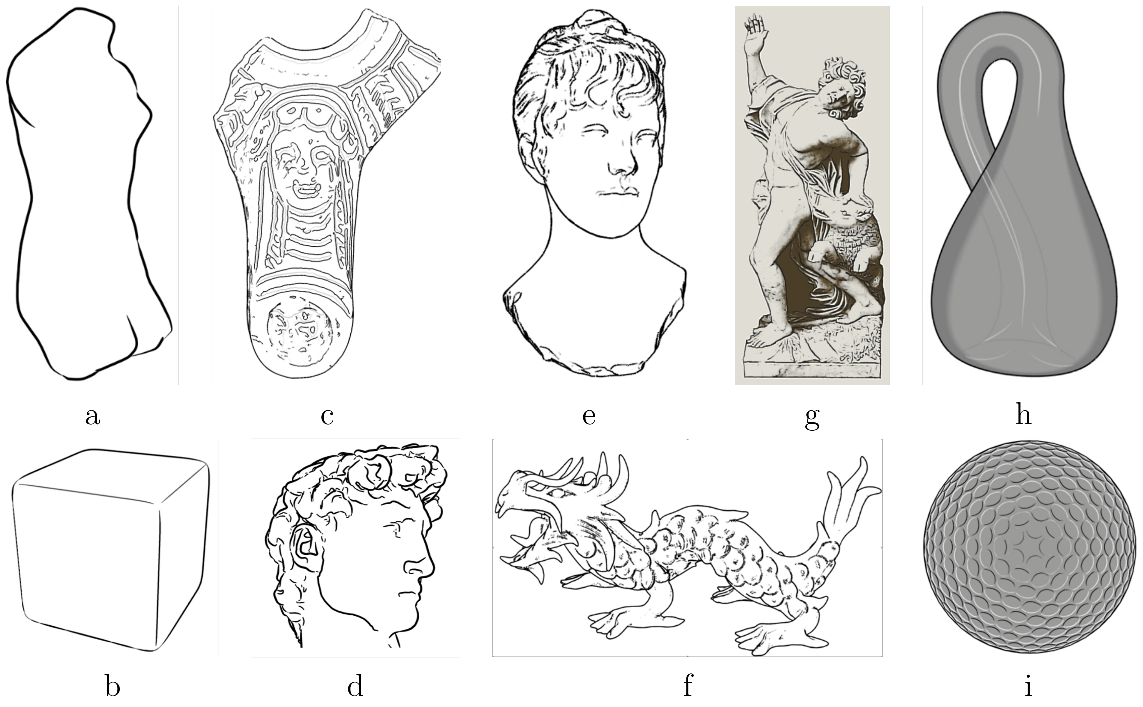


Figure 2.3: *Computer-generated line drawings.* Various mathematical definitions for lines, shown in their most favorable situations: (a) occluding contours [29], (b) geometric ridges [54], (c) demarcating curves [43], (d) suggestive contours [17], (e) apparent ridges [37], (f) laplacian lines [83], (g) abstracted shading [45], (h) principal highlights [18], and (i) suggestive highlights [18].

Perhaps the most intuitive are ridge and valley lines (Figure 2.3b), which are formed by local extrema of surface curvature in the principal directions of the shape [32, 73, 56, 54]. Ridge and valley lines may be considered a generalization of sharp creases to smooth surfaces.

Demarcating curves [43] are defined similarly to ridges and valleys, except that they lie at inflection points of principal curvatures, rather than extrema. These inflection points often seem to correspond with an intuitive segmentation of a shape (Figure 2.3c).

Related to occluding contours are suggestive contours [17, 16], which can be intuitively understood as *almost contours:* places where occluding contours appear with a small change in viewpoint (Figure 2.3d).

Apparent ridges [37] aim to combine the best aspects of ridge and valley lines and suggestive contours, by approximating ridges and valleys when the surface is viewed straight-on, and extending occluding contours when the surface is foreshortened (Figure 2.3e). Apparent ridges are defined as ridges and valleys of a view-dependent curvature measurement that takes foreshortening into account.

Laplacian lines [83] are a geometric definition meant to appear in places where the color of a shaded model would change quickly. In other words, they are geometric lines meant to correspond to image edges (Figure 2.3f).

Some line definitions are meant to indicate or abstract shading highlights. These include lines via abstract shading [45], which directly processed a shaded image to produce lines (Figure 2.3g), and principal (Figure 2.3h) and suggestive highlights (Figure 2.3i), which are defined solely in terms of the geometry of the shape and position of the viewer [18].

It is natural to ask which of these many definitions are appropriate for a specific situation, or which more closely resembles what a human would draw. Recent efforts ([37, 45, 18]) include direct comparisons between their results and artists' renderings. However, these comparisons are informal and generally intended to illustrate the inspiration for the work, not to evaluate the results of the algorithm. This thesis provides the first in-depth and formal studies of this topic in Chapters 3 and 4.

2.3 Evaluation of Effectiveness

There are several strategies to assess the effectiveness of NPR algorithms. One approach is to survey users to gather their subjective impressions. Schumann et al. [66], for example, demonstrate that architects prefer sketchy renderings to depict the preliminary nature of a design. Isenberg et al. [34] compared viewers' perceptions of hand-drawn versus computer-generated pen-and-ink illustrations by asking them to perform several subjective ranking tasks, including pile sorting and answering survey-style questions.

A second approach is to measure performance in some task. For example, Gooch et al. [24] compare response times for recognition of faces in photographs and artistic renderings. Heiser et al. [27] use a concrete task, and evaluate automatically generated assembly instructions [1] by recording construction times for assembly of a piece of furniture. Winnemoeller et al. [80] compared the speed with which participants recognized shapes under several different shading styles by presenting them with an animated set of objects and asking them to rapidly select certain specific shapes.

A third approach is to make use of techniques developed in perceptual psychology, such as gauge figure studies [40]. This is the approach followed in the study described in Chapter 4, and will be examined in detail there.

Finally, a fourth approach is available when we are concerned with how the images guide a viewer's attention: examining recordings of eye movements of viewers [19, 65]. A viewer's overt attention is measured from eye movements, which are closely coupled with cognitive processes [28]. The effectiveness of the rendering style in our interactive line rendering system (Chapter 5) is evaluated in this way.

Chapter 3

Where Do People Draw Lines?

The goal of this chapter is to characterize the mathematical properties of line drawings made by human artists. Specifically, we aim to draw relationships between the locations of lines drawn by artists and properties of the surface geometry, lighting, and viewing conditions at those locations. This type of analysis can both guide the future development of line drawing algorithms in computer graphics, and provide artists and observers with a precise vocabulary for characterizing and discussing where lines on a model are drawn.

This chapter describes a study in which art students were asked to make line drawings that "convey the shape" of 3D models shown to them as rendered images. The study balances the competing concerns of allowing the artists to draw freely and of acquiring useful data. Specifically, the artists were asked to draw in two steps: first to draw in a blank area, then to register their drawing to a faint, photo-realistic image of the model. The registered drawings can then be used to study correlations between the locations of human-drawn and computer-generated lines (Section 3.2.2), characterize the differences between specific artists (Section 3.2.3), and provide training data for synthesis of new line drawings (Section 3.2.4).

While some books on art instruction explicitly identify known line types as

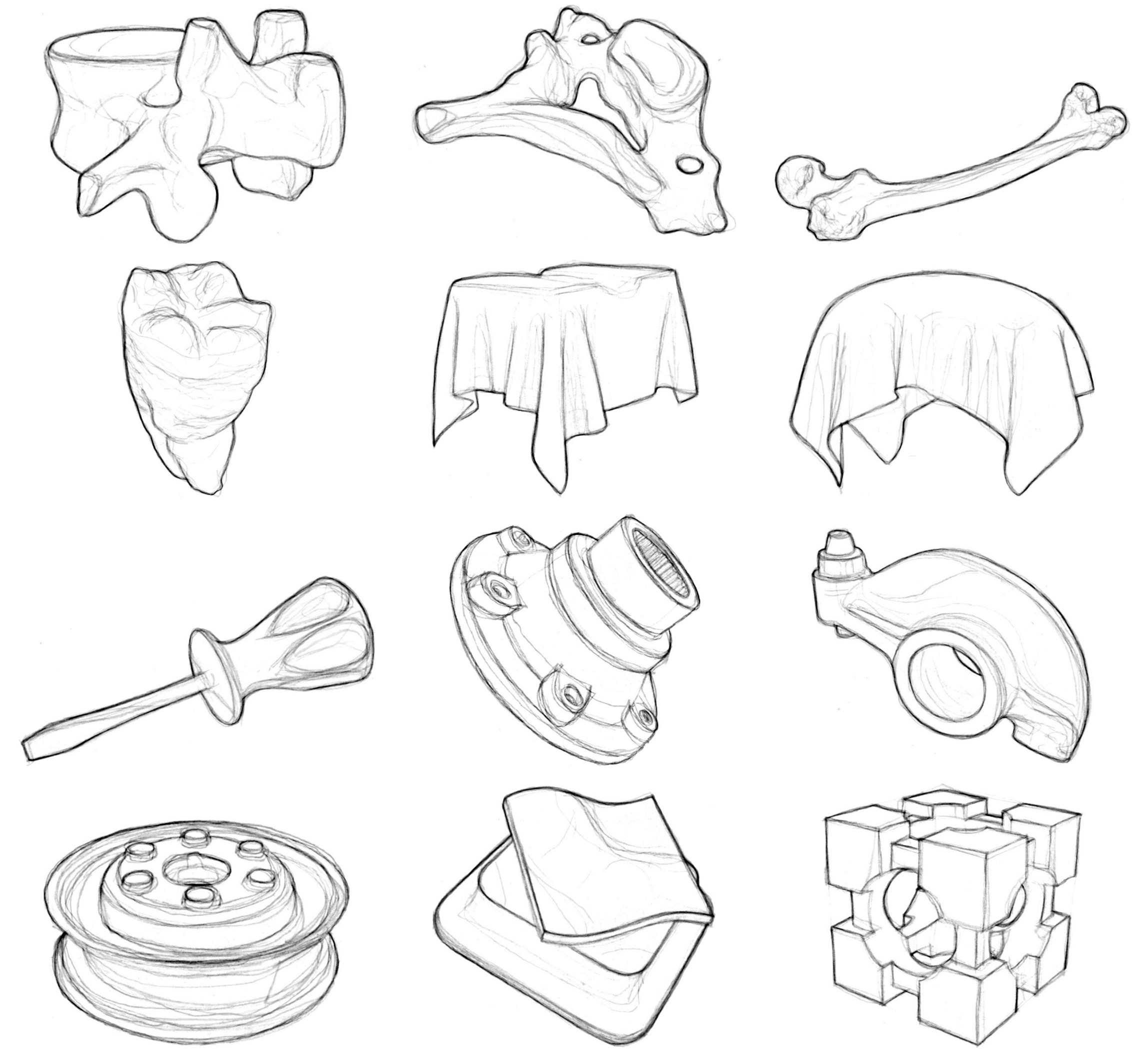


Figure 3.1: *Where people draw lines.* Average images composed of 107 drawings show where artists most commonly drew lines in our study.

candidates for drawing (e.g., contours and ridges [68], or specific feature lines on a known shape such as the nose [57]), little is said about general rules for *where* on a figure to place lines in order to best convey shape. This decision making process seems to be learned through trial and error over years of practice by individual artists.

We provide a statistical analysis of the locations where artists drew lines with the geometric, viewpoint, and lighting characteristics of the underlying 3D scene. The analysis supports several conclusions. First, human line drawings, made under our controlled conditions, are quite consistent with each other. Second, most of the areas where artists consistently drew lines can be described by well-known, simple mathematical properties, such as the locations of occluding contours and large gradients of image intensity. Third, current line drawing definitions can help explain many of the lines that do not lie in those areas, but cannot explain all the artists' lines.

Two recent studies also used drawings by human artists alongside computer renderings of the same models. Isenberg et al. [34] compared viewers' perceptions of hand-drawn versus computer-generated pen-and-ink illustrations. Phillips et al. [58] conducted a study similar to ours, in which artists were asked to draw synthetic, blobby shapes from a range of prompt types. Among other differences from that work, our study includes a separate tracing and registration step that allows greater accuracy in the analysis of artists' lines.

We believe that this study in no way exhausts the possible investigations that can be performed with this data. We therefore make our drawings and models freely available, in hopes that other researchers continue in this line of inquiry.

Overall, this chapter makes the following contributions:

- A study methodology that supports registration of human line drawings with rendered images of 3D models.
- A dataset of 208 line drawings provided by 29 skilled artists covering a dozen 3D models, with two viewpoints and two lighting conditions for each model.
- Results of correlating local properties of 3D surfaces and rendered images with the locations of lines in artists drawings.
- Characterization of which pixels drawn by recent automatic line drawing algorithms are found in human line drawings.
- A method for predicting the probability of an artist drawing at a particular location in an image and using that image to generate new line drawings.

3.1 Study

The study is designed to capture the relationships between the locations where human artists draw lines and the mathematical properties of the of the model's surface and appearance at those locations. To achieve this goal in a way that supports detailed analysis, several important choices must be made: what drawing style to consider, what models, views, and lighting conditions to use as prompts, how to present these prompts to the artists, what instructions to give the artists, and how to scan and process the drawings. The following sections describe each of our design decisions in detail.

3.1.1 Artistic Style

The first challenge in designing the study is to decide on a style of drawing that is narrow enough that all artists have roughly similar intentions while drawing, yet flexible enough for each artist to exercise individual ingenuity.

We balance these goals by focusing on line drawings that include only feature lines, with no hatching or shading (examples appear in Figure 3.4). This choice of style was made for two reasons. First, it is a simple style that is familiar to most artists and yet expressive enough to depict shape. Second, it matches the style generated by several NPR rendering algorithms recently proposed in the computer graphics literature (e.g., [17, 37]). By asking the artists to draw in the same style as the computer algorithms, we can learn both about the human drawings (by using the vocabulary of the algorithms) and the computer drawings (by using statistical correlations with human tendencies).

We give each artist verbal and written instructions to make drawings with "lines that convey the shape" of an object. We do not provide instructions about whether lines should represent shape features, lighting features, or anything else. However, we specifically ask the artists to refrain from including lines that represent area shading or tone features, such as stippling or hatching.

3.1.2 Prompt Selection

A second design decision is to select 3D models and rendering parameters to use when producing prompts (images depicting a shape for the artists to draw). In making this choice, we use the following design criteria:

- **Comprehension:** our first concern is to provide images from which the artists can easily infer shape. This consideration rules out overly abstract 3D surfaces (i.e., shapes unlike anything in common experience), complicated concave shapes (e.g., with lots of occluded surfaces), and surfaces with spatially-varying BRDFs (e.g., textures). It also suggests that multiple views of the shape be provided as prompts, so that ambiguities in one view are resolved by another. Finally, prompt images should be photorealistic, to avoid confusing artists that are not familiar with classic CG rendering artifacts such as hard shadows and lack of indirect illumination.

- **Coverage:** the set of prompts presented to each artist should have pixels that cover a wide variety of mathematical properties (e.g., high image gradients, surface critical points, etc.). This consideration rules out objects containing only large, planar facets (few interesting surface features), convex objects (no concave surface features), and other surfaces with few inflections. Rather, it suggests blobby objects with many curved surfaces.

- **Separation:** the prompt images should have mathematical features of particular interest (e.g., suggestive contours, apparent ridges) in clearly distinguishable positions within the image. This consideration rules out using headlights (a point light centered at the viewer's eye), since many interesting image features line up directly with object-space features in that case (e.g., suggestive contours and image intensity valleys).

- **Familiarity:** the objects shown in prompts must be familiar to the artist (so that he/she can understand it), but not so familiar to the that he/she applies domain-specific knowledge when drawing. This consideration rules out objects with strong semantic features (e.g., human faces) and ones commonly drawn in art classes (e.g., fruit).

- **Simplicity:** the objects must be relatively simple, without much fine scale detail. Otherwise, the artists may be tempted to abstract or simply omit important features.

Based on these criteria, we select 12 models of four object types for our study: (a) 4 bones, (b) 2 tablecloths, (c) 4 mechanical parts, and (d) 2 synthetic shapes (Figure 3.2). We synthesize four prompt images for each model, one for each combination of two different viewpoints and two lighting conditions. The two viewpoints are always 30° apart (so that large parts of each model can be seen from both viewpoints) and are carefully chosen to distribute surface features across the image. By providing prompts with different lighting and different viewpoints for the same model, we can analyze image-space properties in isolation from object-space ones.

We generate our images using YafRay [82], a free raytracing package capable of global illumination using monte carlo pathtracing. The models are rendered using a fully diffuse, gray material, and thus take on the color of the lighting environment. For lighting, we use the Eucalyptus Grove and Grace Cathedral high dynamic range environment maps captured by Debevec [15].

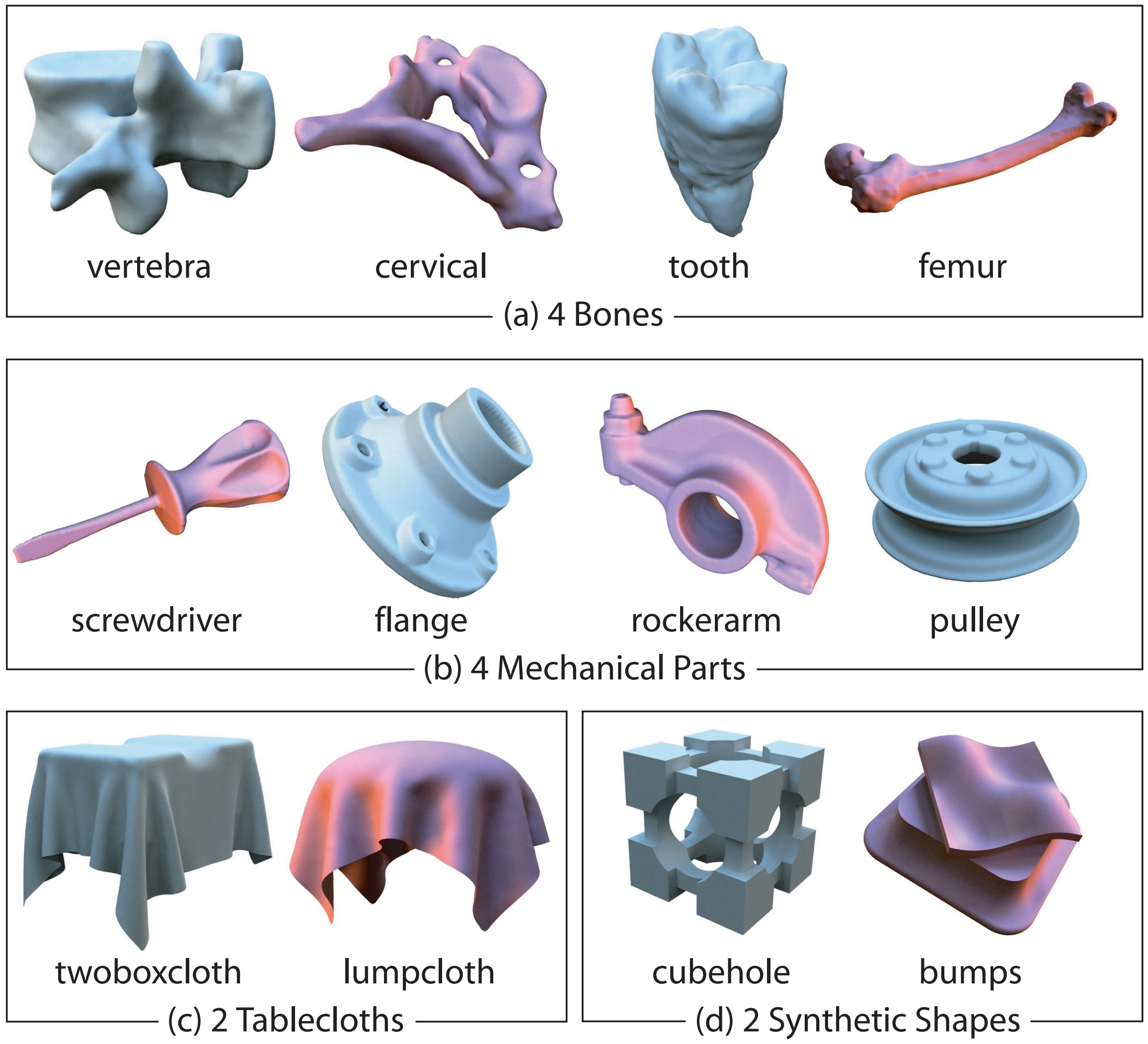


Figure 3.2: *Prompt models.* The twelve models from our study, shown with one of two views and one of two lighting conditions. Groups (a) and (b) are scanned meshes, (c) and (d) are synthetic.

3.1.3 Line Drawing Registration

The final and most difficult part of the study design is to engineer a system that is able to register line drawings made by artists to pixels of a prompt image with great accuracy.

Designing such a system is challenging because there is a trade-off between allowing the artist to draw in a natural manner (e.g., with pencil on a blank sheet of paper) versus including constraints that facilitate accurate registration between prompts and line drawings. On one hand, the drawing process surely must not bias the locations of lines made by the artist, and thus it is not a good idea to have the artist *compose* a drawing directly over the image prompt. On the other hand, the process must provide enough registration accuracy to distinguish between important mathematical properties at nearby pixels in the prompt. This problem is particularly difficult since free-hand drawings can be geometrically imprecise, and the intended location of every line is only known by the artist.

Our design balances these trade-offs with a simple two step process. In the first step, the artist is given a pencil and a blank sheet of paper and then asked to make a free-hand line drawing that "conveys the shape" of the surface in the prompt. In the second step, the artist is asked to re-create the same line drawing by tracing over a faint copy of the prompt, being careful to redraw every line of the free-hand drawing at the position corresponding to its originally intended location.

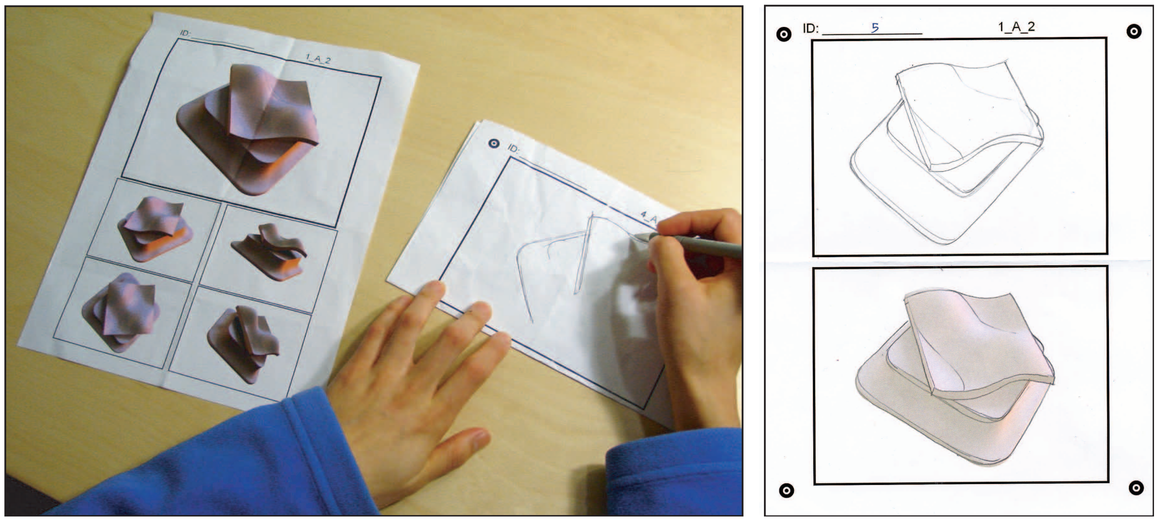


Figure 3.3: *Making a drawing.* With the drawing page folded in half, the artist makes a free-hand drawing while refering to the prompt page (left). The completed drawing page (right) contains a free-hand drawing and a registered drawing.

Specifically, the artist is given two sheets of paper for each line drawing (Figure 3.3). The *prompt page* (shown on the left) contains multiple full color views of the prompt shape, one of which is large across the top and is called the *main view*.

The *drawing page* (shown on the right) contains two boxes, each the same size as the main view. The top box is initially blank, while the bottom box contains a faint version of the main view.

The artist is asked to complete the drawing page by first folding the page vertically in half so that only the blank space at the top is visible (left of Figure 3.3). Using the viewing page for reference, the artist draws the prompt shape in the blank space, just as if they were making a normal sketch. When finished, the artist unfolds the drawing page and copies their freehand drawing onto the faint image on the bottom of the same page. During the copying step, the artist is asked to change the shape of their lines to match the target rendering, but not to change the number or relative position of the lines. In effect, the artist is asked to perform a non-linear warp of their original drawing onto the target shape. A typical result is shown on the right side of Figure 3.3.

We scan the drawing page with a flat-bed scanner, locate fiducials included in the corners of the page, and then use the fiducials to register the traced lines with the 3D model rendered from the main viewpoint. An adaptive thresholding method is used to convert the scanned gray-scale image into a binary image so that all the artist's lines, regardless of strength, are included in the binary image. We then use a thinning operator to narrow the lines in the binary image down to the width of one pixel. The final result is a 1024×768 pixel binary image containing a single pixel wide approximation of the human artist's lines.

While this procedure takes up to twice as long as a single drawing (e.g., it requires the artist to draw every line twice), it achieves a nice balance between the design trade-offs: the line drawings are composed in a free-hand manner familiar to artists, while the intended locations of every line on the 3D surface can be inferred with great accuracy.

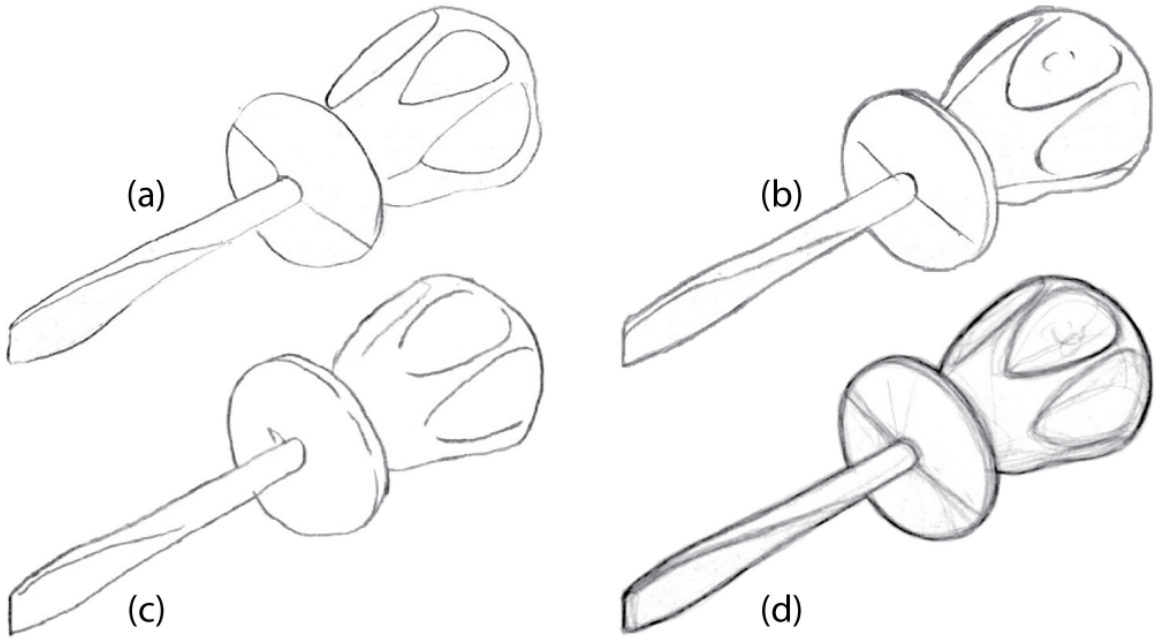


Figure 3.4: *Example drawings.* Three drawings of the screwdriver model from the same view (a,b,c), and the average of 14 drawings of the same view (d).

3.1.4 Data Collection

This line drawing and registration procedure was repeated for 29 artists, most of whom were enrolled in one of four art classes (two composed of middle and high school students, one of adult evening students, and another of college students). Two of the participants were professional artists. Each artist completed up to 12 prompts.

Every participant completed a questionnaire listing his/her gender, age, and number of years of art training. In all, there were 22 females and 7 males. The ages ranged from 10 to 54 years, with an average of 22; and the participants reported an average of 6 years of art training (this number should be taken with a grain of salt, as some participants reported only training at the college level, while others reported all art classes).

Every artist was provided a folder with one page of instructions, twelve prompt pages, and twelve corresponding drawing pages (one for each model). The folders were arranged such that no artist could draw the same model more than once, and prompts for models, viewpoints, and lighting conditions were arranged in shuffled order to reduce effects of training on our analysis.

The artists were given brief verbal instructions ("draw lines that convey shape" and "be sure to copy every line from your free-hand drawing over the faded image below") and then told to complete line drawings at their own pace for as long as they had time. Most of the art classes were scheduled for a two hour block, and each line drawing took 10-15 minutes, on average (with time split around 2/3 for drawing free-hand and 1/3 for tracing lines over the faded image). Each participant completed an average of 7.5 drawings – only one participant (a professional artist) completed all twelve available in his folder.

In all, 208 line drawing images were collected. Generally speaking, the artists followed the directions well, produced line drawings that convey shape effectively, and were careful when tracing lines over the faded image (some example line drawings are shown in Figure 3.4). However, in some cases, the artists clearly were not careful in the registration step, failing to follow even the exterior outline of the shape. Since accurate registration of lines to image features is essential for meaningful results, we cull these tracings from our analysis. To do this in an unbiased way, we assume that inclusion of the exterior outline is common to all human line drawings, and eliminate from our data set any drawings where less than 90% of the exterior is within 1mm of a human-drawn line. The remaining 170 line drawings form the basis for our analysis.

3.2 Results

We can investigate a number of questions by comparing how our captured line drawings overlap with the synthetic images provided as prompts to the artists.

We ask not only how artists' drawings overlap with one another, but also how they overlap with lines generated by computer graphics algorithms, and how they can be predicted from local properties of the underlying surface and rendered image. The latter two topics are of particular interest for computer graphics, as they provide a characterization of artist line drawings in terms of line definitions (e.g., this drawing is X% occluding contours, Y% suggestive contours, Z% apparent ridges, and so on) and differential properties commonly used in the field (e.g., there is a high propensity for lines when the view dependent curvature is large and its derivative is zero). We believe that characterizing the relationships between artists' drawings and these terms is the most interesting aspect of our study.

All comparisons between drawings are based on overlaps of pixels, rather than strokes or lines. This approximation is made for two practical reasons. First, since artists' drawings are scanned after they are complete, we have no robust method to tell where line strokes begin and end (an artist may make several small strokes that merge together into a single line). Second, since it is difficult to establish correspondences between lines robustly, there is no obviously good measure of the overlap between sets of lines. Rather, we compare line drawings based on proximity of pixels, an approximation that is both simple and robust.

3.2.1 How similar are the artists' drawings?

The first and most basic analysis we perform is to measure the similarity between artists' drawings of the same prompts.

We can show consistency between artists qualitatively by superposing drawings on top of each other and visualizing how much they overlap (Figure 5.1). For example, Figure 3.5a shows each artist's drawing in a separate color. In this example, the artists agree very closely with each other in most areas, especially along obvious features such as boundaries and occluding contours, but differ in exactly where they place lines in the right part of the rockerarm.

In order to quantify consistency, we compute a histogram of pairwise distances between artists' drawings (Figure 3.5b). For every pixel in every drawing, we record the distance to the closest pixel in every other drawing of the same prompt, and then observe how often these distances lie within the tolerance of the tracing procedure (1mm). Across all prompts, approximately 75% of human drawing pixels are within 1mm of a drawn pixel in all other drawings for that prompt.

3.2.2 Do known CG lines describe artists' lines?

A natural question to ask is how well currently known line drawing algorithms can describe the human artists' lines. In our analysis, we consider the following line drawing algorithms: image intensity edges [9], geometric ridges and valleys (as defined by [54]), suggestive contours [17], and apparent ridges [37]. For the object space methods (ridges and valleys, suggestive contours, and apparent ridges), we

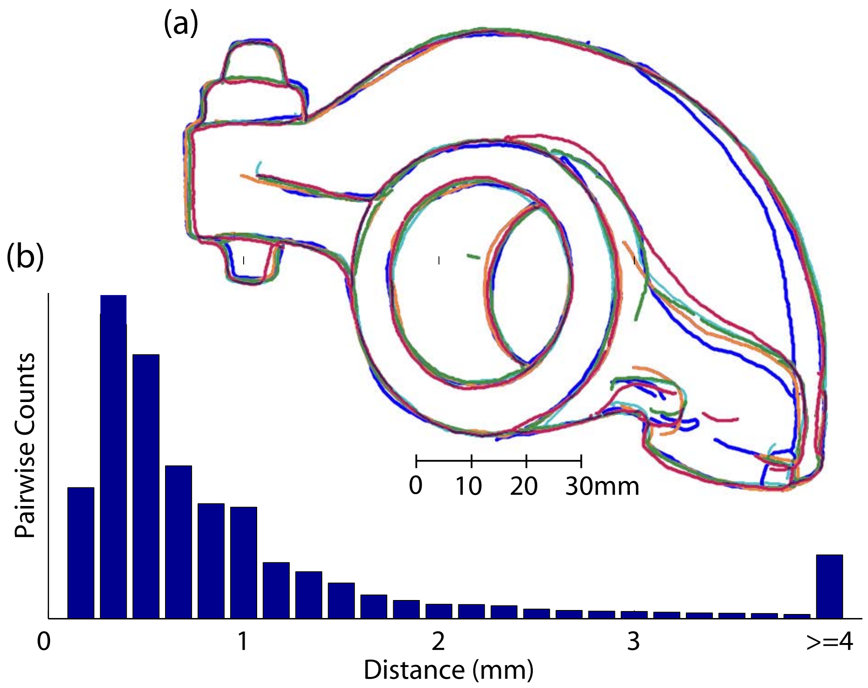


Figure 3.5: *Consistency of artists' lines.* a) Five superimposed drawings by different artists, each in a different color, showing that artists' lines tend to lie near each other. b) a histogram of pairwise closest distances between pixels for all 48 prompts. Note that approximately 75% of the distances are less than 1mm.

always include the exterior boundary and interior occluding contours in the generated drawing. For Canny edge detection we always include the exterior, but not the interior contours, since they are not necessarily image intensity edges.

Quantifying comparisons between drawings

In order to compare an artist's drawing and a computer generated drawing quantitatively, we use the standard information retrieval statistics of *precision* and *recall* (PR). Here, precision is defined as the fraction of pixels in the CG drawing that are near any pixel of the human drawing. Recall is defined as the fraction of pixels in the human drawing that are near any line of the CG drawing. We define "near" by choosing a distance threshold – we use 1mm.

As an example, consider comparing the set of five human drawings shown in Figure 3.5a with the lines generated by the apparent ridges algorithm (Figure 3.6). The output of the apparent ridges algorithm is not only a set of lines, but also a "strength" value at each line point. In general, we expect stronger lines to be more important and thus more likely to match the artists' lines. We thus generate a series of binary apparent ridges images, each consisting of all points with strength above a given threshold. The PR of each drawing compared with this set of images is shown as a dotted pink line in Figure 3.6. As the strength threshold is lowered more lines are produced, typically causing recall to increase and precision to go down, yielding a sloping line in the PR graph. For completeness, we allow the PR plot to extend to $P = 1.0$, $R = 0.0$ (defined as a blank image), and directly downward to $P = 0.0$ from the highest recall obtained by the algorithm. Since each PR curve is defined for $P = [0, 1]$, we can compute an average curve by combining points along lines of fixed precision. The PR values for occluding contours alone are plotted as black dots, and are not averaged.

While computing precision and recall for the other object space definitions

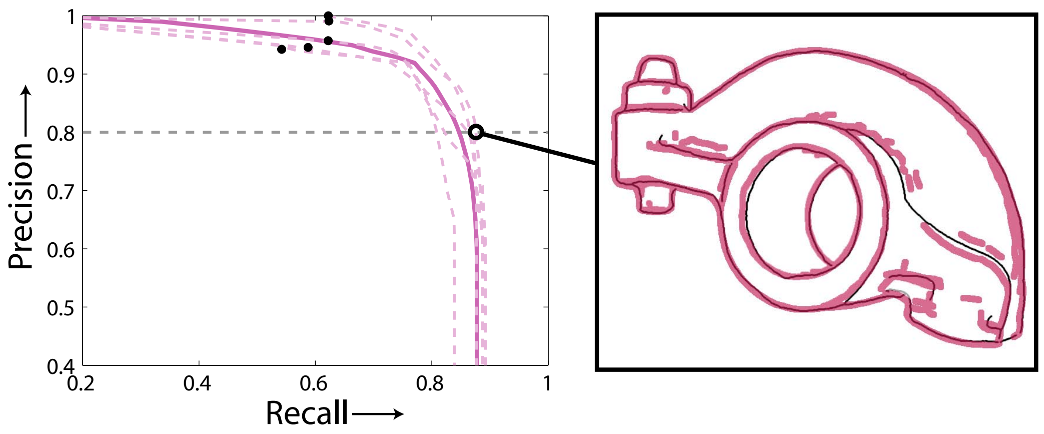
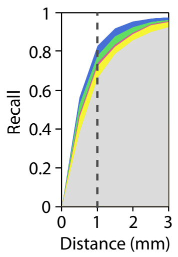


Figure 3.6: *Precision and recall example.* Left: apparent ridges are compared with five artist drawings. Solid line (highlighted) is the average PR for the set of drawings. Black dots indicate contours only. Right: an example drawing with overlapping apparent ridges (widened by 1mm on each side). PR of the example is circled.

is performed similarly, computing PR for the Canny algorithm is slightly more complicated. Canny also has a natural "strength" value (the intensity of the filter response), but the algorithm has three free parameters: the size of the image filter, and the low and high thresholds (l and h). For our analysis, we fix the filter σ at 2 pixels (a value we find produces reasonable results) and set $l = 0.4h$. We then vary h to control the number of line pixels produced by the edge detector.



In choosing the distance threshold, there is tension between achieving high recall and causing nearby, but distinct, line definitions to overlap. The figure inset right shows the cumulative recall over all drawings of lines explained by distinct (colored, as in Figure 3.7) versus overlapping (gray) definitions, as a function of threshold distance. We find that the threshold of 1mm provides a good balance between high recall and low overlap. Importantly, we find that while the exact recall numbers change somewhat with the distance threshold, the qualitative behavior and relative rankings of the CG lines do not.

Comparing CG lines individually

Figure 3.7 shows the average precision and recall for four representative models. The lines drawn on mechanical models such as the flange are classified readily into ridge-like features (green and pink), and are also explained well by Canny edges (yellow). The lines drawn on the cloth and bone models are largely occluding contours (black dots), though suggestive contours (blue) explain relatively more of the lines on these smooth models than on the other sets. The cubehole is almost completely explained by the CG methods. The tightness of the group of occluding contour dots gives a measure of how similar each drawing's PR is to the others, since the unaveraged PR lines (with the exception of Canny) pass through the contour dots.

We find that overall, Canny edges, apparent ridges, and geometric ridges and valleys best match the human lines when taken individually. No single definition matches the artists' drawings perfectly. Even in theory, however, no single CG algorithm with a single free parameter could match all the drawings, because the drawings are different from one another.

To gauge how far the CG algorithms could possibly improve, we imagine a computer algorithm to create an *optimal CG drawing* for a set of artists' drawings. The optimal drawing has the highest recall for a given precision of any possible CG drawing (dotted red in Figure 3.7). Thus, it both describes how closely the original drawings match each other, and puts a conservative ceiling on how well *any* CG algorithm can match the human drawings.

The optimal CG drawing for precision P is created by the following procedure: for each of n binary, thinned drawing images, make an image that contains all the pixels within 1mm of any drawn pixel. Add these images to create an overlap image with values in the range $[0, n]$. Sort the pixels of this overlap image, and choose pixels with the highest value until precision falls below P. The blank image is defined to have $P = 1.0$, so this procedure always produces a drawing. Since the value of a pixel in

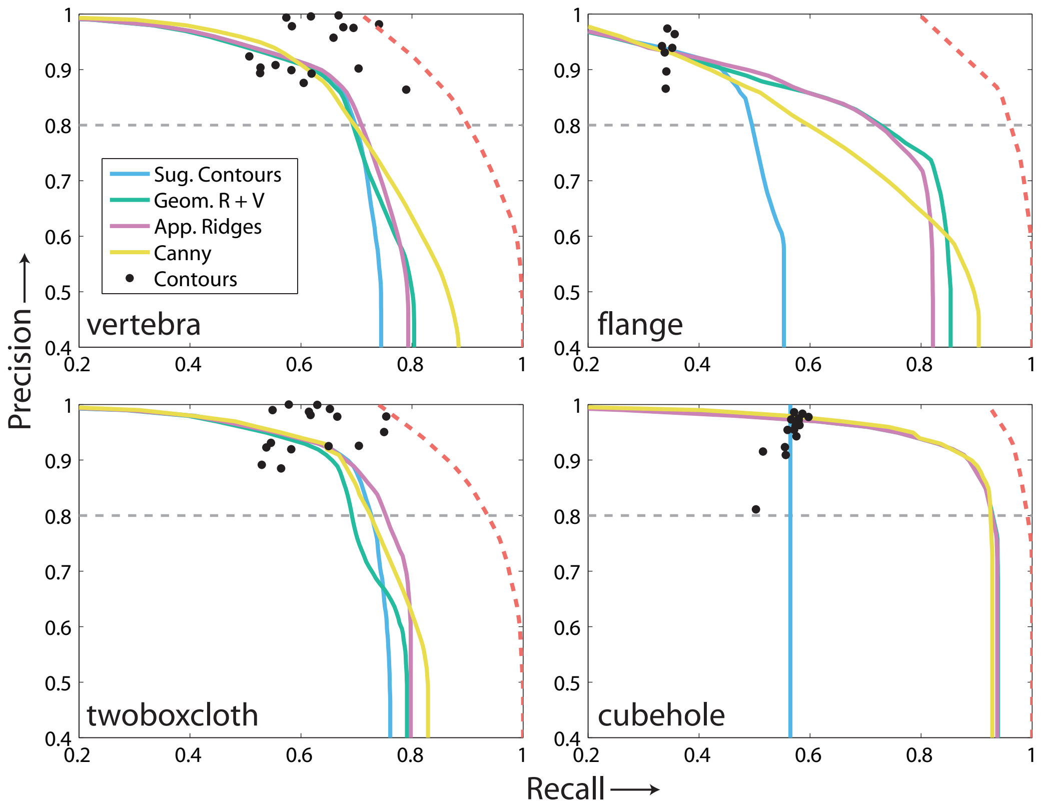


Figure 3.7: *Average precision and recall.* Lines further to the upper right represent better matches between the artists' and CG lines. The black dots represent occluding contours only, and are not averaged. The slope of the curve generally falls off rapidly after 80% precision. Red dotted line indicates theoretical maximum recall. Note: axes begin at 20% recall and 40% precision.

the overlap image is exactly the precision of that pixel if added to the optimal drawing (times n), this procedure will choose a drawing with the largest possible number of pixels for a given precision.

Comparing CG lines in combination

Even with the theoretical ceiling imposed by the optimal CG drawing, the individual CG algorithms do not match the artists' lines particularly closely. In almost all human drawings, however, there are examples of multiple classes of lines. For example, an artist might draw mostly along image intensity edges, but still draw other important features. It is thus interesting to consider how different line definitions might explain the human drawings in combination.

In order to combine line definitions fairly, we use computer generated drawings with a fixed 80% precision. We then classify each pixel in each human drawing by the nearby CG lines. Pixels that lie near a single line definition are considered to be explained only by that definition, while pixels that lie near multiple definitions are considered explained by all the nearby definitions.

To visualize the results we create bar charts that partition the lines into object space definitions (blue), image intensity edges (green), or both (brown). Looking at the results in Figure 3.8a, we find that the large majority of lines are described by both image intensity edges (Canny edges) and an object space definition. Of the remainder, slightly more lines are explained by the combined object space approaches than by image edges alone.

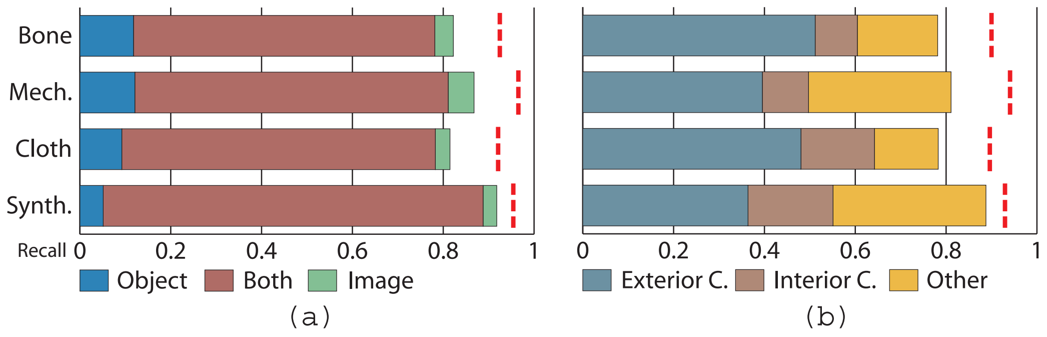


Figure 3.8: *Categorizing artists' lines.* a) fraction of all lines explained by image based lines only, object based lines only, and both. b) fraction of all lines explained by the exterior contours, interior occluding contours, and all other object space lines. Dotted red indicates theoretical maximum recall.

Lines that are explained only by image edges account for at most 5% of all classified lines at 80% precision. We can therefore learn a good amount by examining the object space lines alone. Analyzing object space lines is also more informative than analyzing image edges, since the different definitions correspond to familiar geometric concepts. For example, we can break down the human lines by intuitive categories, such as exterior and interior occluding contours, and everything else (Figure 3.8b). Across all model groups, exterior contours alone account for between 35-50% of all classified pixels. Interior occluding contours account for between 10-20% of all classified pixels, while all other definitions make up 20-35%.

If we look at the object space lines that are not exterior or interior contours, we see that in the mechanical and synthetic models, ridge and valley like features dominate the remaining lines (green, yellow, and pink in Figure 3.9). Both apparent ridges and geometric ridges and valleys contribute alone, but the majority of ridge like features are classified both as apparent and geometric.

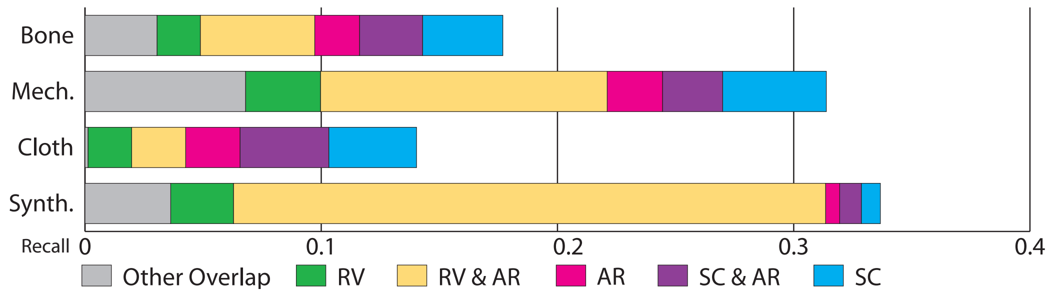


Figure 3.9: *Non-contour lines.* Categorization of artists' lines that are not exterior or interior occluding contours: geometric ridges and valleys (RV), apparent ridges (AR), suggestive contours (SC), and combinations.

For the bone and cloth models, ridges and valleys are less important, though the overall total of non-contour lines is approximately half that of the mechanical and synthetic models. One particularly interesting combination of object space lines is suggestive contours and apparent ridges (purple in Figure 3.9). Both suggestive contours and apparent ridges extend contours, but are largely disjoint elsewhere. Lines that are classified as both suggestive contours and apparent ridges, therefore, are likely to be extensions of occluding contours. As might be expected, the folds in the synthetic cloth lead to a disproportionately high amount of this combination.

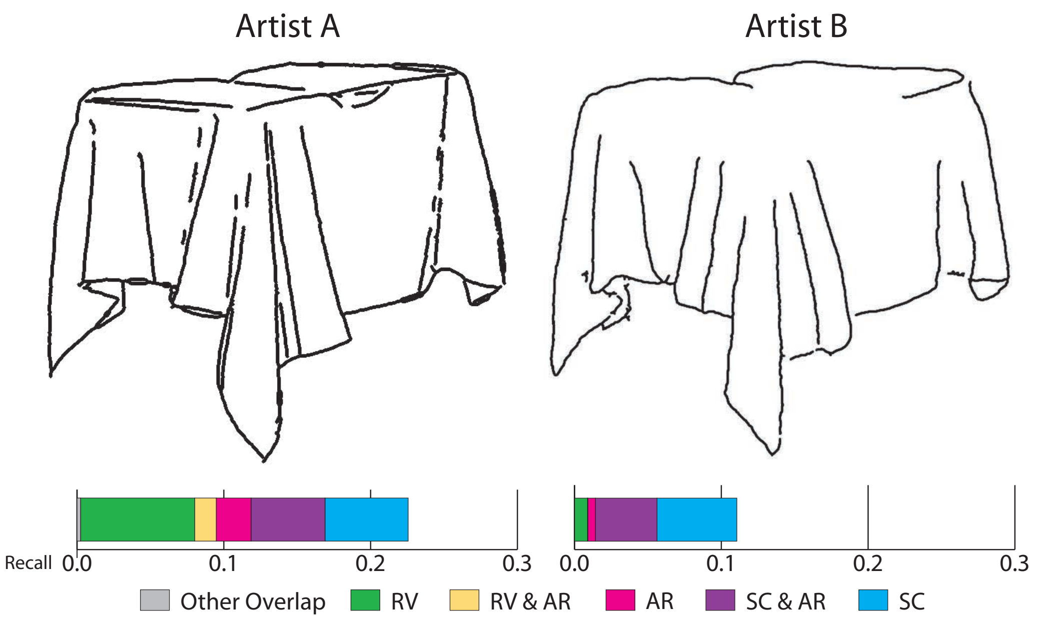


Figure 3.10: *Comparison of two drawings by different artists.* Two drawings of the same prompt show significant visual differences. These differences are reflected in the statistics, especially in the use of ridge-like lines (green).

3.2.3 Can CG lines characterize artists' tendencies?

Given a way of describing an artist's drawing in terms of CG line types, it is possible to investigate whether those descriptions can characterize the similarities and difference between artists' styles or tendencies. For example, it may be possible to characterize whether certain artists tend to draw certain geometric features (e.g., ridges) more than other artists do. In such cases, the CG line definitions provide a vocabulary to discuss features of human line drawings.

Figure 3.10 shows a simple example of this type. Two drawings of the same prompt (twoboxcloth with Grace Cathedral lighting) are compared by the composition of CG line types. As in Figure 3.9, the colored bars indicate the fraction of the drawing made up by each line type. In this case, however, each set of bars represents a single drawing. One immediate difference between the drawings is that artist A drew more lines besides the contours. Non-contour lines account for 26% of artist A's drawing, and only 13% of artist B's drawing. The bulk of the difference between the artists is in the use of ridge-like lines (green, yellow, and pink bars). Artist A drew ridge-like lines along the top of the shape, while artist B did not. This visual difference is evident from the statistics, which show a large fraction of geometric ridges and apparent ridges in artist A's drawing, and almost none in artist B's drawing.

While this analysis is instructive in some cases, we find that some individual artists appear to have consistent tendencies that are not well explained by the CG lines examined here. For example, artist C made seven drawings, in which 16% of the lines are unexplained by the tested CG definitions at 80% precision. Over the same seven prompts, all other artists averaged only 8% unexplained lines. As shown by the examples in Figure 3.11, artist C made consistently distinct drawings. Artists with different styles, such as artist C, may provide valuable data for future research on line definitions and shape perception.

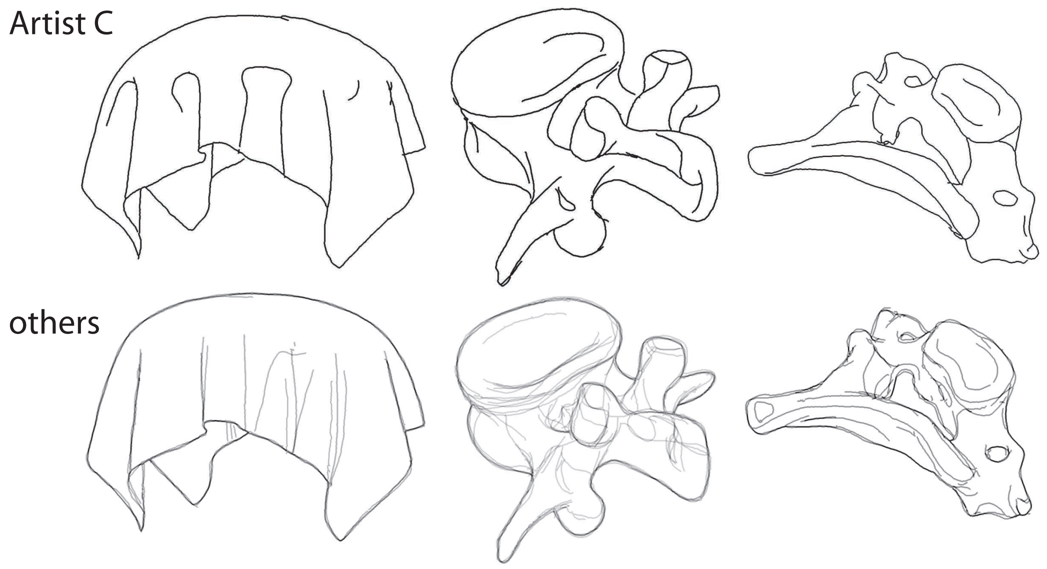


Figure 3.11: *Unusual drawings by an individual artist.* CG definitions explain fewer lines in artist C's drawings than other artists' drawings of the same prompts. However, artist C's drawings are careful and consistent.

3.2.4 Can combined local properties explain lines?

While it is interesting to investigate the relationship between artists' lines and the lines commonly used in computer graphics, a more fundamental question is how artists' lines relate to differential properties of images and surfaces. The analysis above addresses this question indirectly, since each CG definition is based on a set of local properties, but it is restricted to the relationships suggested by the known line drawing algorithms.

To address this question, we take a classic data mining approach. For every pixel of every prompt, we compute: 1) a feature vector (x) of properties derived from the 3D surface and 2D rendered image, and 2) an estimated probability that a line will be included at the corresponding location in an artist's line drawing. Our goals are to learn a function $f(x)$ that estimates the probability p of an artist drawing at a point (regression) and to understand which combinations of properties are most useful for building such a function (feature importance).

Choosing local properties

To build the feature vector for each pixel, we compute 15 local properties of three types commonly used in image processing, computer graphics, and differential geometry. First, we consider four image-space properties of the photorealistically-rendered image prompt: the luminance, the gradient magnitude after a Gaussian blur with σ=2 pixels (ImgGradMag), and the minimum and maximum eigenvalues of the image Hessian (corresponding to the minimum and maximum directional second derivative of luminance (ImgMinCurv and ImgMaxCurv, respectively). In general, we expect that lines are more likely near image edges (ImgGradMag is large) and at ridges and valleys of luminance (where ImgMinCurv and ImgMaxCurve are large).

Second, we consider view-independent, differential properties of the visible point on the 3D surface, including the maximum (κ_1), minimum (κ_2), mean ($(\kappa_1 + \kappa_2)/2$), and Gaussian ($\kappa_1\kappa_2$) curvatures (SurfMaxCurv, SurfMinCurv, SurfMeanCurv, and SurfGaussianCurv, respectively). In most cases, we expect lines to occur in areas where these expressions are large, though it has also been observed that lines are drawn near parabolic lines ($\kappa_1\kappa_2 = 0$).

Third, we consider view-dependent properties that correspond to specific definitions for computer-generated lines. Corresponding to the definition of ridges and valleys, we take the derivative of the largest principal curvature in the corresponding principal direction (SurfMaxCurvDeriv), which is zero at ridges and valleys. Corresponding to occluding contours, we compute the dot product between normal and view vectors ($N \cdot V$). Corresponding to apparent ridges and valleys, we compute the largest view-dependent principal curvature (ViewDepCurv) and its derivative in the corresponding apparent principal direction (ViewDepCurvDeriv), which are large and zero, respectively, at apparent ridges. Corresponding to suggestive contours, we compute the radial curvature (RadialCurv) and its derivative in the radial direction (RadialCurvDeriv), which are zero and large, respectively, at suggestive contours. Finally, corresponding to principal highlights, we compute radial torsion, which is zero at principal highlights.

Finally, we estimate the probability, p, of an artist drawing at a pixel by averaging the registered drawings of all artists for the same prompt and blurring with a Gaussian filter to account for tracing errors ($\sigma = 0.5$mm).

Predicting lines by regression

While several of the computed properties clearly can be used to distinguish pixels where artists draw from where they do not (Figure 3.12), the interesting question is whether combinations of those properties can be used to predict where artists will draw more accurately than any of them alone. To investigate this question we have experimented with several regression models, including linear regression, radial basis functions, regression trees, and several others.

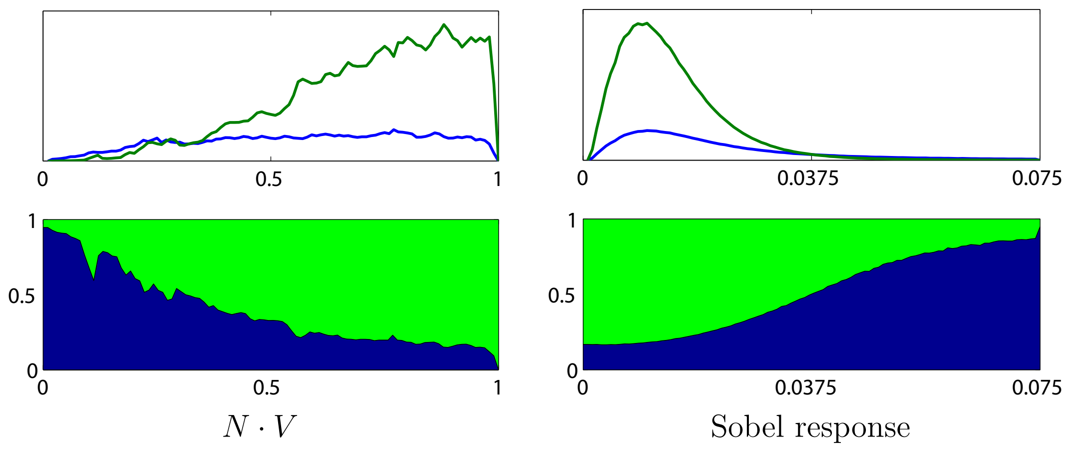


Figure 3.12: *Example local surface features.* Top: the relative frequencies of pixels near artists' lines (blue), and away from artists' lines (green). Bottom: the same data, but shown as probabilities.

As an example, Figure 3.13a shows a regression tree built with the M5P package in Weka [81] to predict the set of line drawings for one view of the twoboxcloth model shown in Figure 3.2. In this visualization, branches of the tree are shown as conditionals proceeding down the tree (indentation indicates level of the tree). If the conditional at a branch is TRUE, then its descendent on the next line is evaluated; otherwise, the one lower in the tree and connected by vertical bars is visited. Leaves are drawn with two text strings, the first one (colored) indicates a predicted value of p, while the second provides the count of pixels mapping to that leaf during training. The two images to the right of the tree show how the tree is used for regression – every pixel is sorted into the tree based on its properties and assigned the value stored at the leaf. Figure 3.13a shows the prediction of p resulting from this simple tree, while Figure 3.13b provides a visualization of which pixels sort into which leaves of the tree (pixels in the image are colored to match the text of the leaf).

In this example, several properties are combined by the decision tree to predict p, including ImgGradMag, RadialCurvDeriv, ViewDepCurvDeriv, SurfGaussianCurv, ViewDepCurv, $N \cdot V$, SurfMaxCurvDeriv, and SurfMinCurv. The set of properties

```
ImgGradMag > 2433.08
| ImgGradMag > 4706.19
| | 0.181 (43,128)
| | 0.0341 (33,744)
| RadialCurvDeriv > 0.02
| | ViewDepCurvDeriv > 0.044
| | | 0.0455 (7,761)
| | | 0.0175 (7,593)
| | SurfGaussianCurv > -0.004
| | | ViewDepCurv > 0.076
| | | | N dot V > 0.782
| | | | | 0.0113 (27,737)
| | | | | 0.0252 (8,641)
| | | | SurfMaxCurvDeriv > 0.014
| | | | | SurfMinCurv > 0.022
| | | | | | 0.0044 (1,602)
| | | | | | 0.0125 (32,420)
| | | | | 0.0023 (617,255)
| | | 0.0229 (6,551)
```

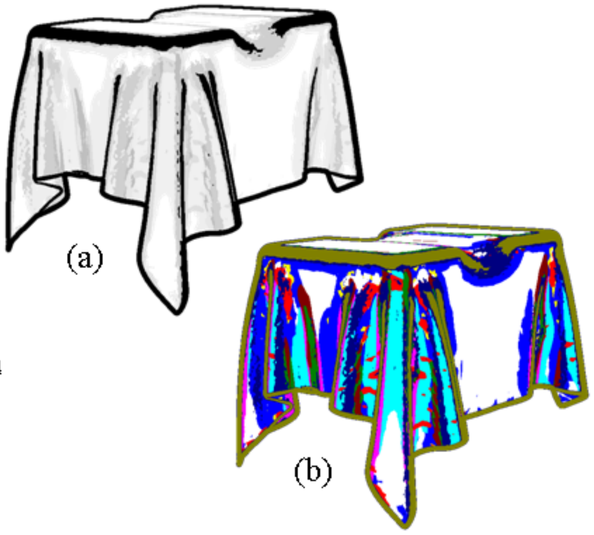


Figure 3.13: *Decision tree for predicting where artists will draw.* (left) decision tree learned from prompts of bones, (a) predicted probabilities of where artists will draw for this view (black is high probability), (b) a visualization of which pixels fall into which leaves of the tree. Note that this tree was purposely kept small for didactic purposes, which causes the prediction to be coarse.

chosen is instructive, as it suggests that they provide the highest incremental value in predicting p (at the start of tree building). Of course, many properties are correlated, and the decision tree may be non-optimal, so an alternative tree may have produced similar or better predictions. None the less, it is interesting to see how non-trivial combinations of local properties can be used to make predictions – even though the tree was purposely kept small in this example, it still is able to provide a plausible (albeit coarse) prediction for where artists draw lines (Figure 3.13a). If we consider deeper trees or other regression models, we are able to predict p from x more accurately.

It is also possible to use image processing tools to find ridges in the predicted probability distribution to produce a new line drawing. In the case shown in Figure 3.14a, we predict the image p (top left) using linear regression. We observe that this synthesis qualitatively resembles the composite of artists' drawings (Figure 3.14b). We then extract lines with a ridge finding algorithm to produce the

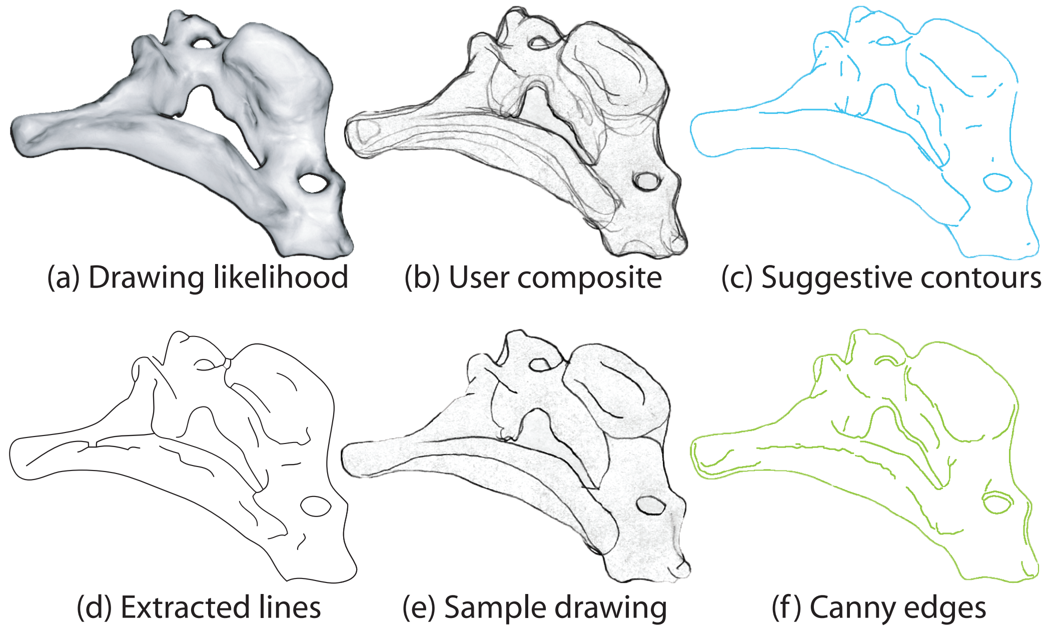


Figure 3.14: *Synthesis.* Linear regression on the cervical model, trained on drawings of other bone models, estimates the likelihood of an artist's line appearing at each pixel (a). The likelihood image resembles the overlaid artists' drawings of the same prompt (b). To synthesize a line drawing, ridges are extracted from the likelihood image(d). This synthesis qualitatively compares with a sample human drawing (e). Note that the synthesis could not have emerged from independently analyzing the CG lines shown in (c),(f)

line drawing in Figure 3.14d. This line drawing is comparable to a sample artist's drawing from the data set (Figure 3.14e) and contains elements from multiple CG line definitions. For comparison, the right two images (Figure 3.14c,f) show suggestive contours and Canny edges for the same prompt with parameter settings tuned to produce approximately the same line density as the artist's drawing. Since the probability predicted by our model combines the differential properties that lay the foundations of these other algorithms, the fact that it exhibits features from more than one of them is not surprising. However, it provides a practical way to combine features from many automatic line drawing algorithms into a single framework where thresholds are learned automatically.

3.2.5 Which local properties are most important?

In our data mining framework, it is not only possible to predict where artists will draw, but also to examine which local features are most important when building such a regression model. For example, Random Forests [6] estimate the importance of every feature to its model by building a large number of decision trees trained on different subsets of the data [6]. For each feature m of each built tree, the error observed in predictions for the "out of bag" data (the part held out of training) is computed and compared to the error that is observed when values of feature m are permuted. The difference between these errors, averaged and normalized, is reported as the "importance" of feature m. Of course, the importance only measures an average over many trees, and so it does not capture the importance of any single feature at any single branch in the tree. Yet, it is interesting to use importance estimates to study how much low-level features contribute on average to predictions of line drawing locations. For this analysis, we make the assumption that almost all occluding contours ($N \cdot V = 0$) are drawn by artists (Figure 3.12),and so exclude any pixel within 1mm of a contour from the training set.

Table 3.1 shows the relative feature importance as computed with the Random Forest implementation of Breiman and Cutler in R [61] for the remaining pixels of all drawings in our study. The first four columns report the importance of features (rows) estimated when training on models from each of one type (bones, cloth, mechanical, and synthetic), while the rightmost column reports the average over the whole dataset.

The results indicate that image-space intensity gradient magnitude is the feature amongst the tested set that is most useful in predicting the probability that an artist will draw at a particular location in our study (e.g., the average prediction error is largest if values of the image-space gradient magnitude are randomized). While image-space discontinuities often appear at the same place as boundary contours and occluding contours ($N \cdot V = 0$), the locations where those contours appear have been excluded from this study. So, this result suggests that image-space intensity gradients away from the contours are also highly correlated with artist line locations. Of course, this is not surprising, as ridges, valleys, and shadow boundaries are commonly drawn by artists. However, it is a bit surprising how all the simple image-space features (which do not require a 3D model to compute) are so important relative to the other more complex properties that have been the focus of recent research in computer graphics.

	Feature	Bone	Cloth	Mech	Synth	Avg
Image Space	ImgGradMag	31.3	36.0	73.8	147.8	72.2
	ImgMaxCurv	38.0	15.8	55.5	64.4	43.4
	ImgMinCurv	15.1	15.3	23.4	56.6	27.6
	ImgLuminance	20.2	19.8	33.9	33.6	26.9
View Dependent	$N \cdot V$	23.6	13.9	31.3	36.9	26.4
	ViewDepCurv	21.5	17.2	49.8	10.1	24.7
	ViewDepCurvDeriv	22.8	14.4	31.9	9.5	19.7
	RadialCurvDeriv	19.2	15.0	29.8	8.0	18.0
	RadialTorsion	14.6	10.3	27.8	7.2	15.0
	RadialCurv	14.8	10.3	26.2	7.2	14.6
View Independent	SurfMaxCurvDeriv	16.9	11.0	27.3	8.9	16.0
	SurfMaxCurv	13.9	8.8	25.1	7.6	13.9
	SurfMinCurv	13.9	8.1	27.0	5.1	13.5
	SurfMeanCurv	14.1	8.9	22.5	7.0	13.1
	SurfGaussianCurv	13.1	8.5	25.7	4.9	13.1

Table 3.1: *Property "importance."* This table shows the relative importance of local properties in predicting the probability of an artist drawing at a particular location. Columns show results for Random Forests trained on different subsets of the data.

3.2.6 Which CG lines are most important?

We can also use Random Forests to compute importance of the computer graphics line definitions studied in Section 3.2.2 for predicting where artists draw lines. For this analysis, we compute a new feature vector for every pixel storing the strength for every CG line definition. Note that strength is only defined at pixels where the algorithm would draw a line (e.g., zeros of maximum curvature derivative for ridges). At all other pixels, strength is always zero. We then recompute the Random Forests with the new feature vectors.

From the results in Table 3.2, we see that strong image-space gradients in illumination (Canny edges) still provide the strongest cues for artists to draw lines, even in relation to other computer graphics line definitions.

Feature	Bone	Cloth	Mech	Synth	Avg
Canny edges	18.2	37.2	50.9	145.0	53.4
Apparent Ridges	8.7	11.2	21.2	77.9	24.8
Ridges & valleys	6.8	7.4	24.4	77.1	24.5
Suggestive Contours	9.8	11.9	17.4	1.6	11.3

Table 3.2: *CG line definition "importance."* This table shows results similar to thoe ones in Table 3.1, but for features derived directly from CG line definitions.

3.3 Conclusion

Overall, we make the following conclusions from this study, some of which are obvious and others of which are not.

First, we observe that artists in our study draw nearly 75% of their lines at a location that is within 1mm of *all* other artists drawing from the same prompt. The overlaps appear mainly at exterior and interior occluding contours, which comprise 57% of all lines drawn.

Amongst the other lines, large gradients in image intensity (as measured by image-space gradient magnitude) provide the best single predictor for where artists will draw under the conditions of our study. Lines generated by Canny edge detection on a prompt image cover 76% of artists' lines with 80% precision. These lines are almost entirely overlapped (95%) by lines predicted by object-space line definitions commonly found in computer graphics. The three object space definitions together cover 81% of the artists' lines at the same precision. We find that each of the four CG line definitions we considered explains some artists' lines that the others do not.

The cumulative output of the four line drawing algorithms considered here cover only 86% of artists lines. We believe that some of the remaining lines could be explained by other line definitions based on local properties (e.g., lines from diffuse shading [45], or a new definition net yet described), or by clever combinations of current line definitions.

In some cases, however, it appears that artists select lines using criteria beyond the local features we examine here. For example, Figure 3.15 shows two cases where artists chose to draw lines on locally weak ridge and valley features, while omitting lines along locally stronger ridges and valleys. This choice is consistent between several artists.

While it may be that local surface features that we have not examined (or a combination of them) could explain the artists' choices in these cases, we believe it is more likely that the artists used non-local criteria for selecting these lines. Indeed, several artists mentioned that, for the flange drawing (Figure 3.15a), they omitted lines that were "implied." Since implied features depend on context, they are not describable with local properties alone.

Limitations and Future Work

We identify the following limitations of our study, which suggest topics for further work:

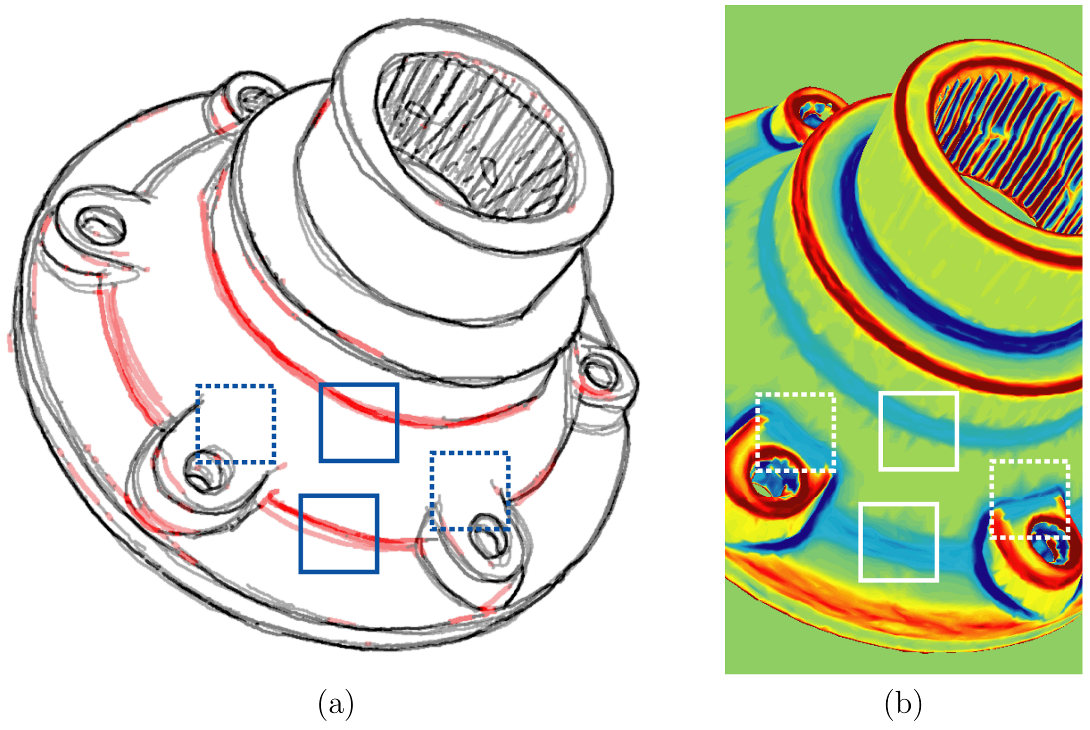
Potential bias. The possibility exists that our results carry some bias due to the way we collected data. We believe that artists generally followed the given instructions and faithfully copied their drawings from the drawing area to the tracing area, but it is possible that they altered their lines when tracing over a faint version of the prompt. Such alteration could contribute to the relatively high importance of image-space features noted in our analysis (Sections 3.2.5 and 3.2.6). Note, however, that the same image-space features in the faint images also appear in the prompt images.

Limited data. This section draws a number of conclusions from the limited data set we have acquired to date. Of course, those conclusions are limited to the conditions of our study and the artists participating. In the future, we hope to expand this study to provide more drawings per prompt and cover a greater range of subjects. More data would reduce noise in the analysis, support a greater range of analyses, and offer greater predictive power in machine-learning approaches to synthesis.

This chapter does not exhaust the possible analyses that could be performed with the data presented here. Other possibilities include:

Local per-pixel analysis. Our analysis has to date only studied per pixel properties of the strokes. We believe that studying such properties along the lengths of strokes will be both challenging and fruitful. Moreover, it is known that humans generally need a *global* view of a line drawing in order to fully understand it [75]. Likewise, Figure 3.15 suggests a global analysis might yield a better model for the human lines.

View-dependence of lines. The two viewpoints selected for our prompts are close enough to allow analysis of how lines move with changing views. One possible way to approach this question would be to reproject lines drawn from one view to compare with lines drawn in the other view.



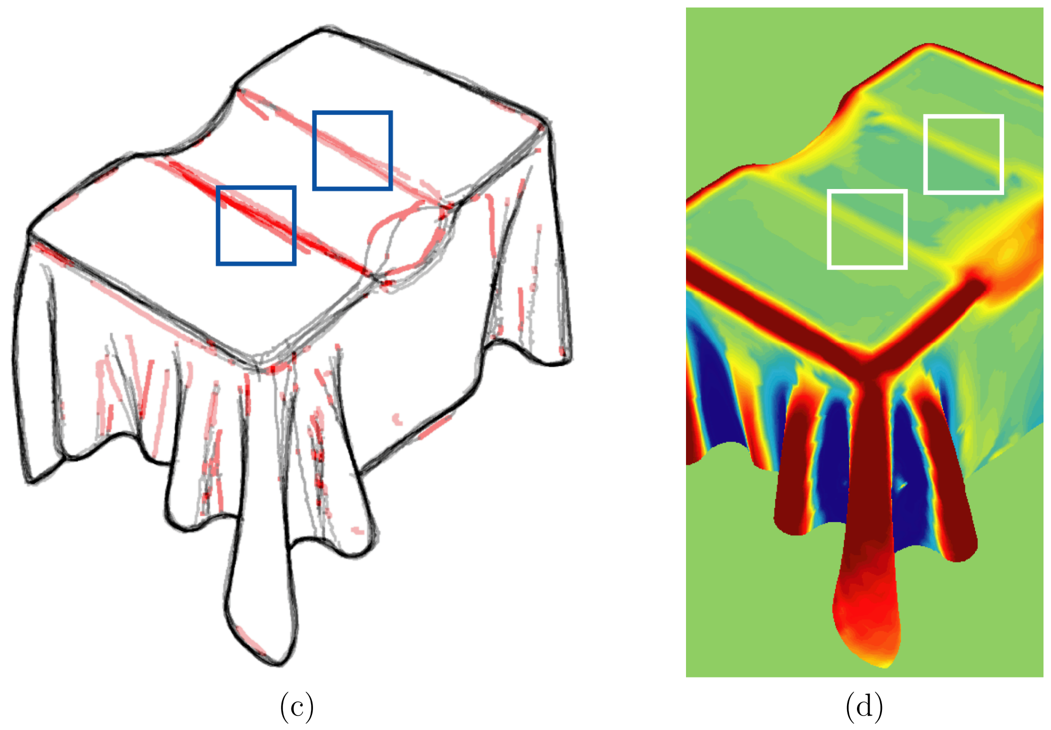


Figure 3.15: *Artists' sophisticated line selection.* The red lines (solid boxes) in this composite (a) are unexplained at 80% precision, but can be characterized as geometric valleys. However, several artists have omitted locally stronger valley features (dotted boxes), as shown by the maximum curvature of the model (b). The red lines in (c) are also ridge like features, but the curvature strength in the area is very low relative to the rest of the model (d).

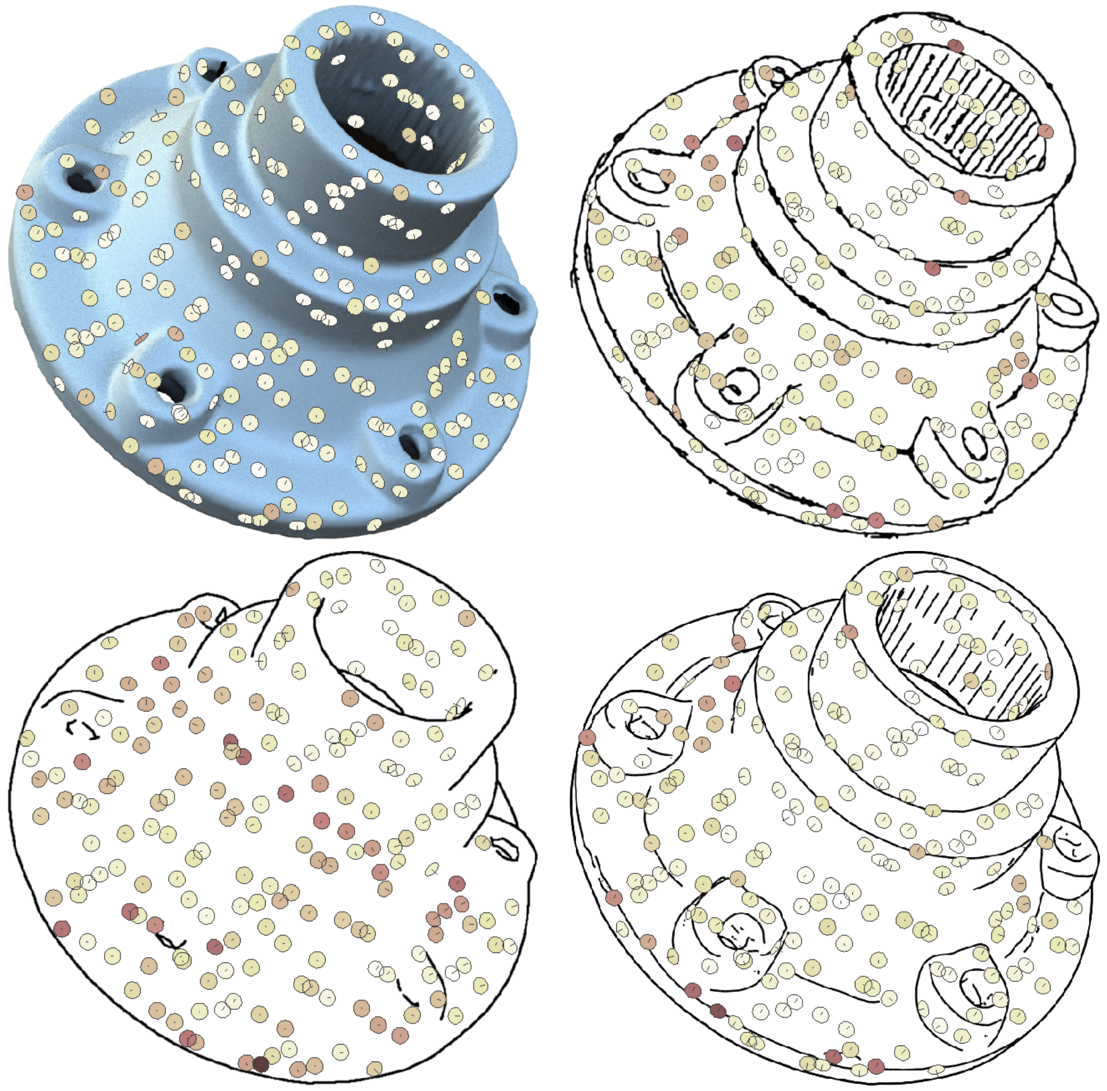
Chapter 4

How Well Do Line Drawings Depict Shape?

Common experience tells us that people generally agree about what shape they see in a wide range of line drawings; that some line drawings are more effective than others at conveying shape; and that some shapes are difficult to draw effectively. However, there is little scientific evidence in the literature for these observations. Moreover, as mentioned in Section 2.2 automatic algorithms for line drawing are a popular topic for reseach, but to date no objective way has existed to evaluate the effectiveness of such algorithms in depicting shape.

At first, it seems difficult to study how well people interpret the shapes depicted by line drawings. Aside from asking sculpters to craft the shape they see, how can we know what shape is in a person's mind? Koenderink et al. [40] have proposed several strategies for providing evidence about perceived geometry, based on the collective results of asking people many simple questions.

This chapter describes such a study, based on the gauge figure method, in which the subject is asked to rotate a *gauge* (see Figure 4.3) until it appears to be tangent to the surface, providing a perceived surface normal at that point. Studies of shape



(a) shaded image (b) human drawing
(c) contours (d) apparent ridges

Figure 4.1: Gauge figure results. In this study different people were shown six different renderings of a shape: (a) a shaded image, (b) a line drawing made from the shaded image by a person, (c) contours, (d) apparent ridges, and (shown in Figure 4.7) ridges/valleys and suggestive contours. Overlaid are representative "gauges" (discs revealing the surface normal) oriented on the images by people in the study, colored by how far they deviate from the ground truth.

using gauge figures and other related methodologies have used photorealistic images of diffusely shaded objects and shiny objects. With photos, researchers find that people see shape accurately (up to a family of shapes related by the *bas-relief ambiguity*).

This study offers the first substantial evidence that people can interpret shapes accurately when looking at *drawings*, and shows this for drawings made by both artists and computer graphics algorithms. Not all drawings are equally effective in this regard. We offer evidence that viewers interpret individual lines as conveying particular kinds of shape features (e.g. ridges, valleys, or inflections). Where different line drawing algorithms place different lines, the algorithms may be more or less effective at conveying the underlying shape. This study offers both statistical and anecdotal data regarding the performance of various algorithms and drawings created by hand, with the goal of informing future development of computer graphics algorithms.

These results have been made possible by two resources unavailable in earlier research. One is the dataset of line drawings described in Chapter 3. The other is the Amazon Mechanical Turk, which allowed us to distrbute the potentially tedious gauge figure task out to more than 500 subjects. Via this service we collected more than 275K gauge samples distributed over 72 images. This chapter only begins to analyze this data, and there is plenty more to learn about it. The data is publicly available to researchers for further analysis.

This chapter makes the following contributions:

- We show that different peoples' interpretations of line drawings are roughly as consistent as their interpretations of shaded images.
- We demonstrate that line drawings can be as effective as photorealistic renderings at depicting shape, and that not all line drawings are equally effective in this regard.

- We provide new evidence that mathematical descriptions of surface features are appropriate tools to derive lines to convey shape in drawings.
- We offer the first publicly available data set of gauge figures placed over a variety of drawings of 3D shapes; we believe this to be the largest gauge figure data set recorded to date.

4.1 Psychophysical Measurements of Shape

In order to understand the shape perceived when people look at a line drawing, we rely on the *gauge figure* technique from visual psychophysics to obtain local estimates of surface orientation at a large number of points spread over a picture [40, 42]. A gauge figure is simply a circle and a line in 3D, parameterized by slant (orientation in depth) and tilt (orientation in the image plane). When properly adjusted, it resembles a "thumb tack" sitting on the surface: its base sits in the tangent plane, and its "pin" is aligned with the outward-pointing surface normal. See Figure 4.3 for an example. Gauge studies can document not only shape interpretation but also the priors, bias, and information used by the human visual system. For instance, the direction of illumination affects the perceived shape [55, 8], and specular reflections can improve the perception of shape [21].

Most gauge figure studies consider diffuse imagery. The only precedent for gauge figure study with line drawings is [39], who presented a single shape rendered as a silhouette, a hand-crafted line drawing, and a shaded picture, and found that the percept was better from the line drawing than the silhouette, and nearly as good as the illuminated version. However, these results must be taken with a grain of salt, since the single shape was a sculpture of a very recognizable shape (a nude female torso).

To interpret our results, we also draw on the larger context of psychophysical research. For example, since people can successfully and consistently locate ridges and valleys on diffusely rendered surfaces in stereoscopic views [59], it seems likely that the visual system represents these features explicitly. Also, perceived shape is likely an interaction between the global organization of line drawings and inherent biases of the visual system [44, 49], such as preferences for convexity over concavity in certain kinds of ambiguous imagery.

A final wrinkle concerns the inherent underdetermination of depth from imagery. Given a shaded image, the 3D surface is determined only up to an affine transformation: individual points on the surface can slide along visual rays as long as planarity is preserved. This is the *bas-relief ambiguity* [4]. Thus, to show the veridicality of human percepts of diffuse images, it is necessary to correct for this ambiguity [42]. In cases where human perception is *not* successful, however, a bas-relief correction is problematic, because it uses the best fit between subjects' percepts and the true scene geometry.

4.2 Study

The study is designed to determine how viewers interpret line drawings and shaded images of the same shapes, and how consistent those interpretations are. The study is also designed to be broad enough to allow general conclusions about the effectiveness of line drawings. To achieve these goals, several issues must be decided: what images and drawings to show, how to sample each image with gauges, and how to structure the presentation to each participant so that a large amount of quality data can be gathered.

4.2.1 Subject Matter

One of the chief subjects of interest for this study is the effectiveness of line drawings by artists compared to computer generated line drawings. To compare human and computer generated drawings we use the dataset collected by Cole, et al. [13].

Besides offering artists' drawings registered to models, the dataset includes a useful range of 3D shapes. Intuitively, it seems that the usefulness of a line drawing varies with the type of shape depicted. For example, boxy shapes with hard edges are easy to depict with lines, while smoothly varying shapes are more difficult. We test this intuition by using the 12 models from the Cole dataset.

The study includes six different rendering styles: fully shaded, occluding contours only, apparent ridges [37], geometric ridges and valleys [54], suggestive contours [17], and a binarized human artist's drawing. The shaded image was created using global illumination and a completely diffuse BRDF with the eucalyptus grove HDR environment map [15]. For the cubehole model, the suggestive contour and apparent ridge styles are identical to the contour only and ridge and valley images, respectively, and are therefore omitted.

We endeavoured to represent each drawing style as kindly as possible. For computer generated drawings, we smoothed the model and chose thresholds to produce clean, smooth, continuous lines. For the human drawings, we chose the subjectively best drawing available for the model (Figure 4.6). Our choice of view for each model was dictated by the view of the best drawing. Example computer-generated and human drawings are shown in Figure 4.2.

4.2.2 Methodology

We follow the gauge placement protocol described by Koenderink et al.[40]. Participants are shown a series of gauges in random order and asked to place each gauge correctly before moving on to the next. The participants have no control over the

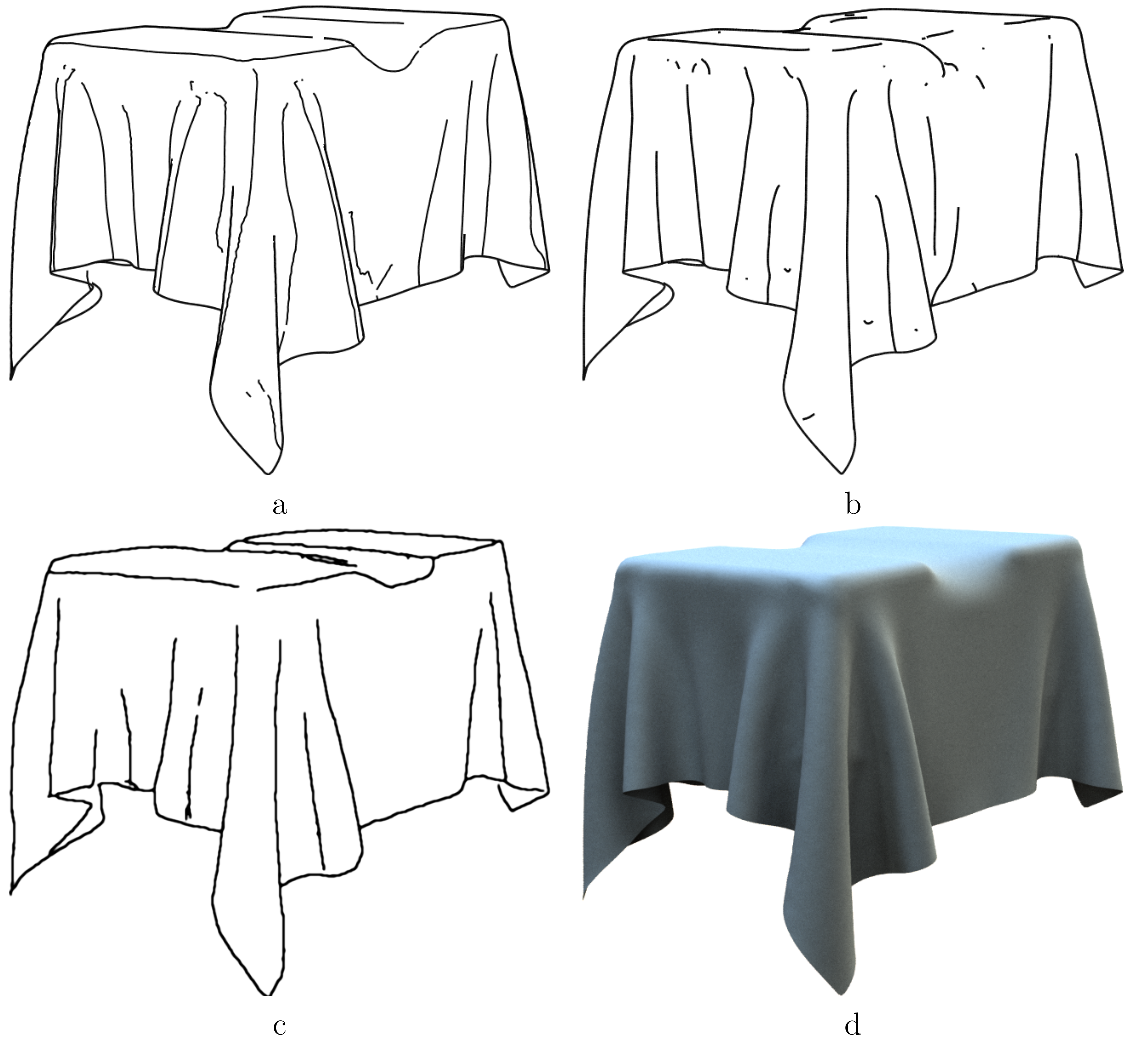


Figure 4.2: *Example prompts.* (a) ridges and valleys, (b) suggestive contours, (c) artist's drawing, (d) shaded image.

position of the gauge, only its orientation. Each gauge is drawn as a small ellipse representing a disc and a single line indicating the normal of the disc. The gauges are superimposed over the drawing images and colored green for visibility (Figure 4.3). To avoid cueing the participants to shape, the gauges do not penetrate or interact with the 3D model at all. The initial orientations of the gauges are random.

Participants were shown a simple example shape that is not included in our dataset in the instructions. The shape had examples of good and bad placement (Figure 4.3). Each time the participant started a session, they were allowed to practice orienting

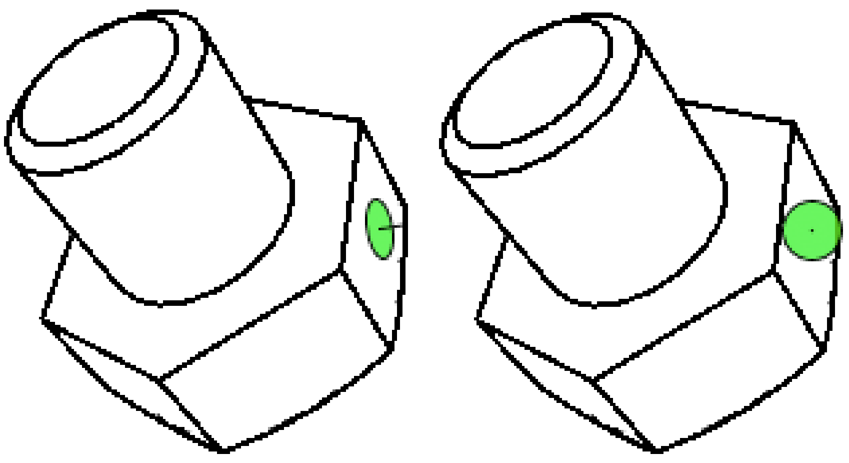


Figure 4.3: *Instructional example.* Participants were shown images such as this pair as part of the instructions for the study. Left: good gauge setting. Right: bad gauge setting.

gauges on the example shape before moving on to the actual task. Participants were asked to orient the gauge by dragging with the mouse, and to advance to the next gauge by pressing the space bar. Participants were shown the total number of gauges to place in the session and the number they had placed so far.

The placement of gauges for each shape is determined in advance and is the same across the different rendering styles. We place gauges in two ways: evenly across the entire model, and in tight strings across areas of particular interest.

The evenly spaced positions are generated by drawing positions from a quasi-random Halton sequence across the entire rectangular image, and rejecting samples that fall outside the silhouette of the shape. All 12 models have at least 90 quasi-random gauge positions. Four representative models – the flange, screwdriver, twoboxcloth, and vertebra – have 180 positions in order to better understand how error is localized.

Four densely sampled lines of gauges (*gauge strings*) are also included in the study, one each on the flange, screwdriver, twoboxcloth, and vertebra. The gauge strings consist of 15 gauges spaced by 5 pixels along a straight line in screen space.

4.2.3 Data Collection

Previous studies of this type have included small numbers of highly motivated subjects. Each subject in the 2001 study by Koenderink et al. [42], for example, was asked to perform approximately twelve hours of work, an impractical amount of time for any but a tiny number of subjects. In previous studies, the authors often provided all the data themselves. Our study has many subjects, but each is asked only a small number of questions. Rather than relying on the motivation of our subjects, we rely upon the robust statistics of many participants.

We used the Amazon Mechanical Turk as the source of participants in our study. The Mechanical Turk is a internet service that allows requesters (such as researchers) to create small, web-based tasks that may be performed by anonymous workers. Each worker is typically paid between \$0.05 and \$0.25 to complete a task. The number of workers on the service is such that particularly attractive tasks are usually completed within minutes. Unfortunately, workers on the Mechanical Turk are generally interested in work that takes around 5-10 minutes. This restriction dictates the number of gauges that we can present to a single participant. We found empirically that workers completed tasks at a satisfactory rate for up to 60 gauges in a session, but with more gauges workers became likely to give up on our task without completing it.

We asked each worker to set each gauge twice in order to estimate their precision. Setting reliability provides a measure of the perceptual naturalness or vividness of the observer's shape interpretation [22]. If the observer clearly and consistently perceives a specific shape in a line-drawing, then setting reliability will be high. If the percept is weak, and the observer has to "guess" to some extent, then reliability will be low. The 60 gauges in a session are two sets of the same 30 gauges, shuffled randomly and presented back-to-back. For simplicity, the sets of 30 are defined as consecutive sets in the Halton sequence for the 90 or 180 evenly spaced gauges. The statistics of each set of 30 are thus approximately equal.

To avoid training effects, each participant is allowed to see only a single style for each model. A participant is allowed repeat the study until they have performed the task once for each set of gauges. There are 52 distinct sets of 30 gauges across the 12 models, so a participant could theoretically perform the study 52 times.

Each worker is randomly assigned a stimulus from the available images each time they begin a new session. To favor more uniform sampling across the dataset, we weight more heavily the probability of receiving stimuli for which there is already little data, as follows. Out of the set of available stimuli we select randomly with probability proportional to $1/(k_i + 1)$ where k_i is the number of participants who have already successfully completed stimulus i.

Participants were told that the estimated time for completion was 5-10 minutes. The maximum time allowed for each session was one hour. The average time spent per session was 4 minutes.

In total, 560 people participated in our study and positioned a total of 275k gauges. The most active 20% of workers (115 people) account for approximately 75% of all data. The median number of sessions an individual performed was 4, but the median for the most active 20% was 28. The average time all individuals spent setting a gauge was 4 seconds.

4.2.4 Data Verification

Since we have no way of ensuring workers do a good job, we have to be careful to filter obviously bad data. Since we are asking for information about interpretation, however, there are no definitively wrong answers. We therefore base our rejection criteria on the reliability with which the worker positions each gauge.

We assume that if a worker is making a good faith effort, each duplicate set of gauges will be aligned in roughly the same orientation. Since each gauge is presented in a random orientation, a worker who simply speeds through our task will end up with their gauges positioned roughly randomly across the hemisphere. Therefore, if a worker positions fewer than 70% of the duplicate gauges within 30 degrees of each other, their data is rejected. These numbers were found empirically during pilot testing and remove grossly negligent workers without removing many good faith workers.

During pilot testing, we also noticed that through guile or misunderstanding some workers oriented all gauges in a single direction (usually directly towards the screen). This data passes our consistency check, but is useless. We therefore add an additional check, whereby a worker's data is discarded if the standard deviation of all gauge positions in a session is less than 5 degrees.

In all, approximately 80% of the data that we gather passes our two criteria and is included in the dataset. Each gauge in the study had an average of 15 opinions. The minimum number of opinions was 9, and the maximum was 29.

4.2.5 Perspective Compensation

The dataset of Cole et al. [13] includes drawings made from a camera with a vertical field of view of 30 degrees. To match the gauges most effectively with the shape, the gauges should be drawn with the same perspective projection. However, drawing gauges with perspective projection gives cues to the underlying shape. Instead, we draw the gauges with an orthographic projection and compensate after the fact for possible error due to projection.

The compensation method is as follows: create gauges for the ground truth normals, project the ellipses of those gauges by the camera projection matrix, reconstruct slant and tilt values from the projected ellipses, and finally reconstruct a normal from slant and tilt in the same way as the gauges set by the partcipants. Our comparisons against ground truth are made against these projected ground truth normals.

4.2.6 Compensation for Ambiguity

Koenderink et al. [42] found that different subjects perceptions of photographs of shape were related by bas-relief transformations [4]. As Koenderink did, we can factor out this transformation before making comparisons between different participant's responses.

The bas-relief transform for an orthographic camera looking down the z-axis maps the surface point $[x, y, f(x, y)]$ to $[x, y, \lambda f(x, y) + \mu x + \nu y]$, given parameters for depth scaling (λ) and shearing (μ, and ν). In determining the best fit of a set of gauges to ground truth data, we need to find appropriate values of μ, ν and λ that map the subject data to the ground truth.

Given a set of bas-relief parameters, we can transform a set of normal vectors (renormalizing them after the transformation). Thus, using a non-linear optimization procedure (which we initialize to the identity, $\lambda = 1$ and $\mu = \nu = 0$), we find the bas-relief parameters that minimize the L^1 norm of angular differences between the (normalized) transformed normals and the ground truth. We found the use of the L^1 norm to be robust to spurious gauges.

4.3 Results

Our data allows us to investigate several broad questions. First, how closely do people's interpretations of the stimuli match the ground truth shape? Second, how similar are two different peoples' interpretations of the same stimulus? Third, when compared with a shaded image, how effective are the line drawings in depicting shape? Fourth, are there particular areas for each model that cause errors in interpretation, and if so, can we describe them?

4.3.1 How well do people interpret shaded objects?

Before examining the results for line drawings, we investigate how well our participants were able to interpret the objects under full shading. We expect that people will perform most accurately on the shaded prompts, so the results for these prompts provide a natural way to determine how successfully the average Mechanical Turk worker performed our task.

Across all the shaded prompts, the mean error from ground truth is 24 degrees, with a standard deviation of 17 degrees. After bas-relief fitting, mean error is 21 degrees, with a standard deviation of 16 degrees (Table 4.1). Histograms of the errors for each style before and after bas-relief fitting are shown in Figure 4.4a and c.

For comparison, a worker placing gauges randomly in each of the same tasks would have a mean error of 66 degrees, with a standard deviation of 31 degrees. After bas-relief fitting, random data would have a mean of 42 degrees with a standard deviation of 19 degrees. A worker rotating all gauges to face the camera (a situation we mark as bad data) would have a mean error of 42 degrees before bas-relief fitting, and 40 degrees after.

The reliability or precision with which the participants positioned gauges can be measured by comparing against the median vector for that gauge. If a gauge has n settings v_i, the median vector is v_k that minimizes the total angular distance to every other v_i.

Given the median vectors for each gauge, we can plot the error from the median (Figure 4.4b and d). In the case of the shaded prompts, the mean error from the median vector was 16 degrees, with standard deviation 14 degrees(Table 4.2). These numbers do not change significantly with bas-relief fitting.

The scatter plots in Figure 4.5 give an alternate visualization of the distribution of errors for the shaded prompts. The orientations are shown using an equal-area azimuthal projection, so the area of each ring matches the area of the corresponding

Model	S	H	AR	RV	SC	C
cubehole	12	18	-	14	-	26
rockerarm	15	21	19	21	23	26
screwdriver	20	25	31	29	27	34
flange	21	26	25	22	32	32
pulley	21	29	27	29	29	30
bumps	22	29	27	27	36	36
femur	22	28	25	25	26	25
tooth	22	32	30	28	29	32
twoboxcloth	23	25	25	26	26	32
vertebra	24	38	42	35	37	42
cervical	25	37	35	35	37	38
lumpcloth	26	27	28	29	28	27
average	21	28	29	27	30	32

Table 4.1: *Mean errors from ground truth for each model and style.* Values shown are after bas-relief fitting. Rows are ordered by the mean error of the shaded prompt. Columns correspond to styles: S, shaded, H, human drawing, AR, apparent ridges, RV, ridges and valleys, SC, suggestive contours, C, occluding contours only.

Model	S	H	AR	RV	SC	C
cubehole	11	15	-	12	-	17
rockerarm	11	13	13	14	13	8
lumpcloth	14	16	16	13	15	14
femur	15	17	16	17	16	15
pulley	15	16	15	16	15	15
flange	16	18	19	16	21	20
screwdriver	16	17	16	15	17	13
bumps	17	18	16	18	14	13
twoboxcloth	17	16	18	18	18	20
tooth	18	22	21	21	21	21
cervical	19	17	18	18	13	12
vertebra	19	20	18	22	20	18
average	16	17	17	17	17	16

Table 4.2: *Mean errors from median orientations for each model and style.* Values shown are after bas-relief fitting. Rows are ordered by the mean error of the shaded prompt. Columns are same as Table 4.1. Note that, unlike the errors from ground, errors from median are sometimes *lowest* for the occluding contours drawing.

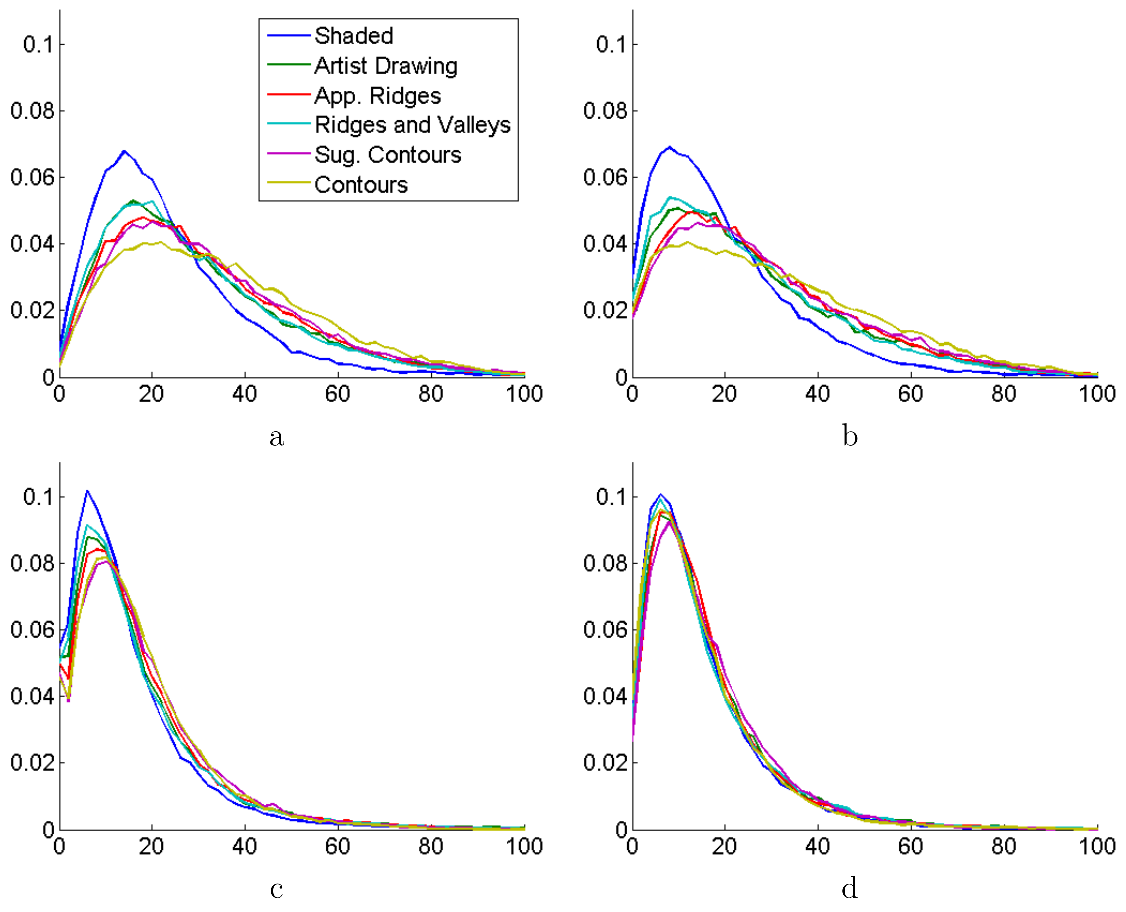
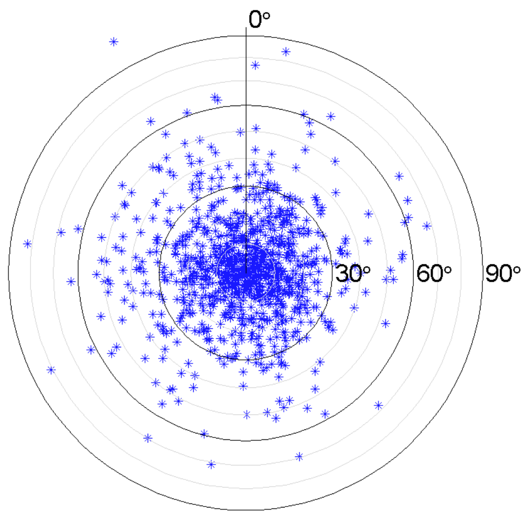


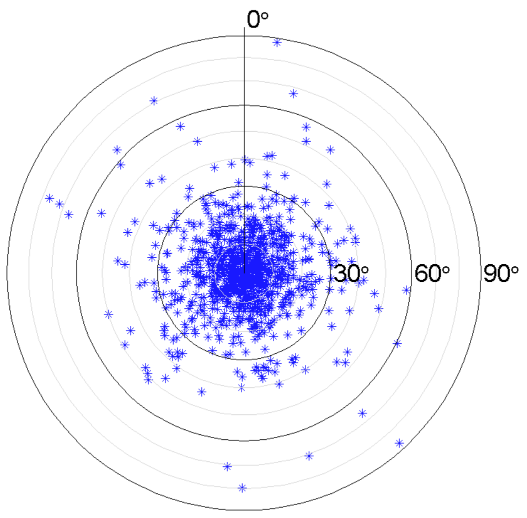
Figure 4.4: *Angular error distributions across all shapes.* Errors from the ground truth for each style of prompt are shown before bas-relief fitting (a) and after (b). Note that the errors for the shaded prompts were considerably lower on average. Errors from the median normal at each gauge are shown before bas-relief fitting (c) and after (d). Compared with ground truth, the deviations from the medians are relatively consistent across styles.

slice of the hemisphere. If the participants were placing gauges randomly, the points would appear uniformly distributed across the disk. The participants' settings, however, are clustered towards the center of each plot: for error from ground, 75% of the samples are within 31 degrees, or 14% of the area of the hemisphere, while for error from the median, 75% are inside 23 degrees, or 8% of the area of the hemisphere.

There is variation in the accuracy and precision with which workers placed gauges when seeing the shaded models, suggesting that some models were more difficult to interpret than others even under shading. The models for which the viewers had most



a



b

Figure 4.5: *Distribution visualization for shaded prompts.* Errors for 1000 randomly chosen settings. (a) errors from ground truth (blue distribution in Figure 4.4c), (b) errors from median (blue distribution in Figure 4.4d). Radial distance indicates magnitude of error, and compass direction indicates direction on the screen. Errors are roughly uniform in all directions, and errors from ground truth are larger than errors from the median.

difficulty are the smooth, blobby shapes such as the lumpcloth and the vertebra. For the obvious shapes such as the cubehole, however, the viewers interpreted the shape very closely to ground truth, lending confidence that viewers were able to perform the task successfully and that errors from ground truth in the line drawings are not simply the effect of negligent or confused participants.

4.3.2 How similarly do people interpret line drawings?

A simple way to examine how similarly our participants interpreted the line drawings is to compare statistics of the distributions around the median vectors at each gauge. We find that when seeing the line drawings, our participants set their gauges nearly as close to the other participants' as when seeing the shaded image (Figure 4.4b and d, Table 4.2). This result suggests that the participants all had a roughly similar interpretations of each line drawing, and positioned their gauges accordingly.

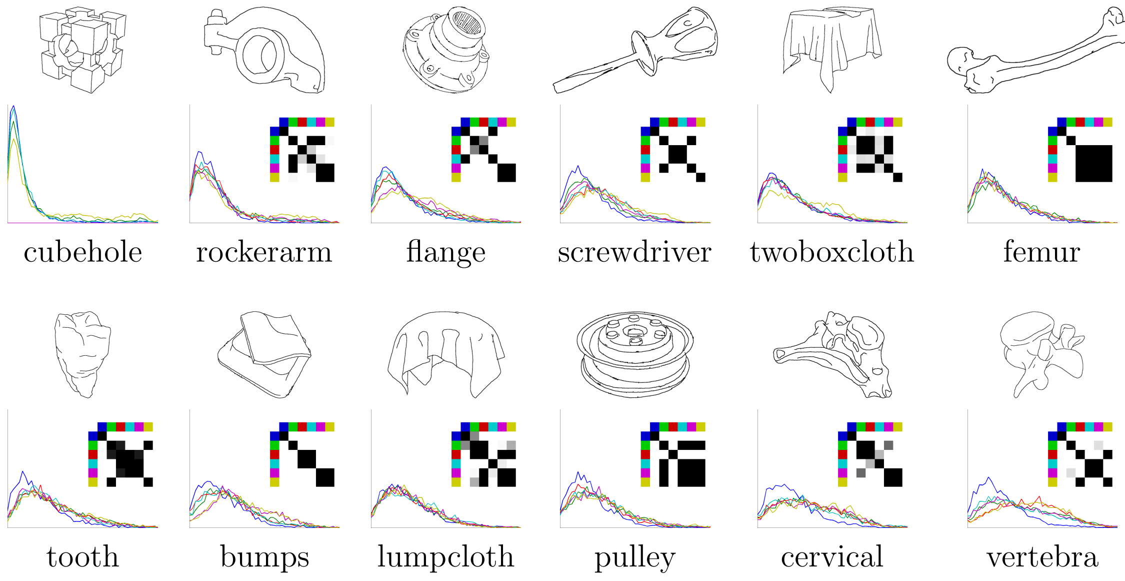


Figure 4.6: *Distributions of angular errors from ground truth for all models.* Colors are as in Figure 4.4. Below the graphs are the human artists' drawings used for the models. Inset in each graph is a visualization of the p-values for significance (black: p-value > 0.05) of difference between distributions, where the colors correspond to the styles in the histogram. The table for the cubehole is incomplete and therefore omitted. Images are ordered by the mean error for the human artist's drawing.

To get a more complete picture of the differences between the error distributions, we perform the Wilcoxon / Mann-Whitney non-parametric test for comparing the medians of two samples. This test yields a p-value. The full pair-wise comparisons of all error distributions are visualized in the inset of Figure 4.6. The colors match the colors in the legend of Figure 4.4. Black indicates a p-value > 0.05, meaning that we can not disprove the null hypothesis that those two distributions are the same. We find a statistically significant difference in error between most pairs of line drawings. This result suggests that while the interpretation of each drawing was similar across viewers, the viewers usually had different interpretations for each drawing.

4.3.3 Do line drawing interpretations match ground truth?

Unlike precision, the accuracy with which our participants interpreted shape from the line drawings varies considerably with the type of drawing and the model

(Figures 4.6). In general, the performance of the occluding contours alone was considerably worse than the other drawing types, while the performance of the other drawing types (apparent ridges, ridges and valleys, suggestive contours, and the human drawing) were roughly comparable, though still often statistically significantly different.

The types of surfaces in the model has a large effect on the accuracy of interpretation. For the cubehole, which is largely made up of flat surfaces joined with ridges, the error from ground truth for all but the contours drawing is small: approximately 15 degrees on average. For the vertebra and cervical models, which are smooth and not obviously good candidates for line drawings, the errors for the best drawings are much larger: 35-40 degrees on average. In these cases, even the human artists were unable to effectively convey the shape with only lines.

Examining the statistical significance between distributions, in almost all cases we find that the lit image did provide a statistically significant improvement over any line drawing, suggesting that some information is lost in the translation from model to line drawing. A notable exception is the flange model, for which the errors in the shaded and apparent ridges versions are not reliably different (for a visualization, see Figure 5.1).

4.3.4 Is error from ground truth localized?

Beyond the aggregate statistics for each model, we can inspect the individual gauges to immediately determine if error is localized in a few important places, or if it is evenly spread across the model. If it is highly localized, then it may be interesting to examine high error areas in detail and attempt to form theories for why the errors occured. In order to answer this question convincingly, we chose four representative models and scattered 180 (rather than 90) gauges on their surfaces.

Figures 4.7 and 4.8 show gauges for the four representative models: the flange,

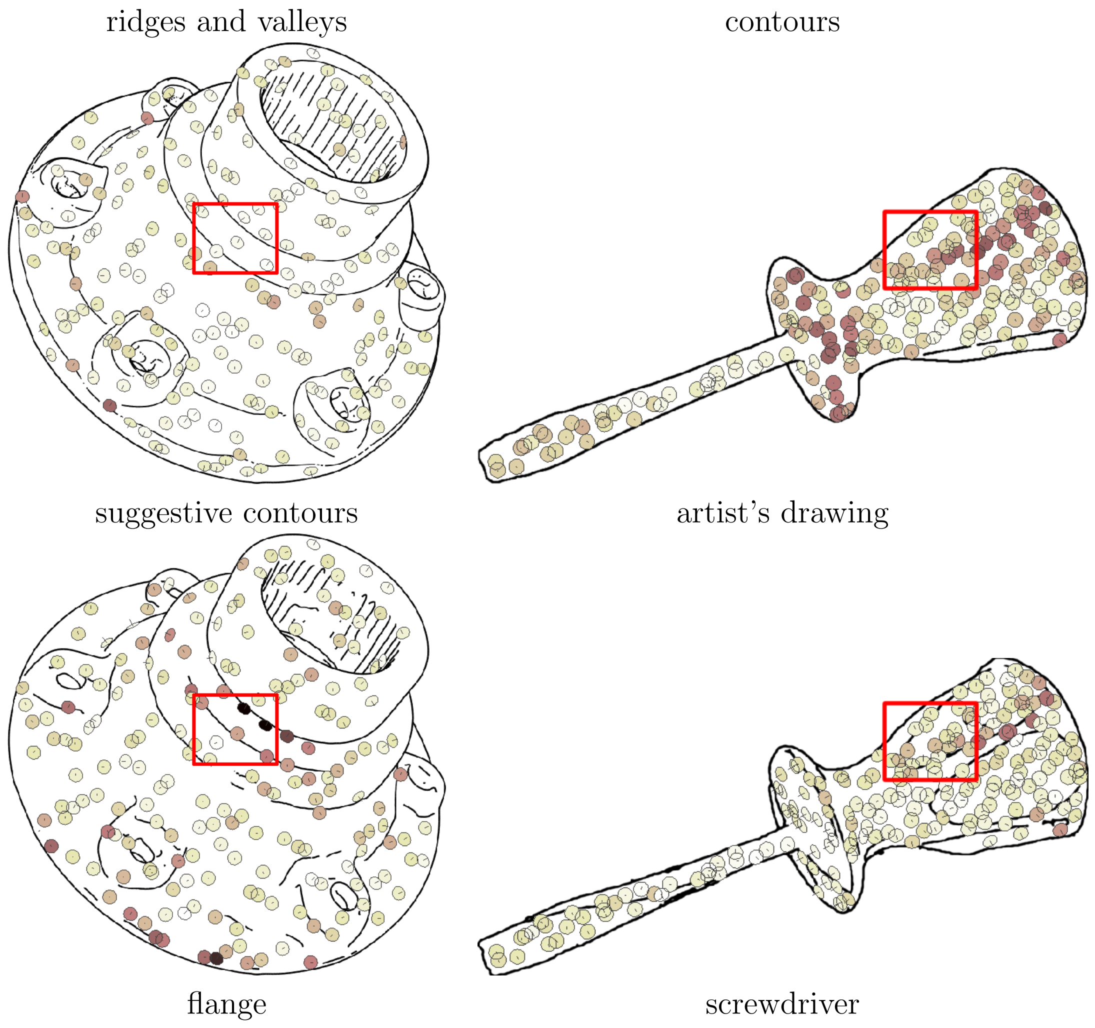


Figure 4.7: *Plots of error of median vector from ground truth (part 1).* Each pair of plots shows a different finding from our study. Flange: lines are in some cases interpreted as ridges and valleys regardless of position, which can lead to error when the lines are not positioned on ridges or valleys (e.g., suggestive contours). Screwdriver: without additional lines, the screwdriver appears to be a cylindrical solid (contours only). A skilled artist can depict the inflection points in the handles (artist's drawing). Red box: area of interest shown in detail in Figure 4.9.

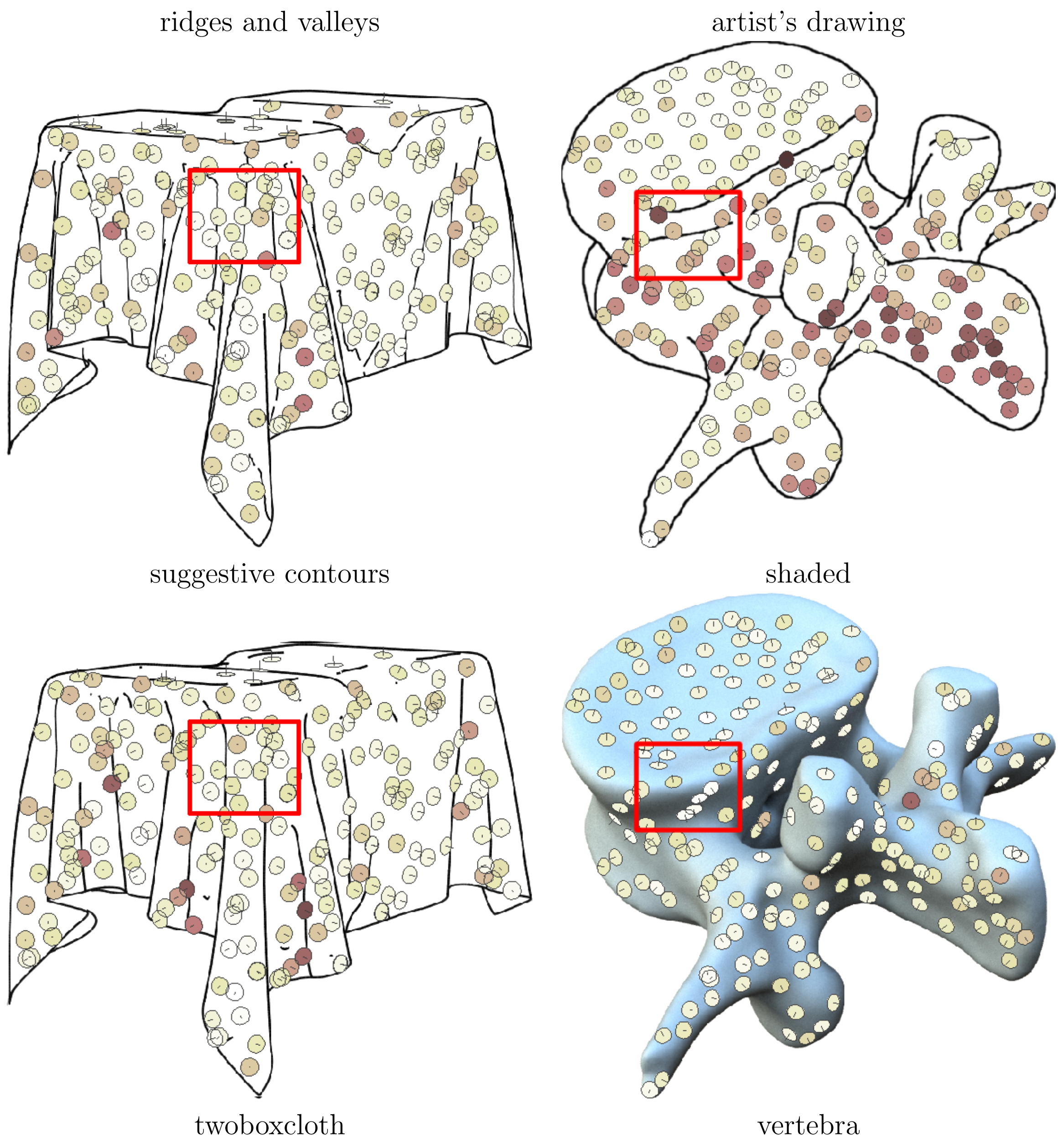


Figure 4.8: *Plots of error of median vector from ground truth (part 2).* Each pair of plots shows a different finding from our study. Twoboxcloth: errors in the folds near the front of the shape appear both with lines (ridges and valleys) and without (suggestive contours). Vertebra: some shapes are difficult for even a skilled artist to depict with only lines. In such cases, the shaded image is significantly superior (though not perfect). Red box: area of interest shown in detail in Figure 4.9.

screwdriver, twoboxcloth, and vertebra. It is immediately apparent from the plots that the errors from ground truth are not uniformly spread across the models, but rather exist in hotspots that vary with the style of drawing. For example, on the flange model we see heavy error in the suggestive contours drawing around the outer rim and the neck of the model. For the screwdriver, error is localized in the handle area. Error in the twoboxcloth model is localized around the folds near the front of the shape, whether lines are drawn there (suggestive contours, upper image) or not (apparent ridges, lower image). Error in the vertebra is large almost everywhere, even for the human artist's drawing, but relatively low in the flat area near the back of the model.

4.3.5 How can the local errors be described?

Once we have established that error is often localized in particular areas of each model, we can closely examine these areas by placing gauge strings. We chose four areas of interest, one on each of the four representative models in Figures 4.7 and 4.8. The median vectors for each gauges string, colored by error from ground truth, are visualized in the left and middle columns of Figure 4.9 (the images are shown magnified for clarity, but the prompts were the same as for the random grids). Surface curvature can be estimated by differentiating along the string and is shown in the right column of Figure 4.9. Because our model of bas-relief ambiguity is global, we do not apply a bas-relief transformation to the gauge string data. Global fitting applied only to a small patch of local data is not well constrained, and can erroneously flatten the surface (set $\lambda = 0$) if the gauge settings are inaccurate.

Looking at the gauge strings in detail, we can conjecture what types of shape interpretation errors occur with each drawing style.

Flange: The errors on the flange model suggest that lines can be interpreted as representing particular geometic features, regardless of their exact location. The neck area of the flange is roughly a surface of revolution and includes a ridge and valley across the circumference of the shape. When presented with the ridge and valley drawing (Figure 4.9b), viewers interpreted the shape about as accurately as the shaded version. They were also quite consistent with each other, except where the gauge string crosses the valley line. When presented with the suggestive contour version (Figure 4.9a), however, viewers did poorly. It appears that viewers often interpreted the two suggestive contour lines as a ridge and a valley. The median absolute deviation for the suggestive contour gauge string is between 10-30 degrees, however, suggesting that the viewers held strongly differing opinions.

Screwdriver: The gauge string on the screwdriver lies across an inflection point in the surface. Without a line to depict the inflection point (Figure 4.9d), the change in curvature is lost – the surface appears as an area of uniform positive curvature, similar to a cylinder. The human artist managed to depict the inflection point effectively (Figure 4.9c). Both these interpretations are relatively consistent: median absolute deviation for each gauge in both drawings is 10 degrees or less.

Twoboxcloth: The fold on the front of the twoboxcloth provides a counterexample to the misidentification effect on the neck of the flange. Here, viewers interpreted both the the suggestive contour drawing (Figure 4.9e) and the ridge and valley drawing (Figure 4.9f) roughly correctly, though they performed better on the suggestive contour drawing. The median orientations indicate that viewers intepreted the ridge and valley drawing roughly as a triangle, with gauges oriented similarly on either side of the middle ridge. In the suggestive contour example, the viewers interpreted the lines correctly as inflection points, leading to a more accurately rounded shape. Viewers were roughly equally reliable in both of these interpretations.

Vertebra: The string on the vertebra is an example where the artist appears to have included a ridge line and an inflection point line (Figure 4.9h), but the depiction is not successful. Viewers interpreted the shaded image (Figure 4.9g) roughly correctly, but the drawing is completely misinterpreted. Viewers were also relatively unreliable when interpreting the artist's drawing: the median absolute deviation for the drawing gauges is between 10-20 degrees, approximately double the value for the shaded image.

4.4 Conclusion

This study allows us to comment on several issues in line drawing and depiction that have previously been speculative. In particular, we now have a partial answer to our original question: how well do line drawings depict shape? Further work is necessary, however, to answer other pertinent questions, such as how best to place lines for shape depiction, and the relationship between the aesthetic quality of a drawing and its effectiveness in depiction.

4.4.1 Main Findings

For about half the models we examined, the best line drawings depict shape very nearly as well as the shaded image (difference in mean errors < 5 degrees). This is true of the cubehole, rockerarm, flange, twoboxcloth, and femur. In the case of the flange, we did not find a statistically significant difference in errors between the shaded stimulus and the ridge and valley stimulus, while in the cases of the other models we did find significant difference, but the difference was small. These results suggest that viewers interpreted the shaded and drawn versions very similarly in these cases.

In other cases, such as the cervical and the vertebra, viewers had trouble interpreting the shaded image (mean error 24-25 degrees), but were completely lost

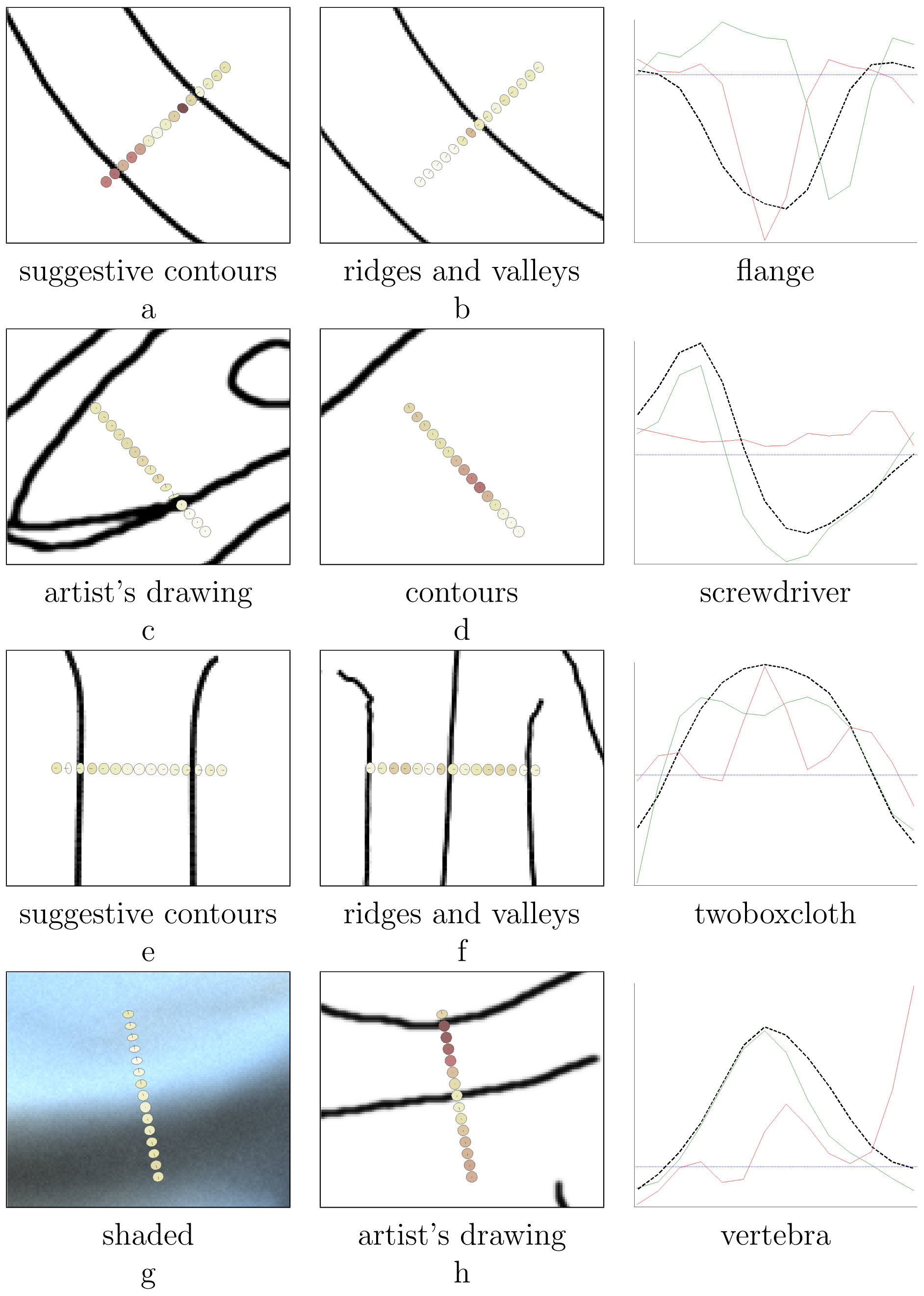


Figure 4.9: *Gauge strings.* Right column: curvatures along the string (green: left image. red: right image. dashed: ground truth. dotted: zero curvature). Four points of interest were studied in depth by placing tight strings of gauges. Flange: the curvature valley for suggestive contours (a) is translated along the gauge. Screwdriver: the artist's drawing (c) depicts inflection points, while the contours only drawing (d) does not. Twoboxcloth: suggestive contours (e) depict a smooth shape, ridges and valleys (f) depict a pointy shape. Vertebra: the artist's drawing (h) fails to depict the shape, while the shaded image (g) depicts it closely. Note: bas-relief fitting is *not* used for the strings.

on the line drawings (mean error 35 to 42 degrees). Such shapes tended to be smooth, blobby, and relatively unfamiliar to the eye. Viewers could not interpret these shapes accurately even with a drawing made by a skilled human artist. While not conclusive, this result supports the intuition that certain shapes are difficult or impossible to effectively depict with a line drawing.

Even when viewers interpreted the shapes inaccurately, however, their interpretations were similar to each other. For some shapes, including the rockerarm, cervical, and vertebra, the average error from ground for the occluding contours drawing was 50-75% higher than for the shaded prompt, but the error from median was slightly *lower*. Only for the cubehole model was the average error from median and from ground similar (11 and 12, respectively), suggesting that for most prompts, the viewers shared a common interpretation of the shape that differed from ground truth.

This study also indicates that line drawings based on differential properties of the underlying surface can be as effective in depicting shape as the drawings made by human artists. In fact, the best computer generated drawing had a slightly lower mean error in every case except the screwdriver. However, different mathematical line definitions can be effective or ineffective depending on context. For example, suggestive contours appear to be confusing in the case of the gauge string on the flange (Figure 4.9a), but quite effective in the case of the string on the folded cloth (Figure 4.9d). The human artists drew lines in similar locations to the computer algorithms, but appear capable of selecting the most appropriate lines in each case.

4.4.2 Limitations and Future Work

The gauge orientation task suffers from several limitations. First, there is no way to distinguish between errors due to misunderstanding the orientation of the shape and errors due to misunderstanding the orientation of the gauge. Second, our analysis rewards line drawings that register very closely with the ground truth shape. An artist could depict a shape feature extremely well, but if the feature appeared 10 pixels from its true location, it would have high error. In other words, it is impossible to distinguish a slightly offset but correct interpretation from a wrong one. Finally, we use only global bas-relief fitting in our implementation of the gauge figure study. Todd et al. [74] suggests that local bas-relief fitting may give a more accurate model of perception of shape, but we collected too few settings at each gauge for each viewer (two settings) to make this approach practical.

We would also like to extend this study to more drawings and more styles of depiction. Our selection and generation of drawings is subjective, and while we endeavoured to make the best drawings we could, we had no knowledge *a priori* what features would be successful. It is possible that different artists' drawings, or slight changes in algorithm parameters, could change our results. Beyond including more drawings, however, we would like to include wide-ranging depiction styles. It is possible, for example, that a different shading scheme could improve upon even the global illumination renderings, since viewers had trouble interpreting some of the shaded renderings.

As with many studies of this type, the results and analysis we have presented are purely descriptive. A natural area for future work is to investigate prescriptive results: for example, given a new shape, attempt to predict what lines will depict the shape most accurately. This study indicates that line definitions such as ridges and valleys and suggestive contours are effective in some cases and not in others, but it does not formalize where each type is effective. Formal rules of this type depend on an interpretation model of lines, which is an important long-range goal that this data may help support. Developing such as model would help us determine *how*, not just *how well*, line drawings depict shape.

Finally, artists create drawings to satisfy both functional and aesthetic goals. A study of this type cannot comment on the latter; it may be possible that the "best" drawings for shape depiction are also "ugly." Many factors can contribute to an aesthetic judgement and these factors are difficult to tease apart, but such data would be of tremendous value.

Chapter 5

Interactive Line Drawing

Ironically, mimicking the easy, loose strokes of a hand-drawn sketch is a difficult technical challenge. Even once the positions of the lines are determined, usually via one of the definitions described in Section 2.2, it still remains to draw those lines in an attractive and effective style.

While graphics libraries such as OpenGL provide basic line drawing capabilities, their stylization options are limited. Desire to include effects such as texture, varying thickness, or wavy paths has lead to techniques to draw lines using textured triangle strips (*strokes*), for example those of Markosian, et al. [50]. Stroke-based techniques provide a broad range of stylizations, as each stroke can be arbitrarily shaped and textured.

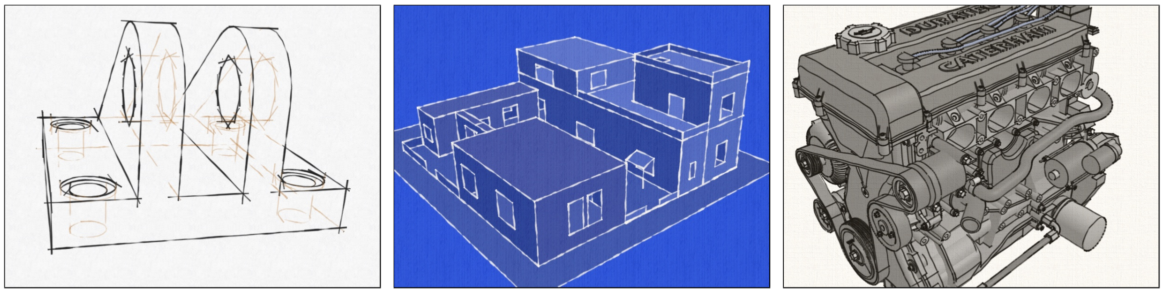


Figure 5.1: *Examples of models rendered with stylized lines.* Stylized lines can provide extra information with texture and shape, and are more aesthetically appealing than conventional solid or stippled lines.

This chapter describes a system for drawing stylized strokes with a variety of rendering effects. As described in Chapter 2, artists have developed a set of principles by which they adjust rendering qualities such as line density and contrast in order to emphasize some areas of an illustration and de-emphasize other areas. Researchers advocating non-photorealistic rendering (NPR) techniques often cite the ability to direct the viewer's attention as a benefit of NPR [23, 70]. Few systems, however, have offered this ability automatically and with local control. DeCarlo and Santella [19, 65] developed a system to automatically create abstract renderings from photographs, based on where people tend to look in the photos. However, to date no equivalent work has been performed for 3D rendering. This system provides control of emphasis for 3D models using an effect called *stylized focus*, and does so efficiently enough and with sufficient frame-to-frame coherence that it is suitable for rendering complex models interactively.

The technical description of this system is broken into three sections: computing line visibility (Section 5.1), controlling level of detail through line elision (Section 5.2), and combining rendering effects to produce stylized focus (Section 5.3). We also include the results of an eye-tracking experiment designed to verify that stylized focus actual performs its intended purpose (Section 5.3.3).

Applications for this system include any context where interactive rendering of high-quality lines from 3D models is appropriate, including games, design and architectural modeling, medical and scientific visualization and interactive illustrations. An implementation of the system, called *dpix*, can be downloaded on the web.[1]

5.1 Visibility Computation

A major difficulty in drawing strokes is visibility computation, which has been the subject of research since the 1960's. Appel [2] introduced the notion of *quantitative*

[1]http://www.cs.princeton.edu/gfx/proj/dpix

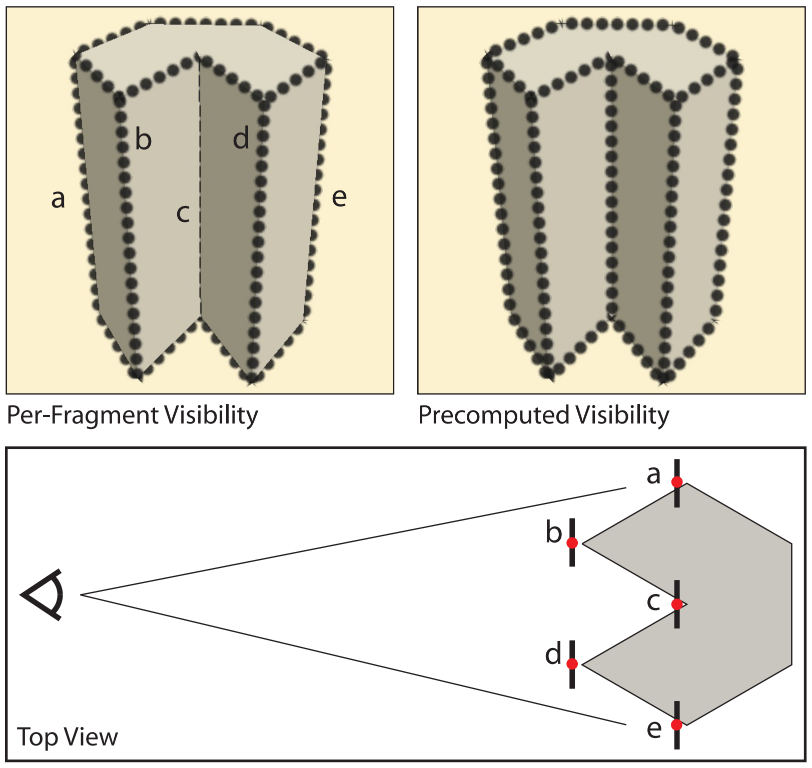


Figure 5.2: *Per-fragment visibility vs. precomputed visibility.* When drawing wide lines using a naive per-fragment visibility test, only lines that lie entirely outside the model will be drawn correctly (b and d). Lines a, c, e are partially occluded by the model, even when some polygon offset is applied. Visibility testing along the spine of the lines (red dots) prior to rendering strokes solves the problem.

invisibility, and computed it by finding changes in visibility at certain locations such as line junctions.

The invention of the z-buffer made computing visibility straightforward for thin, untextured lines (such as provided by OpenGL), but per-fragment visibility testing is insufficient for wide, stylized strokes (Figure 5.2). Thus, Appel's algorithm was further improved and adapted to NPR by Markosian et al. [50], who showed it could be performed at interactive frame rates for models of modest complexity.

Appel's algorithm and its variants can be difficult to implement, however, and are somewhat brittle when the lines are not in general position. To address these problems, Northrup and Markosian [53] adapted the use of an *item buffer* (which had previously been used to accelerate ray tracing [76]) for the purpose of line visibility, calling it an "ID reference image" in this context. In separate work, we improved the item buffer algorithm by including multiple layers and a notion of *partial visibility* for lines [11].

The item buffer approach, however, does not map well on to current graphics hardware, and suffers from aliasing artifacts even in its multi-layer implementation. The system described here introduces the first stroke visibility computation techniques to make use of current graphics hardware, providing up to a 50$\times$ speedup over the item-buffer for comparable quality (see Section 5.1.3). The two methods presented here are a simple shader program (algorithm 1), which can be applied to a conventional line drawing pipeline with minimal modification, and a new pipeline (algorithm 2) using a *segment atlas*, a new data structure that stores visibility information for each stroke.

5.1.1 Algorithm 1: Spine Test

Our first algorithm is simple to implement and provides good quality in many cases. The method requires only a single pass to draw the depth buffer and a single pass to draw the lines, so it can be easily added to an existing line rendering implementation. However, the algorithm does not support some important stylization options. In particular, because it generates an independent stroke for each line segment, it cannot properly parameterize stroke paths with multiple segments such as the curved paths seen in Figure 5.1; such paths require a parameterization if they are to be rendered with texture.

The algorithm begins with a set of 3D line segments extracted from the model. Most of our experiments have focused on lines that are fixed on the model, for example creases or texture boundaries. However, our system also supports the extraction of silhouette edges from a pool of faces whose normals are interpolated (e.g. the horizontal lines at the top of the clevis model shown on the left in Figure 5.1).

The line segments are passed to the GPU using standard OpenGL drawing calls with the primitive type GL_LINES. A geometry shader turns each line segment into a rectangular stroke and assigns texture coordinates to each vertex. A fragment shader then tests visibility at the nearest point on the spine of the stroke. This visibility test can be a single depth probe or an average of many nearby probes. Finally, the alpha value of each fragment is set to its visibility value.

Stroke Generation

Newer graphics processors such as NVIDIA's 8800 series support geometry shaders, which are GPU programs that execute between the vertex and fragment stages and have the ability to add or remove vertices from a primitive. Geometry shaders are thus a natural choice for creating stroke geometry on the GPU.

In the spine test algorithm, a geometry shader takes line segments as input and produces rectangles as output, represented as triangle strips. The shader also determines the screen-space length of the rectangle and assigns texture coordinates so that the stroke texture is scaled appropriately. The examples in this chapter use 2D images of marks in the style of pen, pencil, charcoal, etc., and are parameterized uniformly in screen space. Uniform parameterization in screen space requires perspective-correct texturing to be disabled, and control over perspective-correct interpolation is provided by the GL_EXT_gpu_shader4 extension, and by OpenGL 3.0.

To limit crawling artifacts, we use the simple strategy of fixing the "zero" parameter value at the screen-space center of the stroke. A more sophisticated strategy that seeks temporal coherence from frame to frame was described by Kalnins et al. [38].

While not a specific contribution of our method, we note that generating strokes in this manner makes it very easy to rapidly extract silhouette edges from a portion of a mesh whose normals are interpolated, such as the rounded top of the clevis on the left in Figure 5.1. While generating a stroke for an edge, neighboring face normals may be checked for a silhouette condition (one front-facing and one back-facing polygon). If the edge is not a silhouette, it is discarded and no stroke is generated.

Unfortunately, when drawing stroke paths with many segments, there is no way to know at the geometry shader level the proper parameterization of each segment, since each segment is processed independently and in parallel. It is therefore impossible to texture the entire path as one continuous stroke. This drawback is not very noticeable for models with many long, straight strokes, but is objectionable for models with many curving paths and short segments. In contrast, the segment atlas algorithm described in Section 5.1.2 supports computation of arc length and avoids this problem.

Visibility Testing

Aliasing in the visibility test for lines can cause severe visual artifacts, especially under animation. Unlike the item buffer approach, the spine test algorithm is not vulnerable to interference among lines, because each line tests the depth buffer independently. However, there are still two potential sources of aliasing error: aliasing of the per-sample depth test, and aliasing in the depth buffer with respect to the original polygons. Both these sources of aliasing can be addressed with supersampling.

In order to perform depth testing at the spine of the stroke, the depth buffer must be drawn in a separate pass and loaded as a texture into the fragment shader. The visibility of a sample is computed by comparing the projected depth value of the sample with the depth value of the nearest polygon under the sample, much like a conventional z-buffer scheme, except that fragment shader computes the nearest point on the spine of the stroke and tests the fragment's depth at that point. Using a single test for this comparison produces aliasing (Figure 5.3a). Adding additional

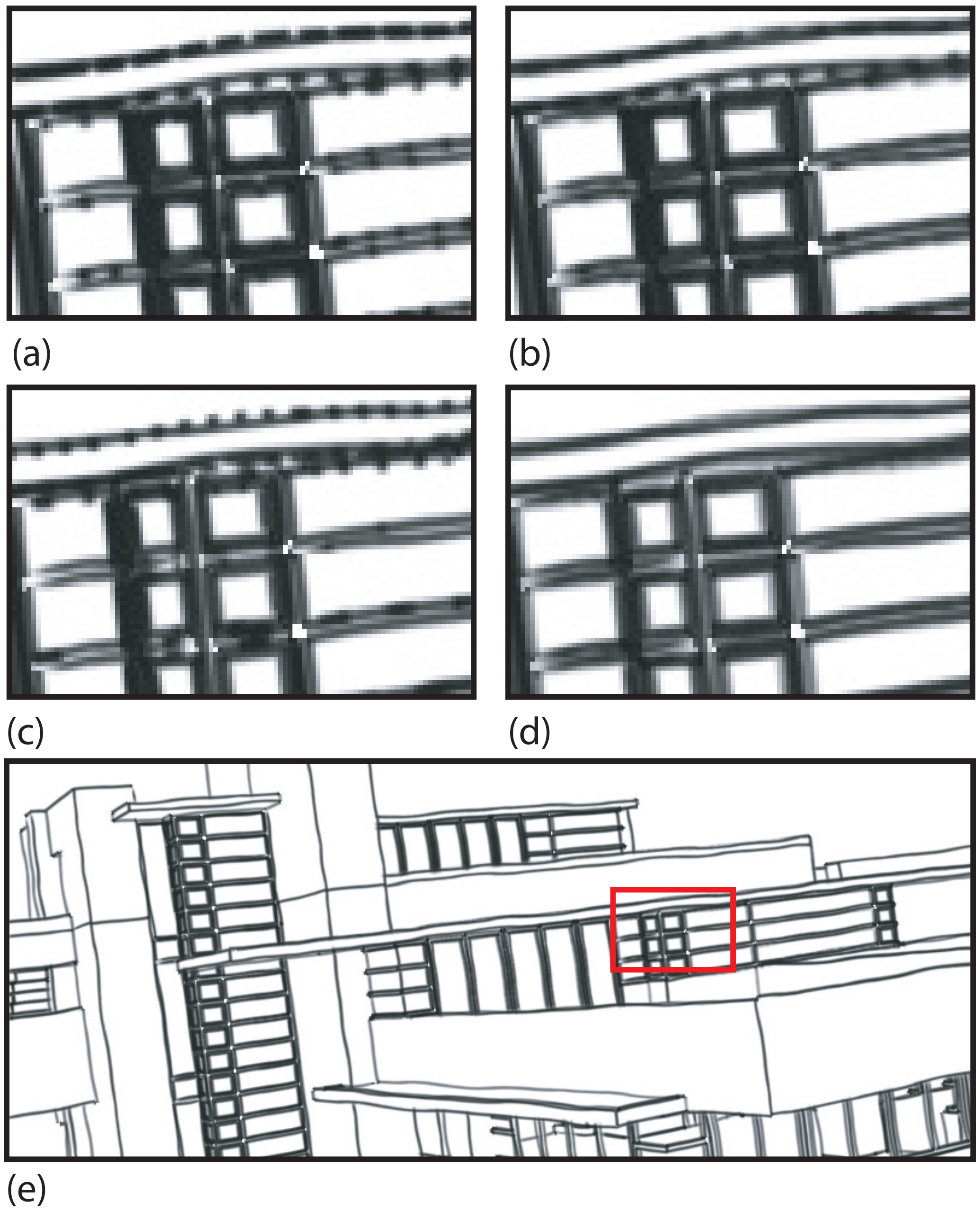


Figure 5.3: *Aliasing in visibility test.* Results for varying number of samples and scale of depth buffer. (a) 1 sample, 1x depth buffer. (b) 16 samples, 1x depth buffer. (c) 1 sample, 3x depth buffer. (d) 16 samples, 3x depth buffer. Red box in (e) indicates location of magnified area in the Falling Water model.

probes in a box filter configuration around the sample gives a more accurate occlusion value for the fragment (Figure 5.3b). Additional depth probes are usually very fast (Table 5.1), but can become expensive on limited hardware. The relative cost of additional probes increases with the width of the lines.

Any number of depth probes will not produce an accurate result if the underlying depth buffer has aliasing error. While impossible to eliminate entirely, this source of aliasing can be reduced through supersampling of the depth buffer by increasing the viewport resolution. Simply scaling the depth buffer without adding additional depth probes for each sample produces a marginal increase in image quality (Figure 5.3c), but combining depth buffer scaling and depth test supersampling largely eliminates aliasing artifacts (Figure 5.3d). Since typical applications are seldom fill rate bound for simple operations like drawing the depth buffer, increasing the size of the buffer typically has little impact on performance outside of an increase in memory usage.

5.1.2 Algorithm 2: Segment Atlas

The second visibility algorithm uses a new data structure called a *segment atlas*. The algorithm is designed to support stylization effects such as endcaps, haloes, and stylization for curved strokes, all of which require some non-local stroke information. For example, each segment in a curved stroke must have texture coordinates based on the entire arc length of the stroke. This information is costly to compute with a single-pass algorithm such as the spine test, because much of the computation is redundant across segments. The same observation holds for endcaps or haloes: while in principle each fragment could check a large neighborhood to determine the closest visibility discontinuity, it is much more efficient to store the visibility at the spine in a structure such as a segment atlas. Additional effects can also be achieved by precomputing visibility, and are explained at the end of this section.

The segment atlas algorithm is designed to efficiently compute and store the visibility information for every stroke in the scene. The input includes 3D line segments, as with the spine test algorithm, but also line strips (stroke paths). The output of the algorithm is a segment atlas containing visibility samples for each projected and clipped stroke, spaced by a constant screen-space distance (usually 2 pixels).

The algorithm has three major stages, illustrated in Figure 5.4: line projection and clipping, creation of the segment atlas, and visibility testing. All stages execute on the GPU, and all data required for execution resides in GPU memory in the form of OpenGL framebuffer objects or vertex buffer objects.

Projection and Clipping

The first stage of the pipeline begins with a set of candidate line segments, projects them, and clips them to the viewing frustum. Ideally, we would use the GPU's clipping hardware to clip each segment. However, in current graphics hardware the output of the clipper is not available until the fragment program stage, after rasterization has already been performed. We therefore must use a fragment program to project and clip the segments, using our own clipping code.

The input to the program is a six-channel framebuffer object packed with the 3D coordinates of the endpoints of each segment $(\mathbf{p}, \mathbf{q})$ (Figure 5.4a). This buffer must be updated at each frame with the positions of any moving line segments. The output of the program is a nine-channel buffer containing the 4D homogeneous clip coordinates $(\mathbf{p}', \mathbf{q}')$ and the number of visibility samples l (Figure 5.4b). The number of visibility samples l is determined by:

$$l = \lceil ||\mathbf{p}'_w - \mathbf{q}'_w|| / k \rceil \tag{5.1}$$

where $(\mathbf{p}'_w, \mathbf{q}'_w)$ are the 2D window coordinates of the segment endpoints, and k is a screen-space sampling rate. The factor k trades off positional accuracy in the visibility test against segment atlas size. We usually set $k = 1$ or 2, meaning visibility

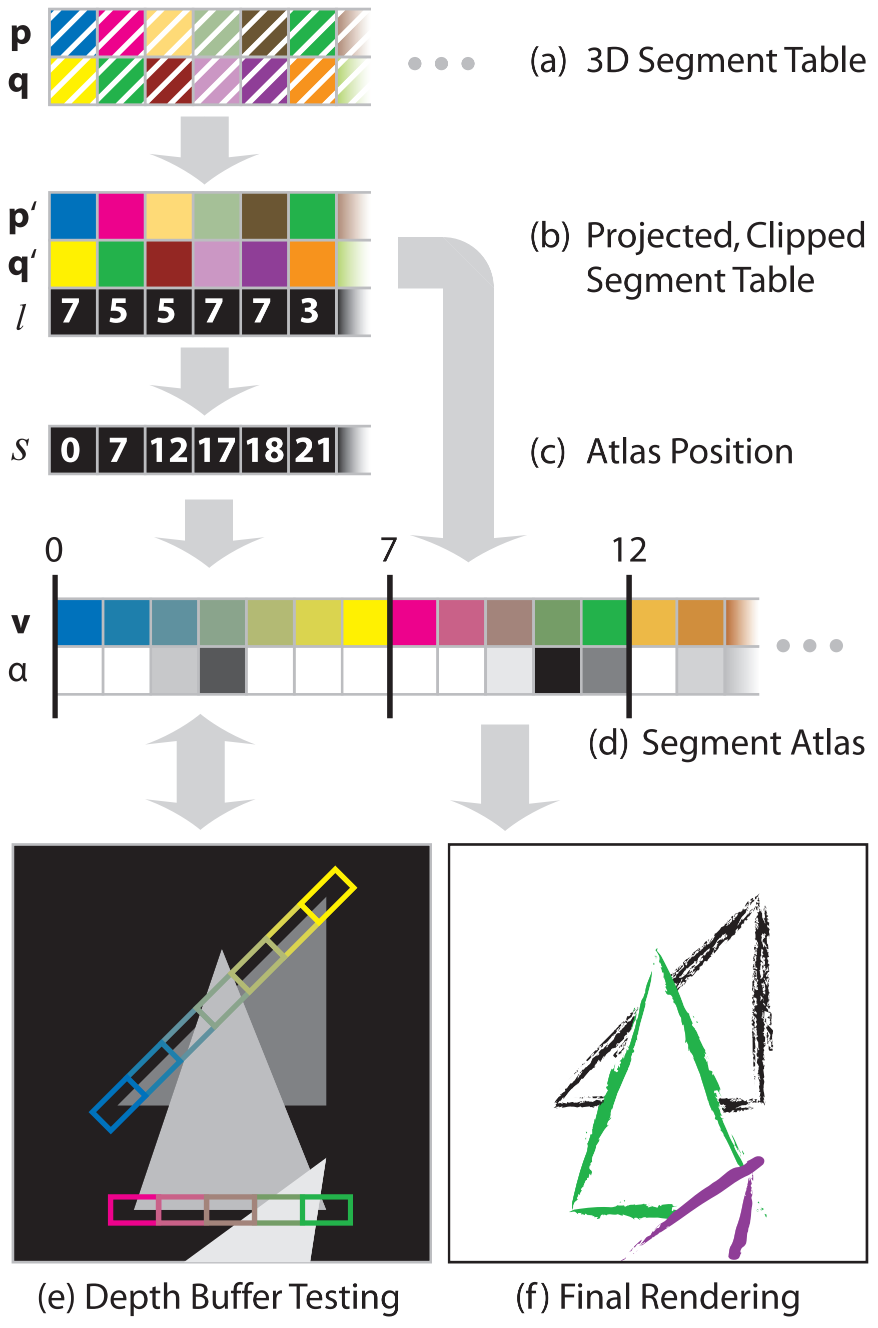


Figure 5.4: *Pipeline.* (a) The 3D line segments $(\mathbf{p}_i, \mathbf{q}_i)$ are stored in a table. (b) A fragment program projects and clips each segment to produce $(\mathbf{p}'_i, \mathbf{q}'_i)$, and determines a number of samples l_i proportional to its screen space length. (c) A scan operation computes the atlas positions s from the running sum of l. (d) Sample positions $\mathbf{v}$ are interpolated from $(\mathbf{p}', \mathbf{q}')$ and written into the segment atlas at offset s. Visibility values α for each sample are determined by probing the depth buffer (e) at $\mathbf{v}$, and are used to generate the final rendering (f). Note the schematic colors used throughout for the blue-yellow and pink-green segments.

is determined every 1 or 2 pixels along each line; there is diminishing visual benefit in determining with any greater accuracy the exact position at which a line becomes occluded.

A value of $l = 0$ is returned for segments that are entirely outside the viewing frustum. Segments for which $l \leq 1$ (i.e., sub-pixel sized segments) are discarded for efficiency if not part of a path, but otherwise must be kept or the path will appear disconnected.

Silhouette edges may also be extracted during the projection and clipping stage by loading face normals alongside the vertex world coordinates and checking for a silhouette edge condition at each segment. if the edge is not a silhouette, it is discarded by setting $l = 0$. This method is similar to the approach of Brabec and Seidel [5] for computing shadow volumes on the GPU.

Segment Atlas Creation

The segment atlas is a table of segment samples. Each segment is allocated l entries in the atlas, and each entry consists of a clip space position $\mathbf{v}$ and a visibility value α (Figure 5.4d). The interpolated sample positions $\mathbf{v}$ are created by drawing single-pixel wide lines into the atlas, using the conventional OpenGL line drawing commands. A fragment program performs the interpolation of $\mathbf{p}'$ and $\mathbf{q}'$ and the perspective division step to produce each $\mathbf{v}$, simultaneously checking the visibility at the sample.

Before the segment atlas can be constructed, we need to determine the offset s of each segment into this data structure, which is the running sum of the sample counts l (Figure 5.4c). The sum is calculated by performing an exclusive scan operation on l [67]. Once the atlas position s is computed, each segment may be drawn in the atlas independently and without overlap.

The most natural representation for the segment atlas is as a very long, 1D texture. Unfortunately, current GPUs do not allow for arbitrarily long 1D textures, at least as targets for rendering. The segment atlas can be mapped to two dimensions by wrapping the atlas positions at a predetermined width w, usually the maximum texture width W allowed by the GPU ($W = 4096$ or 8192 texels is common). The 2D atlas $\mathbf{s}$ is given by:

$$\mathbf{s} = (s \bmod w, \lfloor s/w \rfloor) \tag{5.2}$$

The issue then becomes how to deal with segments that extend outside the texture, i.e., segments for which $(s \bmod w) + l > w$. One way to address this problem is to draw the segment atlas twice, once normally and once with the projection matrix translated by $(-w, 1)$. Long segments will thus be wrapped across two consecutive lines in the atlas. Specifically, suppose L is the largest value of l, which can be conservatively capped at the screen diagonal distance divided by k. If $w > L$, drawing the atlas twice is sufficient, because we are guaranteed that each segment requires at most one wrap. Drawing twice incurs a performance penalty, but as the visibility fragment program is usually the bottleneck (and is still run only once per sample) the penalty is usually small.

For some rendering applications, however, it is considerably more convenient if segments do not wrap. In this case, we establish a gutter in the 2D segment atlas by setting $w = W - L$. The atlas position is then only drawn once. This approach is guaranteed to waste $W - L$ texels per atlas line. Moreover, this loss exacerbates the waste due to our need to preallocate a large block of memory for the segment atlas without knowing how full it will become. Nevertheless, the memory usage of the segment atlas (which is limited by the number of lines drawn on the screen) is typically dominated by that of the 3D and 4D segment tables (which must hold all lines in the scene).

Visibility Computation

Once the position of each segment in the atlas is determined, the visibility of each sample of the segment can be computed. The visibility test for each sample is performed during the rasterization of the segments into the segment atlas. While drawing the atlas, a fragment program computes an interpolated homogeneous clip space coordinate for each sample and performs the perspective division step. The resulting clip space z value is then compared to a depth buffer.

The visibility test itself is similar to the test in the spine test algorithm, with the same configuration of multiple depth probes and supersampled depth buffer. Since visibility is only tested once per spine sample, however, rather than once for every fragment along the width of the stroke, even more depth probes can be efficiently computed.

Stroke Rendering

After visibility is computed, all the information necessary to draw strokes is available in the projected and clipped segment table and the segment atlas. The most efficient way to render the strokes is to generate, on the host, a single point per segment. A geometry shader then uses the point as an index and looks up the appropriate $(\mathbf{p}', \mathbf{q}')$ in the projected and clipped segment table. The geometry shader outputs a quad for the segment, taking into account the pen width and proper mitreing for multi-segment paths. A fragment shader then textures the quad with a 2D pen texture and modulates the texture with the corresponding 1D visibility values from the segment atlas.

Filtering and Additional Effects

By storing visibility for the entire stroke, the segment atlas also provides the opportunity to filter the visibility information to fill small holes or remove short

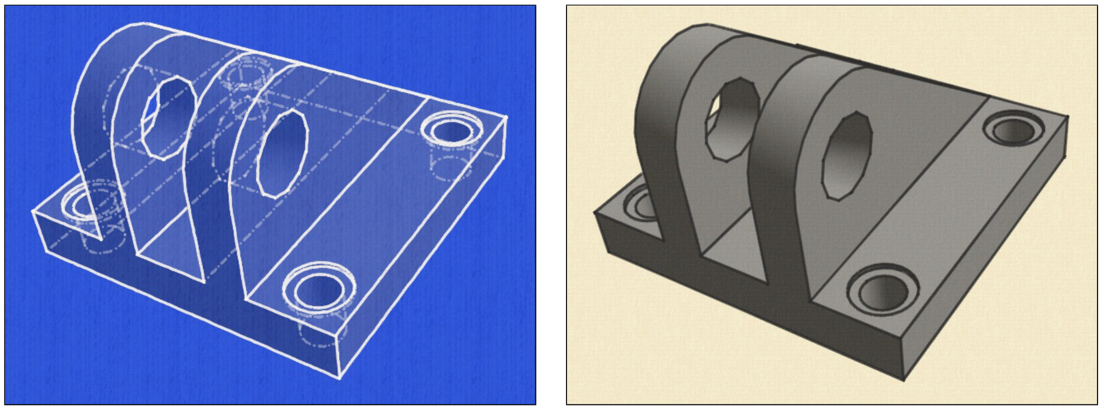


Figure 5.5: *Variation in style.* A different texture may be used for lines that fail the visibility test (*left*), allowing visualization of hidden structures. Our method also produces attractive results for solid, simple styles (*right*).

spurious sections. Other image processing operations can be performed on the atlas as well. For example, erosion and dilation can produce line overshoot or undershoot (haloing) effects (Figure 5.1). For operations such as dilation, it is necessary to add padding around each segment in the atlas, so that the segment can dilate beyond its normal length. Padding can be added easily by increasing the number of samples when computing the atlas offset.

The segment atlas also can be used to store any type of per-sample information, not just visibility. For example, it can store a measure of the density of lines in the local area, as produced by stroke-based line density control scheme (Section 5.2).

Readback

The segment atlas can be read back to the host for any processing that cannot be performed on the GPU. Reading back and processing the entire segment atlas is inefficient, however, since for reasonably complex models the vast majority of line samples in any given frame will have zero visibility. Thus we apply a stream compaction operation [30] to the segment atlas visibility values. This yields a packed buffer with only visible samples remaining, which is suitable for readback to the host.

An added benefit of the this approach is that the samples in the compacted buffer are ordered by path and segment. For models of moderate complexity, compaction and readback adds an additional fixed cost of $\sim$ 20 ms per frame.

5.1.3 Timing

We implemented the two algorithms using OpenGL and GLSL, taking care to manage GPU-side memory operations efficiently. For comparison we also implemented an optimized conventional OpenGL rendering pipeline using line primitives, the item buffer approach of Northrup and Markosian [53], and the improved, multi-layer item buffer approach [11] (not included in this thesis). We did not use NVIDIA's CUDA architecture, because the segment atlas drawing step uses conventional line rasterization and the rasterization hardware is unavailable from CUDA.

Table 5.1: Frame rates (FPS) for various models and methods.

Model	clevis	house	ship	office	ship+s	off.+s
# tris	1k	15k	300k	330k	-	-
# seg	1.5k	14k	300k	300k	500k	400k
OGL	1000+	300+	42	32	-	-
IBlo	87	24	9.6	7.0	-	-
IBhi	20	3.4	0.5	0.4	-	-
STlo	900+	146	26	28	19	23
SThi	300+	75	24	25	19	21
SAlo	400+	119	33	29	23	24
SAhi	200+	76	25	24	22	21

Table 5.1 shows frame rates for four models ranging from 1k-500k line segments. The clevis, house, ship and office models are shown in Figures 5-8. The "+s" indicates silhouettes were extracted and drawn in addition to the fixed lines. Timings for clevis and house are averaged over an orbit of the model, whereas timings for the ship and office are averaged over a walkthrough sequence. All frames are rendered at 1024×768 using a commodity Dell PC running Windows XP with an Intel Core 2 Duo 2.4 GHz

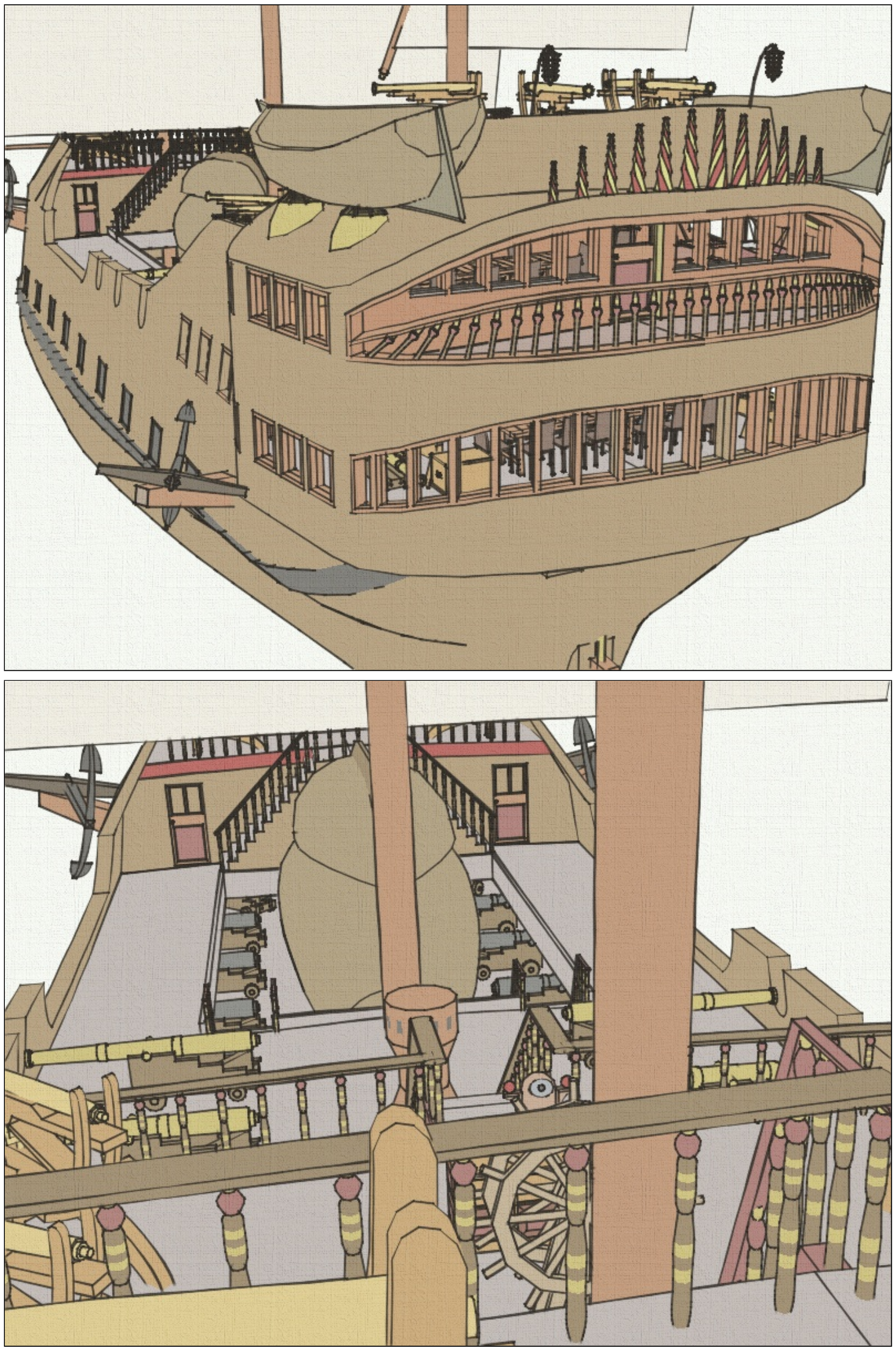


Figure 5.6: *Ship model.* The ship model has 300k triangles and 500k total line segments, and can be rendered at high-quality and interactive frame rates using our method.

CPU and 2GB RAM, and an NVIDIA 8800GTS GPU with 512MB RAM.

We tested the following rendering algorithms: (OGL) conventional OpenGL lines; (IBlo) single item buffer [Northrup2000]; (IBhi) 9× supersampled item buffer with 3 layers [Cole2008]; (STlo/SAlo) spine test shader and segment atlas, respectively, with a single depth probe, which is comparable to IBlo; and (SThi/SAhi) spine test shader and segment atlas, respectively, with 9 depth probes and 2× scaled depth buffer, which is comparable to IBhi. For small models (clevis and house), both the spine test and segment atlas algorithms are slower than conventional OpenGL rendering by factors of $2 - 4\times$, though overall speed is still high. Additional samples and depth buffer scaling also incur a noticeable penalty for these models. For the more complex models (ship and office), the penalty for using either method declines. Both algorithms are within 50% of conventional OpenGL in the high-quality modes (SThi/SAhi). The basic segment atlas (SAlo), which suffers from some aliasing artifacts but still provides good quality, is within 75% of OpenGL on both the office and ship models.

Both of the new methods are always considerably faster than the item buffer based approach, but the most striking difference is when comparing the high quality modes of each method. The item buffer approach with 9× supersampling and 3 layers, as suggested by [11], gives similar image quality to our methods with 9 depth probes and 2× scaled depth buffer. The new methods, however, deliver performance increases of up to 50× for complex models.

Both methods allow for easy extraction and rendering of silhouette edges on the GPU. The last two rows of Table 5.1 show the performance impact when extracting and rendering silhouettes. The increase in cost is roughly proportional to the increase in the total number of potential line segments. We did not implement silhouette extraction for the other methods. However, silhouette extraction can be a costly operation when performed on the CPU.

While accurate timing of the stages of our algorithm is difficult due to the deep OpenGL pipeline, the major costs of the algorithm (∼80-90% of total) lie in the sample visibility testing stage and depth buffer drawing stage. For small models, the sample visibility testing is dominant, while for large models, the depth buffer creation is the primary single cost. Projection, clipping, and stroke rendering are minor costs.

5.2 Line Elision

With visibility testing addressed, the system can now rapidly and accurately render all visible strokes in a scene. As described in Section 2.1, however, artists often vary the complexity of their lines to achieve abstraction or a desired level of tone. Our system so far gives no such control; lines are drawn wherever the mathematical line definition – usually based on local properties – determines they should be drawn. This leads to undesirable effects, such as excessively dense lines in highly detailed areas of the model (Figure 5.7a). Furthermore, simply controlling the density of lines without changing their style can effectively place visual emphasis in a scene (Figure 5.7b).

For these reasons, level of detail algorithms for lines have been a focus of recent research in NPR. In comparision to previous work, the method presented here provides local control over line elision for general models, and temporal coherence suitable for animation.

5.2.1 Background

Most LOD work has focused on reducing detail for efficiency while affecting perception as little as possible [48]. In contrast, the goal in NPR is to change the impression of the image by reducing unnecessary detail. In NPR, unnecessary detail often takes the form of overly dense or cluttered lines. Most line density control schemes are specialized to the type of lines being drawn. For example, Deussen and Strothotte [20]

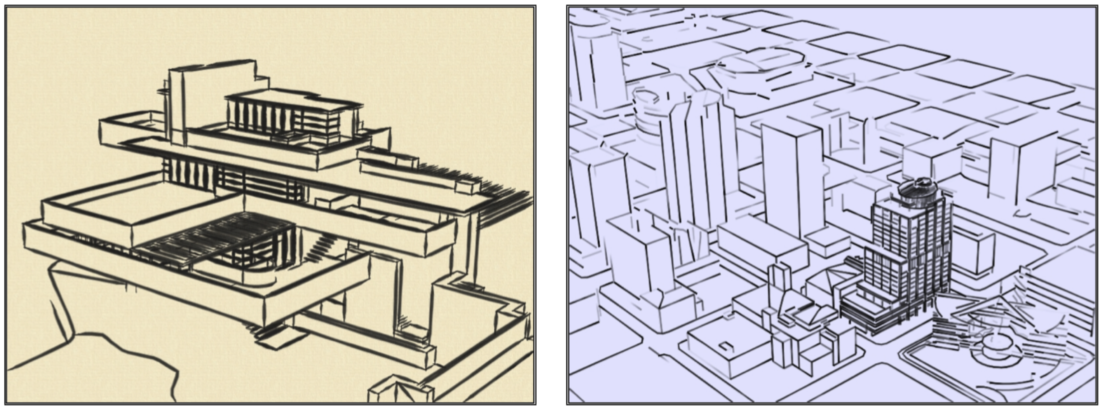

a b

Figure 5.7: *Without and with line elision.* (a) lack of control over line density can produce undesirable blobs of lines. (b) with control of line density, emphasis can be placed while maintaining a uniform line style.

proposed a method to control the drawing of tree leaves and branches. While specialized, their method is effective and provides good temporal coherence.

For rendering architectural models in a pen-and-ink style, the method of Winkenbach and Salesin [79] reduces line clutter using "indication": rather than drawing a complicated texture over an entire surface, only a few patches marked by the user are drawn with high detail. Their method also provides local control over line density, primarily for the purpose of controlling tone. However, such hand-crafted line textures are not amenable to rendering with temporal coherence. The system of Strothotte et al. [71] offers a similar interface – with similar benefits and limitations – to that of Winkenbach and Salesin with the explicit goal of directing the attention of the viewer. Praun et al. [60] introduced a temporally-coherent hatching method based on blended textures that provides control of line density for reproducing tone. These methods are intended to deal with clutter and shading, not abstraction; they are effective at simplifying repetitive or stochastic textures, but not at abstracting larger structures.

Jeong et al. [36, 52] developed a method for abstraction based on a series of representations of a 3D model, with varying complexity (created by simplifying the

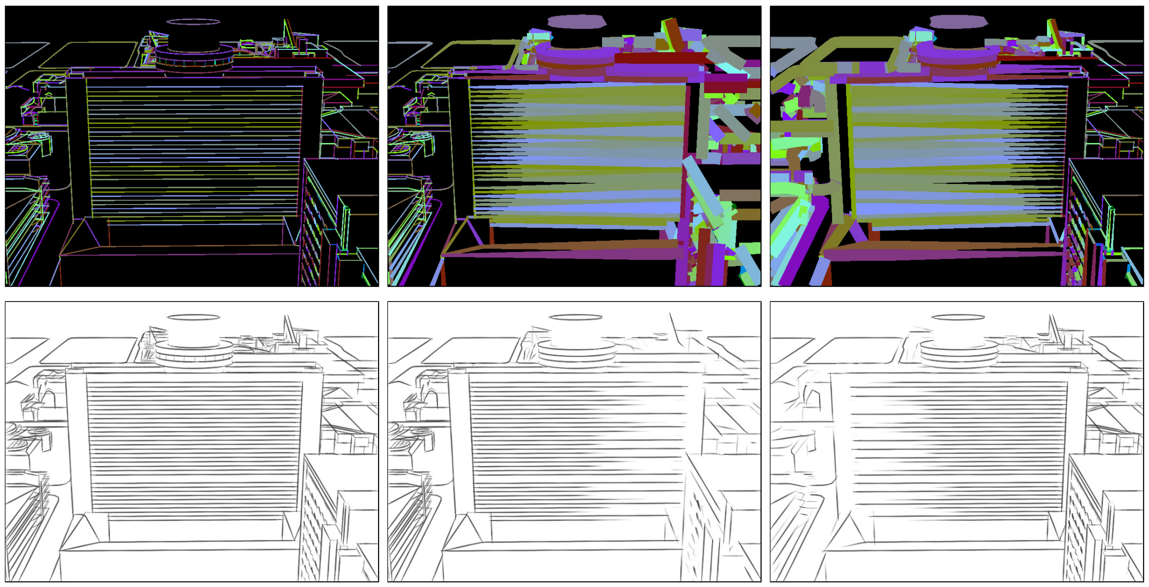


Figure 5.8: Item and priority buffer. The item buffer (top left) determines line visibility (bottom left). The priority buffer (top middle and top right) determines line density (bottom middle and bottom right). Lines are drawn in unique colors in order from low- to high-priority, so the highest priority line in any region will prevail. To emphasize the left side of the image (middle), narrower lines are drawn on the left side of the priority buffer, leading to higher line density. The same approach is used to emphasize the right side of the image (right).

original model). Their algorithm rendered the model with varying LOD, depending on importance or distance. However, such methods do not provide explicit *local* control of line density; rather it emerges as a by-product of mesh simplification. Furthermore, the scheme based on simplification does not perform well on "boxy" meshes such as architectural models. A method introduced by Wilson et al. [78] does offer *local* control of line LOD based on line priority, like the method described herein. The method of Grabli et al. [25] operates similarly to that of Wilson et al., but adds a more sophisticated measure of line density. These two methods only drop strokes in areas of high density. In contrast, the simplification work of Barla et al. [3] simplifies strokes by replacing groups of strokes with fitted, simplified versions. However, none of these methods addressed temporal coherence, inhibiting their use in an interactive setting.

5.2.2 Priority Buffer

Our density control method is inspired by the *item buffer* data structure discussed in Section 5.1. In the item buffer, no more than one line can be present at each pixel. This restriction causes aliasing when the item buffer is used for visibility testing, but provides a minimal level of control over line density. There is, however, no local control; the item buffer essentially sets a global maximum line density. We would like the ability to control this maximum line density at a local level. For example, we wish to stipulate that for a particular region in the illustration, no two lines will be closer than 10 pixels. Furthermore, since we must remove lines to reduce line density, we want to select the least important lines (in some sense) for removal. To achieve this goal, we introduce a new data structure inspired by the item buffer, which we call the *priority buffer*.

The priority buffer shares its general appearance with the item buffer (Figure 5.8): it is an offscreen buffer, consisting of lines colored by ID. The priority buffer departs from the item buffer in two major respects: first, the lines are sorted in the priority buffer by an arbitrary "priority" value, not by depth. Higher priority lines will be drawn on top of lower priority lines, even if the lower priority lines are closer in 3D to the camera. Second, lines in the priority buffer may vary dramatically in width. The width of the priority buffer line is inversely proportional to the desired line density in the region. Thus, in areas of low density, lines will be drawn wider; in high density areas, lines will be drawn narrower. Wide, high priority lines will carve a broad swath through the image, overwriting any other lines in their neighborhood. The exact width of the priority buffer lines is controlled by the user through a transfer function (Section 5.3).

We implement the priority buffer algorithm as follows. After we compute the visibility for each line, we sort the visible lines by *priority.* The priority quantity can be any measure of how important the individual lines are. In our experiments we have used a simple heuristic to assign priority: the length of the original line in the 3D model, before visibility testing has been performed. We assume, in other words, that long lines correspond to important features. This heuristic seems to work well for architectural models, but one can imagine more sophisticated methods that might consider, for example, exterior silhouettes to be of high importance. In our tests the $O(n \log n)$ time required to sort the lines is negligible; however it would be easy to remove this overhead by assigning depths based on priority and using the z-buffer to perform the sort while rendering to the priority buffer.

Once we have rendered the priority buffer, we can use it to decide which visible lines to render and which lines to omit. Similar to the item buffer test, a line should be visible in the final rendering if its associated color is visible in the priority buffer. The item buffer test makes a binary decision about line visibility: either the line is visible, or it is not. If we use a similar test with the priority buffer, we produce effective imagery, but with one caveat – weak temporal coherence. Lines tend to "pop" in and out, particularly when groups of parallel lines alias with rows and columns of pixels in the buffer. Our solution to this problem is straightforward: we test a $w \times w$ region (we use $w = 5$) centered around the projected sample point. We count the number of pixels in this region with the appropriate ID, and divide by the number we would expect to see if the line were fully visible. For example, if the line were three pixels wide, we would expect 3×5 or 15 appropriately colored pixels to appear in our window. The ratio of visible to expected pixels gives us a *degree* of visibility v between 0 and 1. We then multiply the opacity of the line by v when rendering, creating a continuous falloff of visibility. This strategy provides reasonable temporal coherence, while not forfeiting the interactive frame rates available with the priority buffer.

5.3 Stylized Focus

We now have tools for effectively rendering lines with proper visibility and screen-space density. This section explains how we apply these tools, along with several simpler rendering effects, to create stylized emphasis.

The stylized emphasis effect is based on the notion of *focus*, which is a scalar value that determines intensity of each rendering effect. We present two models for calculating focus, and describe how the level of focus impacts colors and lines in the image. Many of the rendering policies described here are motivated by principles developed by artists and codified in numerous texts, for example [26, 47, 51]. Briefly, we control four qualities to emphasize or de-emphasize part of an illustration: contrast, color saturation, line density, and line sharpness. These strategies are of course interrelated; for example, sharper lines can yield higher contrast.

Our system supports a range of styles composed of lines and shading (suggestive of pen and ink combined with watercolor) but many of the mechanisms we describe should be applicable in a spectrum of media ranging from cel animation to colored pencil.

5.3.1 Focus Models

The degree of emphasis at every part of the image is specified by the normalized scalar focus value $f(p)$, which indicates how much emphasis to place at every point p. Following the general framework for expressing focus described by Isenberg et al. [33], our system provides two intuitive focus models:

2D Focal point. In this model, the artist picks a 2D point p_f and the focus $f(p)$ is taken to be the 2D distance $||p - p_f||$. This model produces a foveal effect, which can feel quite natural, especially when used in concert with an animated “focus pull.” Note that when working from still imagery such a photographs (as were DeCarlo and

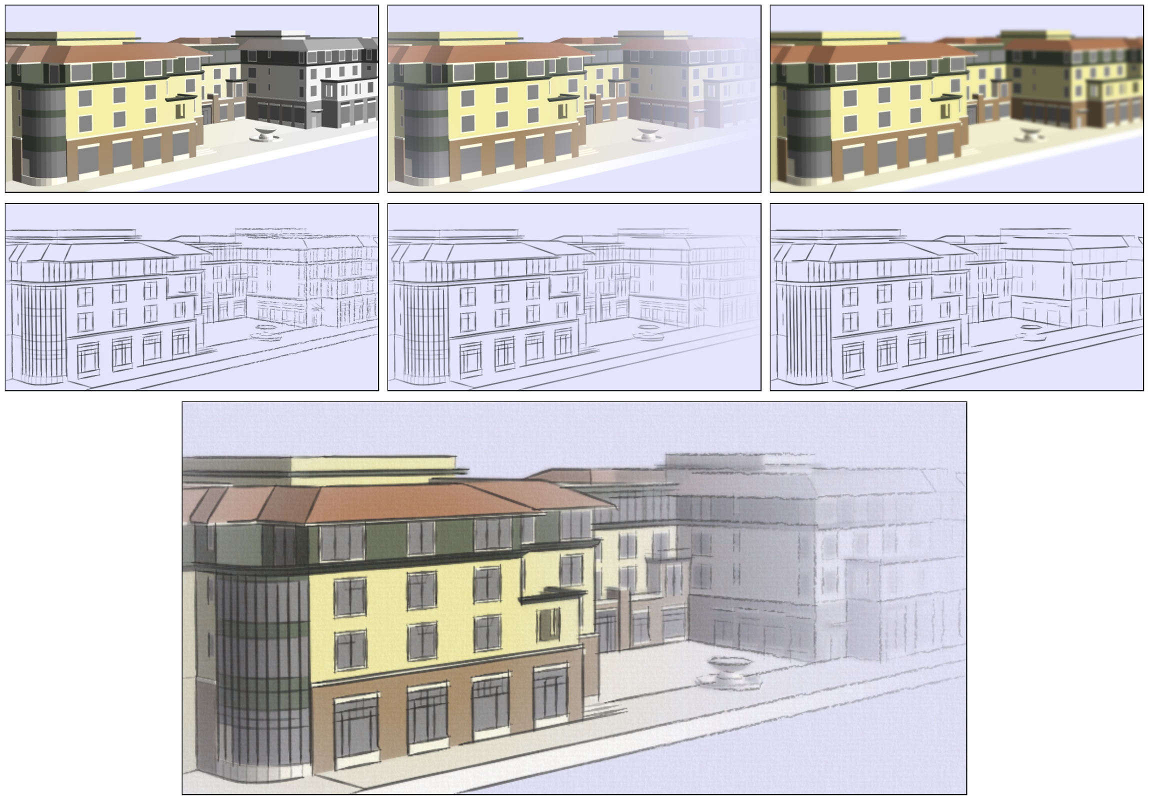


Figure 5.9: From top to bottom: color effects (desaturation, fade, blur); line effects (opacity, texture, density); all six combined.

Santella [19]) this may be the only viable option. However, for some compositions this effect appears unnaturally symmetrical, and we find the next model to be more pleasing.

3D Focal point. In this model, the artist chooses a 3D point p_f from which the focus falls off radially in 3D: $f(p) = ||p - p_f||$. This focal model is perhaps most intuitive for 3D scenes, and distinguishes our work from the bulk of previous methods for placing emphasis in illustrations, as they generally did not have access to 3D information.

5.3.2 Rendering Effects

We implement eight effects that respond locally to emphasis $f(p)$ in the image. Three are color effects (desaturation, fade, and blur) while five adjust line qualities (texture, width, opacity, density and overshoot), some of which are shown in Figure 5.9. These effects are generally used in combination, though their relative impacts are under artistic control. Each effect has its own transfer function from the normalized focus value $f(p)$ to the final intensity of the effect. Aside from the obvious benefit of aesthetic flexibility, varying each effect individually also tends to "break up" the image so that derivative discontinuities are not noticeable. Adjusting the transfer function for each effect is quite intuitive, and the parameters can be saved and restored (together with others settings) collectively in a "style."

Given a model, camera, point of emphasis, and style, the overall rendering process is as follows. First, line visibility and elision are computed as in Sections 5.1 and 5.2. Next, colors are rendered into the frame buffer using a pixel shader. Finally, the lines are drawn over the resulting color image as textured triangle strips. In order to provide emphasis cues, the width, opacity and textures are modulated along the lengths of these triangle strips. The line textures are interpolated among two or more 2D textures τ_i.

The pixel shader for color rendering first renders the color of the model at every pixel. During the same rendering pass, the shader also computes pixel-accurate values for emphasis using one of the focus models described in Section 5.3.1. The shader first desaturates the color, and then fades the color by interpolating towards the background color, in both cases by an amount appropriate to the style and emphasis. Optionally, the shader may blur the image using a variable filter kernel. When applied in combination, these color effects provide a natural cue for emphasis.

5.3.3 Evaluation

The goal of stylized focus is to construct imagery that subtly guides the attention of the viewer to specific places in a scene. A natural question to ask is: how effective are these renderings at achieving this objective? We measure the overt

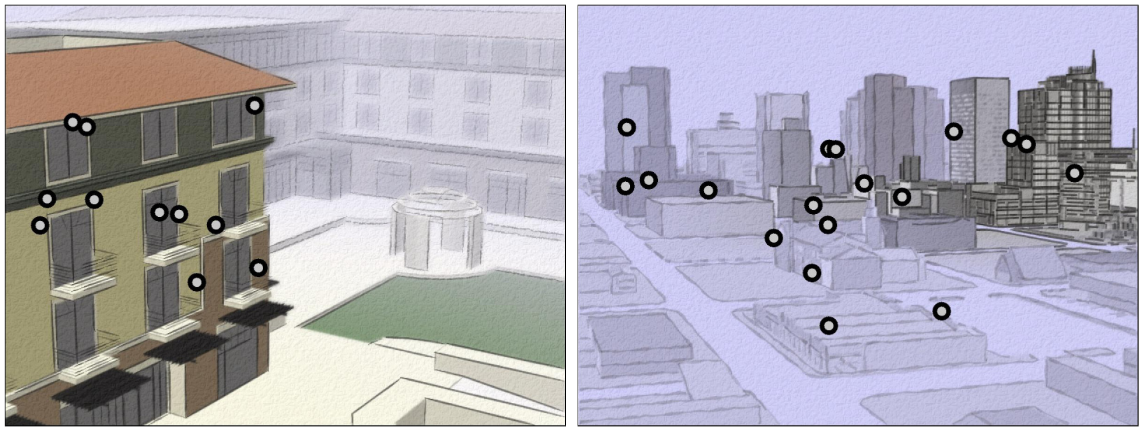


Figure 5.10: *Evaluation.* Tracked eye-movements in the hotel scene (left) follow the emphasis. Eye-movements in the city-scape (right) only loosely respond to emphasis.

visual attention given to emphasized regions using eye tracking, comparing viewings for emphasized and uniform images. Results of this evaluation indicate that viewers examine emphasized regions more than they examine the same location in a uniformly rendered image of the same scene (p-value < 0.001).

Our experimental design follows that of Santella and DeCarlo [65]. Thirteen (student) viewers examined a series of 18 rendered scenes in a variety of styles, with and without emphasis. The viewers were asked to rate how much they liked each image. This task served to motivate viewers to look at the images, but resulting scores were too sparse and noisy to analyze quantitatively. Each image was displayed for eight seconds on a 19-inch LCD display. The screen was viewed at a distance of approximately 86 cm, subtending a visual angle of approximately 25 degrees horizontally x 19 degrees vertically. Eye movements were monitored using an ISCAN ETL-500 tabletop eye-tracker (with a RK-464 pan/tilt camera). The pan/tilt unit was not active during the experiment. Instead, subjects placed their heads in an optometric chin rest to minimize head movements. Viewers saw renderings with:

1. color and lines with uniform level of focus in the scene

2. uniform color only, no lines

3. uniform lines only, no color

4. emphasis of color and lines, based on a single focal point

5. emphasis in color only, with uniform lines

6. emphasis in lines only, with uniform color

7. emphasis in color only, no lines

8. emphasis in lines only, no color

We chose two different focal points in each scene, leading to two versions of each type of emphasized image (types 4-8). This resulted in a total of 13 versions of each of the 18 scenes. Use of two focal points for each scene provides evidence that the effect of emphasis is not limited to the single most natural focal point in the scene. Each viewer saw only one (randomly chosen) version of each scene.

Recall that we have a focus value $f(p)$ at every point p in a rendering. When a viewer fixates at p, the value $f(p)$ tells us how emphasized that region is. We can use this value, sampled under fixations, to measure how much the viewer looked at the emphasized regions. Suppose the viewer's fixations in an emphasized image followed a path $p(t)$ from time t_0 to time t_1. Then we can measure the average emphasis E_f under those fixations by:

$$E_f = \frac{1}{(t_1 - t_0)} \int_{t_0}^{t_1} f(p(t))dt$$

Of course we cannot simply claim success if E_f is greater than the average $f(p)$ in the scene. The viewer might have looked in places of high $f(p)$ independently of the emphasis, perhaps because they were interesting parts of the image. However, for a given focal point we can control for such cases by evaluating E_f using the same $f(p)$ measured over the fixation path from the corresponding uniformly-emphasized image (types 1-3). This provides a (control) measure of how much those emphasized areas are examined *even without emphasis.* If emphasis increases attention on an area, E_f will be greater for the fixation path in the emphasized (test) image than for the fixation path of the uniformly emphasized (control) image. On the other hand, if the viewer looks in un-emphasized areas in a test image, then E_f will not be higher than that of the control. Likewise, if the viewer examines emphasized regions in the test, but also looks in the same regions in the control, then E_f will be similar for both images.

For each type of emphasized image (test images, types 4-8), values of E_f were compared with those in a corresponding unemphasized image (controls, types 1-3), collapsing over all scenes and both emphasis points for each scene. A two way condition cross scene ANOVA for all conditions was conducted followed by multiple comparisons between each test and control condition. Emphasized locations were more heavily examined in *all* styles of emphasis (types 4-8) (p-value < 0.001). As we hypothesized, emphasis of color and lines (stimuli type 5 and 6) each had a significant effect individually. These effects were individually weaker (p-value < 0.001) than their combined effect (type 4). Finally, matching our subjective impression, the effect of emphasizing color with uniform lines (type 5) was stronger (p-value < 0.01) than that of emphasizing lines with uniform color (type 6).

We conclude from this experiment that our method is effective. In all rendering types it shifts viewer attention to the emphasized point. This is a general effect, found in multiple styles and for multiple points of emphasis in the scene. Emphasis works in styles that contain just color or lines alone. It is also effective when images contain both color and lines, but only one is emphasized, though it is most effective when emphasis of both are combined.

5.4 Summary and Future Work

The system described in this chapter allows rendering of high-quality stylized lines at speeds approaching those of the conventional OpenGL rendering pipeline, and interactive control of visual emphasis using a range of rendering effects called *stylized focus*. It provides improved temporal coherence with less aliasing (sparkle) than previous approaches, along with control over line elision, making it suitable for animation of complex scenes. In addition, there is eye-tracking data to indicate that the stylized focus effect actually works.

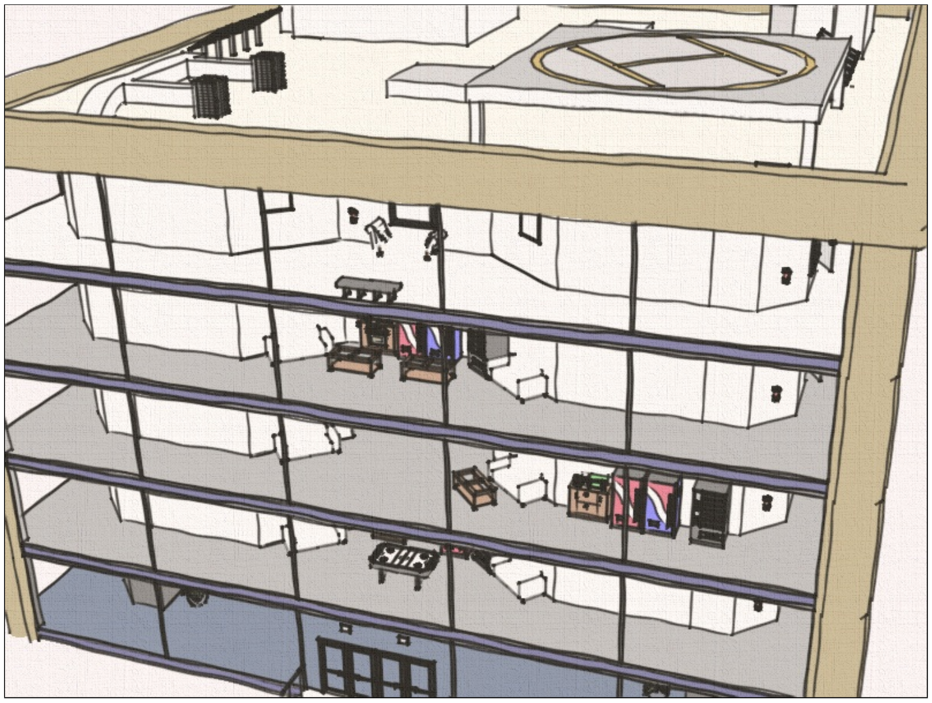
There are several areas for future work suggested by the results of this system:

Stylized focus with feedback. While the system attempts to emphasize some areas of the illustration and de-emphasize others, it does not evaluate whether or not such effects were achieved. For example, when a striking feature exists outside the emphasis in the scene, it may require stronger de-emphasis than would automatically take place. Our system has no mechanism to notice such situations, which may explain the weak result in Figure 5.10 (right). Computational models of visual salience [35] can estimate the prominence of features in an emphasized rendering, and could allow the system to iteratively adjust emphasis until the desired result is achieved.

Other uses for full visibility. The ability to store full visibility information for all lines from one frame to the next affords several possibilities for future work. Just as this system supports stylized focus as an artistic effect inspired by photorealistic defocus effects, we can imagine a "stylized motion blur" effect inspired by photorealistic motion blur. Given the segment atlases from previous frames, we could blur the *visibility values* from consecutive frames rather than the final rendered strokes. Blurring visibility could, for example, allow a disappearing stroke to break up into shrinking splotches of ink, rather than simply fading out.

Improved elision control. Other future work in this area may include improving the line elision control method described in Section 5.2 to operate more effectively on the GPU. Our current implementation exhibits some sparkling artifacts under animation, and causes a hit in performance. One challenge is that these approaches do not take into account partial visibility of lines, which is necessary for smooth animation.

View-dependent lines. While not a direct extension of our method, we would also like it to handle other view-dependent lines such as smooth silhouettes [29], suggestive contours [17], and apparent ridges [37]. Including these line types at a reasonable performance cost may require an extraction algorithm that executes on the GPU. In contrast to lines that are fixed on the model, consistent parameterization of such lines from frame to frame presents its own challenge [38].



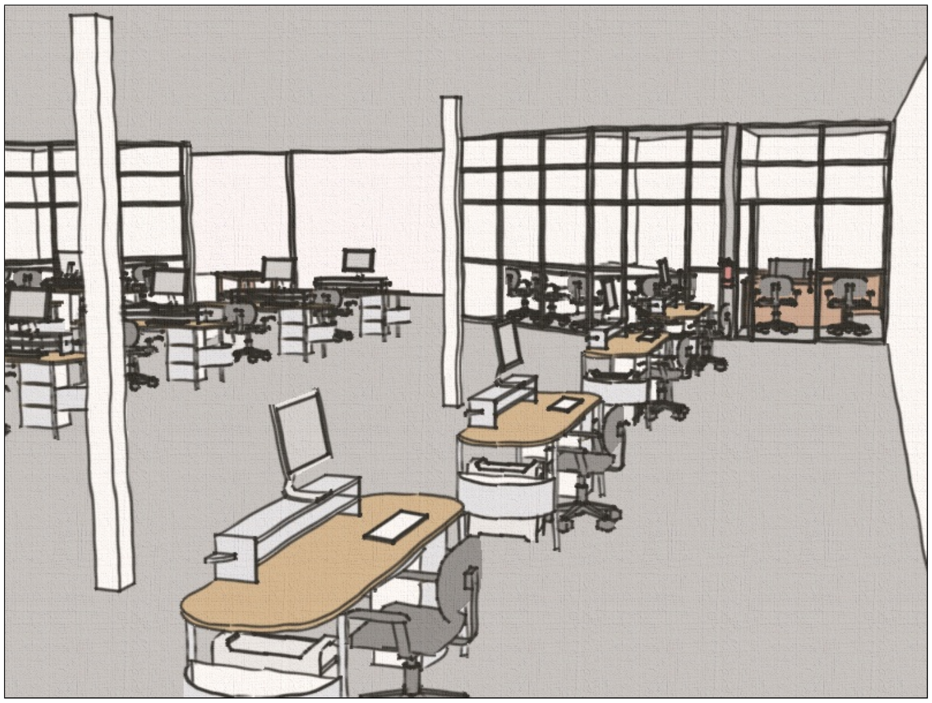


Figure 5.11: *Office model.* The office model has five levels, each with detailed furniture, totaling 330k triangles and 400k line segments.

Chapter 6

Conclusion

This thesis has presented several novel results related to computer-generated line drawings. In terms of impact on future research in computer graphics, the most important are the results of the two empirical studies (Chapters 3 and 4). These two studies are particularly significant for future research into automatic line drawing algorithms, but may also be relevant for studying *how* (as opposed to *how well*) humans perceive line drawings. The new rendering algorithms presented in Chapter 5 are more limited in scope, but will help to make the line drawing style more broadly useful in computer graphics.

The remainder of this section describes the significance of these results and describes several avenues for future work.

6.1 Results for Pure Line Drawings

In the study presented in Chapter 3, we found that the large majority of lines drawn by artists can be explained by known definitions based on local properties of the surface and image. However, the exact thresholds and combinations of definitions for these lines appear to be the result of choices based on the global shape. This is a significant result, because it suggests that further research into local definitions may suffer from diminishing returns. It may be more productive to instead examine better thresholding conditions, or algorithms for intelligently combining different line definitions. Our machine learning algorithm described in Section 3.2.4 is a step in this direction, but considerably more work is required to develop an algorithm that automatically creates artwork comparable to that of a human artist.

The results of the study presented in Chapter 4, however, suggest that this goal may be within reach for certain classes of shapes. The computer-generated drawings used in that study were created with a semi-automatic process: the algorithm decided where to place lines based on current smoothing and thresholding parameters, and the user tweaked those parameters until the desired result was achieved. Remarkably, these semi-automatically generated drawings were often as effective as the human artists' drawings. It is plausible, therefore, that automating the smoothing and thresholding process could produce a complete algorithm for generating effective pure line drawings, at least for the class of models we examined in our study. Automating the choice of thresholds, however, is a challenging problem, and may require building an interpretation model for the human perception of lines.

Our current results are only relevant to how well drawings depict 3D shape, and how well a drawing depicts shape is a very different matter from how "good" a drawing is. For example, human perception may be able to identify and ignore noisy or extraneous lines when considering only shape, even though those lines are considered "ugly." Moreover, a line drawing may be considered a "good" drawing without depicting its subject effectively at all (Figure 6.1). Understanding the differences between an effective shape depiction and a "good" drawing is a challenging direction for future work.

6.2 Future Work in Art and Perception

Our experimental techniques should be sufficient to investigate many further open questions in art and perception. Here are two examples:

When Do Line Drawings Succeed or Fail?

In our gauge study in Chapter 4, drawings of the mechanical parts were more often accurately perceived than drawings of the smooth shapes such as vertebrae. This pattern held even when the subjective quality of the drawings was similar (Figure 6.1). It is not clear whether this pattern is due to the symmetries in the shapes, the familiarity of the shapes, some quality of the line drawings themselves, or some other effect. Future perceptual studies could examine the issue of familiarity by including smooth, but familiar shapes such as eyes, and examine the issue of context by progressively cropping drawings used for stimuli.

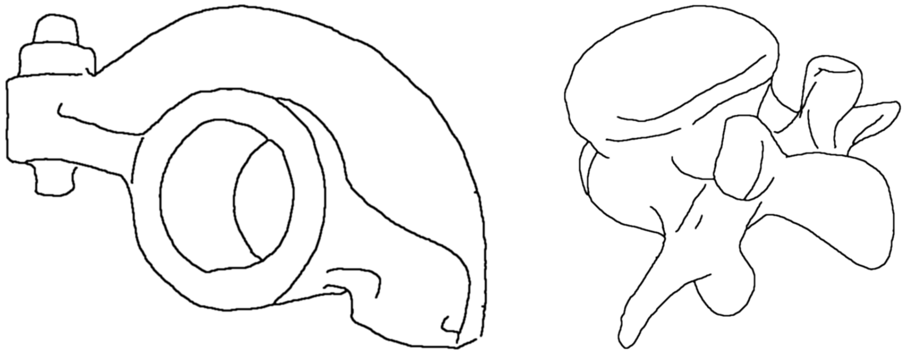


Figure 6.1: *More and less successful drawings.* Subjects achieved approximately twice the accuracy when placing gauges on the rockerarm (left) compared with the vertebra (right). Subjectively, however, the quality of both drawings is similar.

What About Shading?

In our investigations we have tightly restricted artistic style to pure line drawing, using solid, thin, dark, feature lines with no shading. This style is actually uncommon in art in general (Section 2.1), especially when the subject shape is smooth. Even in our studies, artists sometimes chafed against the restrictions on shading.

Effective shading, of course, is perhaps even more difficult than line drawing. Artists have developed techniques such as false light or counterchange to enhance their drawings, and these techniques have been the focus of work in NPR (Figure 6.2). To our knowledge, however, no study has examined the effects of these techniques on perceptual accuracy. Reproducing our drawing collection and perceptual studies with different restrictions on drawing style, or without any restrictions, could begin to answer that type of question.

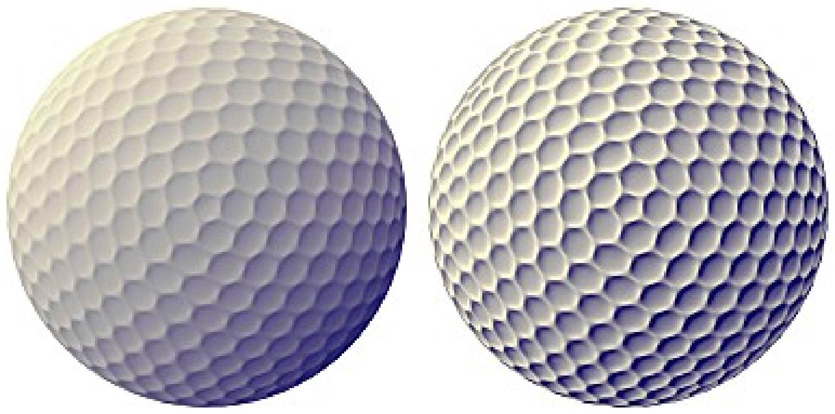


Figure 6.2: *Exaggerated shading.* Techniques such as exaggerated shading [62] are inspired by artistic techniques, but their perceptual impact is unclear.

6.3 Further Technical Challenges

Currently, only a small number of commercial applications (such as Google SketchUp) use the stylized stroke rendering style described in Chapter 5, and none currently use sophisticated line definitions such as suggestive contours or apparent ridges. It seems likely that the lack of acceptance is not due to a lack of demand, but to the technical challenges involved in producing a fast, robust, flexible, high-quality line drawing system.

The new line visibility algorithms presented in this thesis are significant improvements over previous work, and are simple and robust enough to find acceptance in commercial applications. However, visibility testing is only part of the line drawing pipeline. The line elision method presented in Section 5.2 works well in many cases, but suffers when the priority ranking function does not rank lines in order of semantic importance: a poor match between priority and artistic intuition can produce incomprehensible drawings. Future work in this area could examine selection of lines based on this importance.

Additionally, while our empirical studies show that line definitions such as suggestive contours and apparent ridges can be very effective at depicting shape, creating effective drawings using these definitions is currently quite difficult. These definitions depend on accurate and smooth estimation of curvature derivatives, and a dense and regular tessellation for the shape, both of which can be hard to acquire. The best known method is for an expert user (often the researcher who invented the line definition) to hand-tweak the model through careful smoothing and subdivision, producing a model that is just smooth enough so that curvature estimates produce clean results without obliterating necessary detail. Research into more robust curvature estimation could make these line definitions more broadly useful.

6.4 Final Remarks

It is something of a tradition when presenting a new rendering approach to jokingly paraphrase Jim Kajiya, and claim "In ten years, all rendering will be <insert your rendering scheme here>." [1]

But in ten years, computer-generated line drawing will still be a specialized branch of computer graphics, because line drawings are a specialized form of rendering. Even in this specialized branch, it is unlikely that automatically generated line drawings

[1] Kajiya remarked in 1991, "In ten years, all rendering will be volume rendering."

will ever approach the work of the great masters, for the simple reason that no one has yet formalized artistic genius, the way physicists have formalized optics (and thereby paved the way for photorealistic rendering).

However, we can hope that computer-generated line drawings will become more useful for the task to which they are uniquely suited: creating effective, abstract depictions with an aesthetically appealing style. The results presented in this thesis are a step forward, but are by no means a final answer. We hope that the scientific investigations we have presented are closer to the first words on the subject than the last, and believe that the technical innovations we have presented will be superseded by better algorithms in the future. Still, we hope that this thesis will provide useful results for current practitioners, and inspiration for future researchers.

Bibliography

[1] Maneesh Agrawala, Doantam Phan, Julie Heiser, John Haymaker, Jeff Klingner, Pat Hanrahan, and Barbara Tversky. Designing effective step-by-step assembly instructions. *ACM Trans. Graph.*, 22(3):828–837, 2003.

[2] Arthur Appel. The notion of quantitative invisibility and the machine rendering of solids. In *Proceedings of the 22nd national conference of the ACM*, pages 387–393, New York, NY, 1967.

[3] Pascal Barla, Joëlle Thollot, and François Sillion. Geometric clustering for line drawing simplification. In *Proceedings of the Eurographics Symposium on Rendering*, 2005.

[4] Peter N. Belhumeur, David J. Kriegman, and Alan L. Yuille. The bas-relief ambiguity. *International Journal of Computer Vision*, 35(1):33–44, 1999.

[5] Stefan Brabec and Hans-Peter Seidel. Shadow volumes on programmable graphics hardware. In *EUROGRAPHICS 2003*, volume 22 of *Computer Graphics Forum*, pages 433–440. Eurographics, September 2003.

[6] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[7] Michael Burns, Janek Klawe, Szymon Rusinkiewicz, Adam Finkelstein, and Doug DeCarlo. Line drawings from volume data. In *SIGGRAPH '05: Proceedings of*

the 32nd annual conference on Computer graphics and interactive techniques, New York, NY, USA, 2005. ACM Press.

[8] Franck Caniard and Roland W. Fleming. Distortion in 3d shape estimation with changes in illumination. In *ACM Applied Perception in Graphics and Visualization (APGV) 2007*, pages 99–105, 2007.

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

[10] Forrester Cole, Doug DeCarlo, Adam Finkelstein, Kenrick Kin, Keith Morley, and Anthony Santella. Directing gaze in 3D models with stylized focus. *Eurographics Symposium on Rendering*, pages 377–387, June 2006.

[11] Forrester Cole and Adam Finkelstein. Partial visibility for stylized lines. In *NPAR 2008*, June 2008.

[12] Forrester Cole and Adam Finkelstein. Fast high-quality line visibility. In *Proceedings of I3D 2009*, pages 115–120, February 2009.

[13] Forrester Cole, Aleksey Golovinskiy, Alex Limpaecher, Heather Stoddart Barros, Adam Finkelstein, Thomas Funkhouser, and Szymon Rusinkiewicz. Where do people draw lines? *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 27(3), August 2008.

[14] Forrester Cole, Kevin Sanik, Doug DeCarlo, Adam Finkelstein, Thomas Funkhouser, Szymon Rusinkiewicz, and Manish Singh. How well do line drawings depict shape? In *ACM Transactions on Graphics (Proc. SIGGRAPH)*, volume 28, New York, NY, USA, July 2009. ACM Press.

[15] Paul Debevec. Rendering synthetic objects into real scenes. In *SIGGRAPH '98:*

Proceedings of the 25th annual conference on Computer graphics and interactive techniques, pages 189–198, 1998.

[16] Doug DeCarlo, Adam Finkelstein, and Szymon Rusinkiewicz. Interactive rendering of suggestive contours with temporal coherence. In *NPAR '04: Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering*, pages 15–145, New York, NY, USA, 2004. ACM Press.

[17] Doug DeCarlo, Adam Finkelstein, Szymon Rusinkiewicz, and Anthony Santella. Suggestive contours for conveying shape. *ACM Trans. Graph.*, 22(3):848–855, 2003.

[18] Doug DeCarlo and Szymon Rusinkiewicz. Highlight lines for conveying shape. In *International Symposium on Non-Photorealistic Animation and Rendering (NPAR)*, August 2007.

[19] Doug DeCarlo and Anthony Santella. Stylization and abstraction of photographs. *ACM Transactions on Graphics*, 21(3):769–776, July 2002.

[20] Oliver Deussen and Thomas Strothotte. Computer-generated pen-and-ink illustration of trees. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 13–18, 2000.

[21] Roland W. Fleming, Antonio Torralba, and Edward H. Adelson. Specular reflections and the perception of shape. *Journal of Vision*, 4(9):798–820, 2004.

[22] J. M. Fulvio, M. Singh, and L. T. Maloney. Combining achromatic and chromatic cues to transparency. *Journal of Vision*, 6(8):760–776, 2006.

[23] B. Gooch and A. Gooch. *Non-Photorealistic Rendering.* A K Peters, 2001.

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

[25] Stéphane Grabli, Frédo Durand, and François Sillion. Density measure for line-drawing simplification. In *Proceedings of Pacific Graphics*, 2004.

[26] Arthur Leighton Guptill. *Rendering in Pen and Ink.* Watson-Guptill Publications, New York, 1976.

[27] Julie Heiser, Doantam Phan, Maneesh Agrawala, Barbara Tversky, and Pat Hanrahan. Identification and validation of cognitive design principles for automated generation of assembly instructions. In *AVI '04: Proceedings of the working conference on Advanced visual interfaces*, pages 311–319, New York, NY, USA, 2004. ACM Press.

[28] John M. Henderson and Andrew Hollingworth. Eye movements during scene viewing: An overview. In G. Underwood, editor, *Eye Guidance in Reading and Scene Perception*, pages 269–293. Elsevier, 1998.

[29] Aaron Hertzmann and Denis Zorin. Illustrating smooth surfaces. In *Proceedings of SIGGRAPH 2000*, pages 517–526, 2000.

[30] Daniel Horn. Stream reduction operations for gpgpu applications. In Matt Pharr, editor, *GPU Gems 2*, chapter 36, pages 573–589. Addison Wesley, 2005.

[31] Fil Hunter, Steven Biver, and Paul Fuqua. *Light: Science and Magic: An Introduction to Photographic Lighting, 3rd edition.* Focal Press, 2007.

[32] Victoria Interrante, Henry Fuchs, and Stephen Pizer. Enhancing transparent skin surfaces with ridge and valley lines. In *VIS '95: Proceedings of the 6th conference on Visualization '95*, page 52, Washington, DC, USA, 1995. IEEE Computer Society.

[33] Tobias Isenberg, Maic Masuch, and Thomas Strothotte. 3D Illustrative Effects for Animating Line Drawings. In *Proceedings of the IEEE International*

Conference on Information Visualisation, July 19–21, 2000, London, England, pages 413–418, 2000.

[34] Tobias Isenberg, Petra Neumann, Sheelagh Carpendale, Mario Costa Sousa, and Joaquim A. Jorge. Non-photorealistic rendering in context: an observational study. In *NPAR '06: Proceedings of the 4th international symposium on Non-photorealistic animation and rendering*, pages 115–126, New York, NY, USA, 2006. ACM.

[35] Laurent Itti, Christof Koch, and Ernst Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, 1998.

[36] Kyuman Jeong, Alex Ni, Seungyong Lee, and Lee Markosian. Detail control in line drawings of 3D meshes. *The Visual Computer*, 21(8-10):698–706, September 2005. Special Issue of Pacific Graphics 2005.

[37] Tilke Judd, Frédo Durand, and Edward H. Adelson. Apparent ridges for line drawing. *ACM Trans. Graph.*, 26(3):19, 2007.

[38] Robert D. Kalnins, Philip L. Davidson, Lee Markosian, and Adam Finkelstein. Coherent stylized silhouettes. *ACM Transactions on Graphics*, 22(3):856–861, July 2003.

[39] J. J. Koenderink, A.J. van Doorn, C. Christou, and J.S. Lappin. Shape constancy in pictorial relief. *Perception*, 25:155–164, 1996.

[40] J. J. Koenderink, A.J. van Doorn, and A.M.L. Kappers. Surface perception in pictures. *Perception and Psychophysics*, 52:487–496, 1992.

[41] Jan J. Koenderink. What does the occluding contour tell us about solid shape? *Perception*, 13:321–330, 1984.

[42] Jan J. Koenderink, Andrea J. van Doorn, Astrid M.L. Kappers, and James T. Todd. Ambiguity and the ‘mental eye’ in pictorial relief. *Perception*, 30:431–448, 2001.

[43] Michael Kolomenkin, Ilan Shimshoni, and Ayellet Tal. Demarcating curves for shape illustration. *ACM Transactions on Graphics*, 27(5):157:1–157:9, December 2008.

[44] Michael S. Langer and Heinrich H. Bülthoff. A prior for global convexity in local shape-from-shading. *Perception*, 30(4):403–410, 2001.

[45] Yunjin Lee, Lee Markosian, Seungyong Lee, and John F. Hughes. Line drawings via abstracted shading. *ACM Trans. Graph.*, 26(3):18, 2007.

[46] Wilmot Li, Maneesh Agrawala, Brian Curless, and David Salesin. Automated generation of interactive 3d exploded view diagrams. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 27(3), August 2008.

[47] Albert Lorenz. *Illustrating Architecture.* Van Nostrand Reinhold, 1985.

[48] David Luebke, Benjamin Watson, Jonathan D. Cohen, Martin Reddy, and Amitabh Varshney. *Level of Detail for 3D Graphics.* Elsevier Science Inc., 2002.

[49] Pascal Mamassian and Michael S. Landy. Observer biases in the 3d interpretation of line drawings. *Vision Research*, 38:2817–2832, 1998.

[50] Lee Markosian, Michael A. Kowalski, Daniel Goldstein, Samuel J. Trychin, John F. Hughes, and Lubomir D. Bourdev. Real-time nonphotorealistic rendering. In *Proceedings of SIGGRAPH 1997*, pages 415–420, 1997.

[51] Susan E. Meyer and Martim Avillez. *How to Draw in Pen and Ink.* Roundtable Press, 1985.

[52] Alex Ni, Kyuman Jeong, Seungyong Lee, and Lee Markosian. Multi-scale line drawings from 3D meshes. Technical Report CSE-TR-510-05, Department of EECS, University of Michigan, July 2005.

[53] J. D. Northrup and Lee Markosian. Artistic silhouettes: a hybrid approach. In *NPAR 2000*, pages 31–37, 2000.

[54] Yutaka Ohtake, Alexander Belyaev, and Hans-Peter Seidel. Ridge-valley lines on meshes via implicit surface fitting. *ACM Trans. Graph.*, 23(3), 2004.

[55] James P. O'Shea, Martin S. Banks, and Maneesh Agrawala. The assumed light direction for perceiving shape from shading. In *ACM Applied Perception in Graphics and Visualization (APGV) 2008*, pages 135–142, August 2008.

[56] Mark Pauly, Richard Keiser, and Markus Gross. Multi-scale feature extraction on point-sampled surfaces. *Computer Graphics Forum*, 22(3):281–290, September 2003.

[57] Stephen Rogers Peck. *Atlas of Human Anatomy for the Artist.* Oxford University Press, 1982.

[58] Flip Phillips, Morgan W. Casella, and Brian M. Gaudino. What can drawing tell us about our mental representation of shape? *Journal of Vision*, 5(8):522–522, 9 2005.

[59] Flip Phillips, James T. Todd, Jan J. Koenderink, and Astrid M.L. Kappers. Perceptual representation of visible surfaces. *Perception and Psychophysics*, 65(5):747–762, 2003.

[60] Emil Praun, Hugues Hoppe, Matthew Webb, and Adam Finkelstein. Real-time hatching. In *Proceedings of SIGGRAPH 2001*, page 581, 2001.

[61] R. *R: A language and environment for statistical computing.* R Foundation for Statistical Computing, 2005.

[62] Szymon Rusinkiewicz, Michael Burns, and Doug DeCarlo. Exaggerated shading for depicting shape and detail. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 25(3), July 2006.

[63] John Ruskin. *The Elements of Drawing.* Elibron Classics, 1895.

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

[65] Anthony Santella and Doug DeCarlo. Visual interest and NPR: an evaluation and manifesto. In *NPAR 2004*, pages 71–78, June 2004.

[66] Jutta Schumann, Thomas Strothotte, Andreas Raab, and Stefan Laser. Assessing the effect of non-photorealistic rendered images in CAD. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Common Ground*, pages 35–41, 1996.

[67] Shubhabrata Sengupta, Mark Harris, Yao Zhang, and John D. Owens. Scan primitives for gpu computing. In *Graphics Hardware 2007*, pages 97–106, 2007.

[68] Stan Smith. Looking at line. In Readers Digest Editors, editor, *Complete Guide to Drawing and Painting*, page 13. Readers Digest, 1997.

[69] Harold Speed. *The Practice and Science of Drawing, 3rd. Edition.* Dover Publications, 1972.

[70] T. Strothotte and S. Schlechtweg. *Non-Photorealistic Computer Graphics: Modeling, Rendering, and Animation.* Morgan Kaufmann, 2002.

[71] Thomas Strothotte, Bernhard Preim, Andreas Raab, Jutta Schumann, and David R. Forsey. How to render frames and influence people. *Computer Graphics Forum, Proceedings of EuroGraphics 1994*, 13(3):455–466, 1994.

[72] Ivan Sutherland. Sketchpad: a man-machine graphical communication system. *Simulation*, 2(5), 1964.

[73] Jean-Philippe Thirion and Alexis Gourdon. The 3d marching lines algorithm. *Graphical Models and Image Processing*, 58(6):503–509, November 1996.

[74] James T. Todd, Jan J. Koenderink, Andrea J. van Doorn, and Astrid M.L. Kappers. Effects of changing viewing conditions on the perceived structure of smoothly curved surfaces. *Journal of Experimental Psychology: Human Perception and Performance*, 22:695–706, 1996.

[75] David Waltz. Understanding line drawings of scenes with shadows. In Patrick Henry Winston, editor, *The Psychology of Computer Vision*. McGraw-Hill, New York, 1975.

[76] Hank Weghorst, Gary Hooper, and Donald P. Greenberg. Improved computational methods for ray tracing. *ACM Transactions on Graphics*, 3(1):52–69, January 1984.

[77] John Willats. *Art and Representation: New Principles in the Analysis of Pictures*. Princeton University Press, 1997.

[78] Brett Wilson and Kwan-Liu Ma. Rendering complexity in computer-generated pen-and-ink illustrations. In *NPAR '04: Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering*, pages 129–137, New York, NY, USA, 2004. ACM Press.

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

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

[81] Ian H. Witten and Eibe Frank. Data mining: Practical machine learning tools and techniques, 2nd edition. 2005.

[82] YafRay. *YafRay 0.0.9: Yet another free raytracer*. www.yafray.org, 2008.

[83] Long Zhang, Ying He, Xuexiang Xie, and Wei Chen. Laplacian lines for real-time shape illustration. In *I3D 2009*, pages 129–136, 2009.