

ABSTRACT

LEYBA, JASON. Higher Order Primitives for the Reconstruction of Coarsely Sampled Imagery. (Under the direction of Dr. Benjamin Watson).

While computer-processing power continues to grow at rates predicted by Moore's Law, display technologies have lingered and grown at a much slower pace. The ability to display interactive printer-resolution images will not be achievable for several years at the current growth rate of display technologies. To compensate, a method for generating succinct descriptions of hyper-resolution images should be developed. We present such a method that combines sparse color samples with edge information to reconstruct higher resolution images.

Our framework consists of four major phases. In the first phase, we sample the image plane using a sparsely populated grid. We then use a priori knowledge of the scene geometry to detect and record *edgelets*, each of which is a point sample in the image plane containing the location of a detected geometric edge as well as the edge's orientation. The second phase groups edgelets into continuous contours using a function derived from research on the human visual system and the phenomena of illusory contours. In the third phase, the continuous contours are propagated throughout the higher-resolution reconstruction grid to formulate hypotheses about local image edge structures. In the final phase, each reconstruction point gathers color samples from the coarse grid while respecting hypothesized edge boundaries and subjects the samples to a Gaussian filter to determine its contribution to the reconstructed color.

Although more refinements are necessary for ideal reconstructions, results indicate that combining the sparse edge information with sparse color samples is expressive enough

to reconstruct images with crisp edge boundaries, even under conditions of extreme under sampling.

Higher Order Primitives for Reconstruction of Coarsely Sampled Imagery

by

Jason Leyba

A thesis submitted to the Graduate Faculty of
North Carolina State University
In partial fulfillment of the
Requirements for the degree of
Master of Science

Computer Science

Raleigh, NC

2007

APPROVED BY:

DR. CHRISTOPHER HEALEY

DR. WESLEY SNYDER

DR. BENJAMIN WATSON
CHAIR OF ADVISORY COMMITTEE

BIOGRAPHY

Jason Leyba was born and raised in Maryland. Jason graduated magna cum laude from Clemson University in December, 2004 with a B.S. in Computer Science. He will receive his M.S. in Computer Science from North Carolina State University in May, 2007. While at NC State, Jason was a member of the Design Graphics Lab. After graduation, Jason will begin work as a software engineer for a computer software company in Mountain View, California.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Benjamin Watson, for his guidance, support, and encouragement throughout this project. Thanks are also owed to Dr. Christopher Healey and Dr. Wesley Snyder for serving on my thesis committee. Their insight and comments proved to be very helpful. I would also like to thank Chris Sexton, a fellow member of the Design Graphics Lab, who helped me work through many of the problems encountered during the thesis process.

Lastly, I would like to thank my fiancée, Ashley. Without her willingness to listen to me vent about edges, curves, and the Stanford Bunny, I would not have survived the year with my sanity intact.

TABLE OF CONTENTS

LIST OF TABLES.....	vi
LIST OF FIGURES.....	vii
Chapter I: Introduction.....	1
Chapter II: Related Work.....	4
Adaptive Frameless Rendering.....	4
The Importance of Edges.....	7
Illusory Contours.....	8
Contour Integration.....	11
Edge Detection in Completed Imagery.....	16
The Edge as an Image Primitive.....	20
Exploiting Edges in Rendering.....	22
Digital Upsampling with Dense Edge Information.....	23
Digital Upsampling with Sparse Edge Information.....	25
Chapter III: The Framework.....	27
Sampling.....	27
Edge Detection.....	30
Contour Construction.....	32
Forming Hypotheses.....	39
Reconstruction.....	46
Chapter IV: Implementation and Results.....	49
Conclusion.....	59

References..... 62

LIST OF TABLES

Table 3.1	Algorithm for generating $P(x, y)$ over a given image plane using Whitted ray-tracing.....	28
Table 3.2	Edge-sampling algorithm.....	31
Table 3.3	Algorithm for determining the affinity between two edgelets.....	34
Table 3.4	Modified stable marriage algorithm for finding optimal pairings between recorded edgelets.....	37
Table 3.5	Contour creation algorithm.....	38
Table 3.6	Pseudo code for propagating knowledge of contours throughout the sample grid.....	39
Table 3.7	Reconstruction algorithm.....	48
Table 4.1	Statistics for rendered scenes.....	49

LIST OF FIGURES

Figure 1.1	Bandwidth for various display standards, plotted against their release dates	1
Figure 1.2	Total pixels in various display standards, plotted against release dates ...	2
Figure 1.3	Horizontal pixels per inch for various display standards, versus release dates	2
Figure 2.1	Frames of a dynamically changing scene (<i>a</i>) and a static scene (<i>b</i>) rendered by the AFR framework	6
Figure 2.2	The Kanizsa Triangle	9
Figure 2.3	Fragile contours	10
Figure 2.4	A Konizsa square	10
Figure 2.5	An illustration of the Gestalt law of “good continuation.”	12
Figure 2.6	Examples of stimuli used by Hess, Field, and Hayes	14
Figure 2.7	The relationship between two edge elements.....	15
Figure 3.1	Rendering the same image at different resolutions.....	29
Figure 3.2	Illustration of the two half-spaces for an edgelet.....	33
Figure 3.3	Querying the sample grid for contour information.....	41
Figure 3.4	Examples of contour hypotheses.....	43
Figure 3.5	Visualization of the first three phases of reconstruction.....	44
Figure 3.6	The trouble with infinite slopes.....	46
Figure 4.1	A reconstructed cube.....	53
Figure 4.2	A reconstructed sphere.....	54
Figure 4.3	A reconstructed teapot.....	55
Figure 4.4	The reconstructed Stanford Bunny.....	56

Figure 4.5	The effect of d and the number of edgelets on contour creation.....	57
Figure 4.6	The effect of sampling resolution on reconstruction.....	58

Chapter I: Introduction

From the computer animator rendering the most recent frame for the next *Shrek*, to the enthusiastic digital photographer, to the medical imaging specialist, people from across the digital spectrum have been developing ever more demanding ways to tax their displays. However, the ability to generate increasing amounts of information far outpaces the ability to efficiently display it. Moore's Law [1965] states that computers will become 60 times faster every decade, producing machines capable of generating ever more information. Better yet, dedicated graphics processing units (GPUs) have proven to advance at a much quicker rate, growing 1000 times faster per decade [Akeley, 2004; Luebke, 2004; Owens et al., 2005].

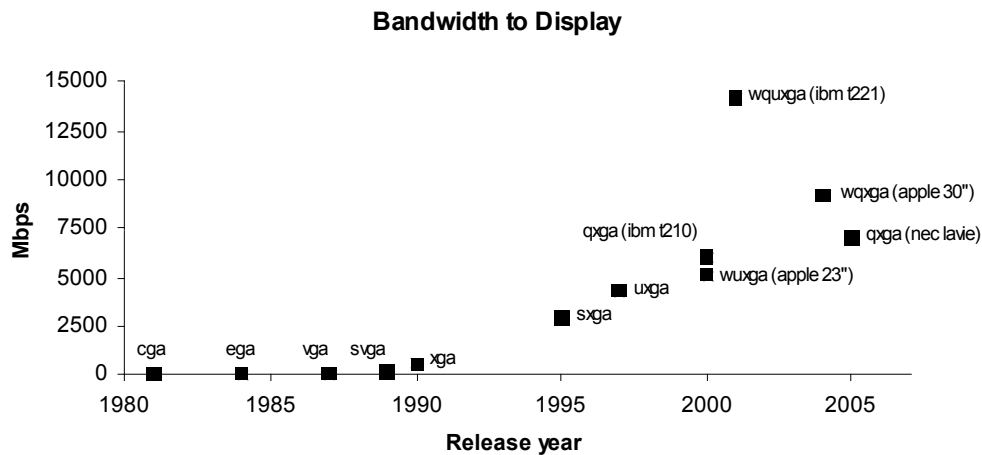


Figure 1.1 Bandwidth for various display standards, plotted against their release dates.

Unfortunately, the displays that are driven by the GPUs have not been able to keep up the pace. In fact, over a 26-year period, display bandwidth has grown by a factor of 43 per

decade (**Figure 1.1**), while pixel count and resolution have only grown by factors of 8.5 and 2 (**Figure 1.2** and **Figure 1.3**). Recent trends are even more alarming. Since 1995, excluding IBM's T221 display (truly before its time and no longer available), bandwidth and pixel count improved by a factor of 3.5 per decade, with resolution advancing by less than a factor of 2.

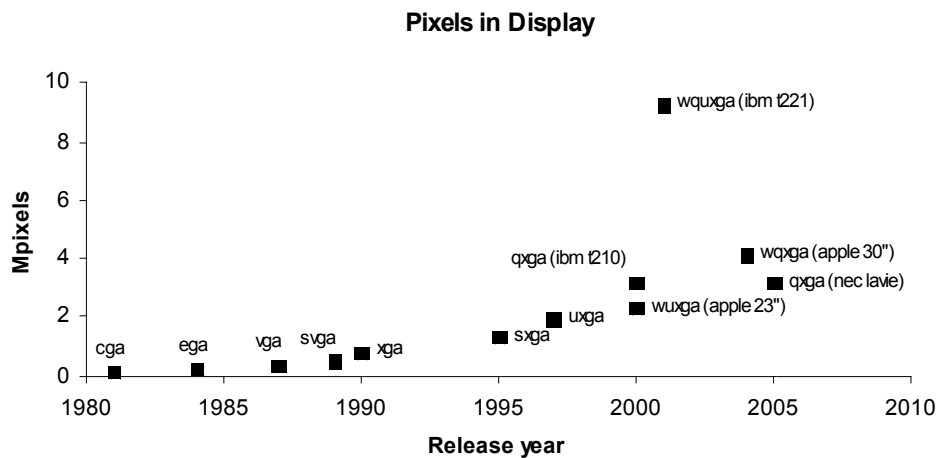


Figure 1.2 Total pixels in various display standards, plotted against release dates.

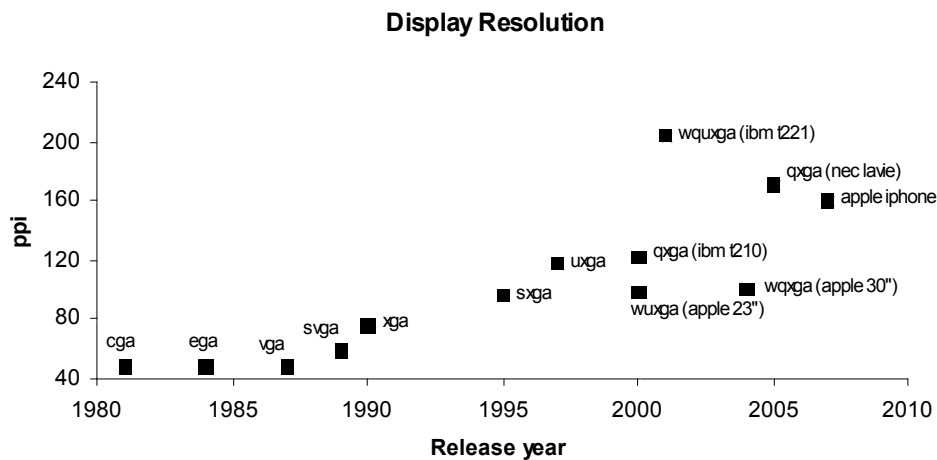


Figure 1.3 Horizontal pixels per inch for various display standards, versus release dates.

Improved display technologies could prove to be amazingly useful. Imagine walls and the tops of desks covered with interactive printer-resolution displays of 600 pixels per inch (ppi) or more. Electronic reading, writing, and working would become a far more pleasant experience, putting less strain on the eyes with crisper fonts, diagrams, and forms. Interactively, the slightest facial expressions could be displayed in videoconferences, improving both business and personal communication. At the current rates, however, the display bandwidth necessary to support a 6'x3' hyper-resolution display at 600 ppi will not be achievable for another 43 years. A larger display of 12'x8' will not be possible for over 50 years. Therefore, software techniques must be developed to overcome these limitations in hardware.

Although it has been a great enabler of video recording, transmission, and playback, rendering techniques based on compression are not suitable for interactive applications. Rather than generating a hyper-resolution image, then compressing, transmitting, decompressing and finally displaying it, a method for directly generating a succinct description of the hyper-resolution image should be developed. Such succinct image descriptions would eliminate the need to create any verbose description before it is local to the display and would bypass compression completely.

The succinct description we propose is based on two bodies of prior work: adaptive frameless rendering and edge perception and detection. We begin with a review of both of these bodies of work, and continue with the development of a new method for reconstructing imagery from a sparse collection of edge samples. We close with an examination of the effectiveness of this method.

Chapter II: Related Work

Adaptive Frameless Rendering

One candidate for succinct hyper-resolution rendering is adaptive frameless rendering. In traditional rendering paradigms, the image data is rendered to a buffer to be displayed on the screen. In a singly buffered environment, a lone buffer is used for both drawing and displaying image data. With high-resolution images, especially in interactive settings, it is often the case that the graphics hardware is unable to fill the buffer with data to draw before it is displayed. This often leads to visible tearing, with the displayed image showing parts of two frames.

To avoid tearing, a back buffer can be used to write image data. Once the back buffer is filled, it is swapped with the front buffer for display on the screen. This improvement does not come without a cost. Since the front and back buffers are not swapped until all of the image data is available, doubly buffered environments suffer from higher display latencies than their singly buffered brethren (all imagery are at least one frame old).

Like the renderers described in Bishop *et al.* [1994], Walter *et al.* [2002], Tolé *et al.* [2002] and Simmons and Sequin [2000], adaptive frameless rendering (AFR) [Dayal *et al.* 2005; Watson and Luebke, 2005] exploits the temporal coherence between samples in an image stream to reuse existing samples for reconstruction of the current image, thus providing a means to break the framed barrier. Instead of sampling the image plane in scan line order and not displaying until all information is available (as in double buffering), AFR randomly samples the image plane and displays the information immediately.

Unlike traditional adaptive rendering techniques, which monitor only sample spatial fidelity within each frame, AFR adds an additional criterion for placing samples by adapting to both the spatial and *temporal* image fidelity. To address the temporal artifacts mentioned above, the AFR framework places samples in a *temporally deep buffer*. This 3D buffer couples a 2D buffer for storing pixel data with a third dimension to store previously sampled pixel data. A sampling controller monitors both the color- and temporal-gradients to place new samples in more changing regions of the image (primarily edges and occlusions); the end result is the image plane being sampled more frequently in dynamic regions and less-frequently in more static image regions.

While AFR produces more temporally accurate information by displaying samples as they become available, it is prone to introducing temporal artifacts into the image since samples taken at different times are displayed together. To compensate, the display of the temporal buffer is throttled, with data being transmitted to a reconstruction buffer at a rate of 60 Hz (this throttling is also required by current display technology, which does not permit true frameless display). The reconstruction module continually applies adaptive space-time filters to reconstruct a coherent image for display.

However, simply applying an arbitrarily shaped space-time filter can have a drastic impact on the reconstructed image. Spatially broad and temporally narrow filters will emphasize relatively recent samples over spatial resolution. Applying these filters to the reconstruction buffer will produce images that reflect the most recently sampled data, but are very blurry due to under-sampling across the image. Conversely, using a temporally broad and spatially narrow filter will increase the display latency as more samples are used to improve spatial resolution. Thus, the spatially broad, temporally narrow filters are more

suited for dynamically changing scenes in which high display latency prohibits the display of spatially accurate information, and temporally broad, spatially narrow filters are more suited for static scenes where the increased latency introduced by displaying older samples does not affect the accuracy of the data displayed. The AFR reconstruction module exploits the strengths of both filters to a given region in the image. Local sampling density is used along with space-time gradients to determine a space-time volume filter to be used in reconstruction.

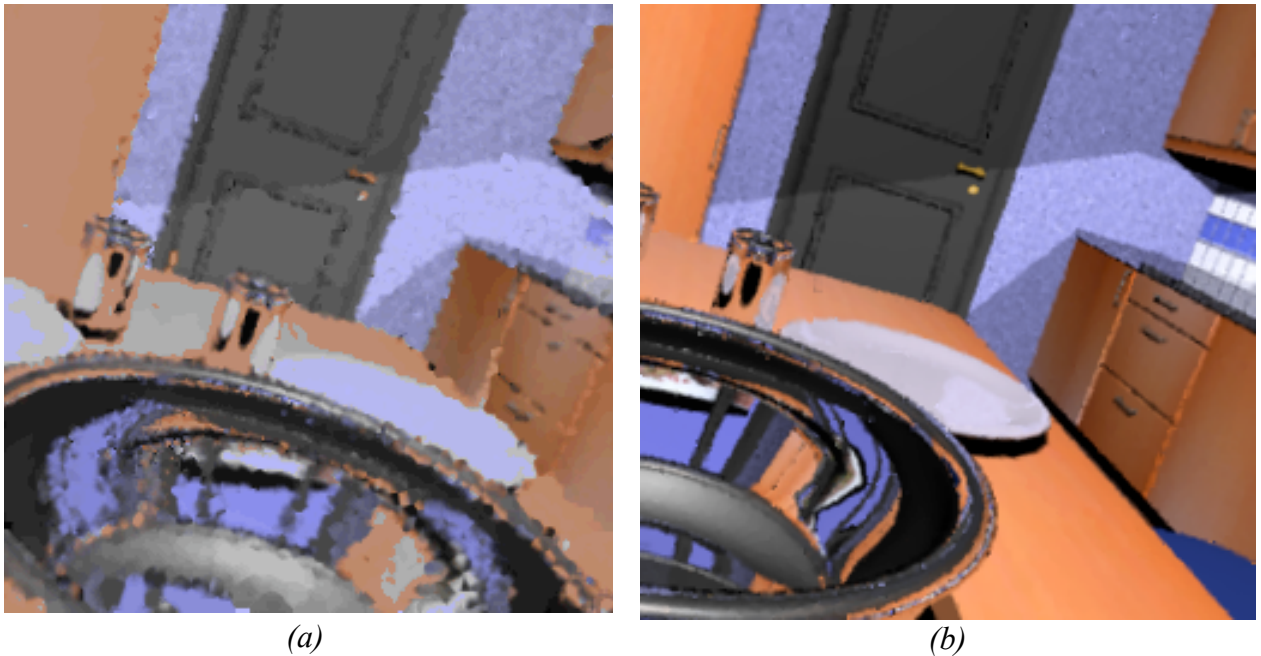


Figure 2.1 Frames of a dynamically changing scene (a) and a static scene (b) rendered by the AFR framework.

Watson and Luebke [2005] were able to compare the performance of AFR and a traditional framed renderer against a zero-delay *gold standard*. Despite only sampling at a tenth of framed renderer's frequency, they found that, in a dynamically changing scene, AFR produced the same root-mean-square (RMS) error as traditional framed renderer.

Nevertheless, the artifacts exhibited by AFR, especially around leading, occluding edges (**Figure 2.1**), make it problematic for hyper-resolution images.

This thesis proposes modifying the rendering paradigm to produce “higher-order” sampling primitives that will permit reconstruction reducing AFR’s artifacts, and enabling the production of good imagery with even fewer samples. Traditional sampling primitives simply describe color at a certain position. AFR already supplemented this information with descriptions of age and local color gradients. We propose the addition of edge information to samples, therefore providing a means of informed upsampling and allowing even an extremely sparse set of samples to express enough information to faithfully reconstruct the image. For the derivation of our primitives we turn to research on the human visual system for inspiration.

The Importance of Edges

AFR is capable of responding to changes in dynamic scenes rapidly and with good accuracy in the resulting imagery, but it introduces blurriness at the edges in dynamic scenes. Is sacrificing sharp edges for low latency an acceptable trade-off? How important *are* edges to the human visual system (HVS)? Although the importance of edges has been the subject of decades of research, we will only review a small fraction of these works.

Beusmans *et al.* [1978] argue that the purpose of any vision system is “to assign the most plausible interpretation to images of the external...world.” However, the projection of 3D space onto a 2D image results in an image that could represent many disparate, yet plausible environmental configurations. Noting that “human observers can often recognize familiar objects from their contours or silhouettes,” Beusmans *et al.* investigated whether this

observation could be combined with the two-stage process to perceptual understanding proposed by Warrington and Taylor [1973, 1978] wherein objects are first categorized by their boundaries and then identified by their surface qualities.

Beusmans *et al.* found that the complex 3D objects could be represented by a more compact construction of simple objects combined through Boolean operations: “solid union (to form humps), solid subtraction (to leave dents), and smoothing (to remove discontinuities).” The resulting deep structures formed a sort of shape-memory. Beusmans *et al.* made no claims that their model was how the HVS operated; nevertheless, they argued that any organism must maintain a similar internal shape memory for representing important scene details. Since it had been demonstrated that the HVS was capable of distinguishing between different categories of objects (*i.e.* a human compared to a dog) by their boundary shape alone, Beusmans *et al.* argued that this was evidence of humans relying on an internal shape memory. Once this memory was accessed to categorize an object by boundary shape, the individual object could only be distinguished through “surface structure.” While Beusmans *et al.* proposed a multi-stage method for object recognition based on first categorizing by object boundaries and then analysis of surface detail, there is ample evidence to suggest that the HVS is capable of using edge information (boundaries) at a much lower level to recognize objects quickly.

Illusory Contours

Beusmans *et al.* argued that 3D boundaries of objects formed the basis for the deep structure of an object. They argued that this structure was similar to the mechanism that allowed humans to first recognize the object and that the surface detail was what was later

used to distinguish between distinct objects with similar boundaries. This reliance on complete 3D boundary recognition, though, contradicts an oft-studied phenomenon in the HVS: illusory contours.

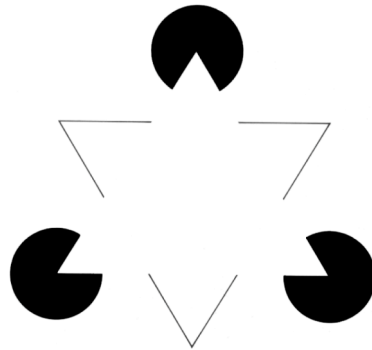


Figure 2.2 The Kanizsa triangle.

Illusory contours are optical illusions where the HVS perceives an edge to exist where there is, in fact, none. The most famous of these optical illusions is the Kanizsa Triangle (**Figure 2.2**), first discovered in 1955 by Gaetano Kanizsa. When viewing the Kanizsa Triangle, the HVS perceives a bright white triangle located on top of three black discs and another triangle. What causes the HVS to interpret the image in this manner is still unknown, but the prevailing belief is that it is a compensatory mechanism for “degraded visual conditions” [Murray *et al.* 2006]. Since the HVS operates on a 2D projection of a 3D scene, there are often discontinuities in image boundaries, or ill-defined differences in luminance on opposite sides of a real edge, causing the edge to be indistinguishable by the viewer. The phenomena of illusory contours compensates by, in essence, connecting the dots to derive a more logical interpretation of a visual scene. In fact, the optical illusion can even overcompensate: in the Kanizsa Triangle, the perceived floating triangle actually appears to

have a brighter color than the triangle beneath it and the background, when it is actually the same color.

Interestingly, von der Heydt *et al.* [1984] were able to show that these powerful optical illusions are extremely fragile. Citing the Gestalt theory that the “whole is greater than the sum of its parts,” they noted that when “one of the disk sectors in [the Kanizsa Triangle] is occluded...the corner of the triangle disappears.” Furthermore, “small changes in configuration can have dramatic effects on the appearance of illusory contours.”

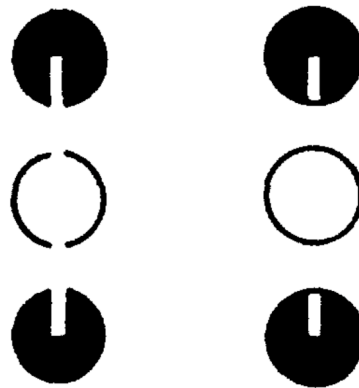


Figure 2.3 Fragile contours. Left: an example of an illusory contour where a white bar is visible above the three dots. Right: by making the small change of closing the openings in the three dots, the visible white bar from the left image disappears. From [von der Heydt *et al.*, 1984].

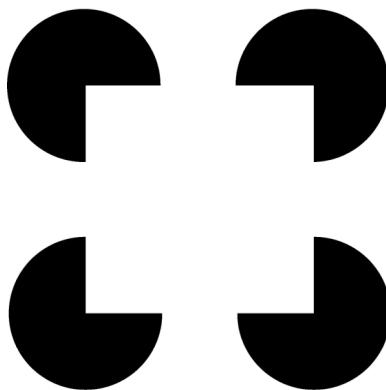


Figure 2.4 A Konizsa square. In the “thin/fat” test, the inducer disks are inverted by $\pm 6^\circ$ to produce illusory contours that make the square appear either squished or bloated. The reader is referred to [Ringach and Shapley, 1996] for additional information.

The deep structure shape memory proposed by Beusmans *et al.* is not completely refuted by illusory contours. Indeed, there is evidence supporting their two-step shape discrimination process. In 2006, Murray *et al.* used the “thin/fat” task [Ringach and Shapley, 1996] to test participant’s abilities to respond to illusory contours and correctly distinguish between shapes based on those contours. Each subject was shown an array of Kanizsa squares (**Figure 2.4**) and was first asked to indicate whether they could distinguish a contour. Next, they were tasked with classifying the perceived shape (while terms of “thin” and “fat” are used to describe the test, these terms are never given to the participants). By analyzing subjects’ electroencephalographs (EEGs), Murray *et al.* found that shape discrimination does occur in multiple stages with the perception of illusory contours occurring 200 ms before shape discrimination. They also noted that contour perception seemed to occur “automatically” with subjects completing the illusory boundaries regardless of their ability to classify the shape correctly.

Contour Integration

It has long been observed that illusory contours are an example of perceptual grouping of local image features [Gibson, 1966; Marr, 1982]. In particular, it is an example of the grouping of edge information, which, as previously discussed, occurs in images at locations of discontinuity in luminance, color, and so on. It is important to note that these elements of discontinuity (*i.e.* edges) are not uniform across the image and can occur with great rarity with respect to total image area. *Contour integration* is the process by which the HVS analyzes these sparse edges and constructs a meaningful interpretation (the illusory

contour). Gestalt psychologists have long argued that, at the most basic level, this process is governed by the law of “good continuation” [Wertheimer, 1923/1958]. That is, the HVS will connect edges into contours in a directional pattern established by existing edge information (Figure 2.5).

Early research into illusory contours suggests that neurons in the visual cortex act as feature detectors, responding to specifically oriented stimuli [Hubel and Wiesel, 1968]. Indeed, this line of research has found that these cells are capable of responding to features at a resolution far greater than their physical configuration: visual acuity has been demonstrated at 30 – 60 *arcsec*, while vernier acuity has been as high as 5 *arcsec* of visual angle [Edelman and Weiss, 1995]. These feature detectors gather information for processing and integration.

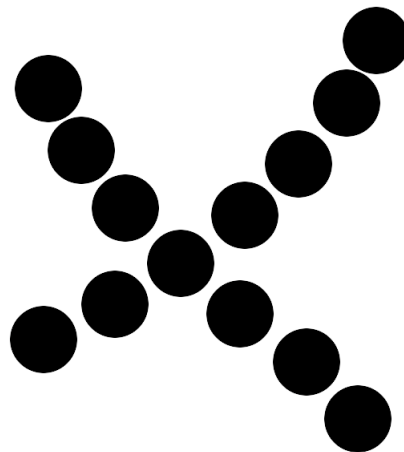


Figure 2.5 An illustration of the Gestalt law of “good continuation.” While the image above could represent many different configurations of a pair of lines, the human visual system will interpret it as two contours: one from the lower-left to the upper-right, and the other from the upper-left to the lower-right.

Hess, Field, and Hayes [1993] tested the physical configuration of these oriented feature detector cells to gain insight into how they utilized “redundancy in continuous, but non-aligned features” in space. They presented observers with arrays of Gabor patches arranged in random orientations. Within this background “noise,” they inserted similarly oriented patches to form continuous paths (**Figure 2.6**). They found that is improbable for any single neuron to be responsible for the detection of continuous contours. Instead, they argued cells worked within a local neighborhood to form an “association field” for responding to continuous contours. These contours were most easily detected when pair wise elements were oriented 60° relative to one another. Observers’ ability to detect a continuous curve was hindered most significantly when elements were orthogonally aligned.

Hess and Field continued their experiments in 1995 with an analysis of the effect of varying depth planes between contour elements. They found that similar to their two-dimensional experiments, element orientation had a large impact on the perception of contour paths. Differences in depth between elements, however, had little impact, suggesting that the HVS integrates edge information across disparate depth planes when constructing contours. Indeed, in a later study [2000], they found that differences in polarity across edges had a larger impact than the distance between individual elements. The HVS shows a higher sensitivity to contour paths where “the phase value of all elements is the same,” regardless of what the phase may be.

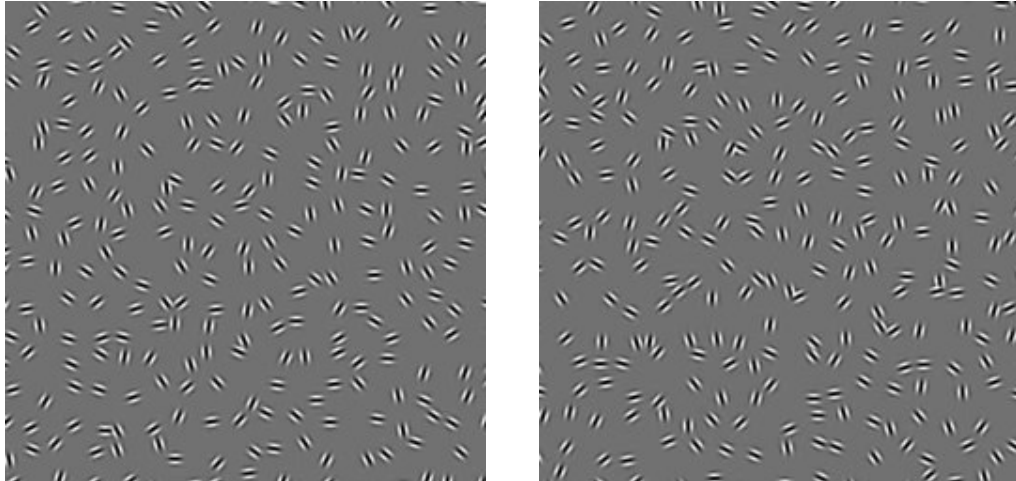


Figure 2.6 Examples of the stimuli used by Hess, Field, and Hayes [1993].

Perhaps the most beneficial insight to be gained from research on contour integration can be found in the work of Geisler *et al.* [2001]. Geisler thoroughly analyzed the co-occurrence statistics of edges and contours in natural images to develop a means of predicting grouping in contour integration by the HVS. They found that the probability of two edges being grouped together in a contour could be characterized by three parameters (**Figure 2.7**): the distance between edges, d , the orientation difference between elements, θ , and the directional angle of the second element in the pair relative to the first, ϕ .

These relationships provided a mathematical basis for predicting the associative fields studied by Hess and Field. In fact, the parameters d and θ have already been accounted for by Hess and Field. The third, while not explicitly dealt with in Hess and Field's work, can be intuitively inferred from the Gestalt law of "good continuation:" if the relative angle between two edge elements (ϕ) is large, they would not form a continuous contour, regardless of their similarity in orientation (θ) and proximity to one another (d).

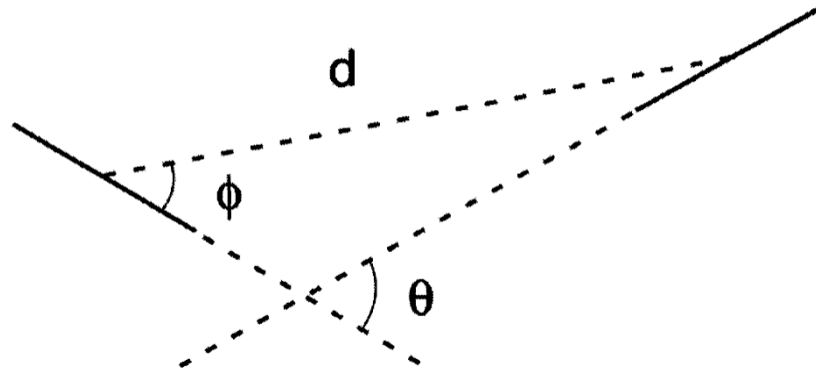


Figure 2.7 The relationship between two edge elements. The two edges are likely be integrated if the distance between them, d , is small, they are similarly oriented, θ , and the directional angle of the second element relative to the first, ϕ , is small. From Geisler *et al.* [2001].

The edge co-occurrence statistics extracted from natural images by Geisler elucidated the aligned and parallel structures of contours preferred by the HVS. Yet, to complete their theory of contour integration, a fourth parameter was added to their structures, that of transitivity. If edge element A connects to element B, which in turns connects to element C, then it follows that a single contour can be formed as the union ABC. Geisler *et al.* demonstrated the strength of their analysis by building a Bayesian contour detector that was shown to reliably detect the same “naturally-shaped contours in complex backgrounds” as a human observer.

According to Hoffman [1998], the HVS is continuously performing this process of contour integration. The HVS, he states, constructs “continuous lines and surfaces...from discrete information.” Given a discrete retinal image, the HVS identifies edges, corners, and junctions [Hubel & Wiesel, 1968; Das & Gilbert, 1999; Hansen & Neumann, 2004]. Then, in accordance with Gestalt perceptual principles such as the law of “good continuation,” it integrates these disconnected primitives into continuous 2D contours [Geisler *et al.*, 2001;

Hess *et al.*, 2003; Field & Hayes, 2004] and eventually 3D percepts [Beusmans *et al.*, 1987; Biederman & Cooper, 1991].

We submit that a similar process can be envisioned for reconstructing crisp, high-resolution images from a sparse collection of color and edge information. Before formulating such a process, though, we investigate existing systems that extract edge information from completed imagery.

Edge Detection in Completed Imagery

Early work in edge detection focused on filtering an image with a smooth function and then sampling the image for large luminance discontinuities, signaling the presence of an edge. In his seminal paper, Canny [1986] first smoothed the image with a Gaussian filter to remove noise. Next, he sampled the image gradient to identify areas with a magnitude greater than some threshold (identified by predetermined image statistics). In these areas of high gradients, Canny sampled the second derivative in the direction of the gradient to identify “zero-crossings.” Canny argued that these zero-crossings signaled local maxima and minima in the image, corresponding to edges in the image.

Canny’s edge detection has proved to be quite reliable, yielding low error-rates (limited false-positives and false-negatives), low spatial errors (detected edges are minimally distant from the true edge), and minimizing multiple responses to unique edges. Nevertheless, Canny’s algorithm relies only on the physiological observation that edges correspond to areas of discontinuity and makes no use of corners or edge junctions, points that are often referred to as *keypoints*. As noted by von der Heydt *et al.* [1984], “corners and line ends play a role in the formation of contours because these picture elements are

frequently produced by interposition of objects, that is, when an object partially occludes others.” Many have proposed techniques to improve Canny’s process to include these important perceptual features.

Rosenthaler *et al.* [1994] argued that the fundamental flaw in Canny’s algorithm is that is “intrinsically one-dimensional,” and therefore incapable of detecting keypoints (corners, end points, and edge junctions). They proposed a method of detecting one-dimensional variations in intensity (the general edges of Canny) and a means for detecting two-dimensional variations, indicating the presence of a keypoint feature.

As in Morrone [1988] and Perona [1990], Rosenthaler *et al.* use oriented gradient filters to sample the image gradient, identifying general edges at local maxima. However, Rosenthaler *et al.* permit general edges to have “arbitrary profiles.” The end result is that their filters are “contrast independent” and will only respond to general edges (*i.e.* they will not mislabel keypoints as general edges and vice versa). Since each general edge has an arbitrary profile, Rosenthaler found that traditional linear filtering was incapable of properly localizing an edge. Instead, they used pairs of even and odd symmetrical filters, each a Hilbert transform of the other, to characterize local energy.

Using these filters to sample the directional derivatives, Rosenthaler *et al.* were able to identify general edges where the derivative was near zero. On the other hand, local extrema in the directional derivative would signal the presence of a keypoint. At keypoint locations (corners, line ends, and T-junctions), a unique edge orientation would be undeterminable due to the second directional derivative exhibiting a higher rate of change. Therefore, to sample the orientation of keypoints, Rosenthaler *et al.* measured the differences in the directional derivatives parallel *and* orthogonal to the oriented edge. While they were

able to demonstrate that an analysis of the first and second directional derivatives could reliably detect and differentiate between general edges and keypoints, they were unable to develop specific models for identifying the different types of keypoints.

Using similar methods, Heitger *et al.* [1994] were able to detect not only edge characteristics, but to group them into contours. Influenced by von der Heydt's bottom-up, multistage models of contour detection mechanisms in the HVS [1984], Heitger *et al.*'s model integrates "end-stopped cell response over [local neighborhoods] in the image" to infer local occlusion-induced contours. The model builds two distinct representations of the image. The first is a collection of edges and lines, built (like Rosenthaler) using oriented filters to detect "1-D signal variations" along the image gradient. Similar oriented filters sampled the first and second directional derivatives of local gradients to detect keypoints. The second representation groups edges and lines using the assumption that keypoints signal illusory contours caused by foreground objects occluding background objects. Such keypoints will respond to similar end-stopped operators. With this approach, the Kanizsa triangle (**Figure 2.2**) represents a white triangle in front of three black disks against a white background. Keypoints are located where the triangle occludes the edges of the discs near the white background. The resulting system groups these keypoints and reconstructs the missing edges (*i.e.* the illusory contours perceivable by the eye).

The edge detection techniques discussed so far rely on image gradients calculated at a fixed image resolution. In contrast, Park *et al.* [1996] point out that the intensity variations that signal an edge can occur at different resolutions within the image. Since the models presented thus far require input parameters to be fine-tuned for a given resolution, they would not handle more complex images containing these regions of varying edge scale. Park

et al.'s method adaptively adjusts the size of the evaluation filters based on the statistical characteristics of the input image. First, the method divides the input image into many different regions of varying spatial resolutions. The local resolution for each pixel is determined by a discontinuity measure evaluated over the nearby regions. Each pixel is considered "to be the center of several neighborhoods of $(2p + 1) \times (2p + 1)$ windows," where p specifies the size of the neighborhood under consideration. Pixels within the window with similar resolutions are grouped into homogenous regions. Once individual regions are determined, Park *et al.* apply a traditional edge detection technique such as Canny, to evaluate the region at its local resolution.

While Park *et al.*'s technique was less prone to false-positives than other techniques, it was not truly adaptive to multiple resolutions. Their system merely divided an image into multiple regions and then applied the single-resolution techniques that they originally criticized. Law *et al.* describe what may be a more adaptive edge detection technique based on fuzzy-reasoning [1996]. Much like Canny, they begin with a Gaussian filter to remove noise and enhance edges. This filter, however, is adaptive to local edge information and avoids smoothing across edges. Law *et al.* found (like Rosenthaler *et al.*) that image gradients provide a means for detecting but not *classifying* keypoints (corners, T-junctions, etc.). They perform classification after filtering by evaluating the "gradient, symmetry, and straightness" near detected edge candidates. Points with a high degree of symmetry (orthogonal to the gradient), gradient strength, and edge straightness are edge points. Points with high gradient and symmetry, but low straightness are corners, and points with high gradient, but low symmetry and straightness are T-junctions. Finally, the technique iterates over the identified edge candidates and joins them into consistent contours. It builds straight

contours where points are well aligned and close together; corners and T-junctions where two or three straight edges have a strong likelihood of connecting with a common keypoint.

This survey of edge detection literature reveals the pervasive influence of the HVS system. Edge elements are detected in image regions of high gradient, echoing the importance of polarity and contrast studied by Hess and Field. Where edge elements are analyzed to infer a higher-order structure, the tenets of perceptual vision were applied, most notably the emphasis placed on keypoints and the enforcement of the law of “good continuation.” All of these papers, though, focus on analyzing existing imagery as a means to model and better understand how we perceive the world around us. This thesis, on the other hand, aims to utilize the principles of human perception to succinctly generate an image at a low sampling frequency and then utilize sparse edge information to digitally upsample and produce an image of higher-resolution, much in the same way the HVS infers illusory contours and other complex information from sparse visual stimuli.

The Edge as an Image Primitive

Many techniques exist for the purpose of *detecting* or *extracting* edges from an image, but this thesis explores the use of edge primitives as image descriptors. We are certainly not the first to consider this possibility: Elder [1999] asked if edges alone were enough to represent an image. For such a possibility to hold, Elder asserted that edges would have to be: explicit enough to represent an image more succinctly than a standard array of pixels and complete enough to represent the image without the loss of “information of potential perceptual relevance.”

To test his theories, Elder reduced the representation of an image from an input pixel array to an array of localized edges. Information is only recorded at the locations where an edge has been detected, providing a more explicit representation of an image than the input pixel array. To ensure completeness, each edge point recorded four parameters: two for the asymptotic image intensity on either side of the edge, one for the blur scale of the edge, and a final parameter to encode the direction of the edge gradient. The image intensities are necessary for reconstructing tonal values in the image, and the edge gradient is necessary to guide this reconstruction. Elder argues, intuitively, that contrary to the step functions used in most edge detection papers, natural edges are not sharply defined. Rather, they are characterized by a gradual distortion in luminosity (*i.e.* the soft shadows caused by diffuse surface reflections). To compute this blur parameter, Elder bi-linearly interpolates across the edge gradient by a distance determined by the standard deviation in a Gaussian blur kernel.

After encoding an image with the four parameters described above, Elder applies a two stage reconstruction algorithm to decode the image. In the first stage, the encoded image intensities are interpolated throughout the image along the image gradients. To reduce the complexity of this initial interpolative step, Elder operates under the assumption that the Laplacian of the original image's gradient will be zero at all non-edge points. Thus, the task of interpolating the encoded intensities is reduced to minimizing the Laplacian over the image. Elder notes that the tradeoff to this simplification is that the Laplacian sharpens all of the encoded edges and all of the soft shadows are lost. To compensate, a second phase was added to reconstruction in which the Laplacian was once again minimized over the image, but this time over the encoded blur factors and not the edge intensities. Elder found that his edge representations were able to compact the encoding of images and produce high-fidelity

reconstructions. Nevertheless, his technique lost information in image regions of great change, most notably in patterned textures such as hair.

Exploiting Edges In Rendering

Edges are crucial in perception and they are quite promising as an image primitive, but can they be found *during* the image creation process to facilitate reconstruction at a frequency much higher than sampling frequency? We are not the first to attempt to use edges in such a manner.

Early work in ray tracing found that, while it was quite expressive, it was prone to aliasing because of its reliance on point sampling [Whitted, 1980]. Increasing the sampling density significantly reduced these aliasing artifacts, but significantly increased rendering time. Mitchell [1987] traded rendering speed for rendering quality with adaptive anti-aliasing, which samples image regions with high image gradients more aggressively. Mitchell notes that, just as in edge detection for image processing, edges are likely to occur in areas of high gradient and are prone to aliasing artifacts, thus requiring denser sampling. Mitchell's was only the first of many adaptive rendering systems.

Edges are also extremely important in non-photorealistic rendering (NPR) techniques, which mimic artistic renderings, including cartoon, pencil and pen. Two important examples are the works of Salisbury *et al.* [1996] and DeCarlo *et al.* [2003]. Salisbury *et al.* developed image representations that could be used to generate pen-and-ink-like renderings for any display resolution. Salisbury found that the quality of his pen-and-ink renderings was especially dependent on well-defined input edges. At increasing resolutions the approximate tonal variations of his output ink strokes and hatches would degrade faster than the tones in

traditional images, blurring the edges and altering the interpretation of surface texture, making his renderings seem less “pen-like”. To avoid this impression, he embedded “discontinuity edges” into his input imagery along with the uniformly spaced image samples. During reconstruction, pen-and-ink style hatching patterns would be applied in such a way that the discontinuity edges remained sharply defined while tonal variations were smoothly interpolated everywhere else.

DeCarlo *et al.* emphasize edges and contours in an image to more effectively convey the shape and surface detail of 3D geometric objects. DeCarlo exploits the perceptual tendencies of the HVS to identify the contours in a 3D model that meaningfully convey its shape. Echoing Beusmans’ argument that silhouette information alone can only identify a broad category object, DeCarlo’s rendering also detected “suggestive contours,” points on an object’s surface where slight variations in viewing angle would produce a silhouette. DeCarlo argued that combining these suggestive contours with true contours provided enough information for the HVS to meaningfully interpret both the shape and surface detail of an object.

Digital Upsampling with Dense Edge Information

In traditional pixel array representations of digital imagery, each pixel represents a fixed region in the image. These pixels are an inherently limited representation of the scene, as each pixel can only approximate the image intensities in the region it represents. Mitchell’s other adaptive renderers strive to overcome this limitation by sampling areas of high intensity variance more finely. Nevertheless, without explicit representation of edges, perceptually crucial contour information is lost.

Tumblin and Choudhury [2004] address this loss of information by integrating scene boundary information directly with traditional pixels, which they called *bixels*. In this image representation, the most important perceptual features are encoded directly into the bixel. Like pixels, bixels are arranged in a uniform sampling grid and contain a color point sample approximating the continuous image plane. Additionally, the bixel encodes local scene boundaries at sub-pixel precision. Not only are these boundaries recorded at sub-pixel precision, but they represent infinitely sharp edges that can be arbitrarily enlarged without the loss of information. Each bixel is encoded as a 2x2 array of tiles. Scene boundaries are detected and recorded within each tile with the following constraints:

- 1) Each tile can contain at most one boundary position.
- 2) At most, one boundary may cross a tile border.
- 3) No boundary segment may cross or intersect the corner of a tile.

These constraints ensure that all bixel-recorded boundaries will represent straight lines. To display a bixel, a bilinear filter [Tomasi and Manduci, 1998] is applied that interpolates the intensity across each tile separately. In tiles containing a scene boundary, the color intensity is only interpolated across the side of the boundary containing the sample. For the remaining side, the color to interpolate is in a neighboring tile (which may or may not be a member of the same bixel).

Bala *et al.* [2003, 2006] integrate a similar edge description into their renderer. For their purposes, Bala *et al.* define an edge in a scene as the location of an important color discontinuity, such as silhouette edges, occluding edges, and penumbral and umbral shadow edges. First, their renderer rasterizes silhouettes, feature edges and shadow edges into a high-resolution buffer at sub-pixel (8x8) precision. Next, it sparsely populates the same

buffer with traditional color point samples. Finally, the renderer reconstructs the resultant “edge-and-point” image using a process similar to bixel reconstruction.

Digital Upsampling with Sparse Edge Information

In making heavy use of edges as an image primitive, our research draws heavily from the work of Elder, Tumblin and Bala. Unlike Elder, and like Tumblin and Bala, we supplement explicit edge information with color samples. Unlike Tumblin and Bala, though, we assume that available edge information is sparse and disconnected, a natural assumption at hyper-resolutions. To permit upsampling in the face of such sparseness, we draw inspiration directly from Geisler’s research on contour integration.

The framework presented in the remainder of this thesis consists of four major phases. In the first phase, we sample the image plane using a sparsely populated grid. We then use *a priori* knowledge of the scene geometry to detect and record *edgelets*, each of which is a point sample in the image plane containing the location of a detected geometric edge as well as the edge’s slope. Note that edge location is recorded at hyper-resolutions much finer than the sparse grid.

The second phase groups edgelets into continuous contours. We first compute the strength of association between all edgelet pairs using a function derived from the co-occurrence analyses of Geisler *et al.* [2001]. Next, we pair edgelets using an adjusted stable roommates algorithm and apply Geisler’s rule of transitivity to form continuous contours. Knowledge of these contours is then propagated locally into the sparse sampling grid.

In the third phase, each point on the upsampled reconstruction grid queries the nine coarse samples surrounding it in the image plane for contour information and extracts the two

closest contours. From these contours the reconstruction point will form a hypothesis about its local contour structure to guide the gathering process in fourth phase: there may be a single contour, an L-junction of two contours, a T-junction, or two contours may cross.

In the final phase, each upsampled reconstruction point queries the coarse sampling grid for points to use in reconstruction. Each candidate sample is tested against the hypothesized contours from the third phase. Only those located on the same side of the contour(s) as the reconstruction target are gathered. A Gaussian filter is applied to the gathered samples to determine their contribution to the reconstruction target. Finally, the weighted contributions are summed together to reconstruct the final color of the point in the image.

Chapter III: The Framework

Sampling

A two-dimensional image is defined as a continuous function of light intensity $I(x, y)$ over a plane located in world space. Computers are incapable of directly representing the continuous domain of I . Instead, a rectilinear array of discrete point samples is used to approximate I at a given resolution. For an image displayed with a resolution of $width \times height$, there will be $width$ discrete samples across the horizontal axis (left-to-right), and $height$ samples across the vertical axis (top-to-bottom).¹ As the resolution of an approximation of I is decreased, its grid will yield a coarser sampling of the image plane. To generate a sparse set of color-point samples for use in reconstruction, we combine this basic property of computer imagery with a simple Whitted ray tracer [1980].

The domain of I is defined over an image plane whose position in world space can be specified by three sets of coordinates: its upper-left corner UL , its upper-right corner UR , and its lower-left corner LL . If P is the rectilinear set of samples approximating I , then at a given resolution $width \times height$, each grid cell in P will cover a region in I characterized by the following two vector equations:

$$\mathbf{dX} = \frac{(\mathbf{UR} - \mathbf{UL})}{width}, \quad \mathbf{dY} = \frac{(\mathbf{LL} - \mathbf{UL})}{height}$$

¹ It is important to note that while each screen pixel will have a corresponding point sample, the two are not equivalent. The rectilinear array of point samples approximating the continuous function I must be subjected to a reconstruction filter to interpolate between the samples and fill in the geometric area of a screen pixel. The various methods for performing this operation are beyond the scope of this thesis and the reader is referred to Smith (1995) for more information.

Furthermore, the exact location for a point sample with x and y rectilinear array coordinates in P is defined by

$$P(x,y) = \mathbf{UL} + (x + 1/2)\mathbf{dX} + (y + 1/2)\mathbf{dY}.$$

Take note that the x and y grid coordinates are offset by half a grid cell to place the sample directly in the center of each cell. To display the points in P , each point sample is projected onto the screen as OpenGL `GL_POINTS` primitive under an orthographic projection. By using `GL_POINTS` for displaying each sample, the geometric region covered by each grid cell is filled with the color of the sample in the center of the cell. The pseudo code for generating P over a given image plane and scene is show below in (**Table 3.1**).

Table 3.1 Algorithm for generating $P(x, y)$ over a given image plane using Whitted ray-tracing.

```

RAYTRACE (P, image, scene)
1  for i ← 0 to P.width do
2    for j ← 0 to P.height do
3      P[i, j].position.x ← (1 + 2 * i) / (2 * P.width)
4      P[i, j].position.y ← (1 + 2 * j) / (2 * P.height)
5      ray.origin ← camera.position
7      ray.direction ← UL + ((i + 0.5) * dx) + ((j + 0.5) * dy)
7      ray.direction ← NORMALIZE(ray.direction - ray.origin)
8      t ← TRACE_RAY(ray, scene)
9      if an intersection, t, was found do
10       P[i, j].color ← COMPUTE_LIGHTING(ray, t)
11     else do
12       P[i, j].color ← background_color

```

The algorithm presented is the standard ray-tracing algorithm, except for lines 3-4. These two lines record the location of each point in P in the range $[0, 1]$. This normalization is important for correctly relating the location of a point on the grid of one sampling resolution to another (**Figure 3.1**). The normalized position for a point can be converted to any given grid resolution by applying the following transformation:

$$grid(x,y,w,h) = \left(w\left(x - \frac{1}{w}\right) , \left(y - \frac{1}{h}\right) \right),$$

where w and h are the width and height of the desired grid resolution.

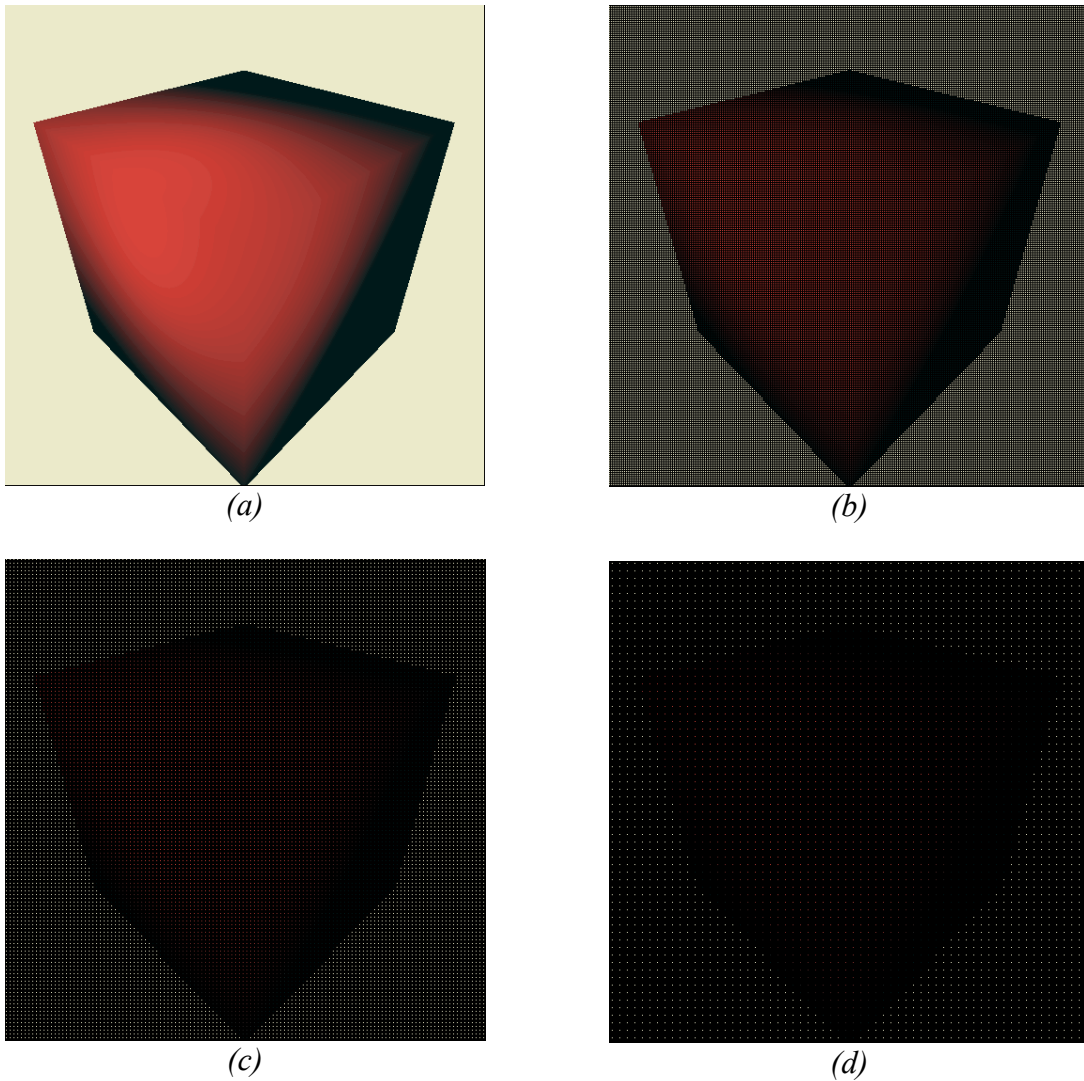


Figure 3.1 Rendering the same image at different resolutions. (a) An image of a cube rendered at a resolution of 512x512. (b) – (d) The same image rendered at 256x256, 128x128, and 64x64, respectively, with the point samples repositioned on the 512x512 sample grid for display. As the resolution lowers, the sampling grid becomes more and more sparse in its coverage of the image plane.

Edge Detection

Once a set of color point samples is generated, edgelet information must be collected from the image plane. An edgelet is defined as containing two pieces of information: (1) its normalized position within the image plane, and (2) its orientation. To determine an edge's normalized position, we first determine the three-dimensional coordinates of where a ray from the viewpoint to the edge intersects the image plain. Next, this position on the image plane is transformed into screen coordinates and normalized. The recorded edgelets were not bound to a sampling grid, thus, their precision was bound only by the floating-point representation of the computer.

Detecting and utilizing geometric edges during rendering remains an active area of research, especially in NPR [Buchanan and Sousa, 2000; McGuire and Hughes, 2004; Wang *et al.*, 2004]. While these techniques and the problem they address are indeed relevant to the research presented in this thesis, they are not the focus of our work and we sought to side-step the problem of edge detection by utilizing the simplest and most reliable edge detection technique possible.

As previously discussed, edges are often perceived at regions of discontinuity in the light intensity of an image. While these discontinuities can occur in various locations in an image, we found that silhouettes were sufficient for our purposes. A silhouette edge is an edge where one of the adjacent triangles is facing the viewer, and the other is facing away from the viewer:

$$\text{sign}(\mathbf{n}_0 \cdot \mathbf{v}) \neq \text{sign}(\mathbf{n}_1 \cdot \mathbf{v})$$

Here, \mathbf{n}_0 and \mathbf{n}_1 are the face normals of the two triangles adjacent to the edge, and \mathbf{v} is a normalized vector from the eye point to a point on the edge.

Table 3.2 Edge-sampling algorithm.

```

SAMPLE_EDGES(edges[1..n], scene)
1 num_edges ← MAX(num_samples * 0.1, n * 0.5)
2 for i ← 1 to num_edges do
3   e ← rand() % num_edges
4   edge_len ← LENGTH(edges[e].end - edges[e].begin)
5   sample_pt ← edges[e].begin +
                NORMALIZE(edges[e].end - edges[e].begin)
                edge_len * rand() / RAND_MAX
6   ray.origin ← sample_pt
7   ray.direction ← NORMALIZE(eye - sample_pt)
8   TRACE_RAY(r, scene)
9   if an intersection was found do
10    i ← i - 1
11  else do
12    screen_begin ← TO_NORMALIZED_SCREEN_SPACE(edges[e].begin)
13    screen_end ← TO_NORMALIZED_SCREEN_SPACE(edges[e].end)
14    edge_pos ← TO_NORMALIZED_SCREEN_SPACE(sample_pt)
15    RECORD EDGELET(edge_pos, NORMALIZE(screen_end - screen_begin))

```

Once the edge candidates have been identified, we randomly sample and record edgelets using the algorithm presented in

Table 3.2. The first line determines the number of edgelet samples that must be recorded. For simple scenes not containing many edge discontinuities, such as the cube in **Figure 3.1(a)**, the number of edgelets recorded is ten percent of the total number of color samples recorded. While these simple scenes do not contain many edges, their edges are arguably more important since there are so few. Therefore, we include this case to ensure that enough samples are recorded. On the other end of the spectrum, we have polygonal scenes containing many more edges than there are color samples. To ensure that these scenes are accurately represented, we record fifty percent of the edges in the scene.

Next, we loop over the list of candidate edges, randomly picking an edge and a random location along that edge to sample. We test to make sure the edge is not occluded by another object in the scene by tracing a ray from the sample point to the eye. If the point is not occluded, we convert it to normalized screen space, compute its orientation, and record the edgelet. The remainder of this thesis operates on entities located within this normalized image space characterized by the upper-left point (0, 0) and lower-right point (1, 1).

The next phase of our reconstruction framework processes the sampled edgelets to produce continuous contours and then formulate hypotheses about edge structures to facilitate reconstruction.

Contour Construction

Geisler's analysis [2001] of the co-occurrence of edges in natural images found that two edge elements were likely to be integrated into a contour if they were located near each other, were similarly oriented, and the directional angle from one element to the other was small. Furthermore, Geisler found that edge pairs could be chained together through transitive relationships. We use these findings to derive a method for unifying our sparse edges into continuous contours.

First, we address Geisler's transitive association between edges: if edge A pairs with edge B , and B pairs with edge C , then the contour chain ABC can be inferred. Under our framework this is achieved by defining an edgelet E as containing two half-spaces, H_0 and H_1 , divided by the vector N perpendicular to E 's orientation O (**Figure 3.2**). Each half-space is capable of forming a pairing with another edgelet. If N can be defined by the standard

equation for a line, $y = f(x)$, then membership in H_0 and H_1 for a given point p can be determined by

$$p \in \begin{cases} H_1, & y - f(p.x) \leq 0 \\ H_0, & \text{otherwise} \end{cases} .$$

Like Geisler, we do not define the existence of a pairing between two edgelets as a binary condition. Instead, the likelihood that two edgelets would form a pairing is characterized by their affinity, a value in the range $[0, 1]$, computed using Geisler's parameters d , θ , and ϕ (**Table 3.3**). On line 1, we test for collocated edgelets with differing slopes, which would indicate the intersection of two different contours (this is discussed in greater detail in the next section). Next, the distance weight, d , is computed using a Gaussian kernel defined as

$$G(x_1, y_1, x_2, y_2) = e^{-\frac{(x_2-x_1)^2 + (y_2-y_1)^2}{2\sigma^2}} .$$

The parameter σ^2 controls how quickly the function approaches 0 as the distance between the two points approaches ∞ . The effects of varying σ^2 will be presented in Chapter 4.

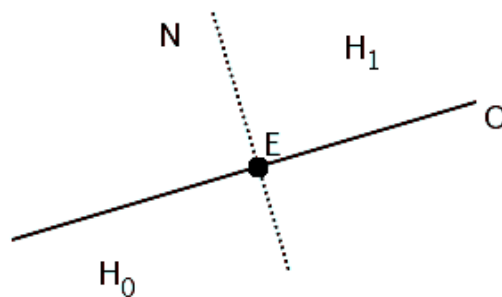


Figure 3.2 Illustration of the two half-spaces for an edgelet.

Table 3.3 Algorithm for determining the affinity between two edgelets.

```
AFFINITY (e, h, f)
1  if e.origin = f.origin and e.orientation != f.orientation
    then return 0

    /* compute d */
2  d ← GAUSSIAN(e.origin, f.origin)
3  if d < 0 then return 0

    /* compute theta */
4  costheta ← DOT(e.orientation, f.orientation)
5  if costheta < 0 then do
6      costheta ← costheta * -1
7  if costheta < cos(30°)
    then return 0

    /* compute phi */
8  toF ← NORMALIZE(f.origin - e.origin)
9  toE ← NORMALIZE(e.origin - f.origin)
10 eToH ← INTO_HALF_SPACE(h)
11 fToI ← INTO_HALF_SPACE(HALF_SPACE(f, e))
12 phi ← 1 - (DOT(toF, eToH) - DOT(toE, fToI))
13 if phi < 0 then do
14     phi ← phi * -1
15 return d * costheta * phi
```

The angle θ is computed on lines 4-7 by taking the dot product between the orientations of the two edgelets. Since the cosine is defined on $[-1,1]$ for $\theta \in [0, \pi]$ and only the orientation of the edgelet is known at this point, not its true direction, the cosine is restricted to the range $[0,1]$. Hess and Fields' early research on contour integration [1993] found that the HVS is most likely to form an illusory contour between elements whose orientations are within $\pm 30^\circ$ of each other. Therefore, if $\cos\theta < \cos(30^\circ)$, the edges' orientations are outside of the range established by Hess and Field and they are not permitted to form a pair.

The remainder of the algorithm is devoted to computing ϕ . For edgelet e , ϕ_e is the angle between the vector pointing along e 's orientation and into the half space containing f and the vector point from e to f (ϕ_f is similarly computed for f with respect to e). The final

value of ϕ is 1 minus the difference of $\phi_e - \phi_f$, resulting in higher affinities for smooth edgelets forming smooth contours without high-rates of change in the curvature of the contour. There are three important observations for our affinity algorithm:

1. It computes the affinity between edgelets with respect to the half-space containing the partner.
2. The half-spaces for an edgelet are disjoint; if $f \in H_{0e}$ and the affinity between e and f with respect to H_{0e} is non-zero, then the affinity between e and f with respect to H_{1e} will be zero.
3. Affinity is symmetric: $\text{AFFINITY}(e, f) = \text{AFFINITY}(f, e)$.

While affinity dictates the bonding strength between a pair of edgelets, it only serves as a heuristic for a larger function (**Table 3.4**) for finding the optimal pairings between all edgelets recorded during the sampling phase of reconstruction. A modified version of the stable marriage algorithm [Gale and Shapley 1962] was used to find those optimal pairings. The stable marriage algorithm guarantees that, at the conclusion of the algorithm, no two edgelets who are not paired will have a higher affinity for each other than the affinity to their current partner. Our algorithm shares this property, but differs in two ways:

1. The algorithm does not permit two edgelets with affinity equal to zero to partner. Therefore, even if the number of edgelets is even, not all edgelets will have a partner when the algorithm terminates. This results in contours being defined by a lone edgelet (creating a straight line).
2. Stable marriage forms a single partnership between elements in the algorithm. To compensate for the two half-spaces of an edgelet, our algorithm treats each edgelet as both a man and a woman (for H_0 and H_1) and will attempt to form two distinct

partnerships for each edgelet. Intuitively, it follows from this second property that, at the conclusion of the algorithm, each edgelet will either have no partners (form a lone contour), have one partner (is the end point for a contour), or have two partners (is an internal point in a contour chain).

With the optimal pairings between edgelets found, the final step in contour construction is to iterate over the pairs and create the contour chains (**Table 3.5**). Bezier curves are used to connect each pair of edgelets,

$$\mathbf{B}(t) = \mathbf{P}_0(1-t)^3 + 3\mathbf{P}_1t(1-t)^2 + 3\mathbf{P}_2t^2(1-t) + \mathbf{P}_3t^3, t \in [0,1].$$

This results in contour chains akin to a traditional Bezier spline,

$$\mathbf{C} = \begin{cases} \mathbf{B}_1(t_1), t_1 \in [0,1] \\ \mathbf{B}_2(t_2), t_2 \in [0,1] \\ \vdots \\ \mathbf{B}_n(t_n), t_n \in [0,1] \end{cases}.$$

We define each curve such that \mathbf{P}_0 is the location of the first edgelet and \mathbf{P}_3 is the location of the second edgelet. \mathbf{P}_1 is set on the line defined by the first edgelet's origin and its orientation pointing into the half space containing the second edgelet (\mathbf{P}_2 is likewise chosen with respect to the second edgelet). To prevent the curve from looping back on itself, \mathbf{P}_1 and \mathbf{P}_2 are placed a tenth of the distance between \mathbf{P}_0 and \mathbf{P}_3 along their respective tangent lines. An advantageous property of Bezier curves is that the tangent lines to the curve at $\mathbf{B}(0)$ and $\mathbf{B}(1)$ will have the slopes $|\mathbf{P}_1 - \mathbf{P}_0|$ and $|\mathbf{P}_3 - \mathbf{P}_2|$, respectively. These slopes will be the same as the edgelet orientations, but will now have an associated direction. During the creation of the curves, an arbitrary direction is assigned with one edgelet's orientation pointing into the curve and other pointing away from the curve (**Table 3.5**, lines 9 & 26). This arbitrary assignment of direction plays a heavy roll in the application of the hypotheses described in

the following section, however, for now it is important to note that Bezier curves are symmetric; the direction of the curve can be reversed and the tangential relationships to the edgelets' orientations will be preserved.

Table 3.4 Modified stable marriage algorithm for finding optimal pairings between recorded edgelets.

```

OPTIMAL_PAIRINGS(edgelets[1..n])
1  affinityTable[1..n][0..1][1..n]
2  partners[1..n][0..1]

    /* Compute affinities between edgelets */
4  for e ← 1 to n do
5    for h ← 0 to 1 do
6      partners[e][h] ← FREE
7      for f ← 1 to n do
8        if e = f then do
9          affinityTable[e][h][f] = 0
10       else do
11         affinityTable[e][h][f] ← AFFINITY(edgelets[e], h, edgelets[f])

    /* Iteratively search for optimal pairings */
12 pairingMade ← true
13 while pairingMade = true do
14   pairingMade ← false
15   for e ← 1 to n do
16     for h ← 0 to 1 do
17       if partners[e][h] = FREE
18         /* Get list of edgelets, f, with AFFINITY(e, h, f) > 0.
19          This list is sorted in descending order on affinity value */
20         while list of candidates is not empty do
21           f ← next candidate
22           i ← HALF_SPACE(edgelets[f], edgelets[e])
23           if partners[f][i] = FREE do
24             partners[e][h] ← f
25             partners[f][i] ← e
26           else if affinityTable[f][i][e] >
27             affinityTable[f][i][partners[f][i]] do
28               g ← HALF_SPACE(edgelets[partners[f][i]], edgelets[f])
29               partners[partners[f][i]][g] ← FREE
30               partners[e][h] ← f
31               partners[f][i] ← e

32   if partners[e][h] != FREE
33     pairingMade = true
34 return partners

```

Table 3.5 Contour creation algorithm.

```

CREATE_CONTOURS(edgelets[1..n], partners[1..n][0..1])
1  for e ← 1 to n do
2    if e has not been processed do
3      if partners[e][0] = FREE and partners[e][1] = FREE do
4        NEW_CONTOUR(CREATE_LONE_CONTOUR(edgelets[e], edgelets[e]))
5      else do
6        if partners[e][0] != FREE do
7          f ← partners[e][0]
8          g ← partners[f][HALF_SPACE(edgelets[e]) + 1 % 2]
9          c ← CREATE_CONTOUR(edgelets[e], edgelets[f])
10         while g != FREE and g has not been processed do
11           ADD_TO_END(c, edgelets[g])
12           f ← partners[g][HALF_SPACE(edgelets[f]) + 1 % 2]
13           SWAP(f, g)
14
15         if partners[e][1] != FREE do
16           f ← partners[e][1]
17           g ← partners[f][HALF_SPACE(edgelets[e]) + 1 % 2]
18           ADD_TO_FRONT(c, edgelets[f])
19           while g != FREE and g has not been processed do
20             ADD_TO_FRONT(c, edgelets[g])
21             f ← partners[g][HALF_SPACE(edgelets[f]) + 1 % 2]
22             SWAP(f, g)
23
24         NEW_CONTOUR(c)
25       else do
26         f ← partners[e][1]
27         g ← partners[f][HALF_SPACE(edgelets[e]) + 1 % 2]
28         c ← CREATE_CONTOUR(edgelets[f], edgelets[e])
29         while g != FREE and g has not been processed do
30           ADD_TO_FRONT(c, edgelets[g])
31           f ← partners[g][HALF_SPACE(edgelets[f]) + 1 % 2]
32           SWAP(f, g)
33
34       NEW_CONTOUR(c)

```

Another property of Bezier curves that we exploit when creating contours is two curves \mathbf{B}_0 and \mathbf{B}_1 can be chained together if the two points $\mathbf{P}_3 \in \mathbf{B}_0$ and $\mathbf{P}_0 \in \mathbf{B}_1$ are collinear. In fact, the two functions `ADD_TO_FRONT` and `ADD_TO_END` (Table 3.5, lines 11, 17, 19, & 28) set these points to be co-located. `ADD_TO_FRONT` creates a new Bezier curve where \mathbf{P}_3 is equal to \mathbf{P}_0 on the current beginning of the contour chain. Similarly, `ADD_TO_END`, creates a new Bezier curve with \mathbf{P}_3 on the current end of the contour chain being equal to \mathbf{P}_0

on the new chain segment. For both functions, \mathbf{P}_1 and \mathbf{P}_2 are computed as before, on the tangent line into the half space containing \mathbf{P}_3 and \mathbf{P}_0 , respectively.

Forming Hypotheses

The contours constructed in the previous section represent a hypothesis about the true edge structure in the image at a *global* scale. Conversely, we wish to reconstruct the image on a *local* scale while respecting *local* edge boundaries.² To achieve this end, knowledge of the contours must be propagated throughout the sample grid (**Table 3.6**).

Table 3.6 Pseudo code for propagating knowledge of contours throughout the sample grid.

<pre> 1 for each point p on the sample grid 2 for each contour c 3 cp ← CLOSEST_POINT(p, c) 4 if cp within reconstruction radius of p 5 inform p about c </pre>

This algorithm must find the closest point on a contour to a given point, which involves first finding the closest point on several cubic Bezier curves, and then choosing the closest out of that set. For any parametrically defined curve, such a point can be found by solving for t in the equation $(\mathbf{B}(t) - \mathbf{p}) \cdot \mathbf{B}'(t)$.

Unfortunately, for cubic Bezier curves this involves solving for the roots of a 5th degree polynomial, to which there is no closed form solution. To compensate, the curve is

² In this regard we are performing the same task as Bala *et al.* (2003) and Tumblin and Choudhury (2004), except where they use completely defined edge information we use hypothesized edge information derived from a sparse sampling of the real edges.

sampled 100 times over $t \in [0,1]^3$ and the closest of these samples to the point is taken as the approximate closest point.

Once the closest point is found, we make sure it is within the reconstruction radius of p , and, if so, p saves the contour. The reconstruction radius defines the area in the normalized image plain that each point on the reconstruction grid operates over when gathering information for reconstruction. If w_s is the width of the sampling grid, then this radius is set in accordance with the sampling theorem [Nyquist, 1928; Shannon, 1949] to twice the distance between two adjacent points on the sampling grid:

$$r \equiv \frac{2}{w_s}.$$
⁴

At this point in the reconstruction process, the sample grid will hold the information necessary for reconstruction: a collection of color samples on a coarse, uniform grid and a collection of continuous contours constructed from sparse edgelet samples. The first step in transferring this knowledge to the reconstruction grid in order to produce a higher resolution image involves formulating hypotheses about the contours in the image. Later, these hypotheses will dictate how to behave in the final phase of reconstruction.

³ More robust methods exist for approximating the roots of a 5th degree polynomial. The most popular of these recursively subdivide the curve until within some error threshold ϵ . The accuracy achieved with these methods is sacrificed for speed. An additional sacrifice on size efficiency is sacrificed for faster execution by pre-computing and storing the 100 curve samples upon curve creation.

⁴ For this thesis, grid resolutions were restricted to equal dimensions; however, extending this computation to include non-square resolutions is a trivial matter.

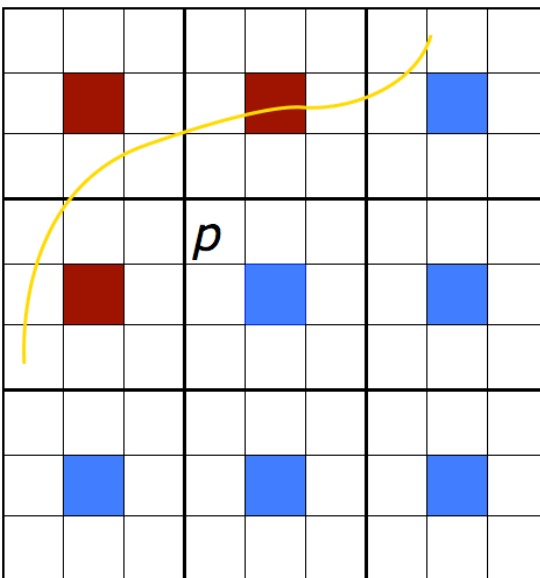


Figure 3.3 Querying the sample grid for contour information. A reconstruction grid displayed on top of a sample grid (bold lines) with three times fewer grid cells (sample colors are located in the center of each sample cell). The point p projects onto the center sample cell and queries the nine sample cells immediately surrounding it for any nearby contours – in this case the golden contour passing through the red sample cells.

Each point on the reconstruction grid projects itself onto the sample grid and queries the nine closest sample grid cells (the center cell it projects onto and the eight surrounding cells) for the two closest contours. When the two closest contours are found, the reconstruction point stores only the segments of the contour within the reconstruction radius. Intended to reduce comparisons against a cubic function during the final phase, this move involves creating a shorter contour that is a subset of the longer chain passing near the reconstruction point in the image; any further references to contours will refer to these sub-contours.

If the nine queried sample cells have no contours in their vicinity, the reconstruction point is given the default hypothesis and will be bound only by the reconstruction radius during the final phase. If a lone contour is found, the point will be given a hypothesis to reflect this contour. During the final phase, the point will only operate over the region of the

image plain within its reconstruction radius *and* on the same side of the known contour. The remaining cases involve analyzing the relationship of the two known contours. If there are two contours within the reconstruction radius, one of three hypotheses can be made:

1. If the ray originating at the end point of one contour and extending in the direction of the tangent line for that point on the curve bisects an internal curve of the second contour, a T-junction hypothesis is formed (**Figure 3.4 (a)**). Under this hypothesis, there are two possible behaviors during the final phase. First, if the reconstruction point is on the opposite side of the bisected contour from the bisecting contour, it will behave as a lone contour hypothesis. Otherwise, it requires that all gathered points be within the reconstruction radius and be on the same side of *both* contours as the reconstructed point.
2. If the two contours do not form a T-junction, a join test is performed. Two contours are joined together if the end-rays intersect each other (**Figure 3.4 (b)**) indicating the presence of a corner, or if the end-rays lie on the same line (**Figure 3.4 (c)**). While conceptually different, the two cases behave similarly in that they both restrict the reconstruction area to the same side of the closest contour.
3. If neither of the two cases above hold, then the two contours are covered by a default hypothesis. This hypothesis is the most restrictive and reduces the reconstruction area to being on the same side of both contours as the reconstruction target. This situation is illustrated in **Figure 3.4 (d)** and **(e)**.

Figure 3.5 demonstrates illustrates the hypothesis formation process followed by a discussion of how to consistent identify the two sides of a contour.

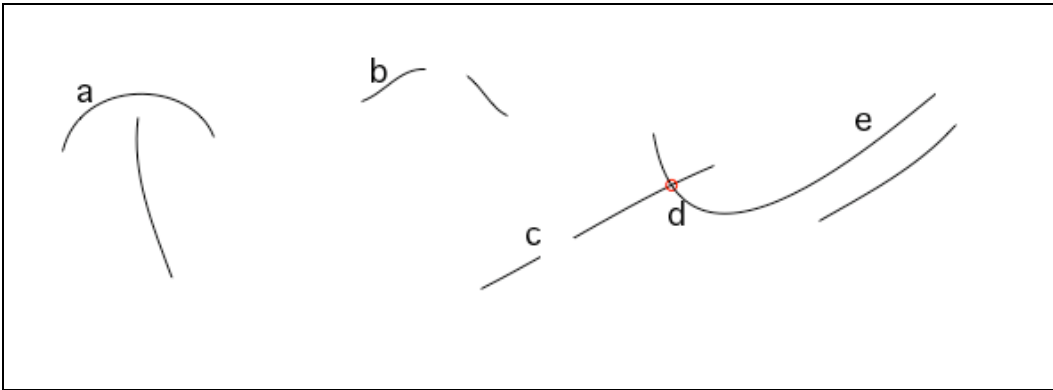


Figure 3.4 Examples of contour hypotheses. (a) A bisection of one contour by another's end-ray indicates the presence of a T-junction. (b) Where two end-rays intersect each other a corner is predicted. (c) The two end-rays do not intersect because they are actually on the same line; therefore the two are treated as an intersection located at the mid point of the end-rays. (d) & (e) The default cases when in the presence of two contours. In (d), the two contours intersect on internal curve segments. In (e), the two contours never intersect, bisect, or form a corner. Therefore any construction performed in the vicinity must respect the boundaries of both contours.

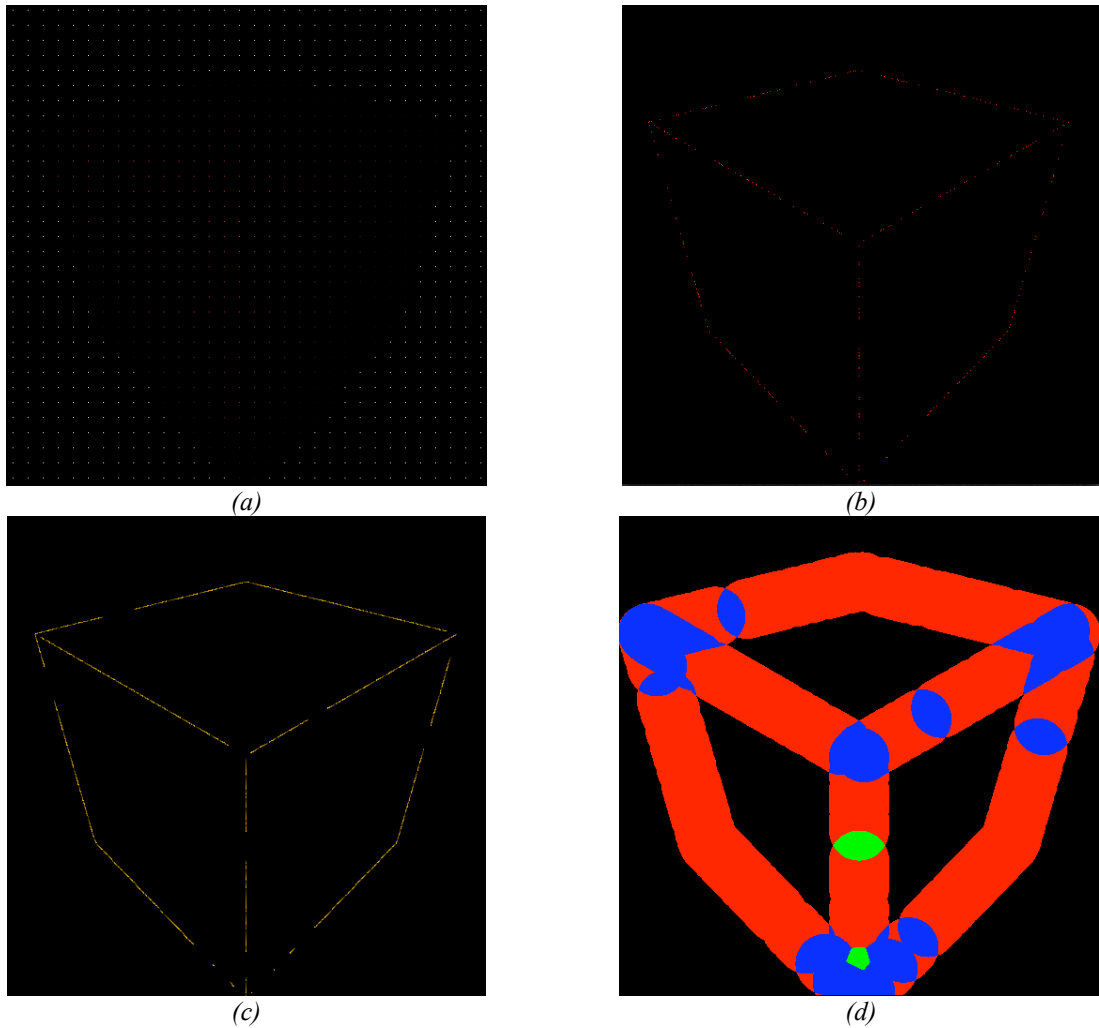


Figure 3.5 Visualization of the first three phases of reconstruction. (a) A 32x32 sample grid displayed on top of a 512x512 reconstruction grid. (b) The recorded edgelets. Thirty percent more edgelets were recorded for the purposes of this visualization. (c) The constructed contours. Each curve in a contour is visualized with the luminosity of the yellow going from dark to bright in the direction of the curve. (d) The 512x512 reconstruction grid with a visualization of the various hypotheses made. Each point in the grid is color-coded to indicate the type of hypothesis. Black: no contours nearby. Red: near a single contour. Blue: two contours that join together either at a corner or in a straight line. Green: two contours that either intersect or do not join or intersect.

Bala *et al.* [2003] and Tumblin and Choudhury [2004] have established the effectiveness of respecting edge boundaries during reconstruction. Their methods, however, simplify edges to straight-line segments encoded at sub-pixel resolutions. As a result, respecting edge boundaries is a matter of identifying cells containing an edge and performing

a standard line test, $0 = y - f(x)$. Although these line tests are sufficient when operating on line segments defined within discrete boundaries, they prove to be difficult to use with parametric curves. Instead, we exploit the first derivative of a Bezier curve to determine sides with respect to a curve. If a cubic Bezier curve is defined by

$$\mathbf{B}(t) = \mathbf{a}t^3 + \mathbf{b}t^2 + \mathbf{c}t + \mathbf{P}_0,$$

where

$$\mathbf{c} = 3(\mathbf{P}_1 - \mathbf{P}_0)$$

$$\mathbf{b} = 3(\mathbf{P}_2 - \mathbf{P}_1) - \mathbf{c}$$

$$\mathbf{a} = \mathbf{P}_3 - \mathbf{P}_0 - \mathbf{c} - \mathbf{b}$$

then its first derivative is

$$\mathbf{B}'(t) = 3\mathbf{a}t^2 + 2\mathbf{b}t + \mathbf{c}.$$

To determine if an arbitrary point \mathbf{P} is “above” or “below” a curve, we first find the closest point to \mathbf{P} on the curve and use the t value at this point to find the associated tangent, $|\mathbf{B}'(t)|$.

This method identifies the appropriate line with which to perform the test, but the standard line fails for curves where the tangent line has an infinite slope as there is no way to label the sides of the line that would be consistent with the rest of the contour. Consider **Figure 3.6** in which a contour ending with a tangent of infinite slope is shown. The two sides of the contour have been designated by “+” and “-” symbols. The sides of the infinite slope have been labeled consistently with the remainder of the contour. Note that as the contour continues (gray dashes), the consistency in side labels is broken. If the contour turns towards the red region all is well, but as if it curves towards the blue, the sides reverse.

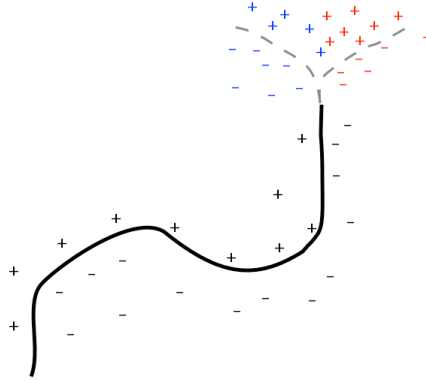


Figure 3.6 The trouble with infinite slopes.

This difficulty is remedied by conducting an angular test as opposed to a line test: given a parametric line defined by $\mathbf{P}(t) = \mathbf{P}_0 + \mathbf{P}_1 t$ where \mathbf{P}_1 is a normalized direction, then an arbitrary point \mathbf{Q} 's side is determined by

$$side = \begin{cases} above, & \tan^{-1}\left(\frac{\mathbf{P}_1 \cdot y}{\mathbf{P}_1 \cdot x}\right) - \tan^{-1}\left(\frac{|\mathbf{Q} - \mathbf{P}_0| \cdot y}{|\mathbf{Q} - \mathbf{P}_0| \cdot x}\right) > \pi \\ below, & otherwise \end{cases}$$

With this definition, the side of a contour containing a given point can be consistently identified by first finding the closest point on the contour to the point (this includes the t value for this point and the curve-segment containing the point). Next, the t value is used to find the tangent of the curve. Finally, the side of the point is determined using the definition above.

Reconstruction

The final phase of reconstruction applies the hypotheses from the previous phase to determine the color of each point on the reconstruction grid. As noted by Dayal *et al.* [2005], reconstruction can be one of two processes. It can be a *gather* process, where each

reconstruction point searches the sample grid for information, or it can be treated as a *scatter* process, with each sample point projecting onto the reconstruction grid and evaluating its contribution to all of the points within some radius. As **Table 3.7** shows, we have implemented reconstruction as the former of the two.

This algorithm loops over all of the points on the reconstruction grid. For each such point, it loops over the points on the sample grid to find the points within its gather radius. For each sample point found, it applies the hypothesis formulated in the previous phase to see if the sample point is valid. If so, it computes the contribution of the sample to the final color of the reconstructed point. While any center-weighted filter function can be used to determine the contribution of a sample, we have chosen to use the Gaussian kernel first introduced during the contour construction phase. As before, this kernel returns a value in the range [0,1] based on the Euclidean distance between the two points. For reconstruction, the parameter σ^2 is set to have the weight fall to 0 beyond the reconstruction radius:

$$\sigma^2 = \frac{-\left(\frac{5}{2w_s}\right)^2}{2\ln(\varepsilon)}$$

Where w_s is the width of the sample grid and ε is near 0.

It was observed that on rare occasions the formulated hypothesis would be too restrictive and reject all samples within the radius of a reconstruction point. For these points, the hypothesis is rejected and a second pass is made to reconstruct the point without a hypothesis. Finally, the reconstructed colors are normalized by the sum of the weights from each sample contributing to its color.

Table 3.7 Reconstruction algorithm.

```
RECONSTRUCT(sample_grid, recon_grid)
1 radius ← [2 * recon_grid.width / sample_grid.width]
2 for each point on recon_grid r
3   for each point on sample_grid s
4     if s.position within radius of r.position do
5       if TEST_HYPOTHESIS(r, s) do
6         weight ← GAUSSIAN(r, s)
7         r.color ← r.color + s.color * weight
8         r.summedWeight ← r.summedWeight + weight
9   if r.summedWeight = 0 do
10    for each point on sample_grid s
11      if s.position within radius of r.position do
12        weight ← GAUSSIAN(r, s)
13        r.color ← r.color + s.color * weight
14        r.summedWeight ← r.summedWeight + weight
15  r.color ← r.color / r.summedWeight
```

Chapter IV: Implementation and Results

In this section we present results from our reconstruction framework. Unless stated otherwise, all reconstructed images are 512x512. The framework was ran on a 2.16 GHz Intel Core 2 Duo with 2GB SDRAM. We begin with a presentation of images reconstructed from a set of 32x32 color samples, yielding 1 color sample for every 256 reconstructed points. **Table 4.1** lists detailed information for each rendered scene.

Table 4.1 Statistics for rendered scenes.

<i>Scene</i>	<i>Triangles</i>	<i>Silhouettes</i>	<i>Edgelets</i>	<i>Contours</i>
Cube	12	6	108	15
Sphere	512	32	108	9
Teapot	1992	297	155	23
Stanford Bunny ¹	16301	1199	608	46

The cube (**Figure 4.1**) and sphere (**Figure 4.2**) are simple geometric objects containing a significantly fewer silhouettes than the number of color samples in the image. As discussed in Chapter 3, basing the number of edgelet samples to take on this small set would result in only a few edgelets to describe the shape of the object; if we were to sample 50% of the silhouettes in view, we would have a paltry 3 edgelets for the cube and 16 for the sphere. Instead, we found that it was more appropriate to make the number of edgelets proportional to the number of color samples. More specifically, if n is the number of color samples (in these images $n = 32^2$), then the number of edgelets recorded e is $e = n/10$. The teapot (**Figure 4.3**) and bunny (**Figure 4.4**) on the other hand are considered geometry

¹ Stanford University 3D Scanning Repository, <http://graphics.stanford.edu/data/3Dscanrep>
Due to our un-optimized ray-tracer, a simplified version of the model was used to reduce render times.

dominant objects. While there may still be fewer silhouettes than color samples (as is the case with the teapot), the objects are complex and cannot be adequately described by the few edgelets recorded for the cube and sphere. For these objects, the number of edgelets recorded was determined by the equation $e = s/2 + s\%10$, where s is the number of silhouettes in view and $\%$ is the mod operator. The $s\%10$ term was added after empirical observations that for some random samplings of the silhouette edges, $s/2$ was just shy of enough edges to describe the silhouette boundary.

Interestingly, none of the reconstructed images made hypotheses for T-junctions or intersecting contours. An explanation is found in the reliance on silhouette edges. While internal silhouettes are possible, as seen in the teapot and bunny, these edges primarily describe the outer most boundary of an object. Intuitively, these boundary contours should not intersect. Furthermore, T-junctions would be expected to occur around occlusions. Since T-junctions were not predicted on occluded regions of the teapot (where the handle is occluded by the body) or the bunny (around the front legs and ears), however, it is suspected that a bug still lingers within our framework.

With respect to the reconstructed images, our framework performs best in regions of relative uniformity and continuous edge information. When edge information is not present, as on the upper-left corner of the cube, reconstruction “leaks” are evident. While many of these artifacts could be eliminated by either increasing the number of edgelets recorded or by increasing the parameter d during contour construction, the random nature of edge sampling prevents any guarantee on the prevention of leaks occurring.

Another cause for leaks that is a concern are the application of the joined-contour hypotheses. For these hypotheses we opted for simplicity over correctness and assumed that

contours that intersect at a corner or line will always be equidistant from their intersection points. This assumption proves fatal when one contour is closer to the intersection point than the other (as evident in the corner of the cube at the middle of the bottom edge in **Figure 4.1 (f)**). Similar leaks are also present along the contours of the sphere (**Figure 4.2 (f)**) where the background color appears to be spraying into the interior the cube.

Figures 4.1 – 4.4 exhibit leaks near edges in the reconstructed image that could be attributed to one of two causes, either a gap in the available edge information, or malformed hypothesis. In the case of gaps in edge information, two remedies are available. First, the number of edgelets recorded can be increased. Ideally, more edge information available would result in more detailed and accurate contours, resulting in better reconstructed imagery. Alternatively, the maximum area permitted by the Gaussian kernel used to determine d during contour construction can be increased. By increasing the maximum permissible distance between edgelets in a contour, the system will make more aggressive pairings, resulting in longer contours at the expense of accuracy. The trade-off between these approaches is illustrated in **Figure 4.5**.

The number of edgelets recorded, as a percentage of total silhouettes in the scene, is increased in each image from top-to-bottom. The images in the left column have set the maximum permissible distance to the same Gaussian kernel used in reconstruction – allowing edgelets to be at most one reconstruction radius away before d falls to 0. In the right column, the spread of the Gaussian kernel has been doubled allowing edgelet pairs to be separated by two reconstruction radii:

$$G(x_1, y_1, x_2, y_2) = e^{-\frac{(x_2-x_1)^2 + (y_2-y_1)^2}{2\sigma^2}}, \quad \sigma^2 = \frac{\left(\frac{5}{w_s}\right)^2}{2\ln(\epsilon)}.$$

Surprisingly, these images show that our framework offers the best performance when the number of edgelets is low and d is kept small. As additional edgelets are recorded, the image gets more complex and the contour construction algorithm has a more difficult time forming consistent contours. On the other hand, as d is increased the contour construction becomes more aggressive, pairing edgelets further away. In these images, the degradation in quality results from complexity added to the hypothesis formation phase.

We conclude with a discussion of the effects of varying sampling resolutions. In **Figure 4.6** the top row of images are all reconstructed at a resolution of 256x256 while the bottom row is reconstructed at 512x512. From left-to-right the number of color samples generated for each image is 16x16, 32x32, and 64x64. The same seed was used for randomly sampling the edges in all six images. The same reconstruction artifacts observed in the 256x256 reconstructions are also present in the 512x512 images, suggesting that the original sampling resolution determines the final performance of reconstruction. This is further evidenced by the decreased emphasis of hypothesis errors as the sampling resolution is increased. As the sampling resolution increases, the threshold for d , which is held proportional to the sampling frequency, is reduced. Therefore, while all of the images use the same set of sampled edgelets, increasing the number of color samples leads to more conservative contour estimations and fewer leakages.

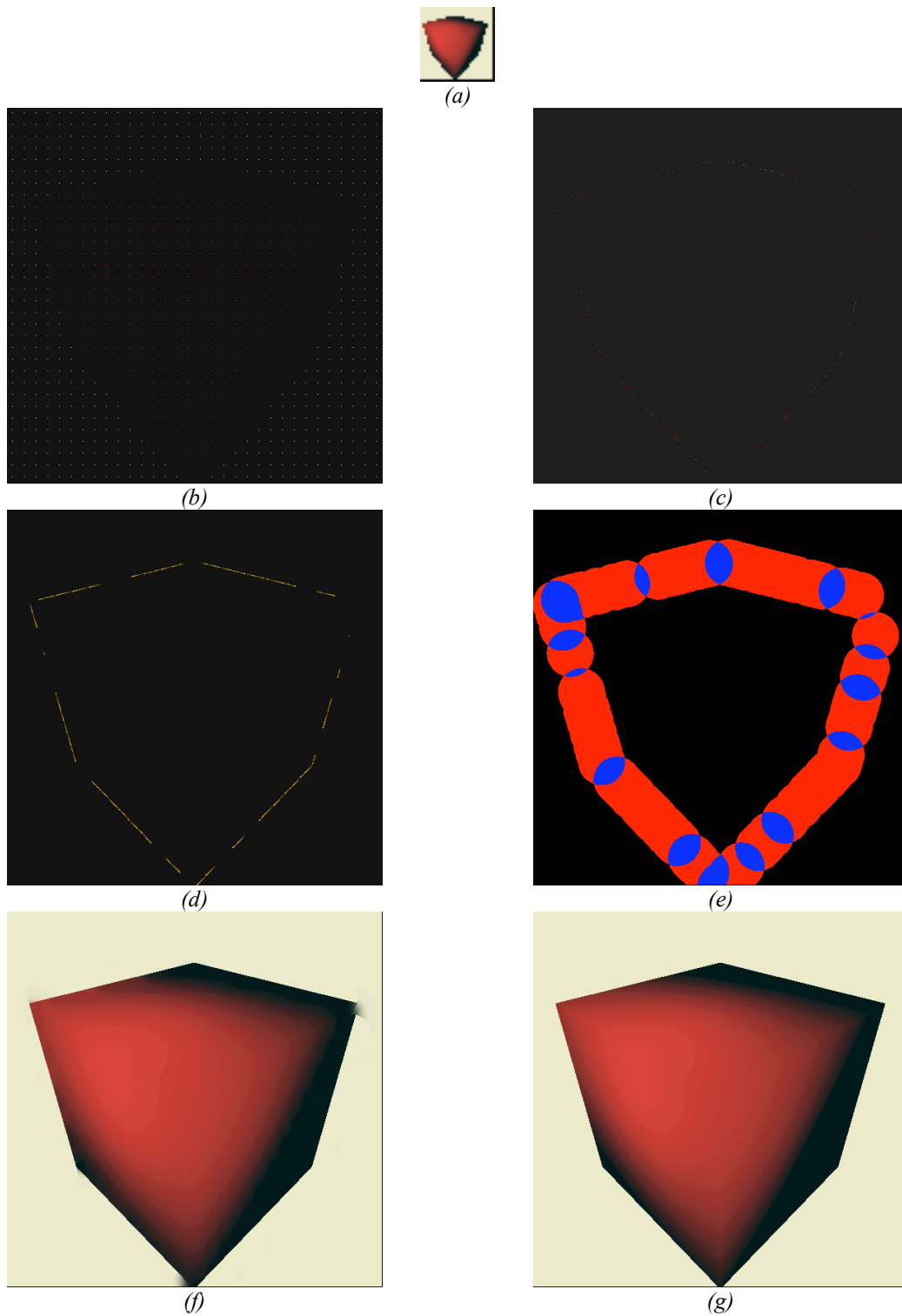


Figure 4.1 A reconstructed cube. (a) The initial color samples at native resolution. (b) The color samples displayed on the reconstruction grid. (c) The recorded edgelets displayed on the reconstruction grid. (d) The constructed contours. (e) Hypotheses formulated based on the constructed contours in (d). The reconstructed image (f) displayed next to an ideal rendering (g).

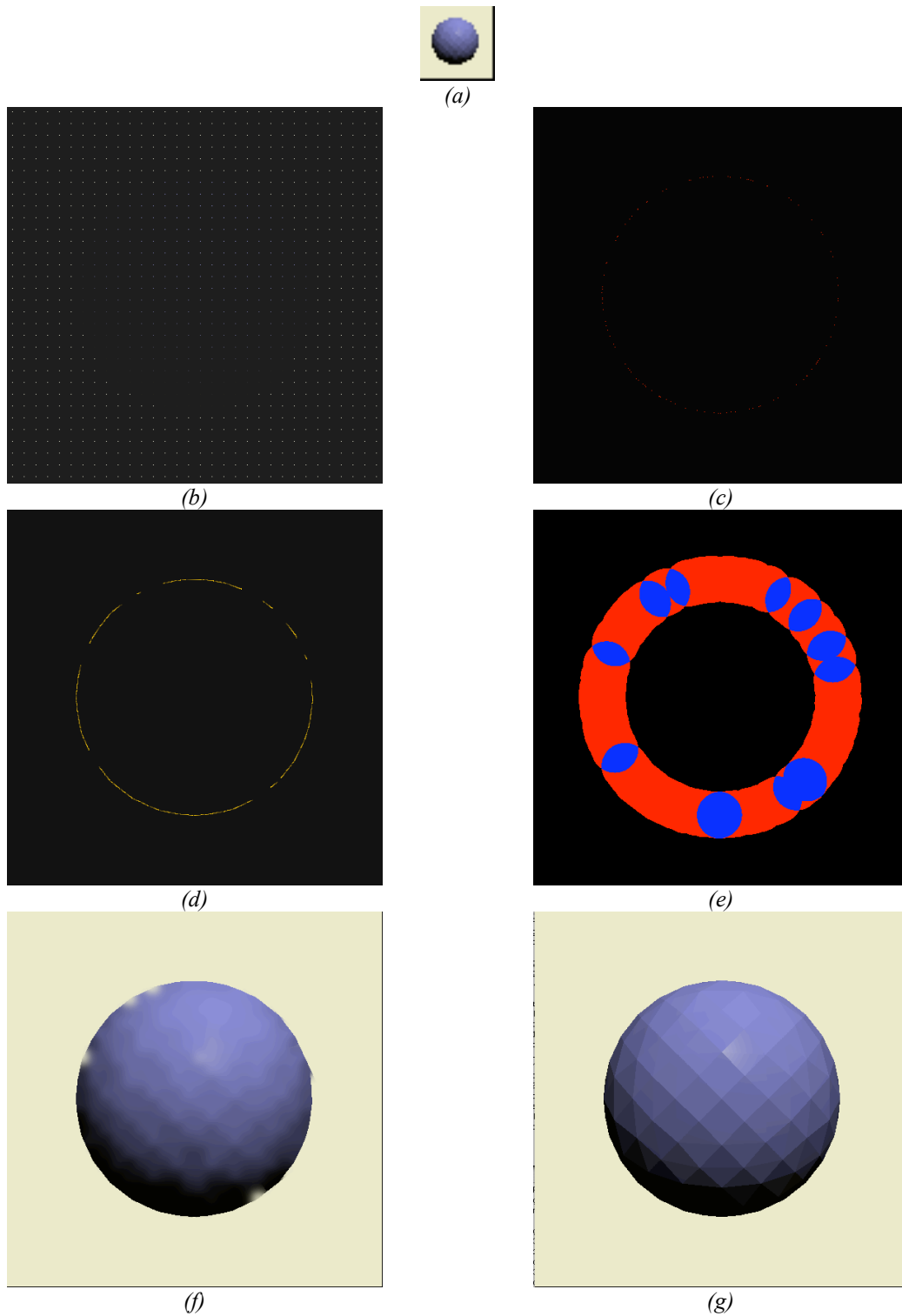


Figure 4.2 A reconstructed sphere. (a) The initial color samples at native resolution. (b) The color samples displayed on the reconstruction grid. (c) The recorded edgelets displayed on the reconstruction grid. (d) The constructed contours. (e) Hypotheses formulated based on the constructed contours in (d). The reconstructed image (f) displayed next to an ideal rendering (g).

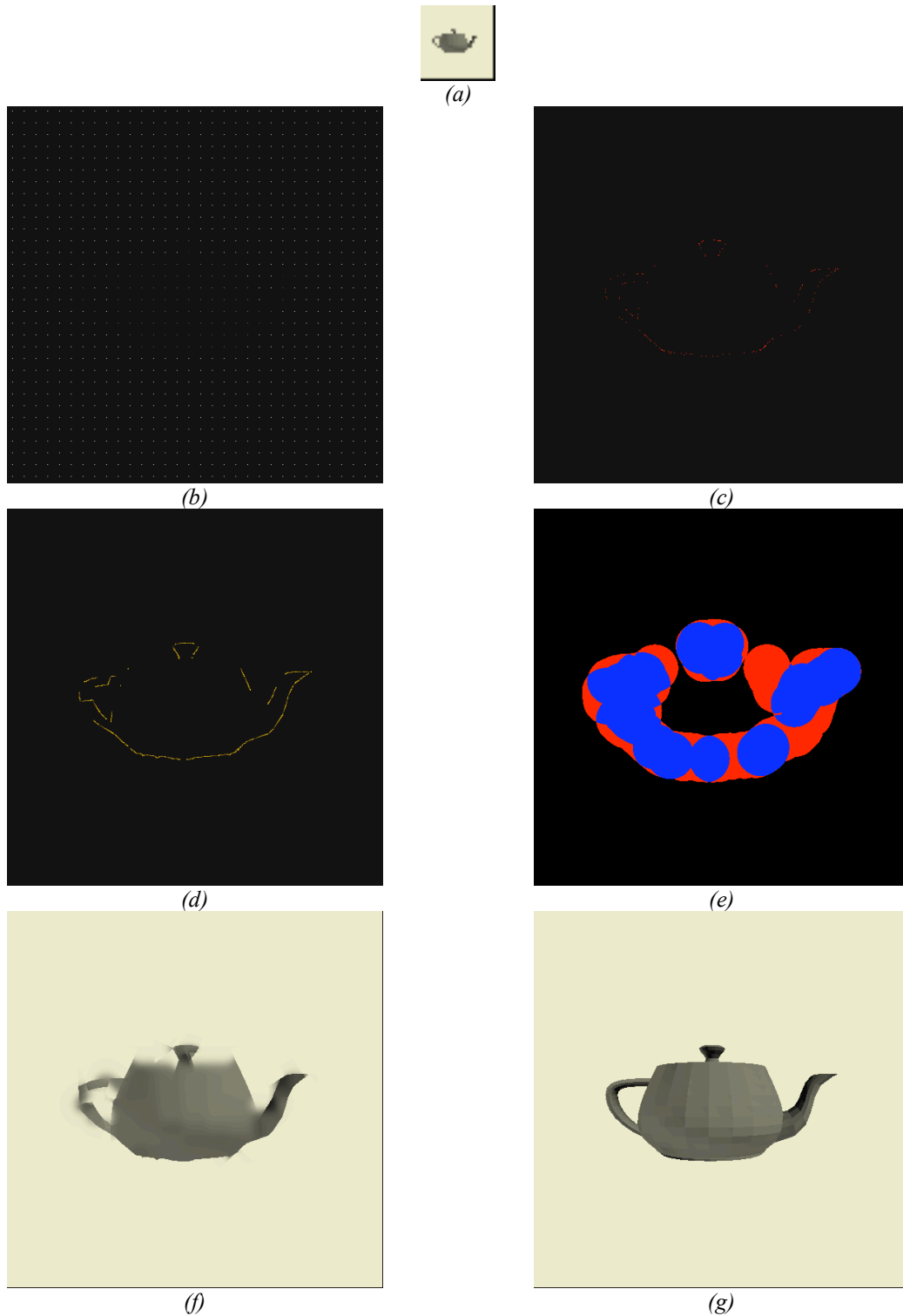


Figure 4.3 A reconstructed teapot. (a) The initial color samples at native resolution. (b) The color samples displayed on the reconstruction grid. (c) The recorded edgelets displayed on the reconstruction grid. (d) The constructed contours. (e) Hypotheses formulated based on the constructed contours in (d). The reconstructed image (f) displayed next to an ideal rendering (g).

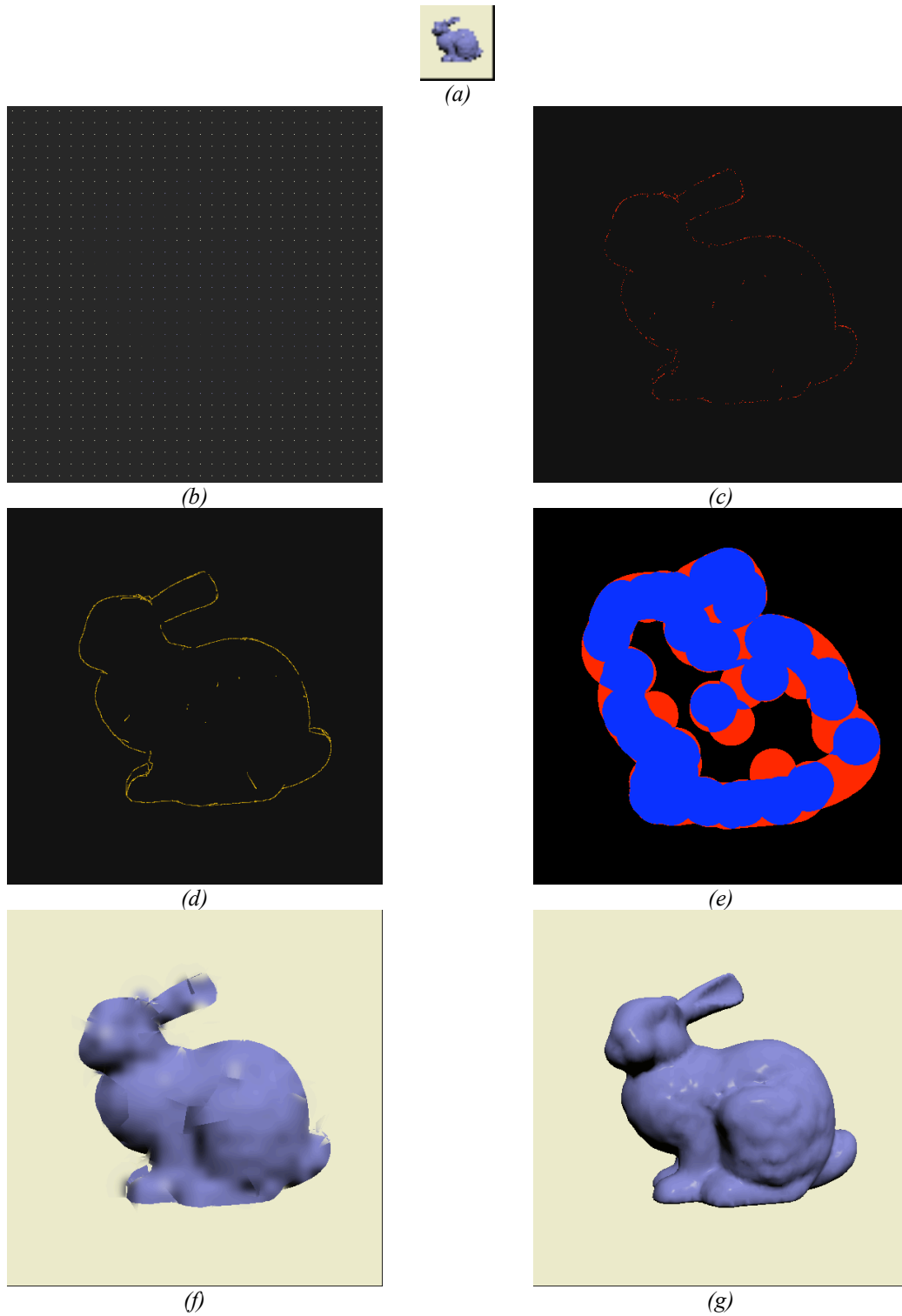


Figure 4.4 The reconstructed Stanford Bunny. (a) The initial color samples at native resolution. (b) The color samples displayed on the reconstruction grid. (c) The recorded edgelets displayed on the reconstruction grid. (d) The constructed contours. (e) Hypotheses formulated based on the constructed contours in (d). The reconstructed image (f) displayed next to an ideal rendering (g).

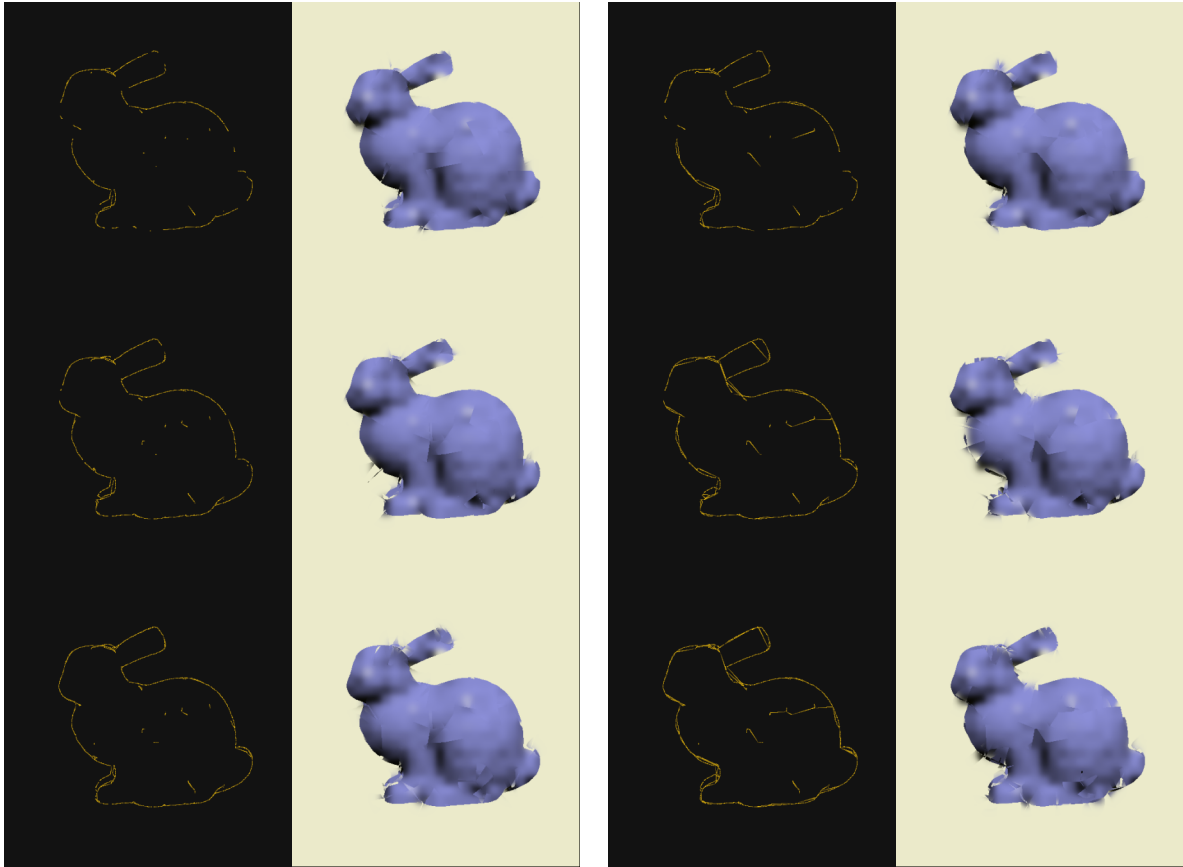


Figure 4.5 The effect of d and the number of edgelets on contour creation. From top-to-bottom the number of edgelets recorded is 25%, 50%, and 100% of the total number of silhouettes in view. In the left column the same Gaussian kernel used during reconstruction determines d . In the right column a Gaussian kernel with twice the radius of the one used in reconstruction determines d .

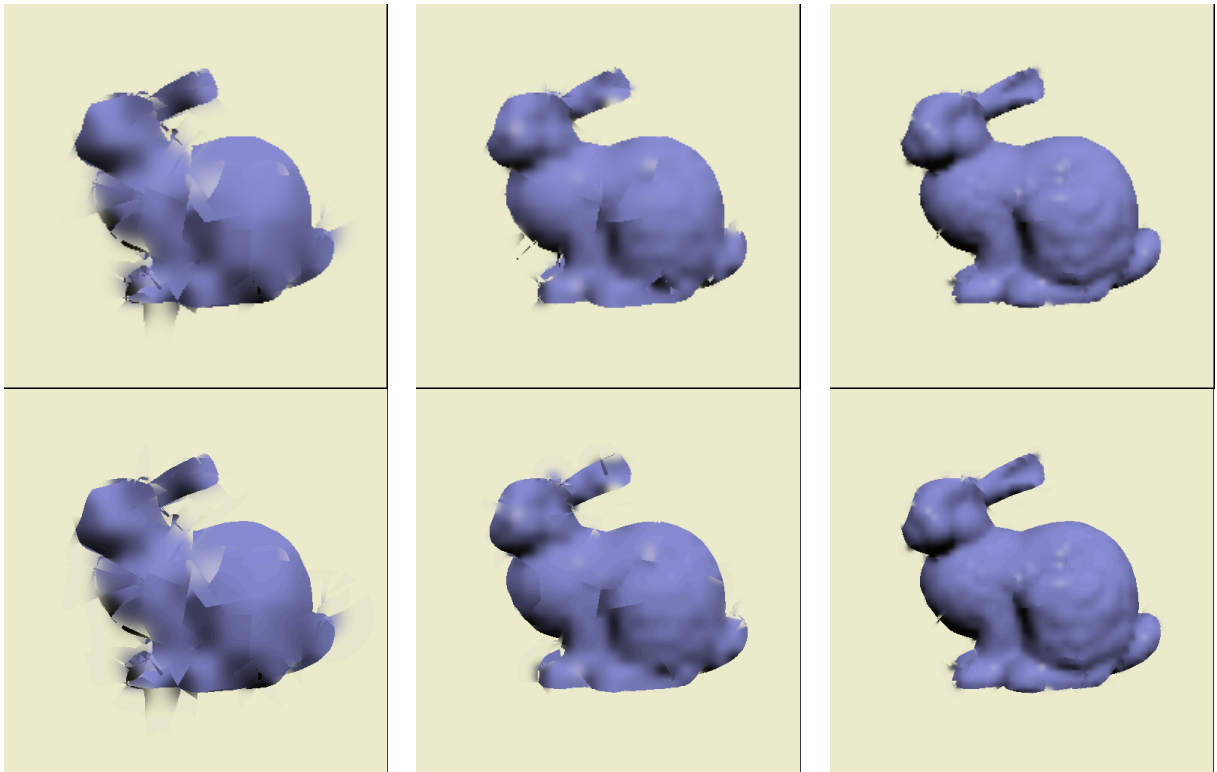


Figure 4.6 The effect of sampling resolution on reconstruction. From left-to-right, images sampled at resolutions of 16×16 , 32×32 , and 64×64 . Images in the top row were reconstructed at 256×256 while the bottom row was reconstructed at 512×512 .

Conclusion

Results thus far have been encouraging, but we believe there are many paths that can be taken to improve our reconstruction framework. The framework should be tested against more complex scenes, especially scenes with lots of occluding edges. This will verify the behavior of T-junctions, and may even identify unexpected flaws in our design.

Along similar lines, by permitting leaks near corners, the joined-contour hypothesis has proven to be an unacceptable generalization of line intersections and corners. These two cases should be separated into distinct hypotheses by adding a new hypothesis designed to specifically handle the corner case. We also foresee the necessity to develop a 3-way intersection hypothesis. While uncommon in natural images, such corners are highly prevalent on man made objects (*i.e.* the corner of a box or desk).

Additional possible refinements exist in how we detect and handle edges. Currently, we are only using the silhouettes of an object. This should be extended to incorporate additional geometric discontinuities, such as crease edges in geometry and edges caused by projected shadow volumes. Alternatively, the current approach of globally sampling edges could be abandoned for a more adaptive technique. One such possibility could involve applying an image-based edge detection algorithm to the coarse sample grid to identify regions of the image that should be sampled more finely. This process could iterate until an edge was detected.

In the long term, there are two large deficiencies that must be addressed. In its current state, the framework is only capable of recognizing edges related to the polygonal geometry. Edges contained within high frequency textures (hair, fur, marble, etc.) or edges

on parametrically defined surfaces would be undetectable. The ability to handle these cases may exist in an alternative rendering paradigm such as point rendering or even in existing image synthesis research.

Ultimately, the primary challenge for future work is making the transition to an interactive system. Depending on the initial sampling resolution or the complexity of the constructed contours (such as those in the Stanford bunny), a single reconstruction can take up to two hours to complete. For this thesis we intentionally focused on correctness over speed, resulting in the long execution times, but this does not mean that vast improvements to efficiency do not exist. To achieve interactive rates, the programmable nature of graphics hardware should be exploited. Most notably, the OpenGL library could be combined with programmable shaders to handle each phase. The challenge in designing such a system would lie in efficiently storing and passing information between the different phases of reconstruction.

The framework presented in this thesis decomposed image reconstruction into four major phases. In the first phase, the image was sampled using a coarse, uniform grid. Once color samples were collected, the sampling phase exploits a priori knowledge of the scene geometry to detect and record *edgelets*, point samples containing the location of a geometric edge in the image as well as its orientation. In the second phase, a function inspired by research on the HVS was applied to unify the sparse and disjoint edgelets into continuous contours, akin to the illusory contours humans perceive. In the third phase, the continuous contours are used to formulate hypotheses about the edge structure in the image. Finally, the framework loops over the points in the reconstruction grid and applies a Gaussian filter to

determine the contribution of each color sample while respecting the edge boundaries hypothesized in the third phase.

Although more refinements are necessary for ideal reconstructions, results indicate that combining the sparse edge information with sparse color samples is expressive enough to reconstruct images with crisp edge boundaries, even under conditions of extreme under sampling. These results are an important first step in developing a technique for generating succinct descriptions of hyper-resolution imagery without taxing the limited bandwidth of the target display.

References

- BALA, K., WALTER, B. and GREENBERG, D. 2003. Combining edges and points for interactive high-quality rendering. In *Proceedings of ACM SIGGRAPH 2003*.
- BALA, K., VELÁZQUEZ-ARMENDÁRIZ, E., LEE, E., and WALTER, B. 2006. Implementing the render cache and edge-and-point image on graphics hardware. In *Proceedings of the 2006 Conference on Graphics Interface*, 211 – 217.
- BEARD and AHUMADA. 1998. A technique to extract relevant image features for visual tasks. In *Proc. Human Vision and Electronic Imaging III*, SPIE 3299, 79 – 85.
- BEUSMANS, HOFFMAN, and BENNETT. 1987. Description of solid shape and its inference from occluding contours. , *J. Opt. Soc. Am. A*, 4, 7.
- BIEDERMAN and COOPER. 1991. Priming contour-deleted images: evidence for intermediate representations in visual object recognition. *Cognitive Psychology*, 23, 393 – 419.
- BISHOP, G., FUCHS, H., MCMILLAN, L., and SCHER ZAGIER E. J. 1994. Frameless rendering: Double buffering considered harmful. *Computer Graphics*, 28, 175 – 176.
- BUCHANAN, J. W. and SOUSA, M. C. 2000. The edge buffer: a data structure for easy silhouette rendering. In *Proceedings of the 1st International Symposium on Non-photorealistic Animation and Rendering*, ACM SIGGRAPH, 39 – 42.
- CANNY, J. 1986. A computational approach to edge detection. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8, 6, 679 – 698.
- DAS, A. and GILBERT, C.D. 1999. Topography of contextual modulations mediated by short-range interactions in primary visual cortex. *Nature*, 399, 655–661.
- DAYAL, C. WOOLLEY, B.A. WATSON, B., and LUEBKE, D. 2005. Adaptive frameless rendering. In *Proc. Eurographics Symposium on Rendering*, 265 – 275.
- DECARLO, D., FINKELSTEIN, A., RUSINKIEWICZ, S., and SANTELLA, A. 2003. Suggestive contours for conveying shape. *ACM Transactions on Graphics, Special Issue: Proceedings of ACM SIGGRAPH 2003*, 22, 3, 848 – 855.
- EDELMAN, S., and WEISS, Y. 1995. Vision, hyperacuity. *The Handbook of Brain Theory and Neural Networks*, M. A. Arbib, ed., 1009 – 1011, MIT Press.
- ELDER, J. 1999. Are edges incomplete? *International Journal on Computer Vision*, 34, 2/3, 97 – 122.

- FIELD D. AND HAYES A. 2004. Contour integration and the lateral connections of V1 neurons. *The Visual Neurosciences*, Eds. L. Chalupa and J. Werner, MIT Press.
- GALE, D. AND SHAPLEY, L. S. 1962. College Admissions and the Stability of Marriage. *American Mathematical Monthly*, 69, 9 – 14.
- GEISLER, W., PERRY, J., SUPER, B. and GALLOGLY, D. 2001. Edge co-occurrence in natural images predicts contour grouping performance. *Vision Research*, 41, 711-724.
- GIBSON, J. J. 1966. *The Senses Considered as Perceptual Systems*, Houghton Mifflin, Boston.
- GROSSMAN, J. P. and DALLY, W. J., 1998. Point sample rendering. In *Rendering Techniques '98*, Eurographics, 181 – 182.
- GUO, B. 1998. Progressive radiance evaluation using directional coherence maps. In *Proc. ACM SIGGRAPH '98*, 255 – 266.
- HANSEN AND NEUMANN. 2004. Neural mechanisms for the robust representation of junctions. *Neural Computation*, 16, 1013 – 1037.
- HEATH, SARKAR, SANOCKI and BOWYER. 1997. Robust visual method for assessing the relative performance of edge-detection algorithms. *IEEE PAMI*, 19, 12, 1338 – 1359.
- HEITGER, F., VON DER HEYDT, R., and KÜBLER, O. 1994. A computational model of neural contour processing: figure-ground segregation and illusory contours. *Conference on Perception of Action*, 181 – 191.
- HESS, R. F., FIELD, D. J., and HAYES, A. 1993. Contour integration by the human visual system: Evidence for a local “association field.” *Vision Research*, 33, 173 – 193.
- HESS, R. F., and FIELD, D. J. 1995. Contour integration across depth. *Vision Research*, 35, 12, 1699 – 1711.
- HESS, R. F., FIELD, D. J., and HAYES, A. 2000. The roles of polarity and symmetry in the perceptual grouping of contour fragments. *Spatial Vision*, 13, 1, 51 – 66.
- HESS, HAYES and FIELD. 2003. Contour integration and cortical processing. *J. Physiology Paris*, 97, 105-119.
- HUBEL, D. and WIESEL, T. 1968. Receptive fields and functional architecture of monkey striate cortex. *J. Physiol.*, 195, 215–243.
- HOFFMAN, D. 1998. *Visual Intelligence*. Norton: New York.

- KALAIAH, A., and VARSHNEY, A., 2001. Differential Point Rendering. In *Rendering Techniques '01*, S. J. Gortler and K. Myszkowski, Springer-Verlag, 2001, 139 – 150.
- KALAIAH, A., and VARSHNEY, A., 2003. Statistical Point Geometry. In *Eurographics Symposium on Geometry Processing*, 113 – 122.
- KALAIAH, A., and VARSHNEY, A., 2005. “Statistical Geometry Representation for Efficient Transmission and Rendering,” *ACM Transactions on Graphics* 24, 2, 348 – 373.
- LAW, ITOH and SEKI. 1996. Image filtering, edge detection and edge tracing using fuzzy reasoning. *IEEE PAMI*, 18, 5, 481 – 491.
- LEVI, D., KLEIN, S. and CARNEY, T. 2000. Unmasking the mechanisms for vernier acuity: evidence for a template model for vernier acuity. *Vision Research*, 40, 951 – 972.
- LEVOY, M. and WHITTED, T., 1985. The use of points as a display primitive. *TR 85-022, Computer Science Department, University of North Carolina at Chapel Hill*.
- LUEBKE, D., 2004. General purpose computation on the GPU. *ACM SIGGRAPH Course*.
- MARR, D. 1982. *Vision*. Freeman, New York.
- MCGUIRE, M. and HUGHES, J. F., 2004. Hardware-determined feature edges. In *Proceedings of the 3rd International Symposium on Non-photorealistic Animation and Rendering*, ACM SIGGRAPH, 35 – 44.
- MITCHELL, D. P. 1987. Generating antialiased images at low sampling densities. *Computer Graphics*, 21, 65 – 72.
- MORGAN, M. 1990. Hyperacuity. In *Spatial Vision*. Edited by D. Regan. London: Macmillan; 87 – 113.
- MORRONE, M. C., and BURR, D. C. 1988. Feature detection in human vision: a phase-dependent energy model. In *Proceedings of the Royal Society*, London Series B, 235, 221 – 245.
- MURRAY, M. M., IMBER, M. L., JAVITT, D. C., and FOXE, J. J. 2006. Boundary completion is automatic and dissociable from shape discrimination. *Journal of Neuroscience*, 26, 46, 12043 – 12054.
- NYQUIST, H. 1928. Certain topics in telegraph transmission theory. In *Trans. AIEE*, 47, 617 – 644. Reprinted in *Proc. IEEE*, 90, 2, 2002, 280 – 305.
- PARK, D. J., NAM, K. M., and PARK, R. 1996. Multiresolution edge detection techniques. *Pattern Recognition*, 28, 2, 211 – 229.

- PERONA, P. and MALIK, J. 1990. Detecting and localizing edges composed of steps, peaks and roofs. *UCB Technical Report*, UCB/CSD90/590.
- RINGACH, D. and SHAPLEY, R. M., 1996. Spatial and temporal properties of illusory contours and amodal boundary completion. *Visualization Research*, 36, 3037 – 3050.
- ROSENHALER, I., HEITGER, F., KÜBLER, O., and VON DER HEYDT, R. 1992. Detection of general edges and keypoints. *European Conference on Computer Vision*.
- SALISBURY, M., ANDERSON, C., LISCHINSKI, D., and SALESIN, D. H. 1996. Scale-dependent reproduction of pen-and-ink illustrations. In *Proceedings of the 23rd annual conference on Computer Graphics and interactive techniques*, 461 – 468.
- SHANNON, C. E. 1949. Communication in the presence of noise. In *Proc. Institute of Radio Engineers*, 37, 1, 10 – 21. Reprinted in *Proc. IEEE*, 86, 2, 1998, 447 – 457.
- SIMMONS, M. and SÉQUIN, C. 2000. Tapestry: a dynamic mesh-based display representation for interactive rendering. In *Proc. Eurographics Workshop on Rendering*, 329 – 340.
- SMITH, A. R., 1995. A pixel is not a little square, a pixel is not a little square, a pixel is not a little square! (and a voxel is not a little cube). Technical report, Microsoft, 1995.
- TOLÉ, P. *et al.* 2002. Interactive global illumination in dynamic scenes. *ACM Trans. Graphics*, 21, 3, 537 – 546.
- TOMASI, C. and MANDUCHI, R. 1998. Bilateral Filtering for Gray and Color Images. In *Proceedings of the 1998 IEEE International Conference on Computer Vision*.
- TUMBLIN, J. and CHOUDHURY, P. 2004. Bixels: picture samples with sharp embedded boundaries. In *Proc. Eurographics Symposium on Rendering*.
- TUMBLIN, J., AGRAWAL, A. and RASKAR, R. 2005. Why I want a gradient camera. In *Proc. IEEE Computer Vision and Pattern Recognition*, 1, 103 – 110.
- VON DER HEYDT, R., PETERHANS, E., and BAUMGARTNER G. 1984. Illusory contours and cortical neuron response. *Science*, 224, 4654, 1260 – 1262.
- WANG, A., TANG, M., and DONG, J. 2004. A survey of silhouette detection techniques for non-photorealistic rendering. In *Proceedings of the 3rd International Conference on Image and Graphics*, 434 – 437.
- WALTER, B., DRETTAKIS, G. and GREENBERG, D. 2002. Enhancing and optimizing the Render Cache. In *Proc. Eurographics Workshop on Rendering*, 37 – 42.
- WARRINGTON, E. K., and TAYLOR, A. M. 1973. The contribution of the right parietal lobe to object recognition. *Cortex*, 9, 152 – 164.

- WARRINGTON, E. K., and TAYLOR, A. M. 1978. Two categorical stages of object recognition. *Perception*, 7, 695 – 705.
- WERTHEIMER, M. 1923/1958. Principles of perceptual organization. In D.C. Beardsless and M. Wertheimer (eds.), *Readings in Perception*. Princeton, NJ: Van Nostrand, 115 – 135.
- WHITTED, T. 1980. An improved illumination model for shaded display. *Communications of the ACM*, 23, 6, 343 – 349.