Efficient and Comprehensible Visualization of Complex 3-D Scenes

Michael S. Burns

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

Recommended for Acceptance
By the Department of
Computer Science

Advisor: Adam Finkelstein

January 2011

Abstract

The advancement of computer technology has brought about greater ability to model and render complex 3-D scenes. Highly complex structures in these scenes give rise to large amounts of information that can be hidden using conventional rendering and visibility techniques because of occlusion. In order to expose structural information in a complex model, non-photorealistic visibility techniques must be used.

This thesis presents non-deformative techniques for creating structurally rich visualizations of complex 3-D scenes. Techniques for reducing occlusion globally are combined with an importance specification system to expose and emphasize objects and regions of interest. These occlusion reduction methods are also applied locally according to an occlusion function defined over a cutaway structure. Localized occlusion reduction can be combined with augmentations to the polygonal and volumetric rendering pipeline to create cutaway illustrations that expose and emphasize objects and regions of interest. The presented techniques are made with realtime rendering in mind, and are shown to operate at interactive framerates for dynamic models and scenes. Sample result images are presented for medical and architectural visualization scenarios.

The main contributions of this thesis are a novel method for creating sparse line representations of isosurfaces in volumes, a contextual cutaway structure and associated occlusion function, an efficient and flexible image-based cutaway surface computation algorithm, and several cutaway surface function modifications to support perspective projection correction, edge compression, locally defined angles, and directional constraints.

Acknowledgments

I would like to thank my advisor, Adam Finkelstein, for his enthusiastic support and encouragement throughout my years at Princeton. His amazing attention to detail and thorough reviews have improved my work many times over, and publication of my work would not have been possible without him.

I thank the members of my dissertation committee, Szymon Rusinkiewicz, Doug DeCarlo, Thomas Funkhouser, and Perry Cook, who served as co-authors and collaborators, and who gave valuable feedback in improving both the technical and presentation aspects of this dissertation. I would also like to thank Olga Troyanskaya for serving on my generals committee and helping me emerge from the introductory stage of my graduate career.

The Princeton Graphics Group has been an invaluable resource, providing unwavering support through thick and thin, and especially during Siggraph crunch time. Thanks to Janek Klawe, Chris DeCoro, Joshua Podolak, Jason Lawrence, Diego Nehab, Forrester Cole, Misha Kazhdan, Phil Shilane, Aleksey Golovinskiy, Benedict Brown, and Tim Weyrich, for collaborating with me on projects, papers, and life. Thanks to the entire group from 2003-2008 for your paper critiques and project suggestions.

My time at Siemens Corporate Research connected me to many talented individuals in the field of applied medical imaging visualization, and gave my research an avenue for impact in the real world. Thanks to supervisors Gianluca Paladini and Klaus Engel, close collaborators Martin Haidacher and Wolfgang Wein, and co-authors Ivan Viola and M. Eduard Gröller for fostering my research interests and assisting me in publishing quality work.

I would also like to thank Gordon Kindlmann, Allen Sanderson, Maneesh Agrawala, and Wilmot Li for advice, assistance, and feedback in the development of my research. This work and my graduate studies have been partially supported by the National Science Foundation through grants HLC-0308121, CCF-0347427, IIS-0511965, and by the Princeton PICASso program, in turn supported by the NSF under grand 9972930.

As a final word I would like to thank the many friends who have supported and encouraged me over the years. Of course none of this would be possible without the unrelenting love and support of my parents, Donald and Lorraine Burns, whom I thank profusely and hope to make proud.

Contents

List of Figures

Chapter 1

Introduction

Over the past three decades, the field of computer graphics has evolved considerably. Limitations in computing resources forced early researchers to use their algorithms on simple geometric models. At the time, these crude images could only show the budding promise of computer-based renderings to be as informational and evocative as photography and human artistry. Today, this promise is being realized, as computer graphics are used to entertain, by creating special effects in movies and full-length animated features, and to inform, through medical and architectural illustrations. The widespread use of computer graphics has come about by a maturity in computer graphics rendering techniques and an advancement of computer processing power.

Along with an increase in computing power comes the ability and desire to more accurately model and represent the world around us, by representing objects and materials with greater fidelity and including more objects together in a scene. The developments of high fidelity 3-D scanners, powerful and accessible 3-D modeling tools, and high resolution medical imaging devices have led to increased surface complexity and volumetric resolution, as well as increased structural complexity. The benefits of increased surface complexity and volumetric resolution can readily be realized as computing power increases and more efficient rendering algorithms are used. On the other hand, the benefits of increased structural complexity are not fully realizable using basic rendering techniques, held back by the problem of occlusion. In order to expose

the structural information present in structurally complex scenes, special visualization techniques must be used.

These specialized visualization techniques often eschew the goal of photorealism in favor of a goal of information communication. The specific methods used are widely varied but the desired result is often different from a photorealistic rendering of a scene. Instead, the joint use of photorealistic and non-photorealistic rendering techniques can result in images that are *more* effective at conveying structural information than a photograph of a real scene could convey. Even so, the real value of using computer-driven visualization techniques stems from the ability to produce animations, especially those that can be driven by a user.

The commoditization of computer hardware has brought interactive graphical systems to every person who owns a computer or video game system. Interactivity, as the result of increased computing power and rendering efficiency, has exposed new potential in computer graphics, allowing users to instantly create images that are customized to specific input data and viewing parameters. At present, we have interactive systems to explore 3-D models of buildings, to simulate combat scenarios from a first-person perspective, and to clinically evaluate patient medical data from CT, MRI and ultrasound scanners, just to name a few. Many of these interactive systems currently use photorealistic rendering techniques, or approximations thereof, and could benefit from using specialized visualization techniques to facilitate exploration of structurally complex scenes.

Artists have long created illustrations to depict structurally complex scenes and expose internal structure, including early anatomical drawings by Leonardo DaVinci [Choulant, 1920] and modern day technical and scientific illustrations [Hodges, 2003; Giesecke et al., 2008], such as that shown in Figure 1.1. This thesis presents a suite of computer-aided techniques for generating similarly expressive, informational visualizations of structurally complex scenes. While the techniques are not aimed at exactly recreating one particular artistic style, they do take inspiration from these artists' efforts to express complex structure in architecture, industrial design, and medicine. Furthermore, the techniques described are supported by efficient computational techniques to facilitate interactive exploration and rendering of reasonably sized input models.

Figure 1.1: Artist-drawn cutaway illustration of a cruise ship. Structures on the ship's interior are exposed by removing occluding material from the ship's hull. The hull is only removed in areas near the exposed structures, preserving view of the rest of the hull when possible. The illustration was created in Adobe Illustrator and Photoshop, with a reported illustration time of 960 hours.

The techniques in this thesis leverage an array of existing techniques developed by previous research in the field of smart visibility [Viola and Gröller, 2005] and focus+context rendering [Hauser, 2005]. While some such techniques create multiple images to examine a single dataset [König et al., 1999; Rezk-Salama and Kolb, 2006], these techniques focus on creating a single expressive image. Furthermore, while substantial work has been done in approaches that deform a scene, such as creating peel-away [McGuffin et al., 2003; Correa et al., 2006] or exploded [Bruckner and Gröller, 2006; Li et al., 2008] images, the techniques presented in this thesis focus on occlusion reduction without deformation. By avoiding deformation, features are rendered in

an image in their original geometric configuration, preserving object size, relative distances, and other positional relationships.

The remainder of this introduction further explores the problem of complexity and occlusion (Section 1.1), examines previous work on complex scene visualization (Section 1.2), outlines goals for visualizations (Section 1.3), and introduces new techniques to address these goals (Section 1.4). The remainder of this thesis explores methods of reducing occlusion globally (Chapter 2), incorporating importance specification of regions for selective exposure (Chapter 3), and providing exposure locally (Chapter 4). It also describes realtime rendering techniques for the aforementioned methods (Chapter 5) and considerations for interactive applications (Chapter 6). Finally we examine some application results in medical imagery and architectural visualization (Chapter 7).

1.1 Structural scene complexity and occlusion

Structural information about a scene concerns the positions of objects relative to others and the structures formed by objects. This information can include *positional* descriptions such as, "Object A is next to Object B," and, "Object B is below Object C," or *nesting* descriptions such as "Object D is enclosed by Object E." This type of information is often useful for gaining a semantic understanding of the scene, allowing one to infer the purpose or intended function of objects, as well as their role in any action in the scene.

As the number of objects increases, so does the potential structural information, as each object can be related to every other. In order to communicate a positional or nesting relationship between two objects in a rendering, these objects must be at least partially visible. However, a single 2-D rendering of a 3-D scene can typically only display a fraction of these structural relationships, because the 2-D projection of the 3-D scene introduces visual occlusion. In these situations, the only way to control which structural relationships are visible is to prevent occlusion by positioning the camera.

Unfortunately, having more objects in a scene also introduces more potential occluders. Following this trend, finding a suitable camera position from which to observe positional relationships can become increasingly difficult. Furthermore, for a group of opaque objects, there may be no suitable camera position at all from which to observe nesting relationships. Thus, occlusion becomes more of a problem as structural scene complexity increases. This thesis addresses this problem by introducing techniques to reduce occlusion, enabling the perception of these positional and nesting relationships in structurally complex scenes.

1.2 Previous work

The need to visualize complex 3-D scenes has fueled visualization research in smart visibility [Viola and Gröller, 2005] and focus+context visualization [Hauser, 2005]. Existing techniques range from user-driven tools that assist artists in creating expressive illustrations, to nearly automatic tools that create images on-demand.

Deformative techniques expose occluded features by changing the shape of occluding materials or geometry. Tools such as VolumeShop [Bruckner and Gröller, 2005] allow user-specified manipulation and deformation of 3-D volumes. More automatic tools can generate exploded diagrams of volumetric datasets [Bruckner and Gröller, 2006] and polygonal scenes [Li et al., 2008]. Such exploded diagrams are efficient are showing many components of a scene, but may hide or omit information about structural relationships in doing so.

Other deformative techniques can create peel-away illustrations [Correa et al., 2006; McGuffin et al., 2003], showing the occluding layers around a featured area. These techniques are best for visualizing nested structures, easily conveying such structural relationships as layers are peeled away. They are most appropriate for deforming feasibly pliable occluding material, such as body tissue, and less appropriate for visualizing rigid structures such as mechanical components.

Non-deformative techniques expose occluded features by omitting portions of materials or geometry in renderings. A depth buffer can control image processing that replaces occluding surfaces with semi-transparent representations [Feiner and Seligmann, 1992]. A similar technique

has been used with medical imagery to expose vessels embedded in the body [Straka et al., 2004]. Other work driven by surgical planning uses smart visibility to create ghosted views, suppressing occluding material locally and globally [Krüger et al., 2005]. User-specification of a focus area and material importance can be used to guide the visualization, as shown in the ghosted images generated by the ClearView system [Krüger et al., 2006]. Other systems incorporate automatic ghosting and fading of outer layers to expose nested structures, such as those in medical imagery [Bruckner et al., 2006].

Rather than smoothly fading out material and creating ghost-like images, some systems sharply cut away occluding material in the view direction, resembling hand-drawn cutaway illustrations [Giesecke et al., 2008; Hodges, 2003]. One simple method considers the intersection of two half spaces or an inflated convex hull of feature objects to define a cutaway volume [Diepstraten et al., 2003]. More advanced cutaway shapes can be created by considering user-drawn cutting outlines along occluding surfaces [Coffin and Hollerer, 2006; Liang et al., 2005]. An effective cutaway shape can be generated automatically by extruding the depth footprint of a feature object in the direction of the camera [Viola et al., 2005]. Cutaway illustrations that more closely resemble traditional hand-drawn illustrations can be created by carefully processing and clipping occluding scene geometry according to a set of artist-inspired rules [Li et al., 2007].

The techniques described in this thesis create images that most closely resemble those of the non-deformative cutaway techniques, with specific visual and algorithmic aspects designed to meet several visualization goals.

1.3 Goals for visualization

In order to create efficient and comprehensible visualizations of structurally complex 3-D scenes, the following three goals should be considered. The first is to create specialized renderings that expose scene content and structural information, which can be used to infer object function. The second is to automatically create these renderings without significant user skill, work, or preprocessing, while still allowing user control over simple parameters. The final goal is to generate

images at interactive frame rates, allowing for visualization of dynamic scenes in interactive applications.

Informational visualization The first goal is to create images that selectively communicate more information about scene content and structure than images generated using conventional rendering techniques. One or more objects of interest may be the *focus* of the visualization, in which case it is important they be visible in a rendered image. Additionally, positional and nesting relationships among these objects and other objects in the scene provide *context* that can be used to show global position as well as functional information.

In order to express positional and nesting relationships between objects, images must be able to show distances between objects. Thus, objects and the spaces between them should not be entirely occluded by other objects. However, the problem of occlusion should be addressed while retaining some notion of depth ordering, which is important for perceiving 3-D position.

Not all structural relationships can be made visible in a given image. In order to control which relationships are expressed, objects can be assigned an importance value. Objects of high importance become the focus of the visualization, and structural relationships involving them are more likely to be exposed. On the other hand, objects of low importance serve mainly to provide context and are likely to be de-emphasized and possibly suppressed from rendering.

Simple interface The second goal arises from the intent that these techniques be accessible to users exploring 3-D scenes or datasets. In order to foster usefulness and accessibility, the techniques do not depend on extensive pre-processing of the scene, nor require that the user have high artistic or technical ability. Instead the techniques depend on a small number of easily controllable and simple parameters that can be used to drive the visualization. Such inputs can include the selection of objects to be the focus of the visualization, as well as sliders to control the extent of object emphasis and exposure. By avoiding pre-processing of the scene or viewpoint, these techniques also afford arbitrary camera movement and operation on animated or dynamic scenes, for maximum flexibility.

Interactive performance The third goal comes about because these techniques are meant to be used in an interactive setting. Visualizations are often more effective when a user is given realtime feedback over parameter changes, so that he may adjust them quickly and efficiently according to his desires. Furthermore, visualization systems that respond to feedback quickly correspond to our natural model of physically investigating real objects. As such, the visualization techniques should have implementations with efficient computation methods that can perform at interactive frame rates.

1.4 Contributions and overview of visualization strategies

Each of the existing visualization solutions mentioned in Section 1.2 addresses some aspects of the aforementioned goals. They are all capable of creating rich visualizations by addressing the problem of occlusion and exposing objects of interest, though do so with varying degrees of quality and visual appeal. The nearly automatic techniques are naturally easier to use than more manual techniques that require substantial or skilled user input. Some techniques use efficient computation and rendering techniques to allow interactive visualizations, but the more computationally involved techniques with high visual appeal can be prohibitively slow for dynamic animated scenes with changing viewpoint. This thesis presents a suite of visualization techniques which together intend to address all three of these goals by creating visually appealing images that expose objects of interest with minimal user-effort at interactive framerates for dynamic polygonal or volumetric scenes.

The techniques described herein can be broadly broken down into three parts: reducing occlusion, incorporating importance, and localizing exposure.

Occlusion can be reduced in an image by rendering objects with sparse rendering methods. When applied globally, these methods can help visualize nesting relationships and positional relationships for all objects in a scene, as described in Chapter 2. Included in this chapter is a novel method for creating sparse line drawings from volumetric data.

Objects can be emphasized or minimized through selective application of occlusion reduction methods by considering a notion of object importance, as described in Chapter 3. Emphasis can be made through choice of color, shading, or rendering style, while objects can be exposed using sparse rendering techniques on occluding objects.

Sparse rendering methods can be applied locally, and in a view dependent fashion, to expose objects while maintaining normal rendering methods in peripheral regions for context, as described in Chapter 4. Included in this chapter are novel definitions of a contextual cutaway structure and an associated occlusion function.

Such localized occlusion reduction methods can be used with polygonal and volumetric rendering systems at interactive framerates, as described in Chapter 5. Included in this chapter are a novel image-based cutaway surface computation algorithm, new modifications to the polygonal rendering pipeline for rendering cutaway images, and cutaway surface function modifications to support perspective projection correction and temporal coherence.

These techniques can be incorporated into interactive scene exploration applications, as described in Chapter 6. Included in this chapter are additional cutaway surface function modifications to support locally defined angles and directional constraints.

Finally, the application of these techniques to visualization of medical imaging data and architectural models is explored in Chapter 7.

Chapter 2

Viewing Nested Structures: Globally Reducing Occlusion

Nested structures are commonly found in 3-D scenes but are often difficult to visualize due to the dominant effect of occlusion, with outer shells occluding interior objects. In architectural scenes, the walls that make up the outer structure can entirely occlude the structure of rooms on the interior, while interior walls and floors that form this structure occlude the contents of rooms they define. Omission of these walls and floors from rendering is not an effective option as it leaves interior objects (e.g. furniture) floating in space, seemingly unstructured and cluttered. A similar problem occurs in anatomical scenes where outer skin occludes underlying muscles, which in turn occlude inner vessels and bones. While any one of these tissues can be the focus of the visualization, view of the other components helps establish context and provides a more complete image of the body.

Conveying nesting relationships can be crucial to conveying scene structure, but it can be difficult to depict even the presence of interior objects using basic photorealistic rendering methods. Animations that progressively remove outer layers can be used to show the contents of nested containers, but equally expressive effects can be achieved with single images by globally reducing occlusion. The following sections show how occlusion can be progressively reduced using

simple translucency (Section 2.1), edge highlights (Section 2.2), and line drawings (Section 2.3), according to a desired occlusion reduction factor. Section 2.4 details a novel method for creating sparse line drawings from volumetric data.

2.1 Translucency

Rendering occluding surfaces with translucency is a straightforward way to mitigate occlusion effects by exposing underlying objects through a translucent occluder, as shown in Figure 2.1. When rendering with translucency, a surface is assigned an opacity value between 0 and 1. This opacity value is used to blend the surface material color with the existing color in the destination image. As the opacity value decreases, the occluding surface appears more faint, while objects behind the surface are seen more clearly.

Such variable opacity enables variable occlusion reduction, as specified by an *occlusion reduction factor*, R. This continuously-valued variable is defined such that $R = 1$ corresponds to full occlusion reduction and $R = 0$ corresponds to no occlusion reduction. In the case of occlusion reduction by translucency, an object will be completely invisible with $R = 1$ and completely visible (rendered without translucency) with $R = 0$. As R increases from zero, objects become more translucent and occlude less of the scene behind them.

Implementation Translucent rendering is straightforward to implement by decreasing the opacity of an object's material, α_m, by the desired reduction factor. Applied uniformly to an object, the opacity becomes

$$\alpha = \alpha_m - \alpha_m * R \tag{2.1}$$

When rendering surfaces with translucency, depth testing cannot be used, since translucent surfaces in the scene must not prevent underlying surfaces from being rendered. Furthermore, surfaces must be rendered consistently from front to back or back to front, depending on the blending mode used. This layered rendering is simple to do for volumetric rendering by doing

(a) Opaque rendering

(b) Translucent rendering

Figure 2.1: Occlusion reduction using transluncency. (a) With opaque rendering, only the exterior surface is visible. (b) With translucent rendering, internal structures such as the ear canals, sinuses, esophagus, and lungs are visible through the outer skin surface.

either screen-aligned ray-casting or slice-based rendering, changing the slicing direction based on the camera position. It is slightly more difficult for polygonal surfaces, because individual triangles must be sorted by their projected depth before rendering.

Visualization effect The effect of translucent rendering is shown in Figure 2.1. When used with a small number of occluding surfaces, translucent rendering allows effective visualization of occluded objects. However, with multiple layers of translucent surfaces, it can become increasingly difficult to differentiate underlying surfaces. This well known problem [Interrante et al., 1996; Hauser et al., 2001] occurs because translucent surfaces reduce the effective expressive colorspace of underlying surfaces by tinting colors from underlying surfaces with the color from the overlaid surface. As R is increased, objects behind the surface are tinted less, retaining more of their original color, while the occluding surface appears more faint. It is important then to choose a value of R that maintains a balance between how strong a translucent surface appears and how much it limits the colorspace of underlying surfaces.

2.2 Edge highlights

Edge highlights attempt to improve upon simple uniform translucency by automatically altering a surface opacity for a given point based on the orientation of the surface relative to the camera, as shown in Figure 2.2. The idea is that areas of a surface nearly parallel to the camera do little to show the shape of the surface and cause unnecessary tinting of underlying surfaces. Areas of a surface oriented some angle away from the camera are what define bumps, ridges, and silhouettes that help depict an object's shape. Edge highlights work by rendering portions of a surface nearly perpendicular to the camera nearly opaque, with areas nearly parallel to the camera rendered very translucent.

Implementation Edge highlights can be rendered with a strength R by computing the opacity α for a surface at a point p as follows, where $\hat{n}(p)$ is the surface normal at p, $\hat{v}(p)$ is the normalized

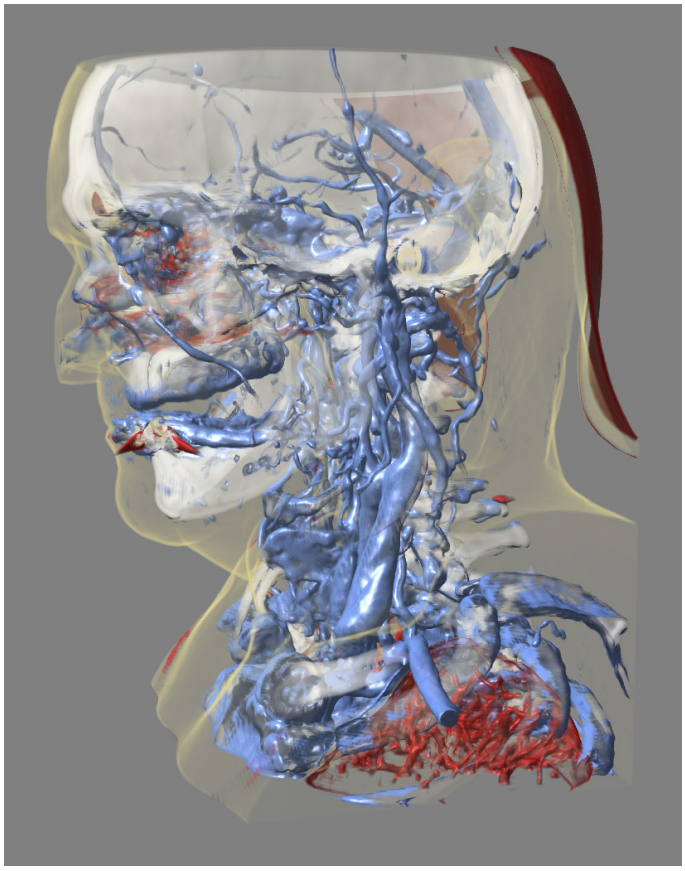


Figure 2.2: Occlusion reduction using edge highlights. The outer skin (yellow) is rendered with edge highlights. Skin surface features are distinguishable in the periphery of the image while broad areas of skin parallel to the camera in the center of the image are suppressed, allowing clear view of the inner bone and vasculature.

view vector from p, and k is a fixed exponential parameter.

$$\alpha(p) = \alpha_m - \alpha_m * R\left(1 - (1 - \hat{n}(p) \cdot \hat{v}(p))^k\right) \tag{2.2}$$

The value of k controls how quickly the opacity decays away from the edges of the surface, and how narrow the highlights appear. A reasonable value of k is between 1 and 4.

The rendering of edge highlights can be carried out in a shader for either volumetric or polygonal rendering. Since edge highlights are basically a non-uniform application of simple translucency, similar rendering requirements apply—depth testing cannot be performed and primitives must be rendered in a consistent front to back or back to front order.

Visual effect The effect of edge highlights is shown in Figure 2.2. In the generic case of a sphere rendered with edge highlights, for high values of R, the surface will be translucent near the center with opacity concentrated at the edges. For low values of R, the entire surface will be nearly opaque, just as with the simple translucency method (Section 2.1). In fact, a surface rendered with edge highlights will always have the same minimum opacity as generated by the basic translucency method for a given value of R. Thus edge highlights can be thought of as an extension of the basic translucency method with more opaque surface definition near the edges.

Edge highlights work best for curved surfaces where the highlights are concentrated near contours and silhouettes. Surfaces with large flat areas will become uniformly opaque in flat areas as they become perpendicular to the image plane. Fortunately these same areas become uniformly translucent as they become parallel to the image plane, reducing their occlusion power. Edge highlights also work best for surfaces with relatively low surface detail, which generate fewer opaque edge highlights and allow more translucency, preventing occlusion of underlying objects.

2.3 Line extraction and rendering

Line drawings are typically used to present sparse, stylized renderings of surfaces but can also be used to represent surfaces in situations when occlusion should be minimized, such as Figure 2.3. A computer-generated line drawing representation typically aims to show important surface features that help define a surface. These lines may be dynamically computed from a surface, such as contours, ridges [Interrante et al., 1995], suggestive contours [DeCarlo et al., 2003], and apparent ridges [Judd et al., 2007], or may be manually defined as part of the modeling process. By rendering lines derived from a surface without rendering the surface itself, a viewer can see a large amount of 3-D surface structure with a relatively small number of rendered pixels.

Implementation For viewing nested surfaces, lines should be rendered without their associated surfaces so that lines from underlying surfaces can also be seen. The line rendering can be

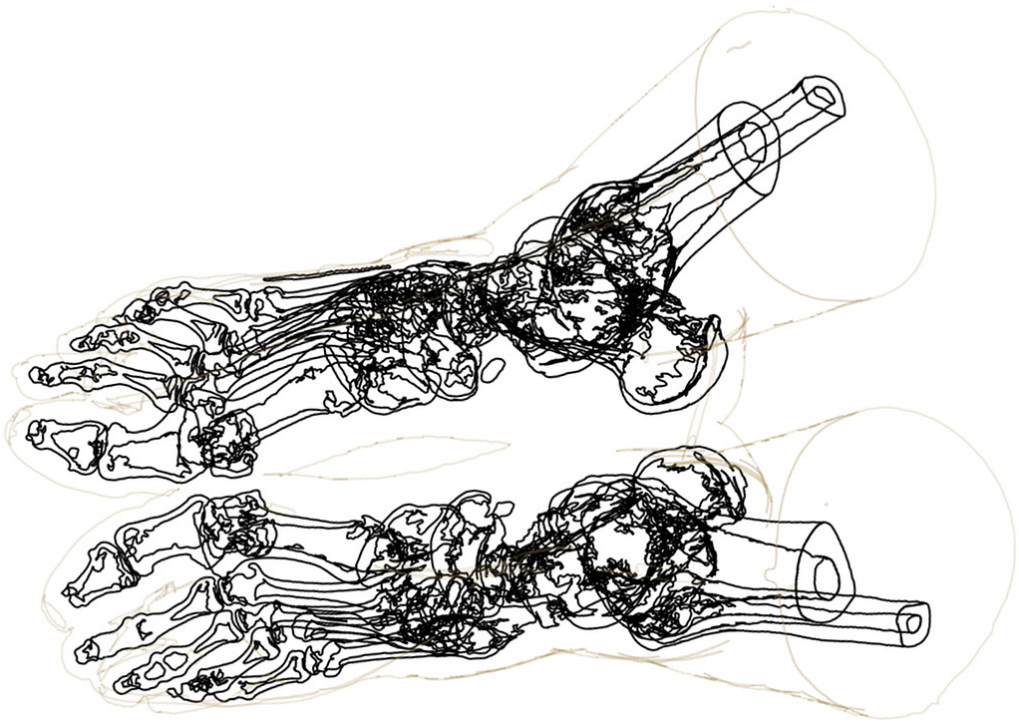


Figure 2.3: Occlusion reduction using line rendering. The bones (black) are visible through the outer skin (tan) because skin lines have a low occlusion footprint. However, too many lines overlapping can cause visual clutter and occlusion, as seen in the ankle areas.

accomplished using standard techniques, once lines are extracted from the polygonal surfaces or volumetric data sets.

Extracting lines from surfaces is a widely studied field, and as a result many different line types have been defined. For a given mathematical definition of lines, there are several ways to extract them from surfaces. Line extraction often involves calculating some function over the surface and defining lines near zero crossings of the function.

For polygonal surfaces, one may evaluate the function per face and extract edges of the mesh between neighboring faces that evaluate to values of alternate sign [Appel, 1967]. This algorithm typically works well for polygonal surfaces with large faces and sharp edges. For well-tesselated smooth surfaces, one may evaluate the functions at each vertex and interpolate along polygon edges to extract new lines that estimate the zero-set [Hertzmann and Zorin, 2000].

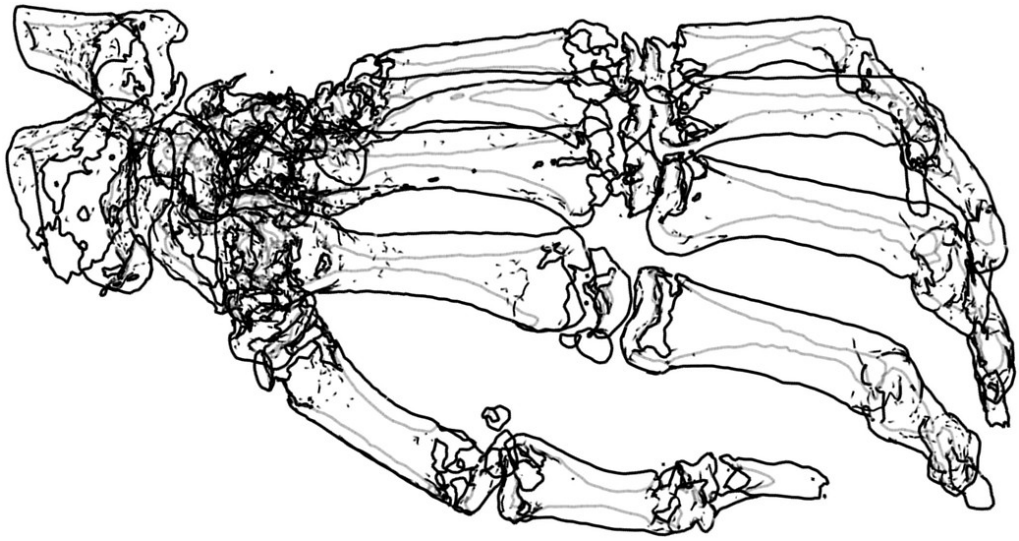


Figure 2.4: Stylized line rendering of occluded surfaces. Lines from interior back-facing surfaces are rendered in grey, aiding distinction from lines from exterior surfaces (black) and helping reduce overall visual clutter.

For volumetric data sets, one can render lines in a variety of ways [Bremer and Hughes, 1998; Nagy and Klein, 2004; Schein and Elber, 2004], including work that extracts smooth lines along zero crossings by adapting smooth polygon extraction methods [DeCarlo et al., 2003] to volumes. Section 2.4 offers a summary of this technique, with a more thorough presentation appearing in Burns et al., 2005.

Visual effect The effect of drawing lines without any surfaces is shown in figure Figure 2.3. The lines themselves are often rendered in solid, opaque black when used in NPR settings but can be rendered in a variety of styles or colors [Kalnins et al., 2003]. Such variety can help aid in visual distinction between surfaces from multiple objects, or provide cues about line-generating surface properties. For example, a lightened or dotted style can represent lines coming from rear-facing surfaces, where lines would be occluded by at least one surface (Figure 2.4). This differentiation can help aid distinction between the structures of front-facing and back-facing surfaces of the same object. Lines can also be styled based on their depth or the number of surfaces by which they would be normally occluded [Fischer et al., 2005]. Rendering lines with halos [Appel et al.,

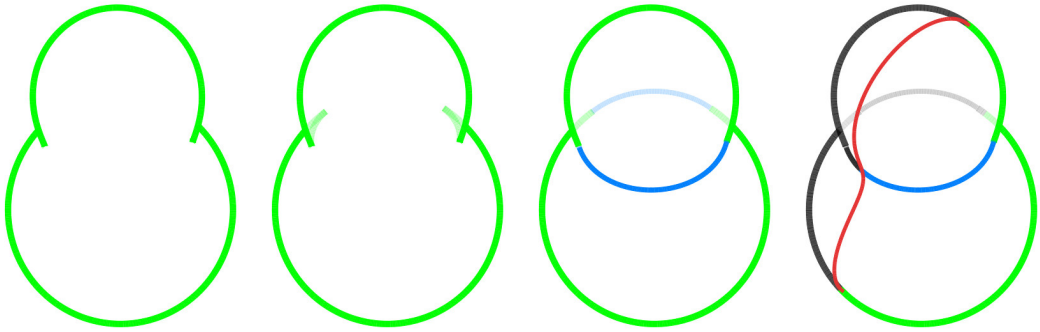
1979], where lines crossing under other lines are broken at their junction, can also be used to help communicate depth ordering.

A rendered surface can be augmented with lines when occlusion reduction is not necessary, and completely replaced by lines when occlusion reduction is necessary. However, when too many lines are drawn over each other, they can occlude themselves, creating visual clutter and hindering comprehensibility. Fortunately, given variable occlusion reduction, lines themselves can be faded as the need for occlusion reduction increases. Thus with $R = 0$ a surface should be rendered with lines at full opacity. For $R > 0$, the surface should be omitted and lines should be rendered with opacity $\alpha = 1 - R$. Thus, when $R = 1$ nothing is rendered, allowing maximal occlusion reduction.

2.4 Line drawings from volumes

This section describes a system that extracts and renders linear features that lie on isosurfaces within a volume. The benefits of rendering or emphasizing linear features such as ridges and valleys or silhouettes in traditional volume renderings have been demonstrated by a number of researchers [Interrante et al., 1995; Ebert and Rheingans, 2000; Lum and Ma, 2002; Nagy et al., 2002; Lu et al., 2003; Svakhine and Ebert, 2003]. Rendering lines alone creates a non-photorealistic line drawing that can communicate surface shape. In general, non-photorealistic techniques have been used to create expressive images from volume data [Interrante and Grosch, 1998; Treavett and Chen, 2000; Csébfalvi et al., 2001; Dong et al., 2003; Kindlmann et al., 2003]. Previous methods for creating line drawings [Bremer and Hughes, 1998; Nagy and Klein, 2004; Schein and Elber, 2004] differ from the following method by supporting fewer line types, requiring preprocessing, or operating in image-space only.

The system uses a randomized, temporally coherent seed-and-traverse algorithm inspired by one originally developed for silhouette extraction from meshes [Markosian et al., 1997]. It exploits not only the sparseness of surfaces within the volume, but also the sparseness of lines on a surface, reducing the amount of data that must be examined while rendering a frame to a one-dimensional subspace of the volume. This reduction allows for efficient computation, realizing



(a) Visible contours (b) Hidden contours (c) Suggestive contours (d) Cutting planes

Figure 2.5: Linear geometric features extracted from volumes. Contours can be classified as (a) visible or (b) hidden, depending on whether or not they are occluded. (c) Suggestive contours and (d) intersections with cutting planes show additional surface detail and provide additional opportunities for styling.

even greater efficiency than accelerated isosurface extraction algorithms, which must traverse at least a significant portion of voxels along a surface [Livnat et al., 1996; Cignoni et al., 1997; Parker et al., 1999; van Kreveld et al., 2004].

Several types of lines are supported (Figure 2.5), including occluding contours (i.e., interior and exterior silhouettes), suggestive contours [DeCarlo et al., 2003], and intersections with cutting planes. Selecting which lines and features to draw allows a flexible trade-off between rendering quality and interactivity. The system achieves interactive rates on commodity hardware for data sets of roughly 512^3 samples, while placing few demands on the graphics card (which could be utilized to provide further stylization or to simultaneously display an extracted mesh).

2.4.1 Fast line extraction

This section describes how to extract sparse linear features from volumetric datasets. Although the discussion focuses on occluding contours (interior and exterior silhouettes), the techniques are general. Their applications to different line types are described in Section 2.4.2.1.

The main algorithmic challenge is achieving extraction of lines in time proportional to the size of the extracted features (i.e., output sensitivity). While it is possible to design pathological datasets of size $n \times n \times n$ that produce isosurfaces of size $O(n^3)$ with silhouettes also of size $O(n^3)$

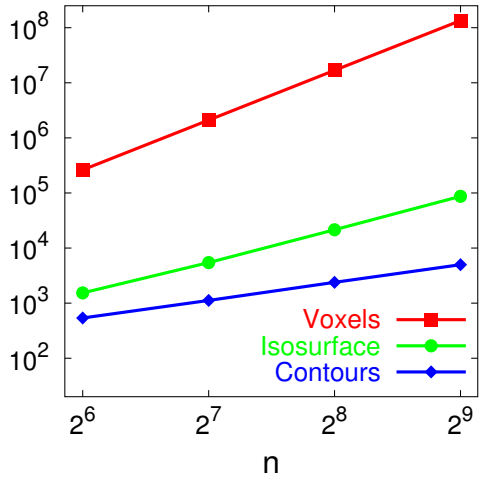


Figure 2.6: Experimental verification of $O(n)$ contour size for several n^3 versions of the aneurysm dataset (Figure 2.10a). The total number of voxels grows cubically with dimension n by definition. Shallower slopes on the remaining lines show that the number of voxels on an isosurface grows quadratically with n and the number of voxels on contours grows linearly with n.

(for example a Peano curve in 3-D), these shapes are not characteristic of typical datasets. In practice, isosurfaces describing a particular bandlimited shape represented by a volume of size n^3 will have size $O(n^2)$. Furthermore, the contours of those isosurfaces will have size $O(n)$. Markosian et al. [1997] effectively argue for the latter statement by showing that silhouettes of a shape have size proportional to the square root of the number of faces representing that shape. Their argument might also be adapted to address the first claim (about the size of the isosurface), although it would be complicated by the nature of the surface extracted by the marching cubes algorithm. Nonetheless, these claims are borne out by experiments, shown in Figure 2.6.

2.4.1.1 Contours in volumes

The input data $\phi(i,j,k)$ is a 3-D matrix of real-numbered data values, positioned at the nodes of a regularly spaced lattice. The following considers the task of rendering contours on isosurfaces of ϕ.

An isosurface F can be defined as the zero-set of the function

$$f(i,j,k) = \phi(i,j,k) - \tau, \tag{2.3}$$

where τ is a threshold within the range of the data values. The equation $f = 0$ implicitly defines a 2-D surface in 3-D space.

Contours on a continuous surface are those locations where the surface normal is perpendicular to the view. That is, they are locations at which $\hat{\mathbf{n}} \cdot \hat{\mathbf{v}} = 0$, where $\hat{\mathbf{v}}$ is the surface normal and $\hat{\mathbf{v}}$ is the normalized view vector (i.e., a unit vector from a point on the surface to the camera). For volume rendering, one could use this definition directly by first extracting an isosurface at some threshold τ, computing the surface normal as $\hat{\mathbf{n}} = -\nabla\phi$, and extracting the contour. Alternatively, one could consider the set of contours on *all* possible isosurfaces of ϕ, which is itself a 2-D surface defined as the zero set C of the function

$$c(i,j,k) = -\nabla\phi(i,j,k) \cdot \hat{\mathbf{v}}(i,j,k). \tag{2.4}$$

The task of finding contours at a specific threshold τ then reduces to finding the intersection of the 2-D surfaces F and C. Generically, this intersection takes the form of a set of 1-D loops in 3-D space.

In order to extract a contour, voxels containing zeros of both the f and c implicit functions are located. Once a relevant voxel is found, its contour segments are extracted using a variant of the Marching Lines algorithm [Thirion and Gourdon, 1996]. This algorithm was originally presented as an adaptation of the well-known Marching Cubes technique [Lorensen and Cline, 1987] to extracting crest lines on surfaces, but can be generalized to extract the intersection of isosurfaces of any two implicit functions.

Briefly, the algorithm finds intersections of the two implicit functions on the faces of the cube, using linear interpolation based on the values at the eight corners (Figures 2.7a and 2.7b). It then finds any intersections between these two sets of lines on each face. The result is a set of points on the faces of the cube, which when connected yield segments of the contour (Figures 2.7c and 2.7d). Note that this method is completely general, in that it can extract the curves formed by the intersection of any two implicit functions on the volume. This property allows extraction of other families of lines in addition to contours, as described in Section 2.4.2.1.

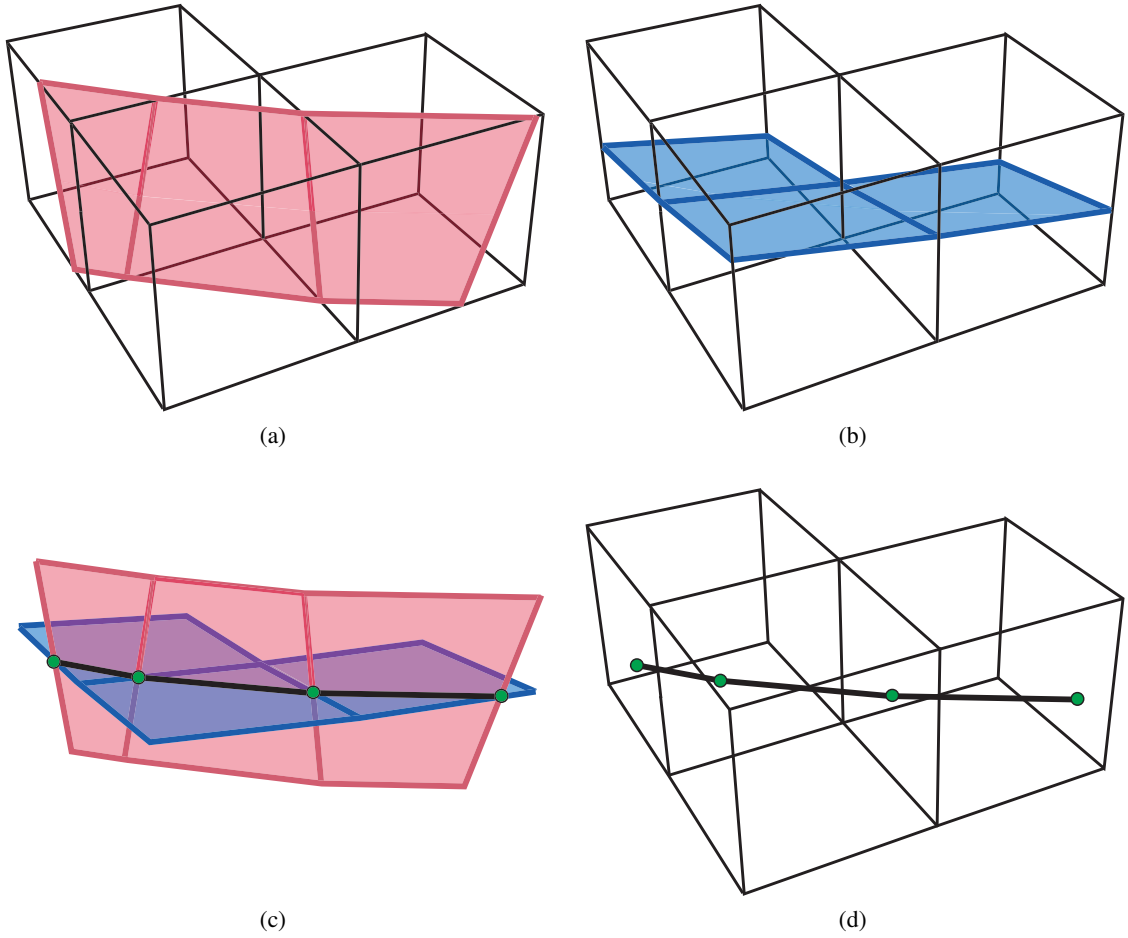


Figure 2.7: Extracting silhouettes as the intersection of two surfaces in voxels. (a) Isosurface F in red traverses three neighboring voxels. (b) Contour surface C is blue. (c) F and C intersect in a curve, which can be traced through the volume (d).

2.4.1.2 Walking contours

The simplest method of finding contours would be to examine all cells in a volume, looking for cubes containing zeros of the implicit functions and extracting line segments in each. However, this approach is impractical for interactive visualization of large datasets due to its lack of output sensitivity. Instead, an accelerated technique that traces lines through the volume can be used to avoid examination of the majority of cells.

Since the intersection of two non-planar, unbounded 2-D surfaces is a closed 1-D loop in the general case, it is reasonable to extract the loop by following it through the volume. Given a

starting *seed* cell, a 1-D segment of the contour is extracted with Marching Lines. Extraction continues in a cell adjacent to a face containing one of the intersection points (Figure 2.7). This process is repeated around the loop until it returns to the starting seed cell. Walking through voxels that contain more than one line segment is performed in an arbitrary but consistent manner, to ensure that complete loops are formed.

Given this algorithm, and supposing knowledge of at least one valid seed cell for every loop in the volume, it is possible to extract all contours in $O(m)$ time, where m is the total length of all loops. Thus, this algorithm is output sensitive, which is convenient since output tends to be sparse for non-pathological cases. In the typical case, m is $O(n)$ in an n^3 volume, as argued above.

2.4.1.3 Exploiting spatio-temporal coherency

The next problem is that of finding seed cells from which to start the contour loop extraction. While contours are view dependent and move along the surface as the viewpoint changes, they usually intersect contours of that surface from nearby viewpoints. To exploit this spatio-temporal coherency, the search for contour seed points for a new frame begins by first examining cells that contained contours in the previously drawn frame, a method previously used for finding contours [Markosian et al., 1997] and suggestive contours [DeCarlo et al., 2004] on surfaces. Since there are $O(m)$ cells containing contours from the previous frame, this search only adds $O(m)$ time to the extraction algorithm, leaving the overall running time as $O(m)$.

To evaluate the performance of this strategy, an experiment in which the number of contour segments found in an animation involving continuous change in viewpoint and isosurface threshold was measured. The red curve at the top of Figure 2.8 shows the number of contour segments extracted by the brute-force algorithm (i.e., "ground truth") at each frame in the sequence. The third curve, drawn in blue, shows the result of using contours from the previous frame as seeds. The two remaining curves involve a random seeding approach, which is discussed in the next section. In all cases, the first frame is initialized with the brute-force algorithm. The results show the temporally-coherent algorithm extracting most of the contours in all frames, with an average

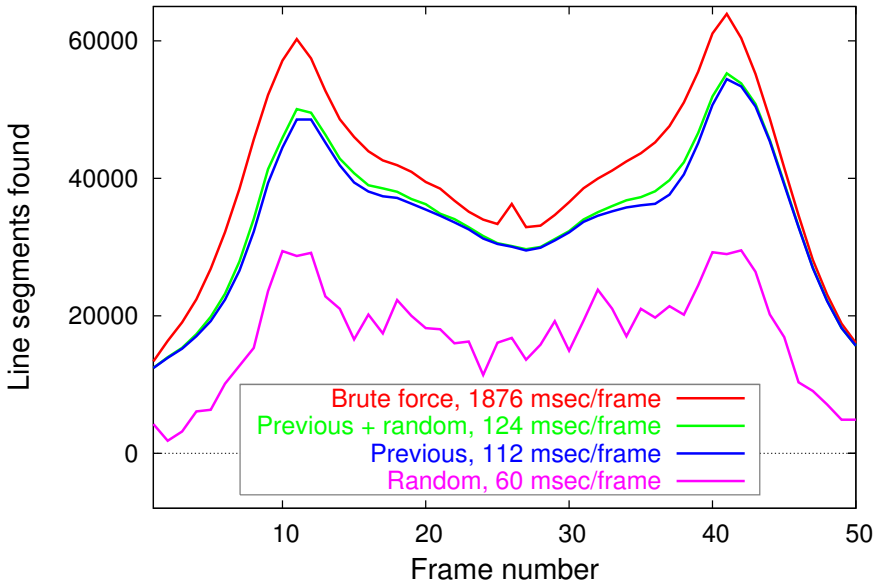


Figure 2.8: Efficiency of extracting contours using different seeding strategies, for an animation with changing viewpoint and isosurface threshold of the aneurysm dataset (Figure 2.10a). A purely random approach finds only a small portion of lines. Using seeds from voxels containing lines in the previous frame finds substantially more lines. The combination of these approaches the effectiveness of the brute force approach, at less than 10% of the computation cost.

of 85% and a minimum of 70%. Although this performance is not perfect, observations show it is not objectionable in practice. First, short segments are more likely to be missed than long ones, so it is unlikely that the most perceptually significant loops will be omitted. In addition, the missing line segments are more likely to occur during periods of quick motion, so they are less likely to be perceived. Finally, when the user stops changing the viewpoint and threshold, the random probing discussed in the next subsection causes the missing segments to be located and drawn, with high probability, within a few frames.

2.4.1.4 Gradient descent for contour location

Contours appear and disappear through changes in view and isovalue. Therefore, in addition to the previous-frame seeding discussed above, new seeds are searched for using an iterative gradient descent method. This locates valid seed cells either near old contours or at random in the volume. Similar stochastic processes have been used for isosurface extraction [Parker et al., 1999; Liu

et al., 2002] to facilitate interactive response times. While not all available features are rendered in an initial frame, subsequent frames find more features and help establish a complete image. As shown in the magenta (lower-most) curve of Figure 2.8, random sampling and gradient descent can be used alone to find a large percentage of new contours. Leveraging information from previous frames complements this process, finding most linear features.

Given a stating point p in the volume, the gradient $\mathbf{g} = \nabla\phi(p)$ and the vector

$$\mathbf{v}_{walk} = \frac{\tau - \phi(p)}{|g|} \frac{g}{|g|} \tag{2.5}$$

are calculated. The vector $\mathbf{v}_{walk}$ is thus a linear approximation of the vector from p to the closest zero of f. The algorithm proceeds to the point $q = p + \mathbf{v}_{walk}$ and repeats this calculation at q using the gradient of the $\hat{\mathbf{n}} \cdot \hat{\mathbf{v}}$ implicit function. This cell is then tested to see if it is a valid seed for a new contour.

This process is repeated by alternately walking along gradients of f and c and testing for contour seeds. Experiments with a wide variety of data sets show that this process typically either finds the surface or fails (by leaving the volume or getting stuck in a local minimum) within four to five iterations. Thus the search can be terminated after five iterations if no seed point is found.

To find new contours, this gradient descent is performed starting either at contour-containing cells from the previous frame or at random cells in the volume. Random probing alone requires potentially a large number of probes to locate most of the contour segments (Figure 2.9).

2.4.2 Comprehensible rendering

To render volume data in a comprehensible manner, any of a number of different families of lines can be extracted using the method described in Section 2.4.1. Extracted segments can be rendered as OpenGL polylines.

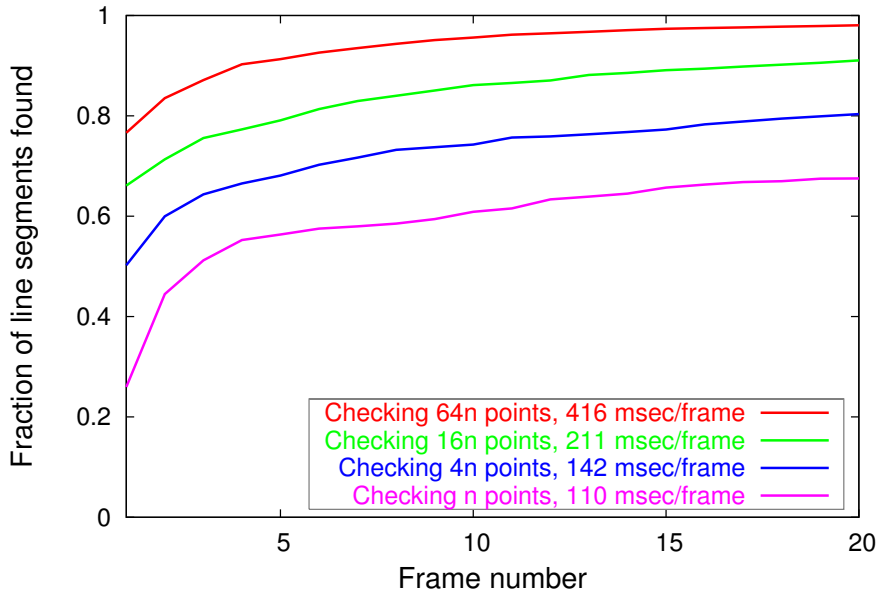


Figure 2.9: Efficiency of randomized probing, together with temporal coherence, for the skull isolevel of the head dataset (Figure 2.10b). The viewpoint and threshold are static. Because contours found on the previous frame are retained, the total number of contour segments found increases over time. We show results for different numbers of random probes, as a multiple of n (where n^3 is the number of voxels).

2.4.2.1 Families of lines

Contours As described in Section 2.4.1, the contour of a surface is found by extracting intersections of the isosurface $\phi - \tau = 0$ and the contour surface $\hat{\mathbf{n}} \cdot \hat{\mathbf{v}} = 0$. The normals are computed as $\hat{\mathbf{n}} = -\nabla\phi$, and are evaluated as needed using a finite differences approximation over a local neighborhood. An alternative would be to precompute normals at program startup, which would increase run-time performance, though with a substantial four-fold increase in memory usage.

Contours arising from interior surfaces can be detected by looking for contours with negative radial curvature [Koenderink, 1984] and specially styled. These interior contours are found by considering extracted contours and computing

$$s(i,j,k) = \nabla\left(\hat{\mathbf{n}}(i,j,k) \cdot \hat{\mathbf{v}}(i,j,k)\right) \cdot \hat{\mathbf{v}}(i,j,k). \tag{2.6}$$

A contour segment is labeled as interior when $s < 0$, since s has the same sign as radial curvature. As with the normal, the gradient is computed numerically on demand.

Suggestive contours Suggestive contours [DeCarlo et al., 2003] can be included to complement contours in line drawings, producing more effective renderings of shape. Suggestive contours are defined as those surface locations where the radial curvature is zero and the directional derivative towards the camera is positive. Thus, suggestive contours are extracted as the zero set of s,

$$s(i,j,k) = \nabla\big(\hat{\mathbf{n}}(i,j,k)\cdot\hat{\mathbf{v}}(i,j,k)\big)\cdot\hat{\mathbf{v}}(i,j,k) = 0, \tag{2.7}$$

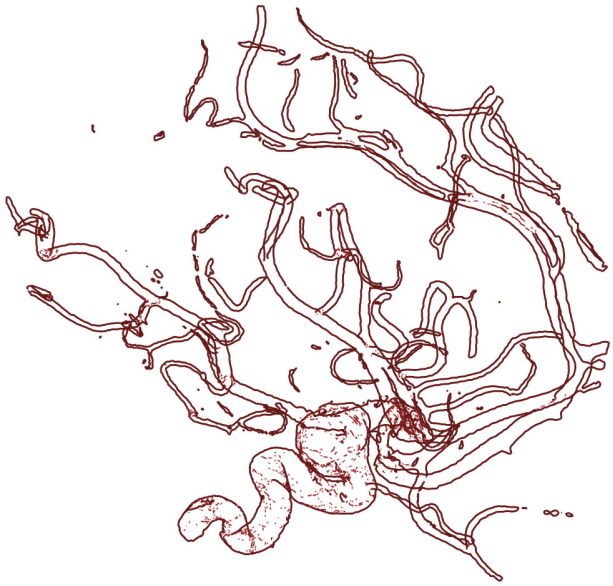
while checking that the derivative of s towards the viewer is positive. In practice, the derivative is compared to a small positive threshold ($D_s > \varepsilon$) to filter out spurious zero crossings caused by noise [DeCarlo et al., 2004]. Furthermore, finding suggestive contours on backfaces requires the sign of the derivative test to be negated ($D_s < -\varepsilon$) in order to keep the analogous lines to those found on frontfaces.

Cutting planes The intersections of isosurfaces with cutting planes can generate additional linear features to communicate surface shape. These intersections can be found by locating intersections of the isosurface $\phi - \tau = 0$ and the zero-set of the plane equation h,

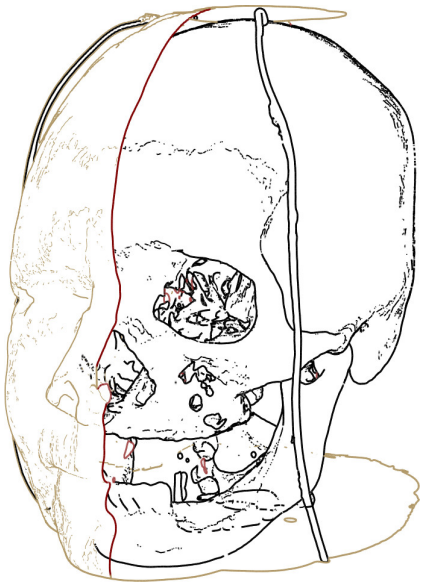
$$h(i,j,k) = Ai + Bj + Ck - D = 0. \tag{2.8}$$

2.4.2.2 Stylization

The system provides the user with full control over which kinds of lines are drawn, and in which style. In particular, the user can control the width and color of rendered lines based on the type of line and the originating isovalue. As an additional option, to provide fast depth cueing without the full cost of visibility, one can render lines with a white halo, as suggested by [Appel et al., 1979]. This visualization cue is most useful under camera rotation and can be performed with virtually no loss of performance over standard rendering.



(a) Aneurysm CT ($256 \times 256 \times 256$)



(b) Head CT ($512 \times 512 \times 209$)

Figure 2.10: Line renderings from volumetric CT scans of an aneurysm and human head. (a) The complex aneurysm contains many thin overlapping features, which are all visible by overlapping lines. (b) The isosurface of the skin and bone are shown separately. In order to prevent interior objects from cluttering the rendering, a visibility test is performed to only render lines on visible surfaces. A cutting plane is then used to suppress skin line rendering to expose the skull.

2.5 Improvements on global occlusion reduction

The aforementioned occlusion reduction methods help convey structural information in complex scenes by allowing more objects to be shown in a single image compared to normal rendering techniques. However, as mentioned previously and evident in Figures 2.1b and 2.3, the application of occlusion reduction to many overlapping components can create visual clutter, making individual objects difficult to discern. Chapter 3 shows how these occlusion reduction methods can be applied selectively to occluding objects in a scene to expose objects of interest rendered with normal occlusion. Chapter 4 shows how this selective application can be localized further according to a view-dependent cutaway structure.

Chapter 3

Controlling Emphasis and Exposure: Incorporating Importance

The techniques in Chapter 2 were described in the context of globally reducing occlusion, where all objects are treated equally in terms of occlusion reduction and visualization focus. Visualization goals are often specific to certain objects in a scene, in which case it is useful to be able to treat certain objects differently from others. These *focus* objects should be the centerpieces of renderings, being shown along with the remaining scene objects that serve as *context*.

Focus objects can be visually *emphasized* by modifying their rendering style to draw the viewer's attention and to differentiate the objects from their contextual neighbors. They can also be *exposed* by applying occlusion-reduction techniques to context objects so that focus objects are as visible as possible in rendered images. The application of emphasis and exposure is controlled by the notion of *importance*, a generalization of focus and context. The following section discusses the concept of importance in multi-object visualizations, followed by a discussion of specifying importance for 3-D scenes in Section 3.2. Emphasis and exposure techniques based on importance values are discussed in Section 3.3.

(a) (b)

Figure 3.1: Artist-drawn illustrations expressing importance through exposure. (a) A car wheel is shown, with a focus on the mechanical brake and suspension components. The wheel and tire serve only as context for the illustration, and are rendered entirely in a line drawing style to minimize occlusion of important mechanical components. (b) The interior electronics of this DSLR camera are shown in the context of the camera body. The front of the lens barrel is allowed to occlude the interior electronics. This occlusion suggests it has a higher importance than the electronics, which in turn have a higher importance than the remainder of the camera body.

Images used with permission, Copyright Kevin Hulsey Illustration, Inc.

3.1 Defining importance

Importance refers to an object's relevance to the broad visualization goal. At its core, visualization is about communicating information through images. When a visualization is constructed to show specific information about an object in a complex scene, this object should be the focus of the visualization and should be considered highly important. An important object may be treated differently from other objects in the scene, in order to draw viewer attention, or to express extra information that would not be possible to express in all objects at once.

Different visualization goals can call for different types of importance. Sometimes, a small set of objects are much more important than all other objects, which are then equally unimportant. Other times, all objects in a scene have a varying degree of importance, and should be visualized

accordingly. Moreover, the most appropriate notion of importance may be some combination of the two. These scenarios are discussed in Sections 3.1.1 through 3.1.3.

3.1.1 Binary importance: objects of interest

Effective visualizations aim to show information about a set of one or more *objects of interest*. These objects may exist in a 3-D scene containing other objects, which become *objects of secondary interest*. The objects of interest become the focus of the visualization, while objects of secondary interest exist to provide spacial and functional context. This type of visualization is sometimes called "focus+context" visualization [Hauser, 2005].

Such two-tiered visualizations suggest a binary notion of importance, where objects of interest are of high importance and all other objects are of low importance. A focused visualization may treat objects of interest specially, making them maximally emphasized and exposed. Accordingly, objects of secondary interest can be de-emphasized and partially omitted in a given rendering.

For example, in the illustration shown in Figure 3.1a, the mechanical components of the wheel, including the brakes, shocks, springs, and suspension link, are the focus of the visualization– objects of interest with high importance. The tire and rim, which surround and would occlude the mechanical components in conventional depictions, are only intended to provide context. Because they are of low importance to the visualization, they are drawn using a very sparse line-drawing representation, to avoid occluding the interior mechanical parts.

With many objects of interest, especially in close proximity to each other, emphasis and exposure techniques become less effective because there are fewer opportunities to de-emphasize and reduce occlusion. Thus this scheme works best for visualizations of a small number of objects of interest, such as a particular isosurface in a volumetric rendering or small group of objects in a complex polygonal scene.

3.1.2 Continuous importance

For a more complex visualization with potentially many objects of interest, a continuous range of importance values can be used. Objects can be assigned an importance value anywhere in the range from the lowest value, 0.0, to the highest value, 1.0. A continuous range allows for objects of moderate importance: objects that should be visible but less so than those of highest importance, and more so than those of lowest importance. The result is a flexible prioritization of objects in a visualization. Objects should be emphasized and exposed according to their importance, such that the most important objects are the most emphasized and most exposed.

For example, the illustration in Figure 3.1b shows the interior electronics of the DSLR camera in the context of the camera body, with much of the occluding portion of the camera body being rendered using a sparse translucent representation. This occlusion reduction suggests that the electronics are of higher importance than the portion of the camera body suppressed from rendering. However, the front of the lens system is shown completely opaque, and is allowed to occlude the electronics, suggesting that the front of the lens is of higher importance than the electronics. The flexibility of continuously defined importance allows such a multi-level importance relationship, and would be necessary to create automated visualizations in the style of this camera illustration.

Continuous importance is more effective than binary importance with many objects in close proximity to each other, and when these objects can be reasonably prioritized. In these situations the continuous importance provides a variable control over which objects should be made visible, and to what extent. Continuously defined importance also works best with emphasis and exposure techniques that have one or more continuously varying parameters to which importance can be mapped. An emphasis or exposure technique of a binary nature may not be able to fully capture the benefit of a continuous importance definition.

3.1.3 Objects of interest with continuous importance

Emphasis and exposure techniques without parameters to which importance can be continuously mapped lend themselves to the binary importance model. While binary importance allows for visualization of objects of interest, it does not allow any variable control over treatment of objects of secondary interest, forcing these objects to all be treated in the same way. In order to reap the benefits of continuous importance with binary emphasis and exposure techniques, one can create a hybrid importance model.

The binary emphasis/exposure techniques can be applied to the entire scene, exposing objects of primary interest, while an additional set of continuous-capable emphasis/exposure techniques is applied to remaining objects, which can have a separately defined continuous importance. Thus the focus objects can be maximally exposed or emphasized while context objects can be prioritized, being exposed or emphasized according to their relative importances.

The hybrid importance model is shown in practice in Figure 7.1, where an Ultrasound plane is assigned the high binary importance and the volumetric CT scan is assigned the low binary importance. In the volumetric scan, various tissues are prioritized by their importance based on a continuous scale. The Ultrasound plane is then the primary focus of the visualization, while classified tissue is viewed in the periphery, with more prominence given to tissue of high importance.

This hybrid model is most useful when objects of primary importance are logically distinct from all other objects, and these objects of secondary importance are in close proximity to each other but have their own relative importances by which they can be prioritized. Using a single binary importance model would prevent individual objects of moderate importance from being seen, as they could be occluded by nearby objects of lower importance. Using a single continuous importance model would do little to show distinction between the objects of primary importance and the rest of the scene. Both of these problems can be addressed by the hybrid model, more examples of which are shown in Section 7.1.

3.2 Specifying importance

Importance is specified per scene entity, to ensure that each polygon or voxel being rendered can be appropriately handled. While there is no required range of importance values, the convenient standard is to use a real-valued parameter between 0.0 and 1.0, with higher values corresponding to higher importance. When a single model of importance is used (binary or continuous), only one set of importance values need be considered. When the hybrid model is used, two sets of importance values may be used and considered in parallel: a binary value to specify the primary objects of importance, and a continuous value to specify importance for the rest of the scene. Alternately, the hybrid model can be encoded into a single importance value with the range $[0.0, 1.0]$ corresponding to the continuous component and low binary importance, with any values greater than 1.0 denoting a high binary importance.

For polygonal scenes, the scene graph can be used to partition the scene into sets, each of which can be assigned an importance value. For volumetric data, a 1-D or 2-D transfer function can define colors and opacities for groups of voxels to designate "objects" in the scene. This transfer function definition can be expanded to include importance, as described in [Burns et al., 2007]. In either case, the importance value of a polygon or voxel should be defined and made available to rendering algorithms before rendering begins.

3.3 Applying importance

Specifying object-level importance is only the first step in creating an importance-driven visualization. Application of importance values in rendering is what allows images to be tailored to different visualization goals. One can visualize particular objects in a scene by using importance values to control emphasis and exposure techniques. The converse is also true—emphasis and exposure can be used to convey the importance of each object to the viewer.

3.3.1 Emphasis

Controlling emphasis with importance has two benefits: it can attract a viewer's attention to important objects and convey the relative importances of different parts of a scene. Painterly renderings have been created from 2-D images using importance data to control emphasis through brush stroke size, stroke frequency, line density, and color saturation [Hertzmann, 1998; DeCarlo and Santella, 2002; Santella and DeCarlo, 2002]. Non-photorealistic 3-D rendering techniques have used similar mechanisms to express importance and attract a viewer's gaze [Barla et al., 2006; Cole et al., 2006]. Among the options for conveying emphasis in 3-D visualizations are variation in color, shading, or rendering style, based on a polygon or voxel's importance value.

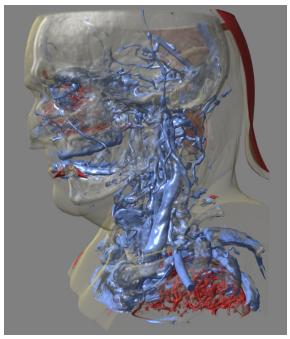
3.3.1.1 Color

The use of color for emphasis is widely used in photography and hand-drawn illustration. One possibility is the use of different hues of color, such as the cool-to-warm color ramp, to denote various importance levels. Another option often found in artistic applications is the use of color saturation. A color c can be desaturated by a constant factor E_c by linearly interpolating between c and the greyscale (intensity) of c, c_i. A color de-emphasized by importance can be calculated by linearly interpolating between the base desaturated color, $c_{desaturated}$, and the original color c.

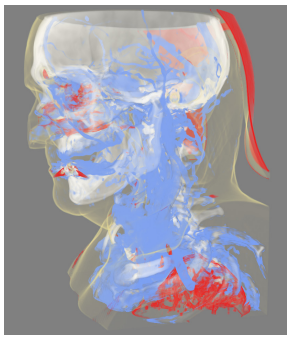
$$c_{desaturated} = c_i * E_c + c * (1 - E_c) \tag{3.1}$$

$$c_{muted} = c * I + c_{desaturated} * (1 - I) \tag{3.2}$$

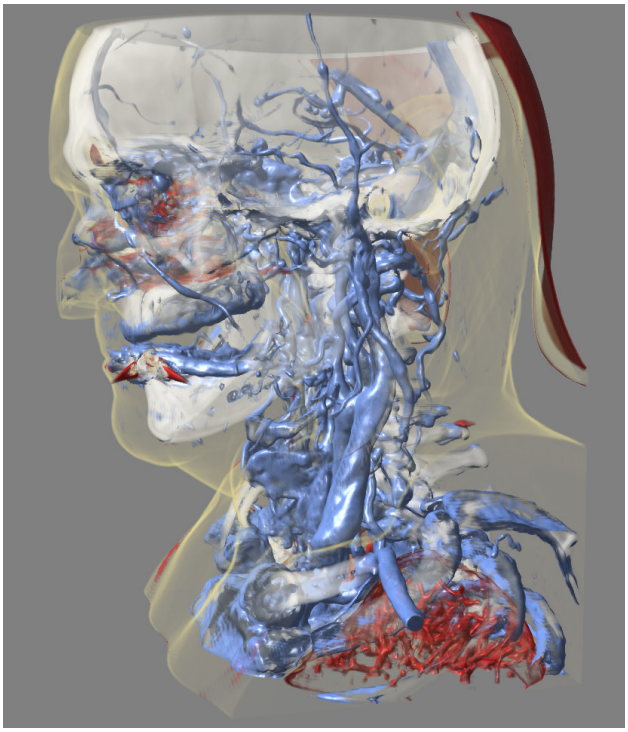
The resulting desaturation proportionately affects unimportant objects, leaving the most important objects normally saturated. The strength of the effect is controlled by the parameter E_c, with a value of 1 causing the least important objects to be completely desaturated, and a value of 0 causing no desaturation anywhere. Important objects are effectively emphasized by de-emphasizing objects of lesser importance.



(a) Normal shading



(b) No shading



(c) Shading according to importance

Figure 3.2: Rendering with emphasis and exposure derived from importance. (a) shows a normally shaded rendering while (b) shows a completely unshaded rendering. Shading by importance combines elements from both of these images, such that the skin (yellow) is unshaded, bone (white) is partially shaded, and vasculature (red, blue) is normally shaded. In addition, the skin is rendered with edge highlights (Section 2.2) to expose the underlying materials.

3.3.1.2 Shading

Shading is used to help convey surface orientation and detail in renderings. Like emphasis using color, emphasis can be made using shading by de-emphasizing objects with low importance by suppressing shading color contributions. A partially shaded color can be calculated as a linear interpolation of a shaded color c_{shaded} and the base color c based on a constant factor E_s. A color de-emphasized by shading can be calculated as a linear interpolation of the partially shaded color and the normally shaded color based on importance.

$$c_{partial} = c * E_s + c_{shaded} * (1 - E_s) \tag{3.3}$$

$$c_{suppressed} = c_{shaded} * I + c_{partial} * (1 - I) \tag{3.4}$$

The strength of the effect is controlled by the parameter E_s, with a value of 1 causing the least important objects to be completely unshaded, and a value of 0 causing all objects to be fully shaded. This technique is used extensively in Section 7.1, and is shown in Figure 3.2.

3.3.1.3 Rendering styles

Emphasis (and de-emphasis) can also be accomplished using special rendering styles. Importance can be used to select individual non-photorealistic surface shading models to emphasize and differentiate between materials of different priorities [Bruckner and Gröller, 2007].

Binary importance can be used to selectively enable rendering styles that naturally emphasize important objects. For example, rendering contours or silhouettes around an object can make it stand out from the rest of the scene. Mapping line rendering to binary importance would effectively outline objects of primary importance in a visualization. Alternatively, if line rendering is performed everywhere in a scene, one can omit surface shading and color in areas of low importance to reduce the appearance of unimportant materials to line drawings. This effect would be very similar to that used by the artist illustrating the engine in Figure 4.2b.

Continuous importance can also be mapped to individual parameters of rendering styles to selectively emphasize objects. For example, continuous importance could be mapped to the stylized focus parameter in the work of Cole et al. [2006] to control line density and texture, and color saturation and contrast, to direct gaze to important objects. Continuous importance could also be used to control how much high frequency detail is shown in objects rendered with Exaggerated Shading [Rusinkiewicz et al., 2006], given enhanced detail to objects of high importance and suppressing detail in objects of low importance.

3.3.2 Exposure

Any objects can be difficult to visualize when occluded, so objects of interest to a visualization should be exposed. Selectively rendering only the most important objects in a scene would be simple, omitting all objects of low or no importance, but such omission would discard the

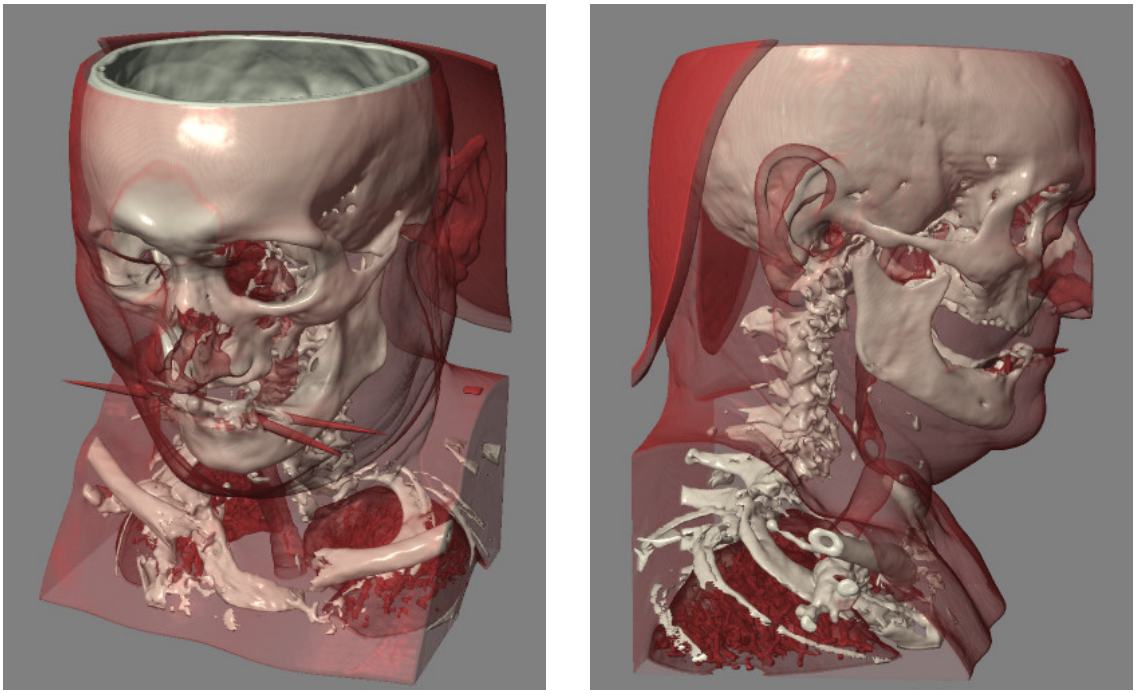


Figure 3.3: Selective occlusion reduction using translucency based on importance. The bone isosurface (white) is given the high importance value (1) and rendered with $R = 0$. The skin and airway isosurface (red) would ordinarily occlude the bone surface but is assigned the low importance value (0) and rendered with high R. As a result, occlusion reduction of the skin via translucent rendering allows the skin to remain seen while exposing the bone underneath.

contextual information provided by these "unimportant" objects. To preserve the contextual information provided by objects of secondary importance, one can selectively apply the occlusion reduction strategies introduced in Chapter 2 to these objects, reducing the occlusion penalty of their inclusion in a scene and facilitating the exposure of important objects.

Each of the described occlusion reduction strategies can be controlled by importance, though some differ in their applicability to binary and continuous importance definitions.

3.3.2.1 Binary importance

The simplest application of importance to exposure occurs when binary importance is considered. In this scenario, one can simply render objects of high importance normally, and apply an occlusion reduction strategy with a high strength ($R \approx 1$) to objects of low importance. With translucent rendering (Section 2.1), objects of low importance are rendered with a low opacity

value, as shown in Figure 3.3. With edge highlights (Section 2.2), objects of low importance are rendered with edge highlights enabled, and possibly with a high fall-off parameter, as shown in Figure 3.2. With lines (Section 2.3), objects of low importance are rendered with lines in place of their surfaces, as shown in Figure 3.1a.

3.3.2.2 Continuous importance

The use of continuous importance allows more flexibility in the prioritization of objects when used by occlusion reduction methods. Importance can be mapped to the reduction strength, R, such that high importance corresponds to low occlusion reduction and low importance corresponds to high occlusion reduction. One simple mapping with such an inverse relationship is $R = (1 - I)$. With translucent rendering, objects of lowest importance are drawn faintly and objects of higher importance are drawn with more strength (and with more ability to occlude). With edge highlights, a similar effect occurs while objects of lower importance retain their edges, as seen in Figure 3.2. With lines, surface rendering is suppressed for objects of all but the highest importance, with lines fading as importance decreases.

3.3.2.3 Varying exposure

Continuous importance is one way to variably apply occlusion reduction, but because importance is independent of scene position, occlusion reduction based solely on importance is applied globally to a scene. Global application of occlusion reduction is satisfactory for some applications, such as entire anatomical systems (e.g. skeletal system, nervous system) inside the context of an entire human body. Because the object of interest, the anatomical system, spans nearly the entire model, it is necessary to reduce the occlusion power of unimportant materials nearly everywhere in the model to provide view of the object of interest. In other applications, when the objects of interest have small footprints compared to the overall model, such as in Figure 1.1, it may be more desirable to localize occlusion reduction, and thereby exposure, around the object of interest. This localization allows the bulk of the scene to be rendered with a conventional appearance, preserving

context. The following chapter describes one such method of localizing exposure based on binary or continuous importance.

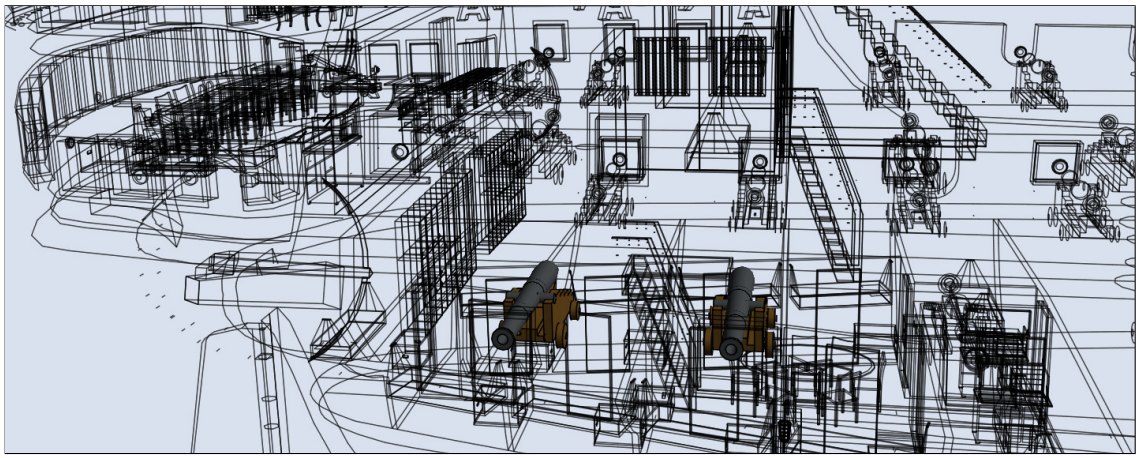
Chapter 4

Preserving Context: Localizing Exposure with Cutaways

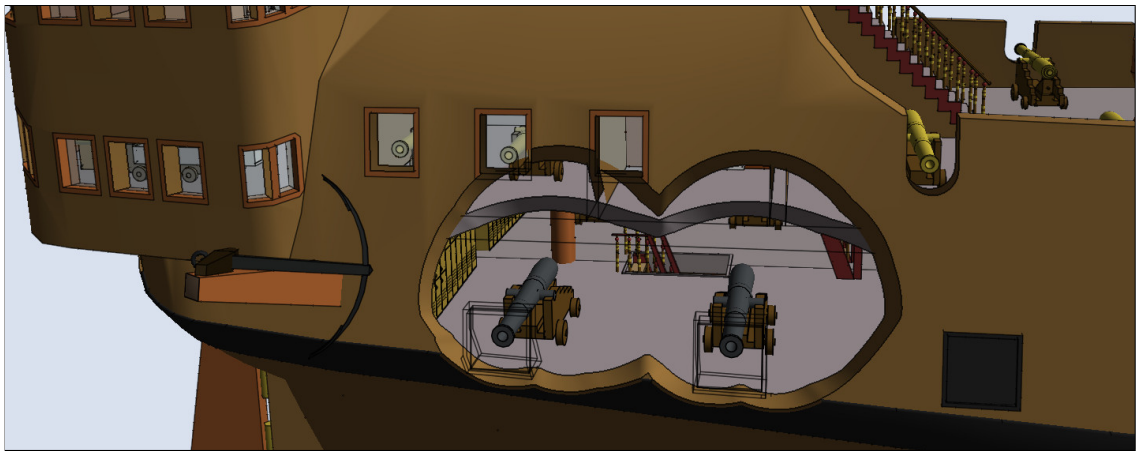
The methods introduced in Chapter 2 for reducing occlusion have thus far been applied uniformly to entire objects based on their importance, as described in Chapter 3. This uniform application allows important objects to be exposed by reducing the occlusion footprint of less important objects regardless of their position in the scene and in 3-D space. The end result is a visualization where objects of interest are shown surrounded by a sparse representation of the rest of the scene. For complex scenes with a large number of objects, the high overlap of sparse representations of context objects can be difficult to parse (Figure 4.1b). To aid in the overall view of the scene, occlusion effects can be preserved in areas of the image where occlusion reduction is not necessary for exposure of objects of interest (Figure 4.1c). This localization of exposure creates images that resemble technical illustrations known as sections or cutaways, such as those shown in Figures 1.1 and 4.2.

Previous research in localized exposure has yielded several visual styles applied in a variety of ways. The earliest examples used depth buffers and image processing to replace occluding surfaces with semi-transparent representations [Feiner and Seligmann, 1992]. More advanced techniques incorporate the specification of cutaway geometry, such as intersecting half-spaces

(a) Normal rendering

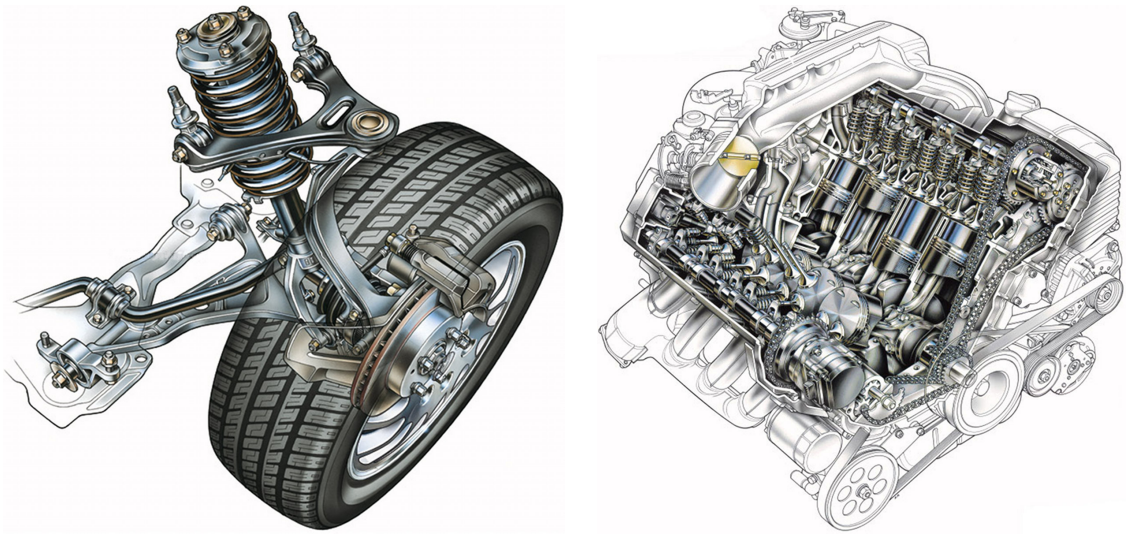


(b) Global application of occlusion reduction



(c) Localized application of occlusion reduction

Figure 4.1: Global and localized application of exposure techniques. (a) The hull of this ship occludes interior objects using normal rendering. (b) Two cannons are exposed by globally applying occlusion reduction to the rest of the scene. The resulting image is almost unrecognizable from the normal rendering, with overlapping lines creating visual clutter. (c) By localizing exposure techniques around the cannons, most of the ship retains a normal appearance.



(a) (b)

Figure 4.2: Artist-drawn illustrations using localized exposure techniques. (a) A car wheel is shown, with a focus on the mechanical brake and suspension components. Like Figure 3.1a, the wheel and tire serve as context for the illustration. Here, exposure is localized to preserve normal rendering in areas of the wheel that do not occlude the mechanical components. (b) An engine is shown cut away to reveal its interior. In addition to localizing exposure to show exterior structure of the side of the engine, emphasis of the interior components is performed by using a simpler rendering style outside the region of interest.

Images used with permission, Copyright Kevin Hulsey Illustration, Inc.

or an inflated convex hull [Diepstraten et al., 2003]. Some techniques, such as VolumeShop [Bruckner and Gröller, 2005] and ClearView [Krüger et al., 2006], are essentially powerful illustration tools, relying on user specification of a 2-D or 3-D focus area to be exposed by cutaway or ghosting. Other manual techniques focus on mimicking real cutting tools, such as those used by a surgeon [Liang et al., 2005]. More automatic techniques use importance information from the scene to automatically suppress or fade occluding materials [Krüger et al., 2005; Straka et al., 2004; Bruckner et al., 2006] or create cutaway illustrations [Viola et al., 2005; Viola et al., 2006].

The techniques presented in this chapter are closest in approach and image appearance to those of Viola et al. and Li et al. [Li et al., 2007]. The presented techniques create a cutaway surface similar to that of Viola et al. but with more control over cutaway shape and a more flexible cutaway structure, while enjoying faster performance and higher fidelity due to an improved computation algorithm. The resulting images are similar in spirit to those of Li et al. but are made

with techniques that do not require preprocessing, working with rapidly changing viewpoint and dynamic animated scenes.

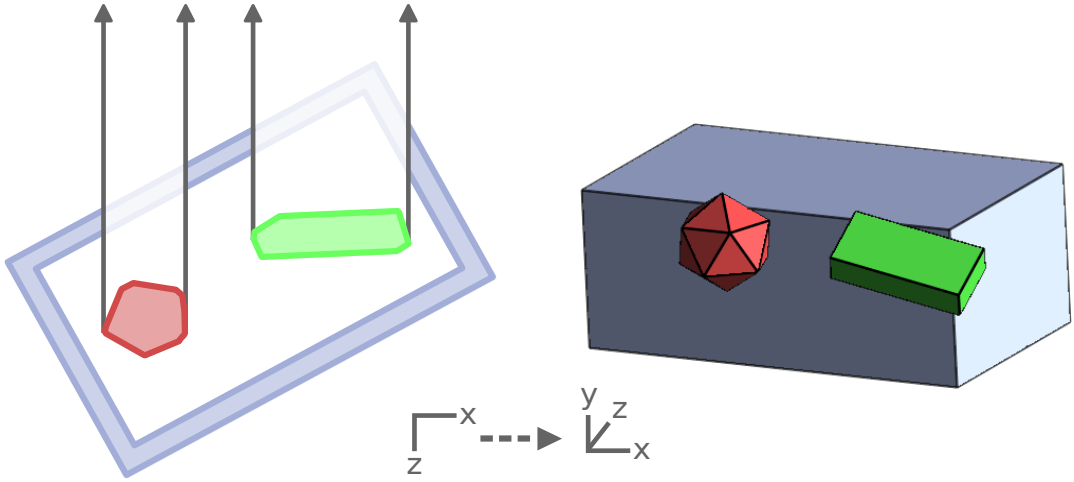
The remainder of this chapter examines how cutaway illustrations can be created by defining cutaway structures that define levels of occlusion for contextual objects (Section 4.1) and using these occlusion levels to selectively reduce occlusion and ensure exposure of objects of interest (Section 4.2). Included in this chapter are novel definitions of a contextual cutaway structure and an associated occlusion function.

4.1 Measuring occlusion

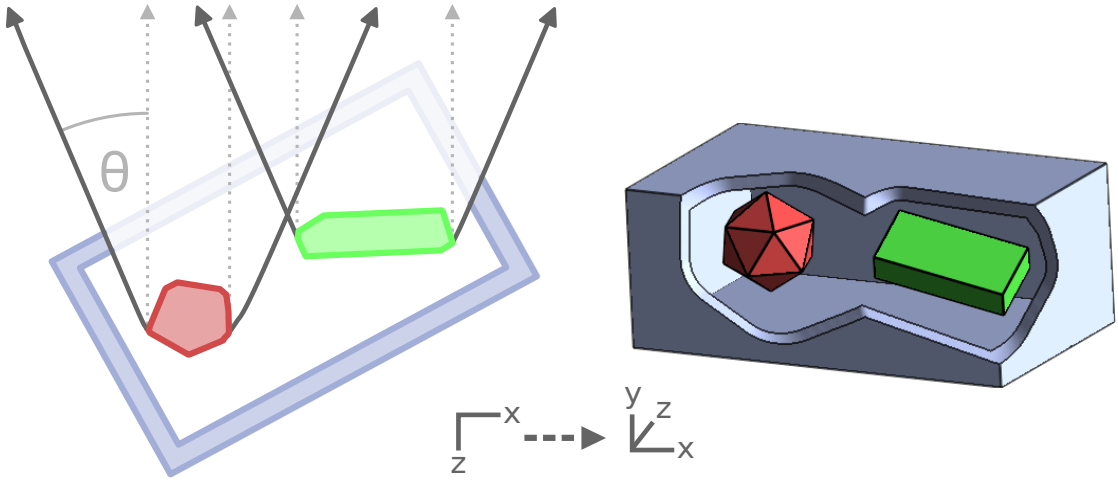
At the core of the cutaway illustration is the *region of interest* to be exposed. In practice, this region contains or is defined by the geometries of objects of interest to the visualization. For the purpose of this discussion, the region of interest can be considered an arbitrary 3-D region of space.

A minimal definition of occlusion space will contain the volume of space between the region of interest and the camera position. From the point of view of the camera, this volume is closer than the region of interest and within the 2-D projection of the silhouette of the region of interest. Any material rendered in this volume is guaranteed to occlude the region of interest to some degree.

By excluding only materials from this volume from rendering, the region of interest is guaranteed to be exposed, as shown in Figure 4.3a. While this exclusion accomplishes the primary goal of exposure, it does so with limited comprehensibility, since it can be difficult to determine the depth of the region of interest relative to depths of the remaining materials. This difficulty occurs because the boundary of the occluding volume is aligned with the viewing rays, causing occluding materials close to the camera to be cut at the same screen-space positions as occluding materials further from the camera. Consequently, materials in the periphery of the region of interest nearest the camera occlude all intermediate layers of material between the camera and the region of interest. The visual absence of intervening layers of material in turn causes the difficulty in depth distinction between the region of interest and objects nearest the camera.



(a) Minimal occluding volume



(b) Expanded occluding volume

Figure 4.3: Diagram of minimal and expanded occluding volumes. (a) When only the minimal occluding material is removed, objects of interest are made visible, but invisible depth discontinuities make them appear in front of their enclosing box. (b) By shaping the wall of the occluding volume outward an angle θ from the view vector, the depth change is made clear, and a cutaway illustration is created.

In order to prevent intermediate layers from occluding each other at the boundary of the occluding volume, the volume must be changed so its boundary is not incident to the viewing rays. By expanding the volume as it approaches the camera, as shown in Figure 4.3b, the boundary of the occluding volume becomes visible to the camera, showing intermediate layers of material that provide valuable context around the region of interest. The boundary of this occluding volume is known as the *cutaway surface*, as it is the surface at which materials are cut away.

4.1.1 Cutaway surface

The shape of the expanded occlusion volume is equivalent to the volume of the union of all possible cones of angle θ with tips anchored on the surface of the region of interest and aligned along the viewing rays. The cutaway surface defined by this volume is thus dependent on the current viewing direction and the angle θ, which becomes the minimum angle between the boundary of the occluding volume and the viewing rays. When $\theta = 0^\circ$, the occluding volume degenerates into the minimal occluding volume shown in Figure 4.3a. As θ increases up to a maximal 90°, more material is included in the occluding volume and the boundary of the occluding volume becomes less perpendicular (and more visible) to the camera. Increasing θ can aide in the visualization of occluding layers, at the expense of removing more contextual material.

In additional to the geometric description above, it is possible to define the cutaway surface in terms of a cutaway surface function C defined over a rendered image plane under an orthographic projection. This definition begins with a rendered depth image I of the rear-hull of the region of interest, which covers some subset R of pixels in I. For each pixel q in R, the function considers a cone anchored at depth $z(q)$ along the z-axis and making a slope m to the xy image plane. For a cone defined with angle θ, this slope is $m = \tan^{-1}\theta$. The depth of this cone at a pixel p in I is calculated as the slope m multiplied with the distance between q and p and subtracted from the depth of the cone's origin at q. The conical surface function c, which gives the depth at p of the cone defined at q with angle θ, is defined as follows:

$$c(p,q) = z(q) - m\|q-p\| \tag{4.1}$$

Since the cutaway surface volume is geometrically defined as the union of volumes of all such possible cones arising from R, the cutaway surface in an image plane is equivalent to the rear-hull of this union of cones. Thus the depth of the cutaway surface at a pixel p can be calculated as the maximum depth at p of all cones emanating from pixels q in R. The cutaway surface function C is then defined as follows:

$$C(p) = \max_{q \in R} c(p,q) = \max_{q \in R} (z(q) - m\|q - p\|) \tag{4.2}$$

where R is the subset of pixels in I covered by the region of interest, $z(q)$ is the rendered depth of the rear-hull of the region of interest at pixel q, and $m = \tan^{-1}\theta$.

4.1.2 Simple cutaway structure

The cutaway surface can be used to define a structure in which occlusion reduction techniques can be applied to generate cutaway illustrations. The simplest version includes a single cutaway surface, which naturally establishes two complementary regions, one on either side of the surface, that span all of 3-D space.

4.1.2.1 Base region

The region behind the cutaway surface can be rendered with conventional rendering techniques as it can never occlude the region of interest. Because this region provides a contextual base for the region of interest, it is called the *base* region. Because occlusion reduction does not occur in the base region, and occlusion reduction of the region of interest should not occur, the region of interest is included in the definition of the base region.

4.1.2.2 Clear region

The complement of the base region is the *clear* region, which lies in front of the region of interest and the cutaway surface, containing the camera and the expanded occluding volume. This region

should be cleared of occluding material to allow maximal view of the region of interest and to expose any intermediate layers along the edge of the expanded occluding volume.

4.1.2.3 Occlusion function

An *occlusion function*, Ω, can be defined within a cutaway structure to denote the degree to which a point in space occludes the region of interest, and to influence how materials are rendered to expose the region of interest. The simple cutaway structure includes only two regions to supplement the region of interest, and thus encourages a binary definition of occlusion. A straightforward definition of occlusion within the simple cutaway structure is $\Omega = 1$ inside the clear region and $\Omega = 0$ inside the base region. Because the occlusion function is uniform within each of the clear and base regions, occlusion reduction based on this function will be applied uniformly.

To create a simple cutaway rendering, it suffices to reduce occlusion in the clear region to expose the region of interest while preserving normal rendering of the base region. Thus occlusion reduction methods (such as those described in Chapter 2) can be applied for a point in space according to the value of the occlusion function, where a value of 1 causes full occlusion reduction and a value of 0 means no reduction.

4.1.2.4 Simple cutaway characteristics

The simple cutaway structure when used with simple occlusion reduction exhibits the following valuable characteristics:

- The surface is view dependent so portions of secondary objects that occlude the region of interest can be detected, and discarded.
- Portions of secondary objects can be discarded in varying amounts depending on their z-distance from the region of interest, allowing viewing of nested structures.

- The amount of material discarded can be controlled by the angle θ of the cutaway as desired, providing a balance between the amount of material removed and the ability to locate objects within nested structures.
- The shape of the cutaway hugs the silhouettes of the region of interest, reducing the amount of material that has to be removed in order to expose the region.
- Multiple regions of interest can define a single cutaway surface, with the cutaway surface exposing all objects of interest together.

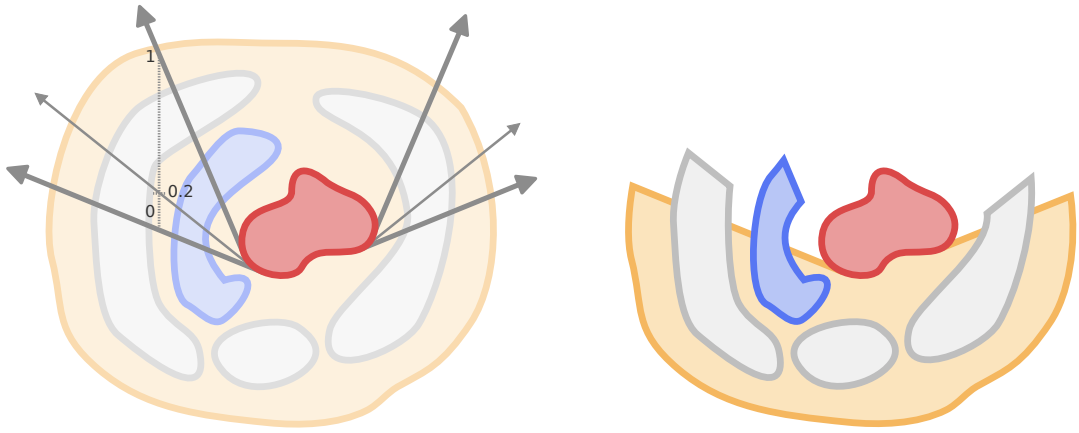
In visualizations of scenes with a binary definition of importance, the binary importance matches up nicely with the binary treatment of occlusion reduction, providing a simple set of visualization options and allowing the creation of basic cutaway illustrations. However, in scenes with more than a binary definition of importance, a more complex cutaway structure can be created to provide more visualization options and different levels of treatment for materials of various levels of importance.

4.1.3 Contextual cutaway structure

When materials can be classified with more than a binary notion of importance, the cutaway structure can be expanded with additional regions to facilitate different treatments for different levels of importance. This section describes two such additional regions, a *transition* region between the clear and base regions, and an *overlay* region within the clear region.

4.1.3.1 Transition region

In the simple cutaway structure, all materials outside the region of interest are treated equally, being trimmed away according to the single defining cutaway angle, θ. This angle can be varied from image to image, to trim away more or less material as desired. However, in situations where a continuous definition of importance exists, materials outside the region of interest may have multiple values of importance. In this case, it may be useful to reduce the occlusion of materials



(a) (b)

Figure 4.4: Diagram of cutaway surface interpolation. Note the camera is looking down from above in this cross section view. (a) Intermediate cutaway surfaces can be defined by interpolating depths between two cutaway surfaces. An interpolation factor of 0.2 defines a third cutaway surface near the outer surface. (b) Blue, grey, and orange material are removed according to the inner, middle, and outer surfaces, respectively. The object of interest (red) is completely exposed in the center while portions of blue and grey objects are exposed in the periphery. Results based on this interpolation in 3-D are shown in Figure 7.1.

based on their relative importance, as in Section 3.3.2. Such variable occlusion reduction could allow relatively important materials outside the regions of interest to be made visible even if they are completely occluded by materials of lesser importance in the base region.

One option for this variable occlusion reduction is to trim away materials at multiple angles, with materials of low importance being trimmed away more than materials of high importance. This variable trimming can be accomplished by expanding the boundary between the clear and base regions to span the volume of space between two cutaway surfaces with different characteristic angles. The cutaway surfaces with the widest and narrowest angles of those to be used to trim materials define the boundary of this transition region. Since an infinite number of angles exist between these bounding angles, there are an infinite number of potential cutaway surfaces in this region. A brute force solution that calculates the exact position of each cutaway surface to be used for trimming could be prohibitively expensive. Instead, a comparable effect can be achieved by calculating positions of only the two bounding cutaway surfaces and interpolating the values for surfaces in between. A 2-D simplification of this technique is illustrated in Figure 4.4.

Because the 2-D cutaway surfaces are linear, two surfaces with angles θ_1 and θ_2 can be represented as lines with slopes m_1 and m_2. Lines with slopes between m_1 and m_2 can be found by a linear interpolation of the heights y_1 and y_2 of the two surfaces along the x-axis. The slope of a given line will depend on the interpolation factor between the heights y_1 and y_2. For example, with an interpolation amount of 0.5, which is the equivalent of averaging the heights y_1 and y_2, the slope will be exactly the average of m_1 and m_2. By varying the interpolation amount between 0 and 1, any surface with slope between m_1 and m_2 can be determined. This property extends to 3-D, in an orthographic project, with the xy plane replacing the x-axis and heights measured along z replacing heights measured along y.

By varying the interpolation amount between the depths of two cutaway surfaces with angles θ_1 and θ_2, the depths of intermediate cutaway surfaces with angles between θ_1 and θ_2 can be calculated. Conversely, for a given point in the transition region, the interpolation factor between the two cutaway surfaces can be determined. This interpolation factor can be used to define a continuous occlusion function within the transition region, as described in Section 4.1.3.3, which can in turn be used to apply varying levels of occlusion reduction to materials of varying levels of importance.

4.1.3.2 Overlay region

Further differentiation in rendering treatments can be made in the area immediately in front of the region of interest, in the overlay region. This region sits in front of the base and transition regions and behind the clear region. It is bounded by the cutaway surface with angle θ_1 and the same cutaway surface uniformly offset a constant distance d towards the camera.

It allows materials of highest importance that may not be part of the region of interest to be treated specially, for example being allowed to partially occlude the region of interest. The thickness of the region should be adjusted according to the density of important structures preserved in the overlay region and the goals of the visualization. As d increases, so does the thickness of the region, along with the potential for material in this region to occlude important features in the

region of interest. As such, small values of d provide the best trade-off between showing material in the overlay region and exposing material in the region of interest.

4.1.3.3 Occlusion function

With the addition of the transition and overlay regions, the binary occlusion function described in Section 4.1.2.3 can be expanded to cover these regions with continuous values. The values $\Omega = 0$ and $\Omega = 1$ are preserved for the base and clear regions, respectively, with Ω smoothly varying in between.

A value t represents the value of Ω at the *transition-overlay* boundary. When both a transition and overlay region are present, t is arbitrarily declared 0.5. In the absence of a transition region, t should be 0. In the absence of an overlay region, t should be 1.

Within the transition region, the values increase linearly from $\Omega = 0$ to $\Omega = t$ according to the interpolation factor between m_1 and m_2, as described in Section 4.1.3.1. Within the overlay region, Ω varies linearly from t to 1, according to the distance along the viewing ray from the start of the overlay region and the thickness d. The specific mechanism for utilizing the occlusion function to reduce occlusion in the contextual cutaway structure is described in the following section.

4.2 Revealing objects of interest

Given the measure of occlusion provided by the occlusion functions described in Section 4.1, objects in the region of interest can be exposed by applying occlusion reduction techniques from Chapter 2 to materials outside the region of interest according to occlusion values where they are rendered. The resulting images can all be considered cutaway illustrations, as they show material being cut away to expose the region of interest. Aside from the occlusion function, there are several important components that can be varied to control the effect of these cutaway illustrations. These variations include the specific mechanism by which occlusion values influence occlusion reduction, and the geometric definition of the region of interest.

4.2.1 Applying occlusion values

Values of the occlusion function can be interpreted as the amount of occlusion reduction needed at a given point to expose the region of interest via a cutaway. For example, occlusion reduction should be performed most in the very interior of the cutaway, in the clear region, where the highest occlusion values are. In the periphery, the base region, the occlusion value is zero and no occlusion reduction is necessary.

Occlusion reduction techniques can be performed with a strength R, as described in Chapter 2, where $R = 1$ implies maximal occlusion reduction and $R = 0$ implies no reduction. In order to take into account the cutaway structure in occlusion reduction, R should be derived from both the value of the importance function I (as described in Chapter 3) and the value of the occlusion function Ω.

In general the function should generate high values of R for the combination of high Ω (occlusion) and low I (importance), preventing materials of low importance from occluding the region of interest. Of course the function should also generate low values of R for the combination of low Ω and high I, leaving materials of high importance visible in certain areas of the cutaway structure. The exact mapping of Ω and I to R can have a significant effect on the visualization and should be determined based on the visualization goals. This mapping is further analyzed and shown in results in Chapter 7.

4.2.2 Region of interest

The primary goal of cutaway illustrations is to expose objects of interest that might normally be occluded. This goal can be achieved by exposing a well-defined region of 3-D space that encompasses the objects of interest, referred to as the region of interest. The heart of the cutaway illustration, the shape and extent of the region of interest has a strong influence over the effectiveness of exposing the objects of interest. Given a set of objects of interest, there are several options for constructing the region of interest that contains them.

4.2.2.1 Object geometry

A simple but effective definition of a region of interest is the region of space occupied by the objects of interest. Its simplicity arises from the fact that the object geometries are readily available, and can be directly used to seed algorithms that calculate the cutaway surface. Effectiveness is ensured because any exposure of the region of interest guarantees exposure of the objects of interest.

This definition is the minimal definition for the region of interest, and results in the minimal amount of material being removed in order to expose the objects of interest, compared to other definitions of the region of interest with the same cutaway parameters. This property is seen as the shape of the exposed region closely follows the shape of the objects being exposed, since the wall of the cutaway extends from the silhouettes of the objects of interest. As a result, only the objects of interest are exposed, while nearby objects or materials may remain occluded or be cut away.

4.2.2.2 Proxy geometry

For visualization of multiple objects of interest in close proximity, the region of interest can be expanded from the minimal definition to include space around and between the objects of interest. This expansion can be carried out by computing a proxy geometry from the objects of interest and using the proxy geometry as the basis of the cutaway shape. This proxy geometry is only rendered as depth to seed the cutaway surface algorithms but is not rendered in the final image. The proxy geometry may be computed using several different methods, depending on the visualization scenario.

Convex hull The convex hull of a set of objects of interest may be used as the region of interest in order to expose all the objects together while creating a smoother cutaway shape. Because the convex hull of objects is geometrically smoother and has fewer features than the geometries of the individual objects, a cutaway surface based on the convex hull will be smoother and have fewer

sharp features than a cutaway based on the geometries of the objects of interest alone, resulting in fewer abrupt changes under animation and view change. However if objects of interest are too far apart, the convex hull can be quite large, resulting in a large amount of material removed in order to expose the region of interest. Thus the convex hull should only be used for objects of interest in relatively close proximity, with few other objects or materials in between them.

Specialized geometry In certain visualization scenarios it is possible to compute a proxy geometry more appropriate to the visualization goal. For example, in architectural visualization one may wish to view the contents of an entire room. In this case, it is beneficial to determine the entire volume of the room and use the resulting geometry as the region of interest. By using the entire span of the room as the region of interest, all objects in the room become visible, and the entire room can be seen in the context of the building. This approach is explored in practice in Section 7.2.

Chapter 5

Implementing Realtime Cutaways: Rendering Techniques

The mechanisms described in Chapter 4 can be used to create cutaway renderings in real-time through the combination of efficient algorithms for computing cutaway surfaces with conventional real-time rendering techniques (e.g. OpenGL, DirectX). Such real-time renderings allow for interactive visualization applications for exploring structurally complex 3-D scenes, as shown in Chapter 7. This chapter details several cutaway surface computation algorithms that can be applied to interactive applications. It also describes extra considerations and modifications necessary to generate full-featured cutaway illustrations using polygonal rasterization or volumetric rendering. Included in this chapter are a novel image-based cutaway surface computation algorithm, new modifications to the polygonal rendering pipeline for rendering cutaway images, and cutaway surface function modifications to support perspective projection correction and temporal coherence.

5.1 Cutaway surface computation

The cutaway surface described in Section 4.1.1 is the basis for the cutaway structures around which cutaway illustrations can be generated. This surface is shaped to expose a region of interest to the camera by carving out materials in front of the region of interest at some angle from the viewing direction. A single cutaway surface never overlaps itself from the point of view of the camera, and as such can be represented as a depth function parameterized by the viewing rays in the scene. In a conventional rendering pipeline, the surface can be represented by a depth image or depth buffer aligned with the rendered image.

In theory the depth image can be created through geometric construction of the cutaway surface and subsequent rendering of the geometric surface into a depth buffer, or directly through evaluation of the cutaway surface function C (equation (4.2)) at every pixel in a depth image. Previous work that generates similarly shaped cutaways [Viola et al., 2004] takes the latter approach, foregoing geometric construction of the surface. This direct approach allows the generation of a suitable depth image for use in cutaway rendering while skipping the potentially complex and costly steps of constructing the geometric surface defined in Section 4.1.1.

5.1.1 Cutaway function

Previous work [Viola et al., 2004] used a chamfer distance transform approximation [Borgefors, 1986] to create a depth image of a basic cutaway surface. The implementation requires rendering the back hulls of the region of interest into a depth buffer, reading this buffer to the CPU to be processed by the chamfer approximation algorithm, then writing the buffer back to video memory.

The chamfer distance transform on a binary image B approximates a Euclidean 2-D distance transform, which determines the minimum distance d at each pixel p to an active (white) pixel q in B:

$$d(p) = \min_{q \in B} \|q - p\| \tag{5.1}$$

Note that minimizing this function is equivalent to maximizing the following function, where d_{max} is a maximum value for d, and $d(p) = d_{max}$ when p is an active pixel in B:

$$d(p) = \max_{q \in B} (d_{max} - \|q - p\|)$$

By using a floating point input image I initialized with variable depth values $z(p)$ instead of d_{max}, and scaling the chamfer kernel by m, the chamfer algorithm maximizes the following function:

$$d(p) = \max_{q \in I} (z(p) - m\|q - p\|) \tag{5.2}$$

Because the depth values in the resulting image decrease linearly with slope m from their anchor points, the surface defined by the image resembles the rear hull of cones with angle $\theta = \tan m$ anchored along the z-axis and positioned over each pixel at the depth $z(p)$. This shape is also precisely the shape of the cutaway surface described in Section 4.1.1. Accordingly, this function matches the definition of the conical surface function c, equation (4.1), and the cutaway surface function C, equation (4.2).

5.1.2 Computation algorithms

The first step of computing an image-based cutaway surface is to define the region of interest. To ensure that the forward-facing cutaway surface reflects the shape of the entire volume of the region of interest, the rear-hull of the region is used to compute an initial seed depth image. The rear-hull can be computed by rendering the geometry of the region of interest into a depth buffer with depth testing set the opposite of normal, favoring far values. In OpenGL this operation can be done using the greater-than test.

Once a depth image has been seeded with the region of interest, each seed pixel can be considered a definition for separate occluding volumes, giving rise to one occluding volume per pixel. Considering each pixel as a point, the occluding volume for each pixel is a cone anchored at the pixel's center with angle θ, aligned along the view vector. An occluding volume for the entire region can be constructed by taking the union of cones defined at each pixel. The cutaway surface for this volume is then the rear-hull of the union of cone surfaces. At a given pixel p, the

depth of the rear-hull is the maximum of the conical surface depth function $c(p,q)$ for each pixel q in the seed image. This value is computed by the cutaway surface function C (equation (4.2)).

There are several ways to compute C over a seeded depth image, the brute force approach, the Chamfer approximation, and a GPU-friendly algorithm based on Jump Flooding [Rong and Tan, 2006].

5.1.2.1 Brute force

The basic brute-force approach offers an exact solution by evaluating $c(p,q)$ at each pixel p for every pixel q in the seed image to find the maximum (furthest) depth. For an n-by-n image this approach requires $O(n^4)$ operations, making it prohibitively slow for real-time application with usable image resolutions.

5.1.2.2 Chamfer approximation

A very fast alternative to the brute-force approach is to calculate the Chamfer approximation [Borgefors, 1986] of the 2-D distance transform of the seed depth image. The algorithm operates in two passes on the image, the first from the top left, left to right, top to bottom, and the second in reverse order. At each pixel p a small k-by-k kernel K of fixed distances is used to find the minimum distance from p to the already-processed pixels covered by K. By operating sequentially on each pixel, distances are propagated downward in the first pass, forming half of the cutaway surface, and propagated upward in the second pass, completing the other half of the cutaway surface. By scaling the distances in K, the slope of the cutaway surface can be increased or decreased, to match the desired θ. Commonly used values for k are 3, 5, and 7.

The Chamfer algorithm has the advantage of running in $O(n^2)$ time for an n-by-n image, but also introduces visual artifacts as a result of the approximation. These starburst artifacts can be seen in Figure 5.1, where the approximation generates triangular features to approximate the smooth cutaway surface. Because the starburst shapes are aligned with the image and not the

objects, they are especially visible under changing camera angle. The approximation improves with larger kernels, but at the cost of increased computation time.

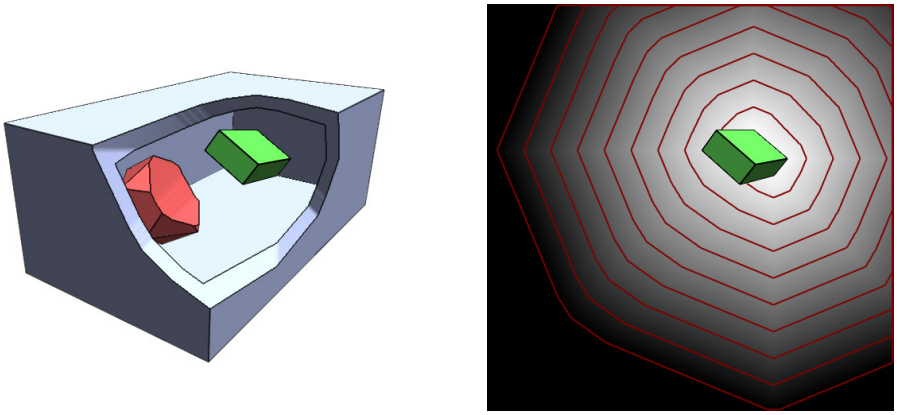
Another disadvantage of the Chamfer algorithm is its serial data dependency. Since distances are propagated from one pixel to neighboring pixels, a given pixel can't be processed until all dependent neighboring values in K are processed. This dependency limits the ability of the Chamfer algorithm to operate efficiently in the massively-parallel GPU environment. As a result, implementations are generally carried out on the CPU, forcing a transfer of the depth buffer from the video memory to the CPU main memory for Chamfer processing, and then back to the video memory for cutaway rendering. These memory transfers can further decrease performance, depending on the memory bandwidth available.

5.1.2.3 Jump flooding

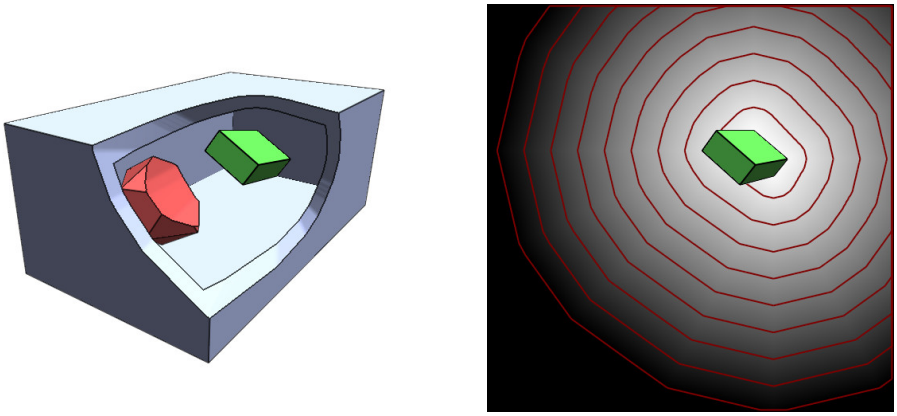
Jump flooding, as presented by Rong and Tan [Rong and Tan, 2006], is a GPU-based version of a parallelizable algorithm for calculating exact 2-D distance transforms by minimizing Euclidean distance (equation (5.1)). This algorithm operates in $O(n^2 \log n)$ time, by performing $\log n$ passes over an n-by-n image. By replacing the 2-D distance function with the conical surface depth function (equation (4.1)) in the jump flooding algorithm, the algorithm can be used to calculate the cutaway surface function. As this algorithm can operate on the GPU, performance is significantly faster than that of the Chamfer approximation, while producing a surface free of approximation artifacts.

Computation begins by rendering the region of interest into a buffer, like the previous algorithms, augmenting stored depth with screen-space coordinates at each pixel. The "greater than" depth function is used so fragments furthest from the camera, comprising the rear hulls, are stored. For an n-by-n image, the algorithm executes $\log n$ iterations over the buffer, with a single per-iteration offset parameter $k = n/2^i$ for iteration i.

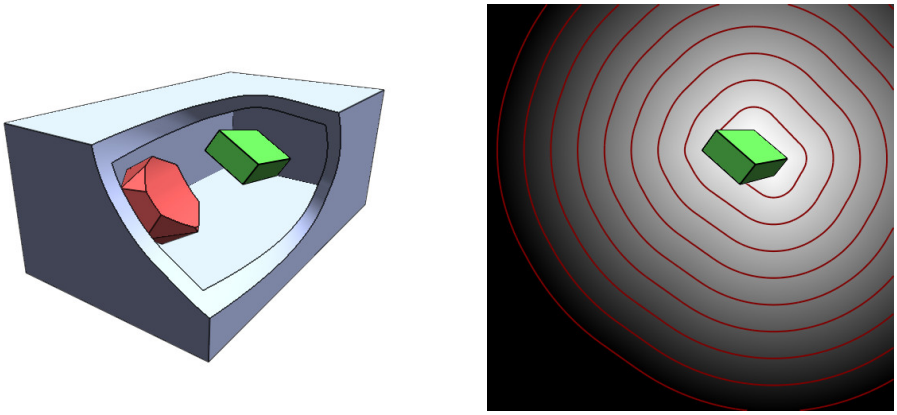
For each iteration and pixel p, the stored depth and coordinates at p, which are the maximum previously found depth and its coordinate location, is compared to $c(p,q)$ evaluated for the 8



(a) Chamfer approximation with 3-by-3 kernel



(b) Chamfer approximation with 5-by-5 kernel



(c) Ground truth calculation

Figure 5.1: Cutaway illustrations and matching cutaway depth images for several calculation methods. The Chamfer approximation with a 3-by-3 kernel in (a) has starburst artifacts that manifest in the cutaway as planar areas. With a 5-by-5 kernel, the approximation in (b) is less jagged, at the cost of increased computation time. A more precise evaluation of the cutaway function in (c) appears completely smooth.

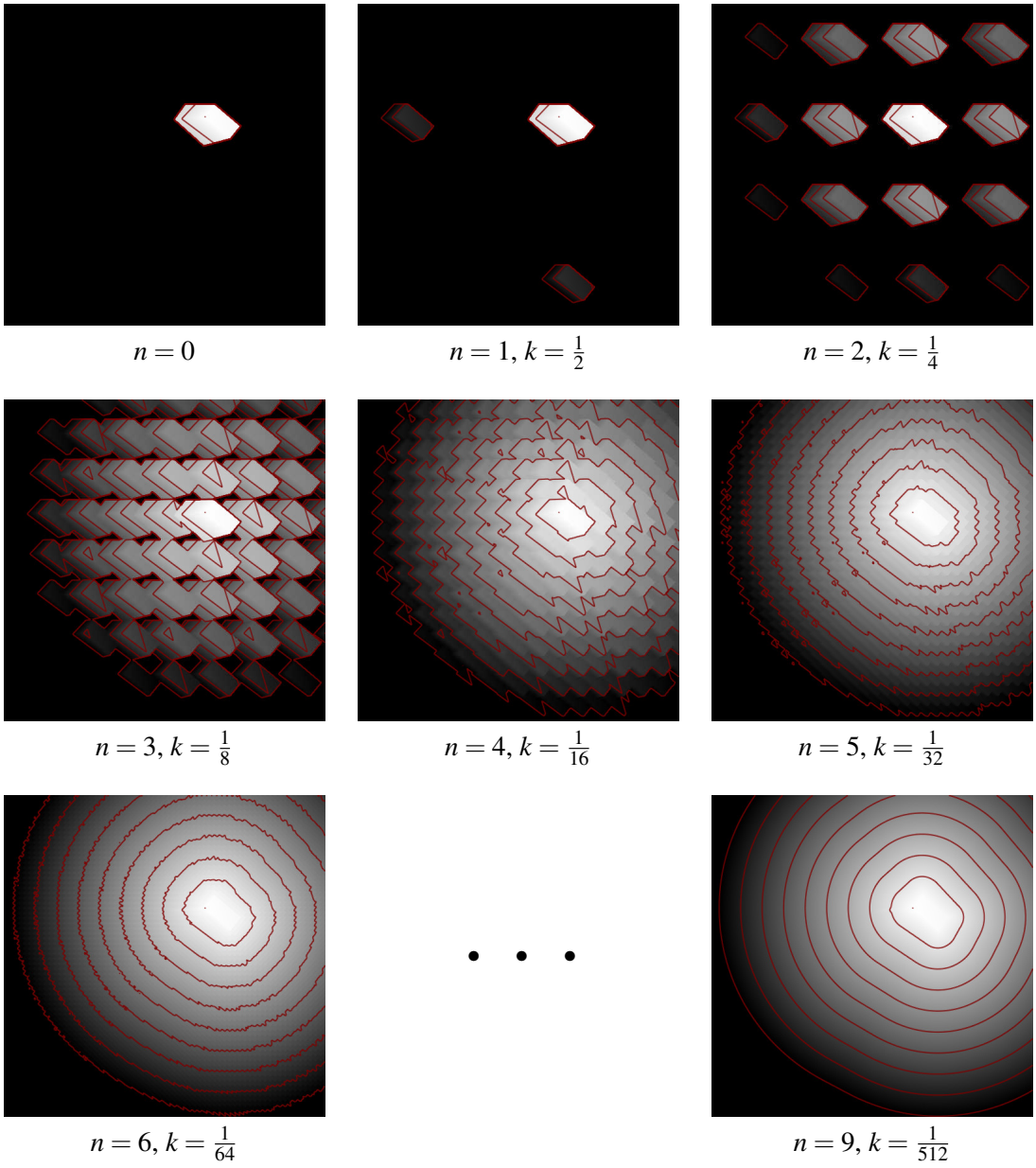


Figure 5.2: An illustration of cutaway surface computation with the jump flooding algorithm. To calculate the surface shown in Figure 5.1c, at a resolution of 512-by-512, 9 iterations are necessary. A depth image of the rear hull of the green box ($n = 0$) is iteratively processed with successively decreasing step sizes, until the resulting surface is smooth and free of jagged artifacts ($n = 9$).

candidate points $q \in \{(p_x + \{k, 0, -k\}, p_y + \{k, 0, -k\}) - p\}$. If a greater depth is found, the new maximum and its coordinate location are stored at p for the next iteration. At the completion of $\log n$ iterations, the depth at p is the maximum value $C(p,q)$ for all q over the image. Evaluating $C(p,q)$ exactly each time avoids approximation artifacts and allows definition of the cutaway function per pixel, affording local control, as described in Chapter 6.

5.2 Rendering cutaways

As described in Section 4.2, cutaway illustrations revealing a region of interest can be generated by applying occlusion reduction techniques to a scene according to the cutaway surface for the region of interest. By computing the cutaway surface as a depth image aligned with the image plane, as done in Section 5.1, the depth of the cutaway surface can be readily determined for each rendered fragment by performing a lookup into a depth buffer containing the cutaway depth image. Given the depth of the fragment and the depths of one or more cutaway surfaces (depending on the cutaway structure), the occlusion function can be evaluated to determine the appropriate occlusion reduction strength.

5.2.1 Region of interest

The cutaway surface is computed from the rear hulls of the regions of interest because only surfaces on the rear hulls contribute to the shape of the cutaway, since surfaces on the front hulls are completely contained by the conical surface emanating from the rear hulls. The regions of interest, existing just in front of these cutaway surfaces, are part of the base region since they are rendered without any occlusion reduction techniques. Their front hulls must then be incorporated into the cutaway surface. Incorporating the regions of interest into the base region ensures an appropriate definition for the overlay region, which should exist in front of the region of interest and the transition region.

The regions of interest can be incorporated into depth images of cutaway surfaces by rendering their geometries into the depth images using a conventional less-than depth test, favoring surfaces

near the camera. This augmentation of the cutaway surfaces should be performed in a separate pass after the cutaway surfaces are calculated and before they are used in rendering the scene.

5.2.2 Simple cutaway structure

In the simple cutaway structure, there are only two regions, *base* and *clear*, separated by a single cutaway surface. The occlusion function is based on which region a fragment is in, and can be reduced to whether or not a fragment's depth is greater or less than the depth of the boundary cutaway surface. This classification can be made while rendering by comparing the fragment's depth p_z to the depth of the cutaway surface $C(p)$ stored in a texture at the same screen-space position. The occlusion function is then:

$$\Omega(p) = \begin{cases} 0 & \text{if } p_z \geq C(p) \\ 1 & \text{if } p_z < C(p) \end{cases} \tag{5.3}$$

This function can be implemented efficiently in a GPU fragment shader using the *step* function [Rost, 2006]:

$$\Omega(p) = 1 - \text{step}(C(p), p_z) \tag{5.4}$$

Once the occlusion value is computed, the occlusion reduction strength R can be computed using the importance I of the fragment, generally determined by which object in the scene or which volumetric material is generating the fragment.

The example images presented here for the simple cutaway structure use a binary importance definition, where the important objects establish the region of interest and all other materials are of equally low importance. In this case, $R = \Omega$ for all materials outside the region of interest, generating a cutaway illustration where objects in the clear region are fully reduced.

5.2.3 Contextual cutaway structure

In the contextual cutaway structure, the two additional regions, *transition* and *overlay*, bring the number of boundary surfaces to three. Fortunately, the *overlay-clear* boundary can be de-

rived from the *transition-overlay* cutaway surface by addition of the overlay depth d. The *base-transition* is a separate cutaway surface defined by a different angle and must be calculated independently of the transition-overlay surface. Thus the contextual cutaway structure requires two depth images of cutaway surfaces, defined by cutaway depth functions C_1, for the transition-overlay boundary, and C_2, for the base-transition boundary.

Given these two depth functions, the occlusion function as defined in Section 4.1.3.3 can be calculated as a two-part linear ramp with 0 at C_2, 0.5 at C_1, and 1 at $C_1 + d$. Given a linear ramp function from a to b:

$$\underset{[a...b]}{\text{ramp}}\, x = \begin{cases} 0 & \text{if } x < a \\ \frac{x-a}{b-a} & \text{if } a \leq x < b \\ 1 & \text{if } x \geq b \end{cases} \tag{5.5}$$

the occlusion function can be calculated as:

$$\Omega(p) = \left(\underset{[C_2(p)...C_1(p)]}{\text{ramp}}\, p_z + \underset{[C_1(p)...C_1(p)+d]}{\text{ramp}}\, p_z \right) \div 2 \tag{5.6}$$

As with the simple cutaway structure, the occlusion reduction strength R can be determined using the importance I of a fragment along with the value of the occlusion function at the fragment's position.

The exact mapping of I and Ω to R can be defined to suit specific visualization goals, but should follow a few universal rules. First, no occlusion reduction should occur in the base region, so R should be zero when Ω is zero. Secondly, R should increase monotonically with respect to Ω for a given I, so that occlusion reduction never occurs less in an area with a high occlusion value than in an area of low occlusion value. This rule follows from the premise that occlusion reduction should occur most in areas with high occlusion values. Finally, R should decrease monotonically with respect to I for a given Ω, so that materials of higher importance do not have their occlusion reduced more than objects of lower importance. This rule follows from the premise that materials of higher importance should be more visible than objects of lower importance.

Within these rules, it is possible to trim objects according to cutaway surfaces defined for angles between θ_1 and θ_2 and to allow some objects to remain visible in front of the region of

interest. In either case it is possible to make sharp cuts in objects by having a sharp change in R, or fade objects out gradually by having a smooth change in R. Specific occlusion reduction functions will be explored in Chapter 7.

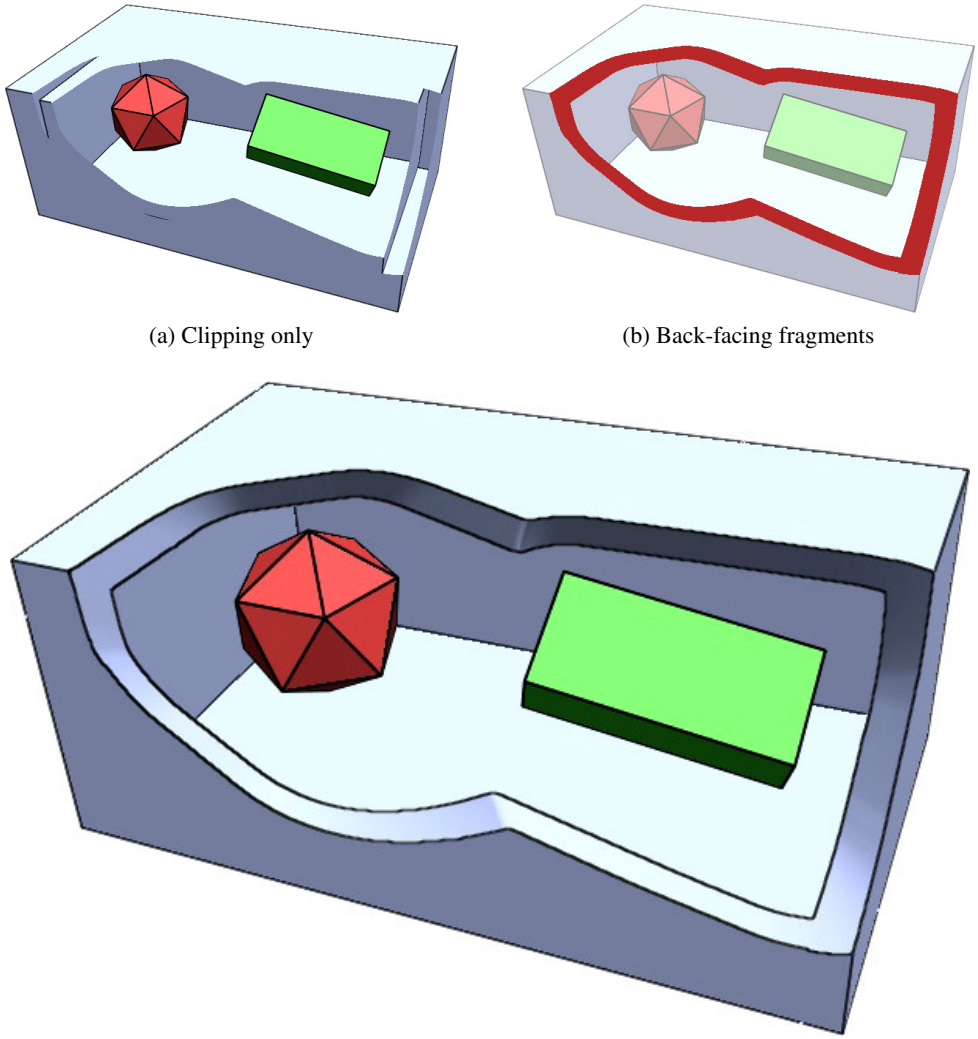
5.2.4 Cutaways and polygonal rasterization

Cutaways of rasterized polygonal scenes can be created by applying occlusion reduction according to the cutaway structure in a fragment shader. While occlusion reduction via translucency or edge highlights can technically be applied in varying degrees to a polygonal object, the best results are had when occlusion reduction is applied fully or not at all, as it creates a sharp cut surface that fits in well with the other solid polygonal surfaces. In the case of occlusion reduction by translucency, a binary function for R equates to ordinary clipping, where fragments with full occlusion reduction are completely omitted during rendering. Performing this clipping without any other consideration accomplishes the goal of exposing the region of interest, but also exposes a shortcoming of polygonal rasterization—the models are left with visible holes, losing their solid appearance.

5.2.4.1 Cut surface

Previous work has performed clipping of polygonal scenes with cutaway surfaces and other arbitrary shapes using stencil buffer algorithms. [Coffin and Hollerer, 2006; Li et al., 2007] These clipping operations involve rendering the cutting geometry in certain areas of the image to represent the *cut surface*. The stencil buffer methods could be adapted to work with a cutaway depth image standing in for the explicit cutting geometry, but an alternate, simpler approach can be taken for rendering the cut surfaces of the proposed cutaway surface.

For a scene composed of non-intersecting water-tight models, partial clipping of models always exposes their interior surfaces. Fragments rendered by these interior surfaces can be detected as coming from back-facing triangles of the model being clipped. Thus, in order to give the appearance of rendered cut surfaces, one can shade all fragments from back-facing triangles using



(a) Clipping only

(b) Back-facing fragments

(c) Back-facing fragments shaded and outlined

Figure 5.3: A cutaway rendering of two objects being exposed from within a sealed box. Conventional clipping alone, shown in (a), exposes the interior of the box as well as the interior of the box walls, causing the walls to appear hollow. Fragments from back-facing surfaces, highlighted in (b), can be shaded using normals calculated from the cutaway surface depth image to represent the cut surface, shown in (c), giving the appearance of solid walls. To fit the line drawing style, the footprint of the back-facing fragments can be outlined to show ridges formed by the cut.

normals calculated from the cutaway surface. The only visible back-facing fragments are those that appear in place of the cut surface. The result of this technique is illustrated in Figure 5.3. The cut surfaces help give the appearance of solid objects being carved away along the cutaway surface and aids in the perception of the cutaway surface shape.

The normal in camera space for a given point on the cutaway surface can be calculated by un-projecting depths of nearby pixels in the cutaway depth image to find nearby points on the cutaway surface in camera space. Three or four of these points can establish two non-parallel vectors along the cutaway surface, which can be crossed to find the normal.

5.2.4.2 Line renderings

Polygonal rasterization is often combined with line extraction and rendering to create non-photorealistic images that mimic artistic techniques. Rendered lines serve the same purpose as those in artistic line drawings, to convey surface shape and object detail. It is possible to extract these lines from the depths of rendered images [Saito and Takahashi, 1990], in which case the cutaway depth image can be incorporated with the rendered depths of geometry to produce an accurate depth image of the cutaway scene. However, most line rendering algorithms start with line extraction from geometry, looking for ridges, creases, and other geometric surface features.

In a cutaway illustration, one such geometric feature that warrants a line is the ridge formed on an object by the cutting action, delineating the cut surfaces. Creating explicit geometry for the cut surfaces is theoretically possible by carefully clipping scene geometry against the cutaway surface, a potentially expensive operation. Alternately, ridges can be found on image-space boundaries of the cut surfaces rendered in Section 5.2.4.1. Cut surfaces can be outlined by recording fragments from back-facing triangles in a separate binary channel and processing this image on the GPU to highlight or mark pixels on the binary image boundaries. Figure 5.3c shows how these outlines correspond to the ridges formed by the clipping, and aid visual perception of the cut surface locations.

This image-space operation can fail to work in situations where the clipped object has little separation between front and back-facing surfaces and the cutaway surface is very steep relative to the camera. In order to address this problem, back-facing fragments can be clipped to a cutaway surface that is slightly narrower, such that at least one back-facing fragment is exposed where a surface is clipped. This surface is created by a dilation operation on the cutaway depth image that finds the minimum value in a 1-pixel neighborhood. The resulting dilated cutaway depth image is used to clip back-facing fragments when detecting cut surface outlines as well as when determining the occlusion value for consistency.

5.2.5 Cutaways with volumetric rendering

Volumetric rendering is often used for scenes and datasets with non-solid objects because materials are rendered by volume rather than by surface alone. As such, smokey, gaseous, and other translucent objects can be rendered by simulating the absorption of light through a scene. Since volumetric rendering already includes high variability in material translucency, it is straightforward to introduce the more variable occlusion reduction techniques like variable translucency and edge highlights into a rendering. Thus, volumetric rendering is more suitable than polygonal rasterization to handle smooth changes in R.

As is the case with polygonal rasterization, occlusion reduction can be applied in a fragment shader before the fragment is rendered. For slice-based volumetric rendering, each fragment must perform a lookup into the cutaway depth image(s) to determine the occlusion value, and the occlusion reduction factor. For raycasting methods, where each ray corresponds to a single screen-space pixel, lookup of the cutaway surface depth(s) may be performed once for each ray (pixel) and can be reused to calculate the occlusion value for each accumulated sample along the ray. This potential optimization, along with the availability of cutaway depths as GPU textures, makes GPU raycasting the recommended rendering method for volumetric cutaways.

For occlusion reduction functions with a sharp binary transition in R, sharp cuts will be generated. Unlike polygonal rasterization, the volumetric shapes will not appear hollow as ma-

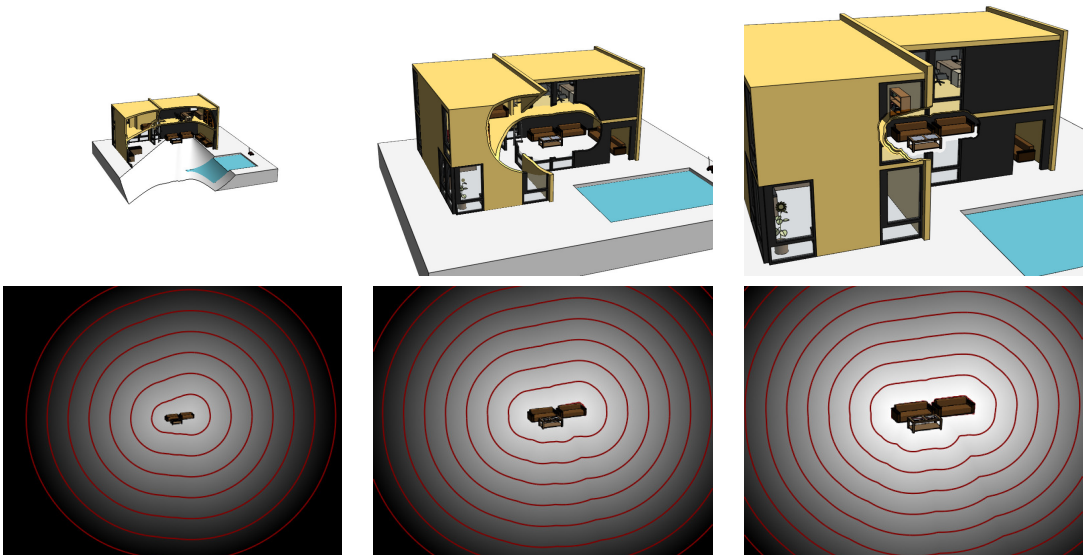
terial is rendered all throughout their interiors. However, the cut surfaces will not be properly shaded, because the volumetric normal used for shading will not reflect that of the newly cut surface. Volumetric clipping and shading techniques that combine the volumetric normals with the clipping geometry normals [Weiskopf et al., 2003] can be used to shade the cut surface according the normal calculated from the cutaway depth image, giving a more appropriate appearance to volumetric cut surfaces.

For occlusion reduction functions with smooth changes in R, materials are faded out without a sharp cut surface. In these cases, no special shading should be performed, as no sharp cut surface will be present.

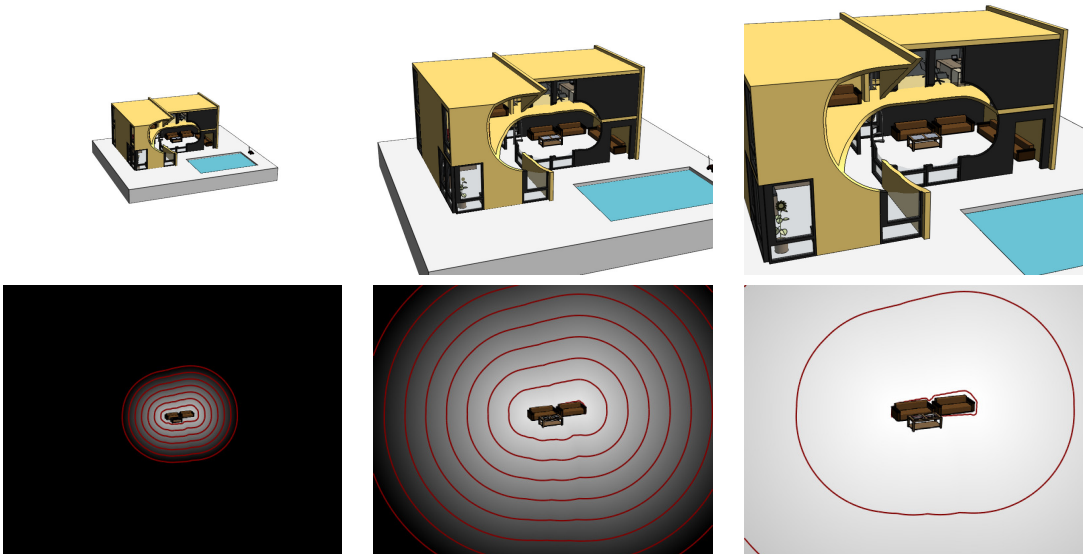
5.2.6 Perspective projection correction

It is important to note that the depth image resulting from the cutaway surface depth function C (equation (4.2)) is generated under the assumption of an orthographic projection. That is, it exactly matches the geometric definition of the cutaway surface only under an orthographic projection. Under a perspective projection, post-projection depth values from the rendered region of interest, which are used as input to C, are distorted by the non-linear perspective transformation. Fortunately, processing these depth values directly with the algorithms of Section 5.1.2 yields a viable cutaway surface with similar geometric properties to the original orthographic cutaway surface.

Since linear surfaces are preserved in the perspective transformation, the surface is still conical with linear edges but instead of the cone axes being aligned along the z-axis, they are aligned along the viewing rays, as shown in Figure 5.4. The precise angle of each cone is no longer constant, since the slope m of a line can change depending on its depth and projected position during the perspective transform. However, the surface generated still exhibits all of the properties enumerated in Section 4.1.2.4, and can very closely match a cutaway for an orthographic projection by modification of the slope m in equation (4.2).



(a) Without perspective projection correction



(b) With perspective projection correction

Figure 5.4: Cutaway illustrations of a house with corresponding cutaway depth images at three different zoom levels. Without perspective projection correction, the size of the cutaway relative to the scene changes significantly as the camera zooms in, while the cutaway surface function changes little. With perspective projection correction, the cutaway surface function adjusts significantly and the relative size of the cutaway is preserved as the camera zooms.

Because the slope m to the image plane is applied in screen space, a fixed slope m can cause cutaways of very different widths (angles) in model space, depending on the depth of the anchor point p. This effect is shown in Figure 5.4, where the cutaway becomes very wide when zoomed out and very narrow when zoomed in. The perspective error can be drastically reduced by applying a scaling factor to the slope, $-(PM_{sz} + p_z)$ where PM_{sz} is the z-scaling factor of the projection matrix, which is defined under OpenGL as $PM_{sz} = (z_{near} + z_{far}) / (z_{near} - z_{far})$. This scaling factor is accurate for the projection of a line passing through the z-axis, and the error of the approximation is larger for lines anchored at p further away from the screen center. The modified slope for a perspective projection is then:

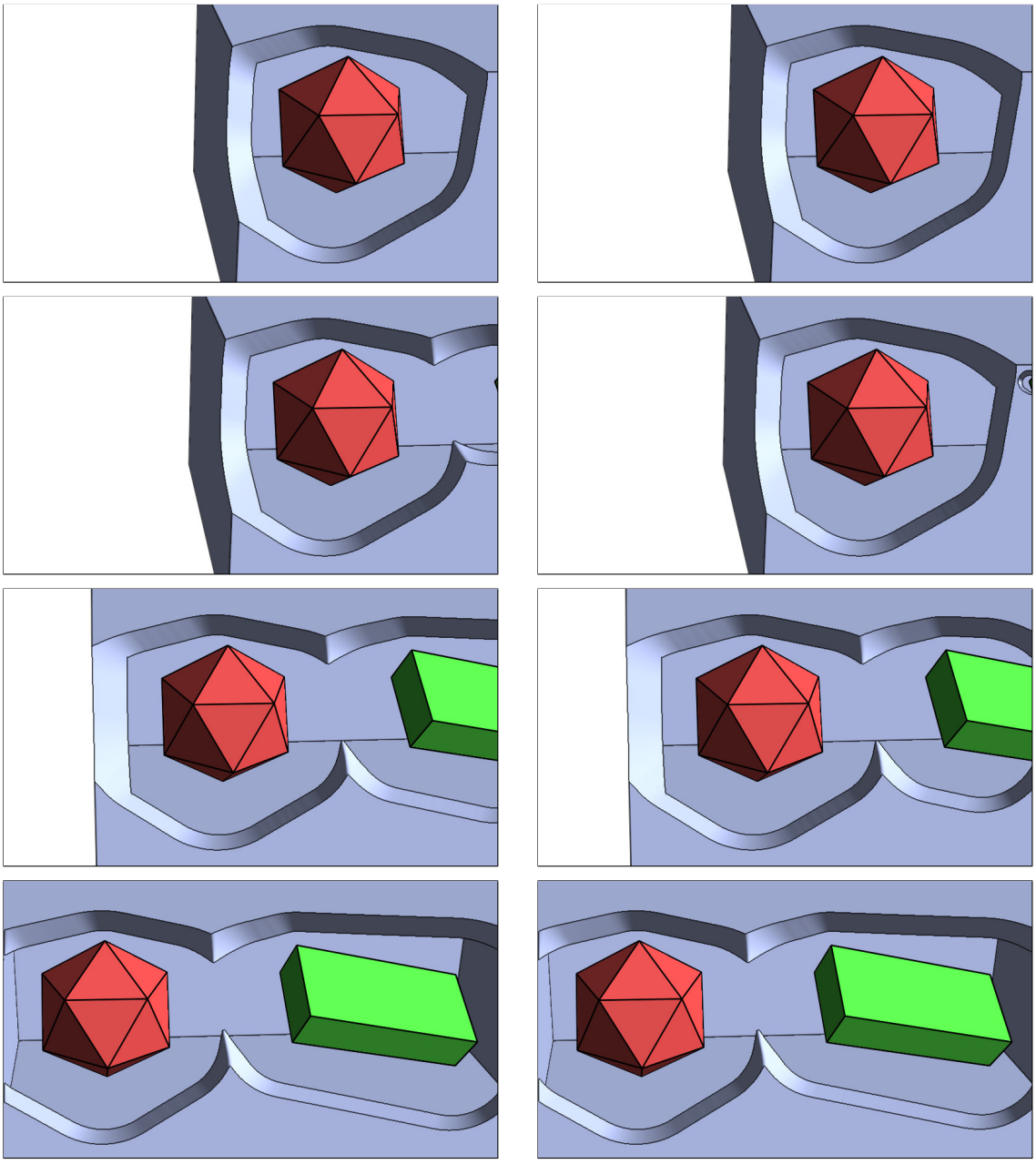
$$m(p) = \frac{-(PM_{sz} + p_z)}{\tan\theta} \tag{5.7}$$

5.2.7 Edge compression

Another limitation of the image-based cutaway surface computation algorithms is that off-screen objects of interest do not contribute to the on-screen cutaway surface. This lack of contribution is acceptable for static images as it makes little sense to carve secondary objects to reveal off-screen objects of interest. However, when objects move on or off screen in interactive or animated applications, a noticeable discontinuity can occur where objects are suddenly carved away, as shown in Figure 5.5.

This limitation could be partially addressed with a cutaway buffer much larger than the screen buffer, so objects outside the rendered screen could still contribute to the cutaway surface. However, the required size of this buffer would increase dramatically as the cutaway angle for the off-screen regions of interest increased, since the increased cutaway angle increases the effective cutaway footprint. Aside from the unbounded increase in memory requirements and processing time, the utility of a region of interest far off screen having a cutaway effect on the image because of its wide cutaway angle is dubious at best.

Given local control over the cutaway function, this problem can be cleanly mitigated by adjusting the effective angles such that pixels near the edge of the screen create a cutaway with



(a) Without edge compression

(b) With edge compression

Figure 5.5: An illustration of edge compression in a cutaway animation. The animation begins with the red icosahedron being exposed (top). As the camera pans right (second from top), a green box just enters the screen, also being exposed. Without edge compression, the edge of the green box creates a large cutaway, creating a visual "pop." With edge compression, the size of the new cutaway is limited to a small region at the edge of the screen. As the green box moves on screen (second from bottom), the cutaway grows to a normal size. Once the green box is completely on screen (bottom), the cutaway shape appears the same regardless of edge compression.

a very narrow angle and pixels sufficiently away from the edge of the screen use a normal angle. Then, as an object moves from off-screen to on-screen, its cutaway footprint increases from zero at the edge of the screen to its normal size once sufficiently on-screen.

Given screen space coordinates for p in the range $(-1,1)$ and a size r of the screen edge region in which compression should occur, this control can be had by defining:

$$\theta_e(p) = \theta \operatorname*{clamp}_{[0\ldots1]}\left(\frac{1-|p_x|}{r}\right)\operatorname*{clamp}_{[0\ldots1]}\left(\frac{1-|p_y|}{r}\right) \tag{5.8}$$

The images in Figure 5.5 use $r = 0.1$, which generally provides a smooth and balanced transition for objects entering and exiting the screen.

This modification of the cutaway function does not require significant user tuning, but other modifications of the cutaway function can incorporate user-specified parameters when appropriate. The next chapter explores additional cutaway function modifications that can improve the quality of rendered images in applications that support interactive exploration.

Chapter 6

Exploring Complex Scenes: Creating Interactive Adaptive Cutaways

The benefits of real-time algorithms supporting cutaway illustrations are not fully realized in pre-rendered images or animations, because they offer little advantage to a viewer over hand-made cutaway images or animations. However, when incorporated into an interactive application, these algorithms empower a user to easily create their own dynamic cutaway illustrations, allowing them to explore complex scenes in real-time. To that end, there are several application features that are useful for controlling cutaway creation. Included in this chapter are new cutaway surface function modifications to support locally defined angles and directional constraints.

6.1 Scene exploration and importance designation

Exploration of complex scenes can be accomplished by selecting objects or volumetric regions of interest and exposing them through cutaways. For a scene with many potential objects of interest, a user can select each of these objects sequentially as they desire them exposed. Additionally, objects and volumetric materials in a scene can be assigned real-valued importance values to control their exposure in a contextual cutaway structure (Section 4.1.3).

For polygonal scenes, the scene can be segmented at modeling time, such that objects can be represented in a scene graph or entity list. A user can then explore the scene by selecting objects in a list to define the current region of interest. This selection widget, which can stand alongside the render window, can also be used to assign importance values to objects. Then, the importance value of an object can be associated with its constituent polygons at render time to influence any occlusion reduction. An application may also support grouping of related objects into user-defined cutaway groups, each defining a region of interest with its own set of cutaway parameters, as described in Section 6.2.

For purely volumetric scenes, similar segmentation of the scene must occur. Common volumetric segmentation algorithms can be used to create geometric surfaces that can then be used to define the region of interest. Importance values must be available at render time for a particular voxel to control occlusion reduction, so a means to define importance per voxel must be established. One possibility is to define the importance values via a transfer function, just as color and opacity are defined [Haidacher, 2007]. The pairing of color and opacity with importance associates importance values with particular volumetric materials, and can provide the distinction necessary for an expressive contextual cutaway illustration.

For hybrid scenes, a combination of the above methods can be used for polygonal and volumetric elements as appropriate.

6.2 Controlling cutaways

The cutaway algorithms described in Chapter 4 and Chapter 5 have several parameters that are best exposed to the user, to be tuned to fit a particular visualization goal. These parameters include the cutaway angles and overlay thickness in the cutaway structures. Because absolute values of the angles and thickness are often unimportant to a user, these values can be effectively controlled with sliders or knobs.

For the cutaway angles, a user can quickly experiment with widening or narrowing the cutaway by adjusting a slider with extents corresponding to 0° and 90°. For the overlay thickness, a simple

knob that increases or decreases the value can allow the user to control how much occluding material is preserved in the overlay. Real-time feedback afforded by efficient rendering algorithms can guide their decision.

While overlay thickness can effectively be applied globally, cutaway angle and shape may be more effective when applied at the cutaway group level. The jump flooding computation algorithm described in Section 5.1.2.3 allows definition of the cutaway function per-pixel, which affords real-time calculation of a cutaway surface with support for locally-defined angles and directional constraints.

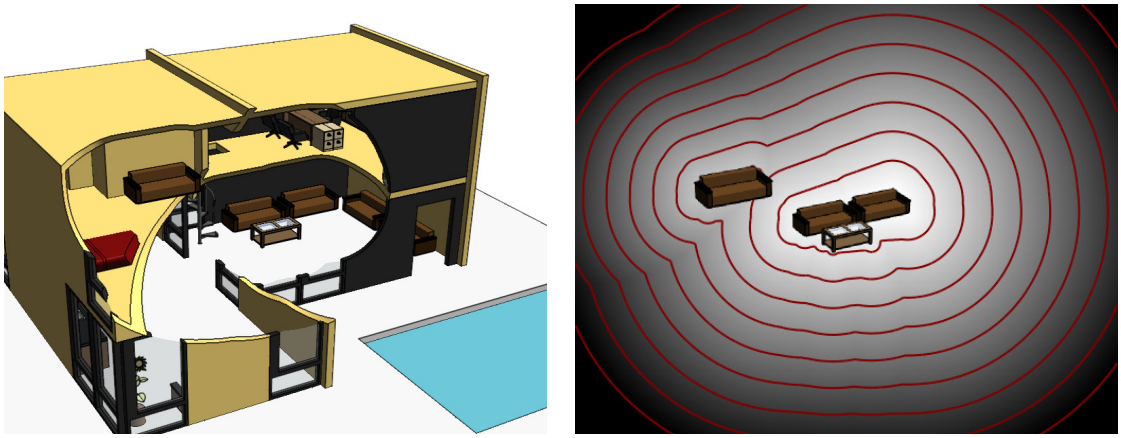
6.2.1 Local angles

By replacing θ in equation (5.8) with a locally defined $\theta(p)$, the cutaway surface can be calculated using locally defined angles. These angles can be defined per cutaway group, and should be constant within a group. The angle values can be written to a texture when the cutaway depth image is seeded with the regions of interest, using the assigned angle value for each region. During cutaway computation, this texture can be referenced by the fragment shader to find the angle at each pixel when evaluating the cutaway surface function.

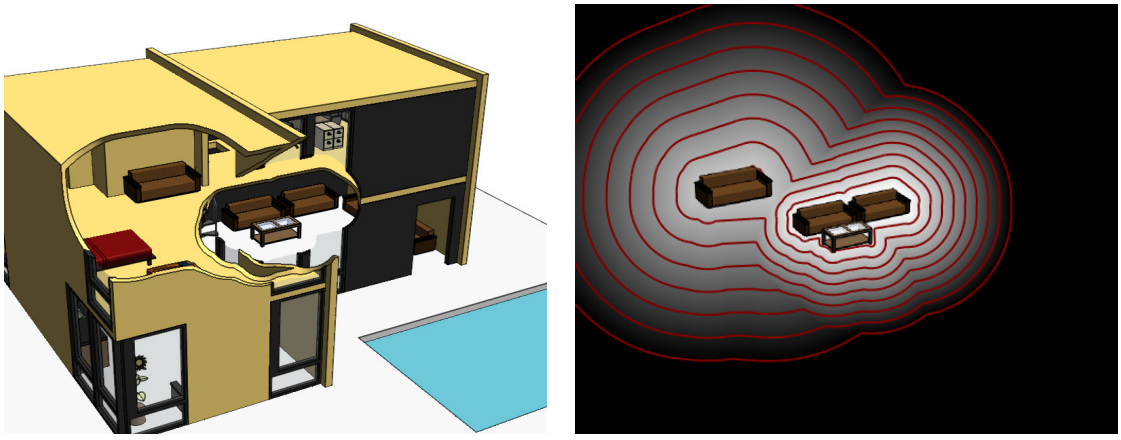
Locally defined angles may be used to de-emphasize certain regions of interest by cutting out less material around them. They may also be used to manually fine tune the cutaway effect for regions of interest in different parts of the screen, when a global angle may produce undesirable results. This effect is demonstrated in Figure 6.1.

6.2.2 Directional constraints

In addition to allowing specification of cutaway angle per cutaway group, per-pixel definition of the cutaway surface function can allow modification of the cutaway shape. Compressing or truncating the conical shape on which the cutaway surface is based can result in differently shaped cuts.



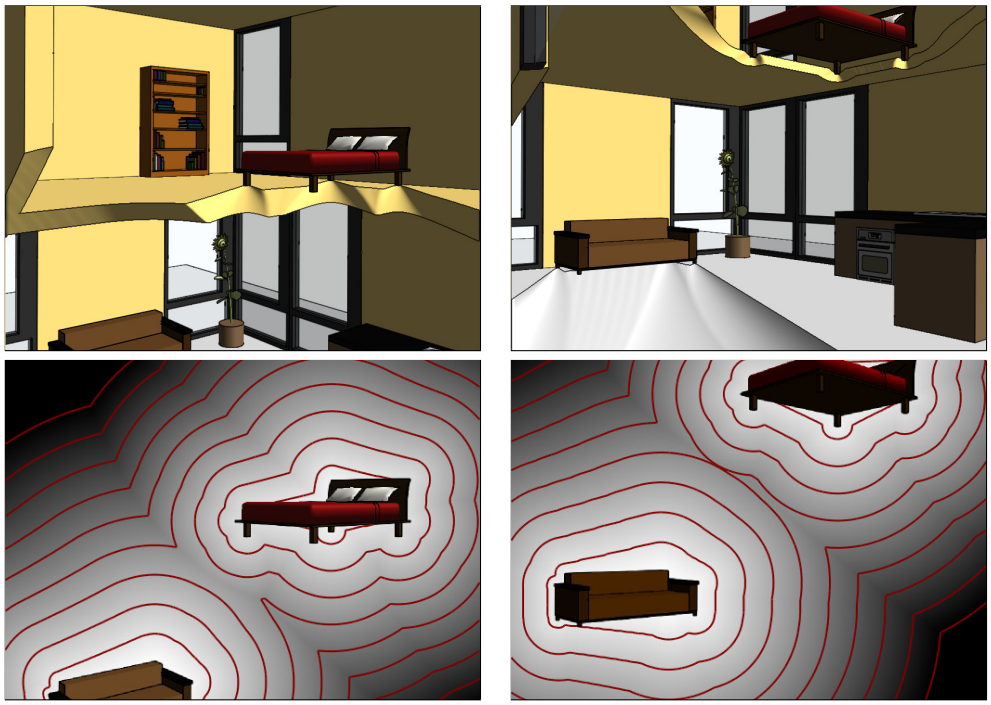
(a) Cutaway with global angle



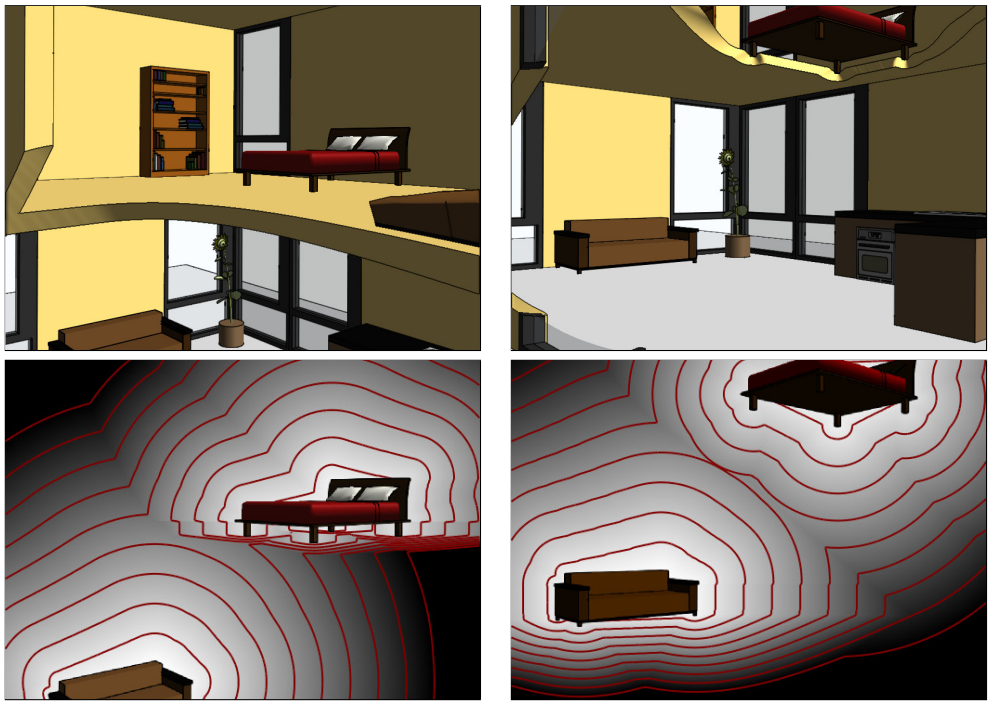
(b) Cutaway with local angles

Figure 6.1: Cutaway illustrations of a house with global and local angles. (a) A large cutaway angle applied to objects on both the upper and lower floors causes the floor beneath the objects on the upper floor to be removed, removing valuable context material. (b) When a smaller cutaway angle is used for objects on the lower floor more contextual material is preserved while sufficiently exposing objects on both floors.

Figure 6.2 shows a potential application for this modification, where a floor exhibits an unnecessary and distractingly large cut surface from objects positioned on the floor. This artifact can arise on any large, flat surface that intersects the cutaway interior nearly parallel to the edge of the cutaway, as the surface is only slightly carved away. It occurs often in views where objects of interest are situated on top of large objects outside the region of interest, such as a floor. To prevent this artifact, one can modify the shape of the cutaway to cut just above the surface of the floor.



(a) Without directional constraint



(b) With directional constraint

Figure 6.2: Objects of interest in the interior of a house rendered with and without a directional constraint. (a) Without the directional constraint, the cutaway surface can pass through the floor plane, causing the bed on the upper floor to lose its supporting floor and causing a visually distracting hole underneath the couch. (b) With the directional constraint, the floor is only removed when necessary to expose the object of interest, such as exposing the bed from below.

This modification requires determining an angle $\phi(p)$, the maximum angle of a cutaway that will avoid clipping a surface with normal $\hat{\mathbf{n}}$. With the (global) normal of the floor direction $\hat{\mathbf{n}}$, and the normalized view vector $\hat{\mathbf{v}}(p)$:

$$\phi(p) = \frac{\pi}{2} - \arccos\left(\hat{\mathbf{n}} \cdot \hat{\mathbf{v}}(p)\right) \tag{6.1}$$

Note this angle is zero when the camera is at the floor ($\hat{\mathbf{n}} \cdot \hat{\mathbf{v}} = 0$), and negative when the floor is viewed from below ($\hat{\mathbf{n}} \cdot \hat{\mathbf{v}} < 0$).

Based on this angle, a cutaway angle that uniformly minimizes undesirable carving of the floor can be calculated. The cutaway angle θ can be clamped to ϕ for camera positions above the floor (positive ϕ) and quickly transition from zero to θ once the camera passes below the floor (negative ϕ). Thus we have:

$$\theta_f(p) = \underset{[0\ldots\phi(p)]}{\text{clamp}}\ \theta_e(p) + \theta_e(p)\ \underset{[0\ldots\frac{\pi}{18}]}{\text{smoothstep}}\left(-\phi(p)\right) \tag{6.2}$$

Application of this angle to the cutaway directly would shrink the cutaway in all screen-space directions as the floor becomes more perpendicular to the camera. However, the cutaway should only be shrunk in screen-space directions where it may cut into the floor, by using θ_f for directions of the cutaway opposite the projected screen direction of the floor normal $\hat{\mathbf{n}}_{proj}$ and θ_e for other directions. α is established as the coherence between the screen space vector from q to p and the projected floor normal, and can be used to interpolate between cutaway angles θ_e and θ_f:

$$\alpha = \underset{[0\ldots1]}{\text{clamp}}\left(\frac{(q-p)}{\|q-p\|} \cdot \hat{\mathbf{n}}_{proj}\right) \tag{6.3}$$

$$\theta_d = \theta_f * \alpha + \theta_e * (1-\alpha) \tag{6.4}$$

Figure 6.2 shows the resulting effect where the cutaway no longer creates large distracting patches of cut surface and automatically adjusts itself with respect to the floor and camera directions.

Chapter 7

Cutaway Visualizations: Results and Analysis

The presented techniques for occlusion reduction and object exposure have been designed to facilitate the visualization of structurally complex scenes. To show how these techniques can be used in practice, the following sections detail two visualization problems that can be solved using cutaway visualization techniques.

7.1 Medical ultrasound and CT scan data

This section describes the motivating application for work on contextual cutaways [Burns et al., 2007] and presents results showing ultrasound data in the context of a volumetric CT scan.

7.1.1 Overview

Non-invasive needle-based medical procedures are designed to interact with organs and tissues inside the body without cutting open the skin. These procedures, which can include biopsy and radio frequency ablation, require that a surgeon navigate the body interior and operate without real visual feedback, which they would have in a normally invasive procedure. As such, the insertion

path of the needle must be carefully planned on a patient's CT scan prior to any procedure, in order to avoid critical structures. During such a procedure, an ultrasound scanner can assist in navigating needles to the planned locations, providing a real-time view of the relevant tissues. For a surgeon who would be otherwise flying blind, a view of the ultrasound scan as well as the CT data on which the needle path was planned is critical.

Traditionally, these views are presented separately, using conventional display techniques for each modality. A volumetric CT scan can be shown either in a 3-D volumetric rendering, or in slices along one of the principal axes of the scan. Medical ultrasound, which is measured in a plane oriented from the ultrasound transducer into the patient's body, is often shown as a live video stream. The separation of these visualizations requires that a user mentally align the ultrasound with the body and the CT scan in order to correlate any features from the two sources.

Using specialized state-of-the-art technology and visualization methods, it is possible to create an integrated visualization that combines the view of the ultrasound data with the context of the CT scan. A magnetic tracking system can be affixed to the ultrasound transducer and the needle(s) to determine the positions and orientations of the transducer and needle(s) relative to the tracking base. The CT scan can be aligned with this coordinate system to provide an accurate position of the ultrasound image and needles relative to the CT volume. A combined visualization can then embed renderings of the ultrasound plane and needles in the same space as the CT volume.

The inherent problem of embedding an ultrasound plane in a volumetric rendering is showing enough data from the pre-operative scan without completely occluding the ultrasound image plane. Ideally, the ultrasound plane should be mostly visible, so that a viewer can see the real-time measurements from the ultrasound image. Critical anatomical structures, such as blood vessels, should also be visible to some extent, to show areas that should be avoided by an inserted needle. Finally, the ultrasound plane should be visually relatable to the patient's body, to show the location of other tissues and organs and to reassure the operator that the tracking alignment is correct. The fused view potentially allows visualization of ablation target volumes and margins and the planned optimal needle path relative to the transducer and/or needle position.

7.1.2 Visualization strategy

The emphasis and exposure techniques of Chapter 3 and Chapter 4 can be used to create fused renderings of ultrasound data in the context of a CT scan. The contextual cutaway structure (Section 4.1.3) can be used with the ultrasound plane as the region of interest to ensure that it is exposed. Tissues from the CT scan can be defined by components of a 2-D transfer function and assigned importance values. The contextual cutaway structure can then ensure that only the most important tissues are seen in front of the ultrasound plane, in the *overlay* region. Other important tissues can be made visible in the *transition* region while the remaining tissues are relegated to the *base* region.

Since the rendering is mainly volumetric, variable translucency can be used as the occlusion reduction mechanism, so that materials are faded out based on their importance and position in the cutaway structure. As mentioned in Section 4.2.1, the specific mapping of I and Ω to R will have a significant effect on the way occlusion is reduced in the cutaway structure. The images shown in this section use the following mapping:

$$\tau_0 = max(2I - 1, 0) \tag{7.1}$$

$$\tau_1 = I \tag{7.2}$$

$$R = \underset{[\tau_0 \ldots \tau_1]}{\text{ramp}}\ \Omega \tag{7.3}$$

When $I = 1$, the *ramp* function degenerates and $R = 0$ by definition, causing materials to never have their occlusion reduced and to be visible in the *clear* region. Materials of high importance ($1 > I > 0.9$) have sharp fades in the overlay region so their cutaway boundaries are easily visible. Materials of moderately high importance ($0.9 > I > 0.5$) fade more gradually from the transition to the overlay regions. Materials of moderate importance ($0.1 > I > 0.5$) are faded gradually and wholly within the transition region. Materials of low importance ($I < 0.1$) are faded near the base region, barely extending into the transition region. Finally, for materials with $I = 0$, full occlusion reduction is performed everywhere except in the base region.

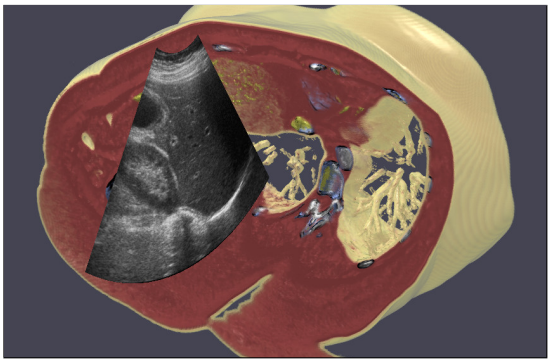
7.1.3 Results

Figures in this section show fused visualizations of ultrasound planes embedded in volumetric CT scans. Needle positions, needle paths, and target ablation volumes are omitted for sake of simplicity but could be incorporated when available using conventional rendering techniques. The ultrasound exams were recorded with the spatial location and orientation of the transducer tracked using a magnetic position sensor. The ultrasound sweeps have been manually registered (spatially aligned) to the tomographic 3-D scans, aided by automatic techniques [Wein et al., 2005]. To show interventional ultrasound in the context of organ vasculature, an early arterial phase CTA scan of a liver is used. Injected contrast agent causes the vasculature to show up with high intensities in the scan, easily distinguishable from surrounding tissue.

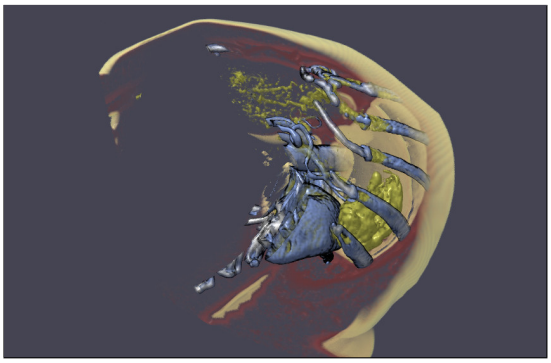
Figure 7.1 shows how a fused visualization is constructed using the contextual cutaway structure. The transfer function defines color values for vessels and kidneys (blue), bone (white), liver (yellow), skin (tan), and unclassified tissue (red). Descending importance values are assigned to components in the same order, from vessels and kidneys (0.99) to unclassified tissue (0.01). As such, skin and unclassified tissue are shown mostly in the base region, never occluding the ultrasound plane. Liver tissue protrudes into the transition region, free of occlusion by surrounding skin and unclassified tissue, but also never occluding the ultrasound plane. Because of their high importance, vessels, kidneys, and bone are shown throughout, extending into the overlay region, where they may occlude the ultrasound plane. The fused visualization provides the desired view of the ultrasound plane with the contextual information from the CT scan.

The benefit of using the contextual cutaway structure over the simple cutaway structure is shown in the progression from Figure 7.1a to Figure 7.1e. The simple cutaway structure consists of only the base region, where vasculature and liver tissue are occluded by unclassified tissue. The additional regions of the contextual cutaway structure expose these tissues while still exposing the majority of the ultrasound plane, making for a richer, more informative visualization.

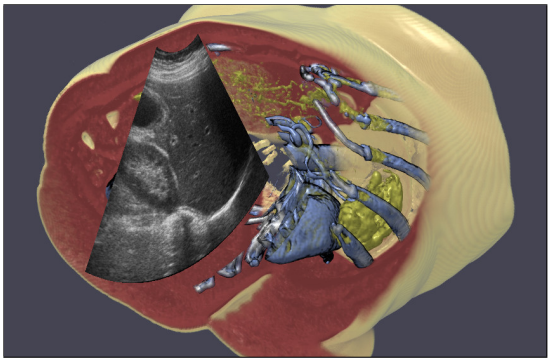
Emphasis via shading (Section 3.3.1.2) using the assigned importance values has been incorporated to guide the viewer's attention to more important tissues such as vessel and bone. Figure 7.2



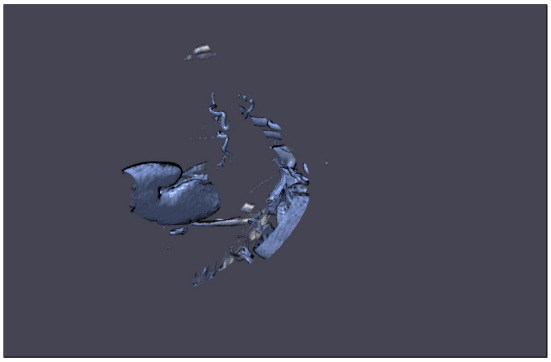
(a) Base region only



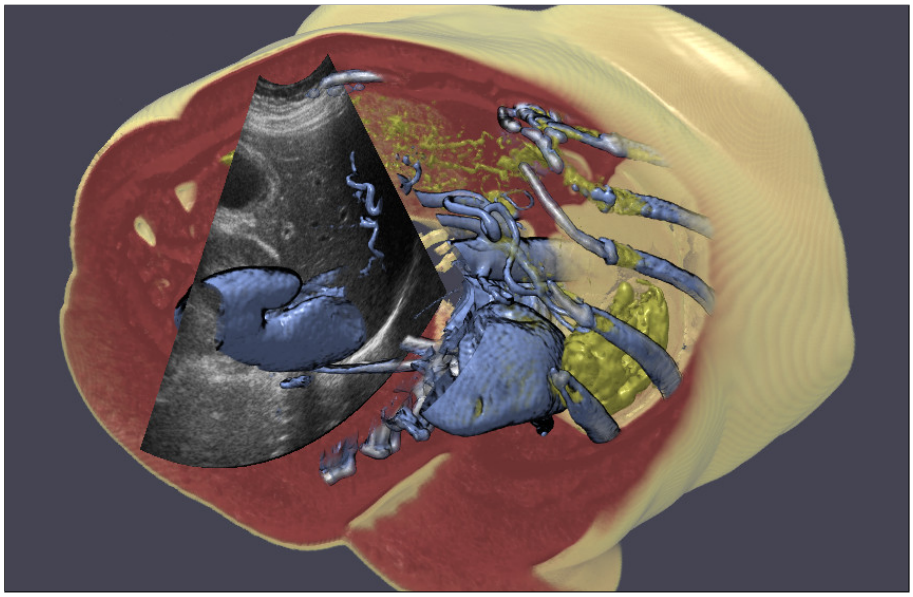
(b) Transition region only



(c) Base and transition regions

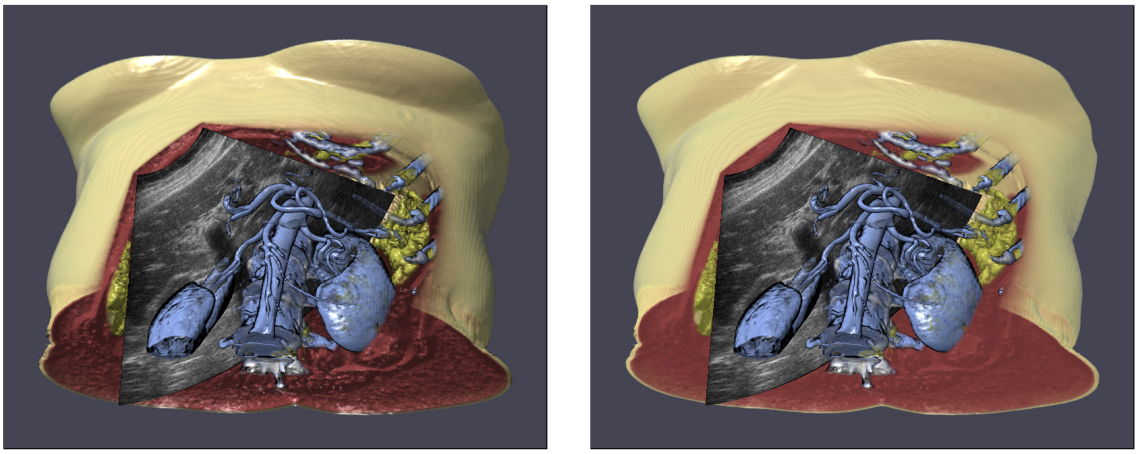


(d) Overlay region only



(e) Base, transition, and overlay regions

Figure 7.1: Illustration of cutaway regions. (a) The base region shows all tissue cut around an ultrasound image. (b) The transition region shows liver (yellow) and kidneys, vessels, and bone (blue/white). (c) Composited with the base, these tissues remain visible, protruding from unclassified tissue (red) and skin (tan). (d) The overlay region contains only important tissues that, (e) when composited with the other regions, may occlude the ultrasound image.



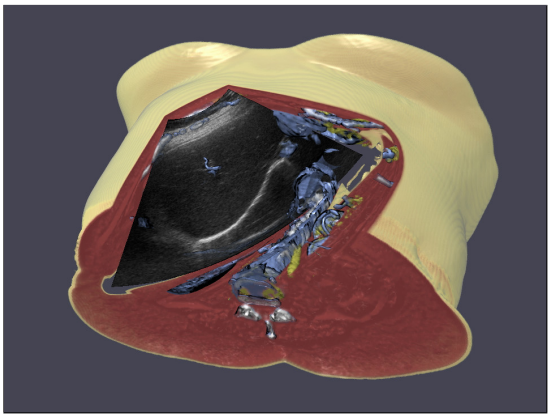
(a) Normal shading (b) Emphasis shading

Figure 7.2: Cutaway images of ultrasound data in a CT scan of a human torso. (a) With normal shading all surfaces are fully shaded. (b) With emphasis shading unimportant materials are shaded less, reducing visible surface detail. Emphasis shading causes skin (tan) and unclassified flesh (red) to be visually de-emphasized relative to materials of higher importance.

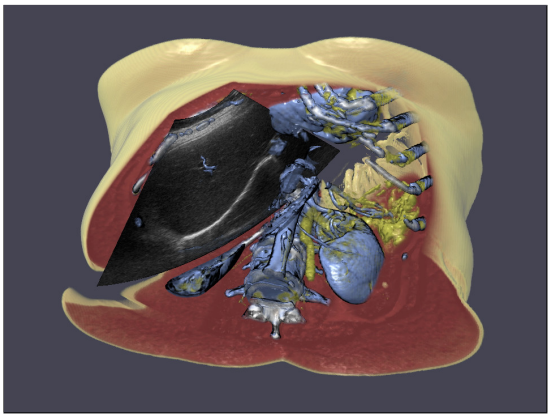
shows how shading detail can be suppressed in the unclassified tissue and skin to reduce the overall complexity of the image. The strength of this effect is controllable, and is shown with $E_s = 0.3$.

Figure 7.3 shows several images from this data set in an interactive visualization session. A user can move the ultrasound transducer to inspect different portions of the abdomen, allowing them to virtually carve away pieces of the body. The user-defined geometric parameters of the cutaway structure (angles, overlay depth) are freely adjustable during the session, allowing the user to see more or less contextual material as desired. The user can also rotate and zoom the camera around the volume, changing the camera position as desired to find a good view of the ultrasound plane. Together these features form a flexible system that allows for intuitive virtual exploration of a patient. When combined with a realtime view of a tracked needle and needle path data, a user would be able to ensure that a needle operation proceeded as planned.

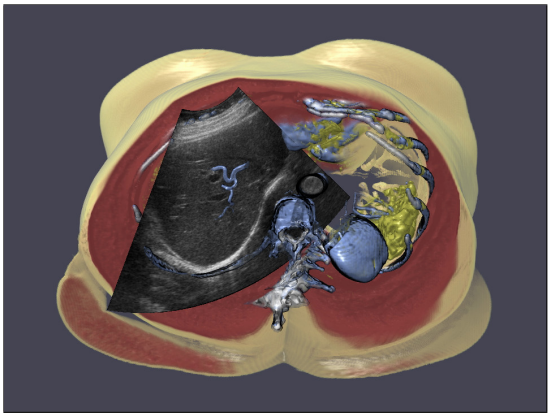
Figure 7.4 shows the result with a non-contrasted CT scan where vasculature is difficult to differentiate from other tissue. In this case, bone is the only tissue of high importance, and provides a structural frame of reference for the ultrasound plane. Because vasculature is missing, this visualization mainly serves to show the position of the ultrasound probe relative to the body.



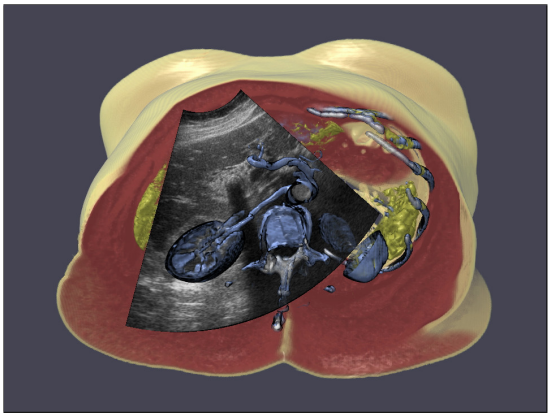
(a) Thin overlay region, narrow transition region



(b) Thin overlay region, wide transition region



(c) Ultrasound sweep forward



(d) Continued sweep forward

Figure 7.3: Cutaway images of ultrasound data in a CT scan of a human torso. A thin overlay region with a narrow transition region (a) shows small amounts of vessel (blue) in front of and in the periphery of the ultrasound plane. A wider transition region (b) exposes bone (white) and liver tissue (yellow). As the ultrasound plane is swept forward (c)-(d) the cutaway structure updates to show important tissues in the overlay and transition regions.

7.2 Architectural models

This section describes results from work that applies cutaway rendering techniques to polygonal architectural models [Burns and Finkelstein, 2008].

7.2.1 Overview

Modern architectural design work is generally performed with computers, using 3-D modeling software to create and refine models that can be rendered to preview a finished building. Architects

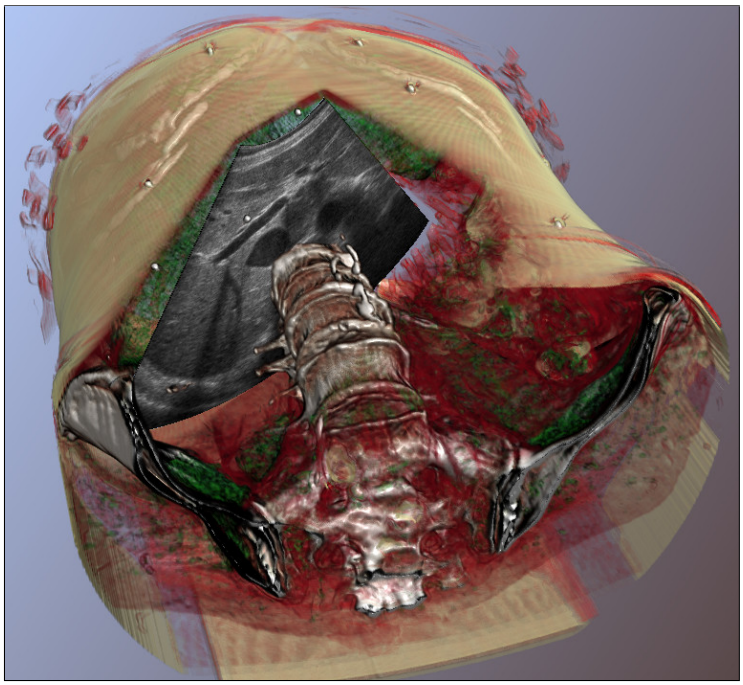


Figure 7.4: A transversal ultrasound image of a patient's liver is shown aligned in the context of a (non-contrasted) CT scan. Bone is allowed to obscure the ultrasound plane and as such helps facilitate spatial perception of the plane.

can go to great lengths to create a detailed model, staging the interior of the building with furniture, appliances, and other structures that show their vision for how a space could be populated and used. As a building increases in size and a modeler puts more effort into populating its interior, the model can become highly complex structurally, with layers of walls, ceilings, and interior objects potentially occluding each other. Such occlusion can make it challenging to create effective renderings showing a building's interior.

One common approach to depicting interior spaces is to place the camera inside the building, recreating the perspective of a building occupant. Such perspectives can be created for static images by manual placement of a camera, or for animations by modeling a camera path in a standard 3-D rendering application. Yet another option is to allow a user to navigate a scene using controls similar to those found in video games. These approaches can all provide interior views with conventional rendering techniques, but require care in placing the camera in order to view a particular area. Additionally, the limited point-of-view of the camera on the interior makes it difficult to determine locations of interior spaces and positions of objects relative to the whole

building. Because of occlusion by walls and floors, this approach also does not allow view of more than a single room.

One option for viewing interiors from exterior camera positions is to use clipping planes to suppress rendering of portions of the building that normally occlude an interior region of interest. For example, it is common when modeling a building's interior to place a clipping plane just below the ceiling of the floor being modeled, so that the contents of that floor are exposed from above. When done manually, this action can become cumbersome as clipping planes must be enabled and disabled when switching from floor to floor, and multiple clipping planes for a multi-level building must be maintained as the model changes. Automatic methods exist to detect floors and provide exploded renderings [Niederauer et al., 2003] but may not work for non-conventional building layouts with multiple adjacent but offset floor levels. Furthermore, these horizontal clipping planes do not solve the issue of occlusion by walls and limit the viewpoint of a floor to one that is looking down from far above. Walls can be addressed by using additional vertical clipping planes, which unfortunately add to the complex and difficult maintenance of a multitude of clipping planes.

An effective visualization of a set of rooms or objects inside the building should ensure that these objects of interest are exposed to the viewer from several camera positions. It is also useful for objects of interest to be locatable in the building, meaning some exterior building structure is shown to relate the objects of interest to the building as a whole. Furthermore, visualization techniques should be capable of handling dynamic models, such as those with animated components, and should respond quickly to changes in building structure brought about by interactive modeling actions, or to changes in camera location brought about by user control. The remainder of this section shows how the cutaway visualization techniques previously described can address these points.

	House	Galleon	Building
Triangles (K)	76	301	530
Lines (K)	45	92	286
Chamfer (fps)	10	10	10
Jump Flood (fps)	60	60	60
Model (fps)	46	17	11
With Cutaway (fps)	41	16	10

Table 7.1: Model sizes and frame rates for the house, galleon, and building models shown in Figures 7.8, 7.7, and 7.6, respectively. Chamfer denotes the frame rate for approximating the cutaway surface using a 5×5 chamfer kernel. The frame rate using the full-featured jump flooding algorithm is significantly faster. Overall performance numbers show that rendering the full model is generally the bottleneck; calculation of the cutaway surface and additional rendering effects impose a relatively small overhead.

7.2.2 Visualization strategy

A scene graph denoting interior objects and room geometries can be used as a selection mechanism to define a region of interest. This region of interest can be exposed through a contextual cutaway structure with a user-specified overlay region depth. The transition region can be collapsed (omitted) because secondary objects are not ranked by importance. A non-photorealistic rendering style that includes lines is chosen to more closely resemble hand-drawn architectural renderings. Given the rendering of lines, occlusion reduction is performed by the surface suppression and line fading mechanism described in Section 2.3. Because objects are not prioritized by importance, R can be derived from Ω alone, with $R = \Omega$. Thus objects will be rendered with their surfaces and lines at full opacity in the base region, and without their surfaces and lines fading out in the overlay region.

7.2.3 Results

Figures in this section show how the aforementioned cutaway techniques can expose interior details of structurally complex architecture models. The models were obtained from the Google 3-D Warehouse, and cleaned up to ensure consistent face orientation, which is necessary for proper cut surface shading (Section 5.2.4.1). The models contained segmented interior structures defined by the scene graph, allowing selection of one or more of objects of interest through a list interface.

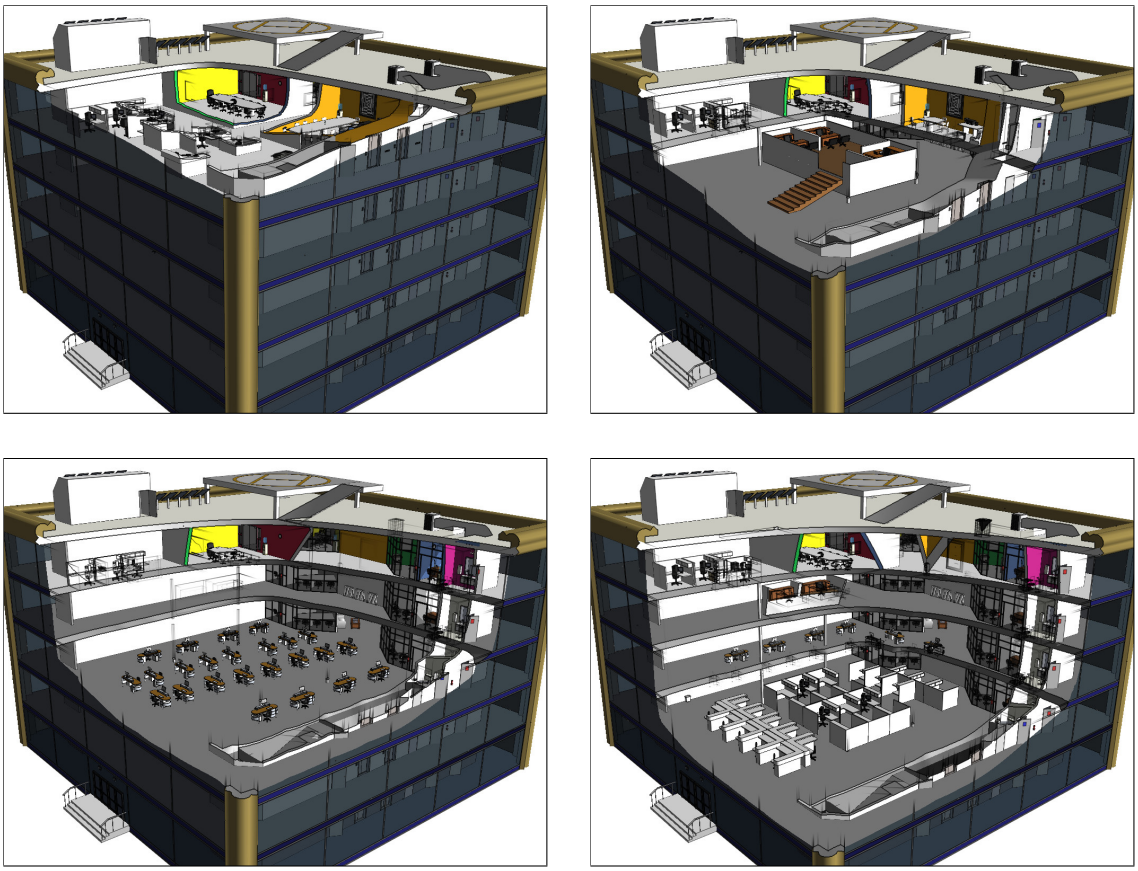
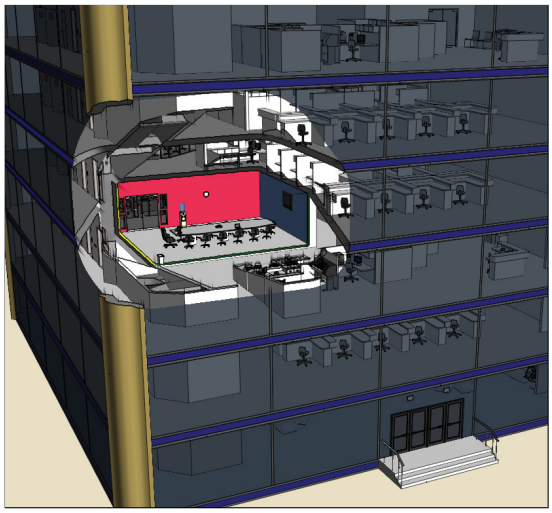


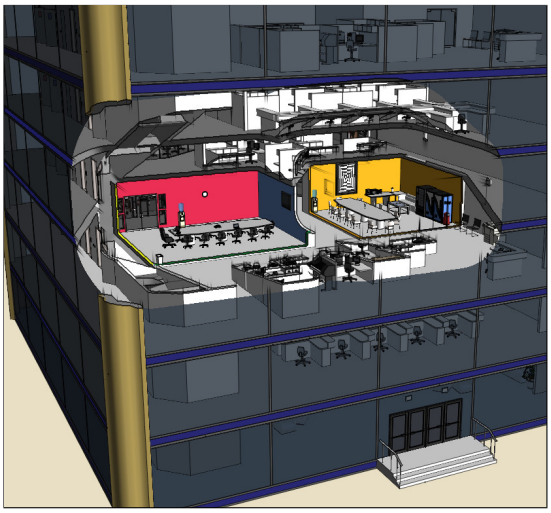
Figure 7.5: Cutaway illustrations of an office building. From left to right, top to bottom, a desk and chairs in a boardroom, a central raised office room, a group of free-standing desks, and a group of cubicles are exposed. By using a wide cutaway angle, cuts are made nearly vertically through upper floors, facilitating visual location of the objects of interest. A directional constraint prevents the floor from being cut away underneath the objects of interest.

Performance was measured using an NVIDIA GeForce 8800 GTS 512 with a framebuffer size of 640×480 and a cutaway buffer size of 512×512 (Table 7.1).

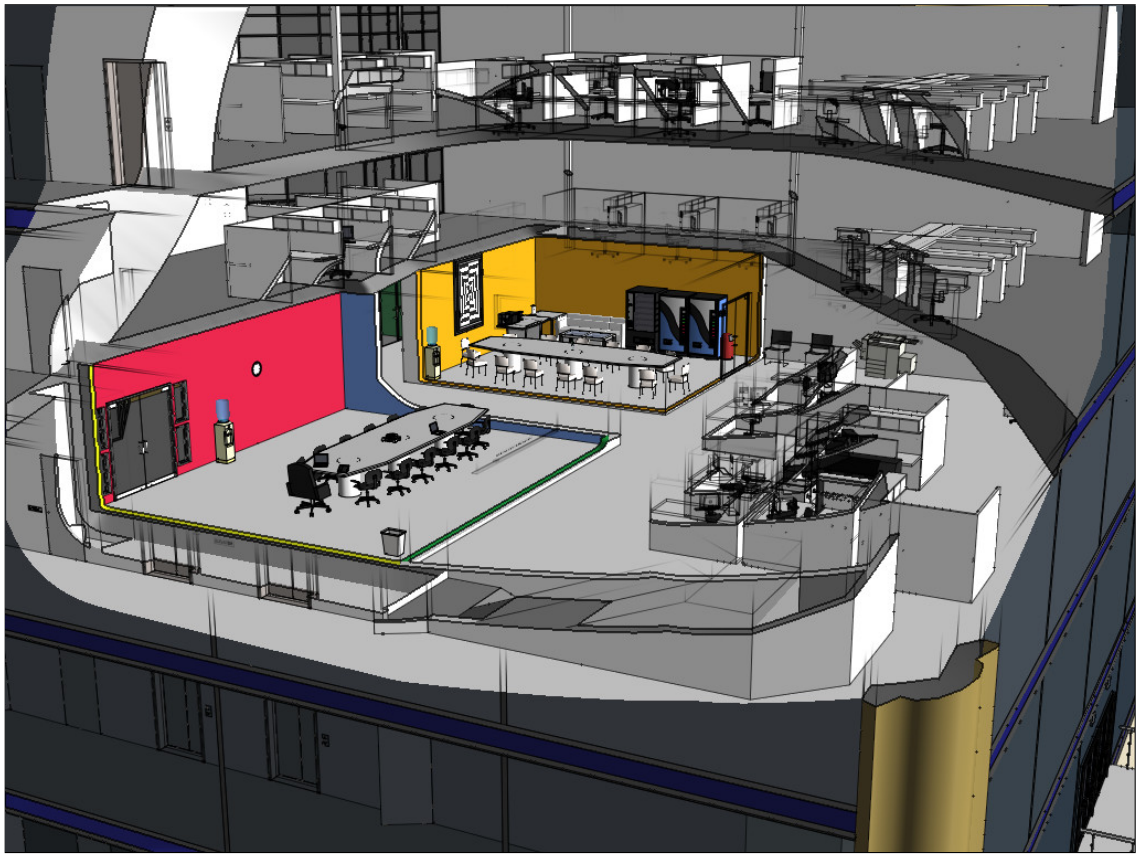
Figure 7.5 shows a series of illustrations of an office building with interior structures on several floors being revealed. Each image reveals a user-specified group of tables, desks, or cubicles that would ordinarily be occluded by the building structure. A wide cutaway angle is used, adjusted so that cuts are made nearly vertical through the overhanging floors. With such a wide cutaway angle, the cutaway shape would normally extend below the floor, causing the objects to float in mid-air. A directional constraint (Section 6.2.2) in the up direction prevents this artifact from occurring.



(a) Single room exposed

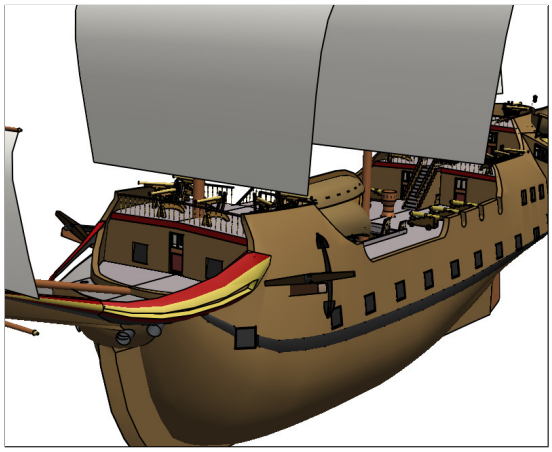


(b) Multiple rooms exposed

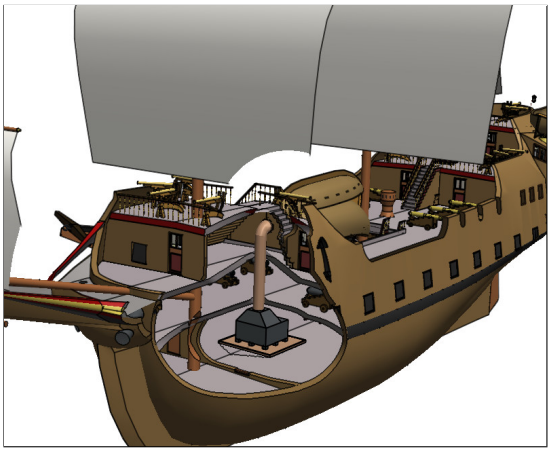


(c) Wider cutaway angle

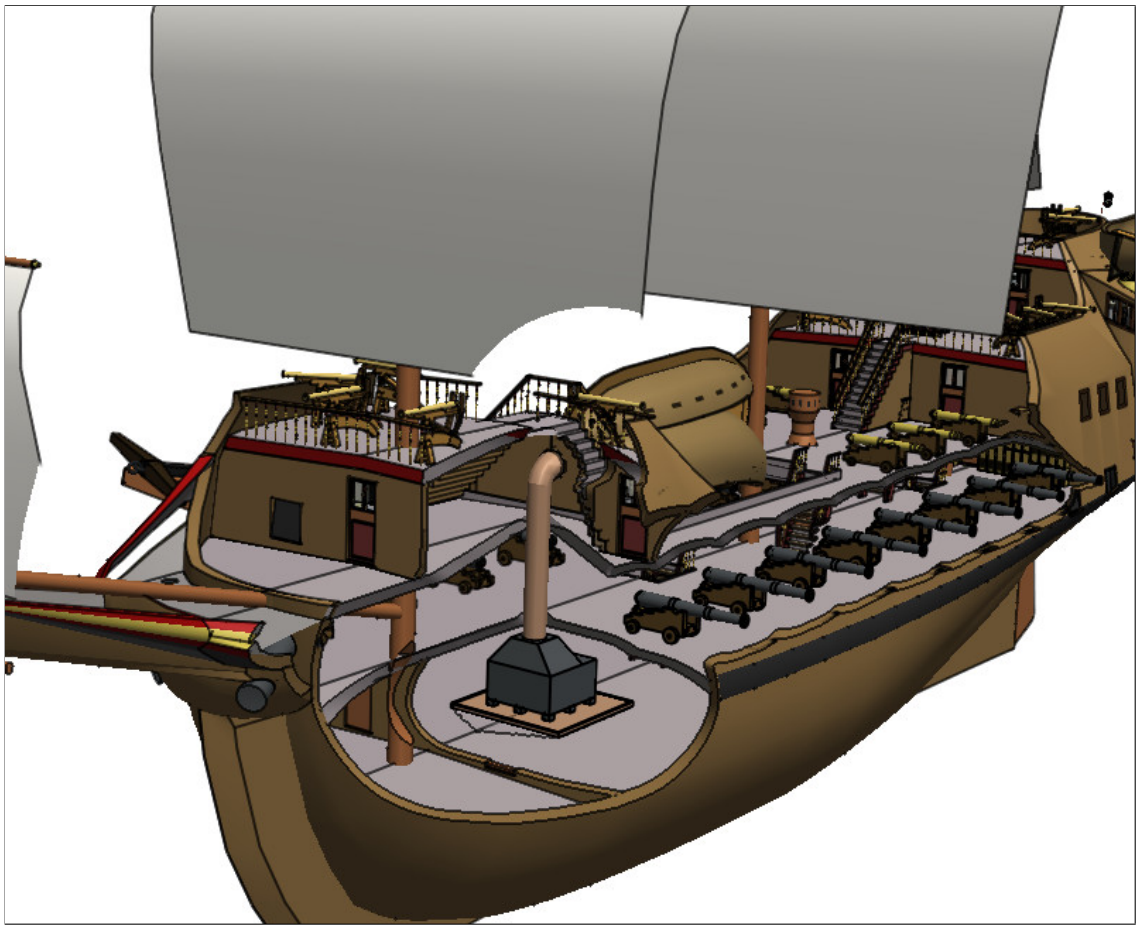
Figure 7.6: Cutaway illustrations using an invisible region of interest. (a) A simple invisible box denotes a boardroom in an office building as the region of interest. Objects contained in the box are considered part of the region of interest and are exposed accordingly. (b) A box representing a break room (yellow) is added. (c) A different camera position and a wider cutaway angle expose the upper floors.



(a) Normal rendering



(b) Furnace exposed



(c) Furnace and cannons exposed

Figure 7.7: Cutaway illustrations of a galleon. (a) Normal rendering techniques show only the exterior of the ship. (b) A furnace spanning three decks is exposed with a cutaway. A wide angle is used to expose the interior deck structure spanned by the furnace. (c) The region of interest is expanded to include cannons on a middle deck. A narrower angle is used around the cannons to preserve structures on the upper deck.

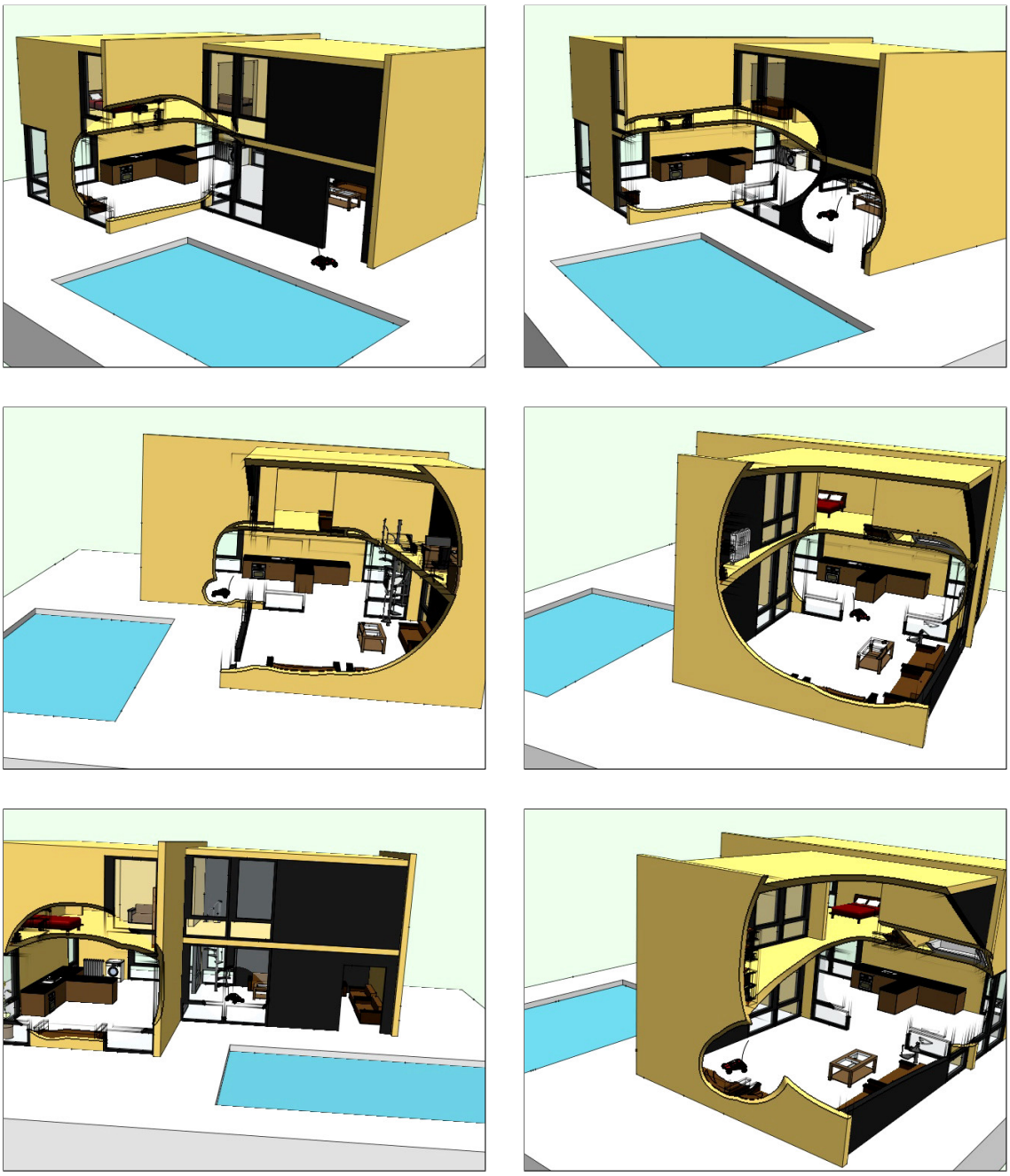


Figure 7.8: Still frames from an animation of a radio controlled car driving through a house. A smoothly varying cutaway keeps the car visible as it drives through the house and the camera rotates about the scene. Faded lines in the overlay region show the partial shapes of cut away structures near the cutaway surface boundary. Efficient computation of the cutaway surface function allows the animation to run in real-time on modest commodity hardware.

The images show the exterior of the lower floors, and the trimmed interiors of the upper floors, facilitating easy recognition of the specific floor on which objects of interest are placed.

As described in Section 4.2.2.2, proxy geometry that does not appear in a rendered image can be used to define the region of interest. Figure 7.6 shows how such proxy geometry can be used to show the location and contents of two boardrooms. The proxy geometry is a simple box in each room that spans almost the extent of each room and includes the contents (table, chairs) that should be exposed. Figure 7.6a shows a single room exposed with a narrow cutaway angle, allowing one to peer into the contents of this room from the outside of the building while maintaining normal view of most of the building exterior. Figure 7.6b adds an additional room to the region of interest, exposing the contents of both rooms simultaneously. Figure 7.6c shifts the camera location and zooms in to show more detail. Note that the contents of the left-side room are not cut by the cutaway extending from the right-side room, since the contents of both rooms are part of the region of interest. A wider cutaway angle is also used to show the contents of two overhanging floors. While the surfaces of most of the cubicles in the floor overhead are suppressed, the remaining lines in the overlay region allude to their existence.

Figure 7.7 shows how a model of a galleon can be explored using the cutaway exposure techniques. The conventional exterior view hides a significant amount of detail on the ship's interior. Cutaways in the subsequent images show a furnace spanning multiple decks and a row of cannons below the top deck. Because the cutaway shape closely follows that of the furnace and the group of cannons, only the area around their footprint is disturbed. The remainder of the ship (sails, supporting masts, rear rooms) are visible as normal. A directional constraint again ensures that the supporting deck below the furnace remains visible despite a relatively wide cutaway angle. A narrower angle is applied to the cannons to limit the material removed from the upper deck. The two groups of objects are exposed by a single cutaway surface that respects their distinct angle parameters.

The exposure methods used readily support dynamic scenes with changing geometry because cutaway surfaces are calculated on the fly every frame. Figure 7.8 shows how a radio controlled car driving through a house can be exposed using a dynamic cutaway. The region of interest

contains the kitchen counter and the RC car. When the car is outside the house, the house structure is only cut to expose the counter. When the car moves inside, the house structure is additionally cut to show the moving car. As the car approaches the counter, the cutaway shape arising from the car smoothly merges with that of the counter. This merging occurs because both objects of interest are contributing to the cutaway surface, and the merged cutaway surface shape reflects the shape of both the counter and the car. When the car moves sufficiently away from the counter, the cutaway surfaces separate as expected. The car is never cut by the cutaway action initiated by the counter, since they are both part of the region of interest. Because no geometry pre-processing is necessary, this animation can run in real-time with little overhead over standard polygonal rendering.

Chapter 8

Conclusion

The images presented in Chapter 7, created using the emphasis and exposure techniques presented throughout this thesis, help address the challenge outlined in the introduction of conveying structural information in complex scenes. By emphasizing and exposing a region of interest, a focus for visualization is conveyed and displayed in the context of the rest of the scene. This exposure allows prominent view of the focus area, which may have been difficult using conventional rendering techniques, and also illustrates structural relationships between the focus area and the rest of the scene. These structural relationships are important to a viewer in gaining a semantic understanding of the scene and inferring the role of focus objects to any action in the scene.

The techniques presented in this thesis fulfill the three goals for visualization set out in Section 1.3: creating informational visualizations with a simple interface and providing interactive performance. While previously developed techniques have addressed these goals, the presented techniques expand upon and overcome several limitations found in previous techniques. In doing so the presented techniques create richer high-fidelity images, requiring little user skill or setup, at interactive framerates for dynamic scenes.

Informational visualization The techniques presented create information-rich visualizations of interior scene structure that would otherwise be absent from renderings created with conventional

visibility techniques. Cutaway illustrations of the ultrasound plane in CT data and the interior objects of architectural models show how these objects of interest can be exposed in situations in which they would normally be occluded. The resulting images show structural information about the focus objects, relating them to their surrounding context and the scene as a whole. Secondary objects of interest, such as the vessels in the medical visualization, are exposed from their surrounding materials and emphasized according to their importance by way of the contextual cutaway structure. In addition to improving upon renderings using conventional visibility techniques, the contextual cutaway structure allows more information to be shown than by the conventional simple cutaway structure.

Simple interface The techniques are flexible enough to create a wide range of visualizations but are automated enough to operate with a simple point and click interface. The only processing of the input data necessary is to define groups of geometry to serve as objects of interest, and to assign importance values to objects to control their priority in the visualization. The former is often accomplished by means of a scene graph created during scene modeling, or by automated volumetric segmentation techniques. The latter can be accomplished using a simple GUI as described in Section 6.1. The remaining parameters, such as cutaway angles, overlay thickness, and emphasis strength, can also easily be controlled by a few sliders in a GUI.

Interactive performance Because performance and interactive feedback was a driving factor in the development of the presented techniques, they use efficient computation algorithms that operate at interactive framerates. When computing cutaway surfaces using the jump flood algorithm, very high frame rates can be achieved with a remotely modern GPU. Performance tests (Table 7.1) show that the cutaway action adds little performance overhead to the polygonal rendering pipeline. Because a volumetric raycaster can cache cutaway surface depth values, the performance impact on the volumetric rendering pipeline is similarly small. As a further improvement over previous techniques, this interactive performance extends to changing viewpoint and dynamic animated scenes.

Future work

While the presented techniques are expressive and reasonably efficient in creating images that serve the stated purpose, they are by no means the final words in this field. From the new cutaway techniques presented, there remains room for improvement in the computation of the cutaway surface, investigation of various cutaway shapes, and the consideration of entity shape geometry in applying occlusion reduction in cutaways. From a broader perspective, there are opportunities for research in additional applications of the jump flooding algorithm and the incorporation of additional emphasis techniques to complex scenes.

Cutaway surface computation Image based cutaway surface computation offers efficient computation with high fidelity but has several weaknesses typical of image-based algorithms. Among these issues is the inability of off-screen objects to contribute to the cutaway shape, necessitating the edge compression method to avoid temporal artifacts as objects move on-screen. Future work may be able to find a more elegant and effective way to handle this problem. Additionally, because calculation of the cutaway shape occurs at a fixed resolution, regions of interest with very thin geometric features can induce aliasing in the cutaway shape. Such high-frequency geometric features likely should not contribute to the cutaway shape, as these small variations in shape are amplified by the cutaway function. As such, processing of the region of interest may be possible to find a shape with a lower level of detail to use in the actual cutaway surface computation. Finally, when very low resolutions are used for the cutaway shape, computation can be very fast but single-pixel differences in the footprint of the region of interest can cause noticeable changes in the cutaway shape as the camera changes. This issue can be overcome by using a higher resolution cutaway buffer, which requires significantly more computation, at the cost of performance. Because the basic cutaway shape is fairly simple and free of high-frequency changes in surface shape, representing it with a high resolution image seems intrinsically inefficient. It may be possible to use a hybrid sampling or computation approach to achieve higher performance while avoiding the aliasing brought about by using a low resolution for computation.

Varying cutaway shape The shape of the cutaway surface is derived from the shape of the region of interest and a user-given angle parameter. Greater user-control of the cutaway surface may be had by further modifying the cutaway surface function. For example, the shape of the cone used as a basis for the surface may be altered to effect the "eccentricity" of the cutout, perhaps cutting out more information in a horizontal direction than in a vertical direction. One might also wish to bias the cutout direction so that it is not heading directly towards the camera, providing greater view of the cut surface in one direction. Investigation into automatic setting of these and other cutaway surface parameters could further lessen the burden on the user and improve the visualization experience.

Entity-based occlusion reduction In the cutaway results shown in this thesis, occlusion reduction is applied per rendered fragment according to the occlusion function and fragment importance. As a result, objects near the cutaway surface are partially suppressed without regard for their object geometry. The work of Li et al. [2007] shows that consideration of object geometry can result in more naturally shaped cuts. Unfortunately their approach is too computationally expensive for real-time rendering of dynamic scenes. However, there may be a way to use the fast occlusion function definition presented in this thesis to drive specialized cuts on specific entities. Such an approach might process points along the surface of an entity's geometry to determine what amount or portion of an entity is causing occlusion, and trim the geometry based on the entity's denoted cut type.

Jump flooding The Jump Flooding algorithm is used to calculate the cutaway surface by minimizing a function that is significantly more complicated than the 2-D Euclidian distance function for which the algorithm was originally designed [Rong and Tan, 2006]. The effectiveness of Jump Flooding in this application is surprising, including to at least one author of the original published paper. It would be interesting to explore the limits of the Jump Flooding algorithm and see if there are other image-based function minimization problems to which it could be applied.

Emphasis in complex scenes While the techniques described here can reveal and emphasize objects of interest, the resulting imagery remains as visually cluttered as the model from which it is rendered. The rendering method of Cole et al. [2006] draws the attention of the viewer based on a desired emphasis target for lines and surfaces in the scene, typically by evaluating distance to the single point of emphasis. Such an emphasis system may be expanded to include information arising from the cutaway structure and the position of the objects of interest. Another option would be to use level-of-detail rendering techniques to render entities of low importance with low surface detail, with or without incorporating information from the cutaway structure. The intent of either approach would be to guide the user's gaze toward the deliberately exposed focus of the visualization.

User study of effectiveness The techniques presented in this thesis create images that expose regions and objects of interest, showing structural information otherwise hidden by conventional rendering techniques. However it is unknown how effective these techniques are in improving perception of structural relationships over hand-drawn illustrations and renderings of other automated techniques. It would be valuable to conduct one or more user studies of the presented techniques, comparing them to other methods of visualization over a variety of model types (e.g. organic vs. mechanical, large vs. small, complex vs. simple, etc.). Because the presented techniques were designed with a focus on interactivity, they may hold an edge over other techniques in interactive settings. Informal experimentation shows that when combined with interactivity, fixing the direction of the cutaway to the model and allowing small camera movements creates a parallax effect that helps better convey the shape of the cutaway. Additionally, the cut surfaces and the cutting action seem to stand out more with camera motion in an animation compared to a static image. A user study may be able to confirm the impact of interactivity on structural perception and the overall effectiveness of the presented exposure techniques.

References

Appel, A. (1967). The notion of quantitative invisibility and the machine rendering of solids. In *ACM '67: Proceedings of the 1967 22nd national conference*, pages 387–393, New York, NY, USA. ACM.

Appel, A., Rohlf, F. J., and Stein, A. J. (1979). The haloed line effect for hidden line elimination. *Computer Graphics (Proceedings of SIGGRAPH 79)*, 13(3):151–157.

Barla, P., Thollot, J., and Markosian, L. (2006). X-toon: an extended toon shader. In *Proceedings of the 4th international symposium on Non-photorealistic animation and rendering*, NPAR '06, pages 127–132, New York, NY, USA. ACM.

Borgefors, G. (1986). Distance transformations in digital images. *Comp. Vision Graph. Image Process.*, 34(3):344–371.

Bremer, D. and Hughes, J. (1998). Rapid approximate silhouette rendering of implicit surfaces. In *Implicit Surfaces 98*, pages 155–164.

Bruckner, S., Grimm, S., Kanitsar, A., and Gröller, M. E. (2006). Illustrative context-preserving exploration of volume data. *IEEE TVCG*, 12(6):1559–1569.

Bruckner, S. and Gröller, M. E. (2005). VolumeShop: An interactive system for direct volume illustration. In *Proc. of IEEE Visualization '05*, pages 671–678. http://www.volumeshop.org/.

Bruckner, S. and Gröller, M. E. (2006). Exploded views for volume data. *IEEE TVCG*, 12(5):1077–1084.

Bruckner, S. and Gröller, M. E. (2007). Style transfer functions for illustrative volume rendering. *Computer Graphics Forum*, 26(3):715–724.

Burns, M. and Finkelstein, A. (2008). Adaptive cutaways for comprehensible rendering of polygonal scenes. *ACM Transactions on Graphics (Proc. SIGGRAPH ASIA)*, 27(5):124:1—-124:9.

Burns, M., Haidacher, M., Wein, W., Viola, I., and Gröller, E. (2007). Feature emphasis and contextual cutaways for multimodal medical visualization. In *Eurographics / IEEE VGTC Symposium on Visualization 2007*, pages 275–282.

Burns, M., Klawe, J., Rusinkiewicz, S., Finkelstein, A., and DeCarlo, D. (2005). Line drawings from volume data. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 24(3):512–518.

Choulant, L. (1920). *History and Bibliography of Anatomic Illustration*. The University of Chicago Press, Chicago, IL.

Cignoni, P., Marino, P., Montani, C., Puppo, E., and Scopigno, R. (1997). Speeding up isosurface extraction using interval trees. *IEEE Trans. on Visualization and Computer Graphics*, 3(2):158–170.

Coffin, C. and Hollerer, T. (2006). Interactive perspective cut-away views for general 3D scenes. In *Proc. of 3DUI 2006*, pages 25–28.

Cole, F., DeCarlo, D., Finkelstein, A., Kin, K., Morley, K., and Santella, A. (2006). Directing gaze in 3D models with stylized focus. In *Proc. of EGSR 2006*, pages 377–387.

Correa, C., Silver, D., and Chen, M. (2006). Feature aligned volume manipulation for illustration and visualization. *IEEE TVCG*, 12(5):1069–1076.

Csébfalvi, B., Mroz, L., Hauser, H., König, A., and Gröller, E. (2001). Fast visualization of object contours by non-photorealistic volume rendering. *Computer Graphics Forum*, 20(3):452–460.

DeCarlo, D., Finkelstein, A., and Rusinkiewicz, S. (2004). Interactive rendering of suggestive contours with temporal coherence. In *Third International Symposium on Non-Photorealistic Animation and Rendering (NPAR)*, pages 15–24.

DeCarlo, D., Finkelstein, A., Rusinkiewicz, S., and Santella, A. (2003). Suggestive contours for conveying shape. *ACM Transactions on Graphics*, 22(3):848–855.

DeCarlo, D. and Santella, A. (2002). Stylization and abstraction of photographs. *ACM Trans. Graph.*, 21:769–776.

Diepstraten, J., Weiskopf, D., and Ertl, T. (2003). Interactive cutaway illustrations. In *Proc. of Eurographics 2003*, pages 523–532.

Dong, F., Clapworthy, G. J., Lin, H., and Krokos, M. A. (2003). Nonphotorealistic rendering of medical volume data. *IEEE Computer Graphics and Applications*, 23(4):44–52.

Ebert, D. and Rheingans, P. (2000). Volume illustration: non-photorealistic rendering of volume models. In *IEEE Visualization 2000*, pages 195–202.

Feiner, S. and Seligmann, D. D. (1992). Cutaways and ghosting: satisfying visibility constraints in dynamic 3D illustrations. *The Visual Computer*, 8(5&6):292–302.

Fischer, J., Bartz, D., and Straßer, W. (2005). Illustrative display of hidden iso-surface structures. In *Proc. of IEEE Visualization '05*, pages 663–670.

Giesecke, F. E., Mitchell, A., Spencer, H. C., Hill, I. L., Dygdon, J. T., Novak, J. E., and Lockhart, S. (2008). *Technical Drawing*. Prentice Hall, 13th edition.

Haidacher, M. (2007). Importance-driven rendering in interventional imaging. Master's thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria.

Hauser, H. (2005). *Scientific Visualization: The Visual Extraction of Knowledge from Data*, chapter Generalizing Focus+Context Visualization, pages 305–327. Springer.

Hauser, H., Mroz, L., Bischi, G. I., and Gröller, M. E. (2001). Two-level volume rendering. *IEEE Trans. on Visualization and Computer Graphics*, 7(3):242–252.

Hertzmann, A. (1998). Painterly rendering with curved brush strokes of multiple sizes. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, pages 453–460, New York, NY, USA. ACM.

Hertzmann, A. and Zorin, D. (2000). Illustrating smooth surfaces. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 517–526.

Hodges, E. R. S., editor (2003). *The Guild Handbook of Scientific Illustration*. Addison-Wesley Professional.

Interrante, V., Fuchs, H., and Pizer, S. (1995). Enhancing transparent skin surfaces with ridge and valley lines. In *Proceedings of the 6th conference on Visualization '95*, page 52. IEEE Computer Society.

Interrante, V., Fuchs, H., and Pizer, S. (1996). Illustrating transparent surfaces with curvature-directed strokes. In *IEEE Visualization '96*, pages 211–218.

Interrante, V. and Grosch, C. (1998). Visualizing 3d flow. *IEEE Computer Graphics and Applications*, 18(4):49–53.

Judd, T., Durand, F., and Adelson, E. H. (2007). Apparent ridges for line drawing. *ACM Trans. Graph.*, 26(3):19.

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

Kindlmann, G., Whitaker, R., Tasdizen, T., and Moller, T. (2003). Curvature-based transfer functions for direct volume rendering: methods and applications. In *IEEE Visualization 2003*, pages 513–520.

Koenderink, J. J. (1984). What does the occluding contour tell us about solid shape? *Perception*, 13:321–330.

König, A., Doleisch, H., and Gröller, M. E. (1999). Multiple views and magic mirrors - fMRI visualization of the human brain. In *Proc. of SCCG'99*, pages 130–139.

Krüger, A., Tietjen, C., Hintze, J., Preim, B., Hertel, I., and Strauß, G. (2005). Interactive visualization for neck dissection planning. In *Proc. of EuroVis'05*, pages 295–302.

Krüger, J., Schneider, J., and Westermann, R. (2006). Clearview: An interactive context preserving hotspot visualization technique. *IEEE TVCG*, 12(5):941–948.

Li, W., Agrawala, M., Curless, B., and Salesin, D. (2008). Automated generation of interactive 3d exploded view diagrams. *ACM Trans. Graph.*, 27(3):1–7.

Li, W., Ritter, L., Agrawala, M., Curless, B., and Salesin, D. (2007). Interactive cutaway illustrations of complex 3d models. *ACM Trans. Graph.*, 26(3):31.

Liang, R., Clapworthy, G., Krokos, M., and Mayoral, R. (2005). Real-time predefined shape cutaway with parametric boundaries. *Computer Graphics, Imaging and Vision: New Trends, 2005. International Conference on*, pages 227–231.

Liu, Z., Finkelstein, A., and Li, K. (2002). Improving progressive view-dependent isosurface propagation. *Computers & Graphics*, 26(2):209–218.

Livnat, Y., Shen, H.-W., and Johnson, C. R. (1996). A near optimal isosurface extraction algorithm using the span space. *IEEE Trans. on Visualization and Computer Graphics*, 2(1):73–84.

Lorensen, W. and Cline, H. (1987). Marching cubes: A high resolution 3D surface construction algorithm. In *Proc. SIGGRAPH 1987*, pages 163–169.

Lu, A., Morris, C. J., Taylor, J., Ebert, D. S., Hansen, C., Rheingans, P., and Hartner, M. (2003). Illustrative interactive stipple rendering. *IEEE Trans. on Visualization and Computer Graphics*, 9(2):127–138.

Lum, E. B. and Ma, K.-L. (2002). Hardware-accelerated parallel non-photorealistic volume rendering. In *Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering (NPAR)*, pages 67–74. ACM Press.

Markosian, L., Kowalski, M. A., Trychin, S. J., Bourdev, L. D., Goldstein, D., and Hughes, J. F. (1997). Real-time nonphotorealistic rendering. In *Proceedings of SIGGRAPH 97*, Computer Graphics Proceedings, Annual Conference Series, pages 415–420.

McGuffin, M. J., Tancau, L., and Balakrishnan, R. (2003). Using deformations for browsing volumetric data. In *Proc. of IEEE Visualization 2003*, page 53.

Nagy, Z. and Klein, R. (2004). High-quality silhouette illustration for texture-based volume rendering. In Skala, V. and Scopigno, R., editors, *Proc. WSCG*, pages 301–308.

Nagy, Z., Schneider, J., and Westermann, R. (2002). Interactive volume illustration. In *Proc. Vision, Modeling and Visualization Workshop*.

Niederauer, C., Houston, M., Agrawala, M., and Humphreys, G. (2003). Non-invasive interactive visualization of dynamic architectural environments. In *I3D '03: Proceedings of the 2003 symposium on Interactive 3D graphics*, pages 55–58, New York, NY, USA. ACM.

Parker, S., Parker, M., Livnat, Y., Sloan, P.-P., Hansen, C., and Shirley, P. (1999). Interactive ray tracing for volume visualization. *IEEE Trans. on Visualization and Computer Graphics*, 5(3):238–250.

Rezk-Salama, C. and Kolb, A. (2006). Opacity peeling for direct volume rendering. *Comp. Graph. Forum*, 25(3):597–606.

Rong, G. and Tan, T.-S. (2006). Jump flooding in GPU with applications to voronoi diagram and distance transform. In *Proc. of I3D 2006*, pages 109–116.

Rost, R. J. (2006). *OpenGL(R) Shading Language (2nd Edition)*. Addison-Wesley Professional.

Rusinkiewicz, S., Burns, M., and DeCarlo, D. (2006). Exaggerated shading for depicting shape and detail. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 25(3).

Saito, T. and Takahashi, T. (1990). Comprehensible rendering of 3-d shapes. *SIGGRAPH Comput. Graph.*, 24(4):197–206.

Santella, A. and DeCarlo, D. (2002). Abstracted painterly renderings using eye-tracking data. In *Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*, NPAR '02, pages 75–ff, New York, NY, USA. ACM.

Schein, S. and Elber, G. (2004). Adaptive extraction and visualization of silhouette curves from volumetric datasets. *Vis. Comput.*, 20(4):243–252.

Straka, M., Červeňanský, M., Cruz, A. L., Köchl, A., Šrámek, M., Gröller, M. E., and Fleischmann, D. (2004). The VesselGlyph: Focus & context visualization in CT-angiography. In *Proc. IEEE Visualization '04*, pages 385–392.

Svakhine, N. A. and Ebert, D. S. (2003). Interactive volume illustration and feature halos. In *PG '03: Proceedings of the 11th Pacific Conference on Computer Graphics and Applications*, page 347. IEEE Computer Society.

Thirion, J.-P. and Gourdon, A. (1996). The 3d marching lines algorithm. *Graphical Models and Image Processing*, 58(6):503–509.

Treavett, S. M. F. and Chen, M. (2000). Pen-and-ink rendering in volume visualization. In *VISUALIZATION '00: Proceedings of the 11th IEEE Visualization 2000 Conference (VIS 2000)*. IEEE Computer Society.

van Kreveld, M., van Oostrum, R., Bajaj, C., Pascucci, V., and Schikore, D. (2004). Chapter 5: Contour trees and small seed sets for isosurface generation. In Rana, S., editor, *Topological Data Structures for Surfaces*, pages 71–86, Reading, Massachusetts. John Wiley & Sons, Ltd.

Viola, I., Feixas, M., Sbert, M., and Gröller, M. E. (2006). Importance-driven focus of attention. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):933–940.

Viola, I. and Gröller, M. E. (2005). Smart visibility in visualization. In *Proc. of Computational Aesthetics in Graphics, Visualization and Imaging*, pages 209–216.

Viola, I., Kanitsar, A., and Gröller, M. E. (2004). Importance-driven volume rendering. In *Proc. of IEEE Visualization 2004*, pages 139–145.

Viola, I., Kanitsar, A., and Gröller, M. E. (2005). Importance-driven feature enhancement in volume visualization. *IEEE Trans. on Visualization and Computer Graphics*, 11(4):408–418.

Wein, W., Röper, B., and Navab, N. (2005). Automatic registration and fusion of ultrasound with CT for radiotherapy. In *Proc. of MICCAI'05*, pages 303–311.

Weiskopf, D., Engel, K., and Ertl, T. (2003). Interactive clipping techniques for texture-based volume visualization and volume shading. *IEEE Transactions on Visualization and Computer Graphics*, 9(3):298–312.