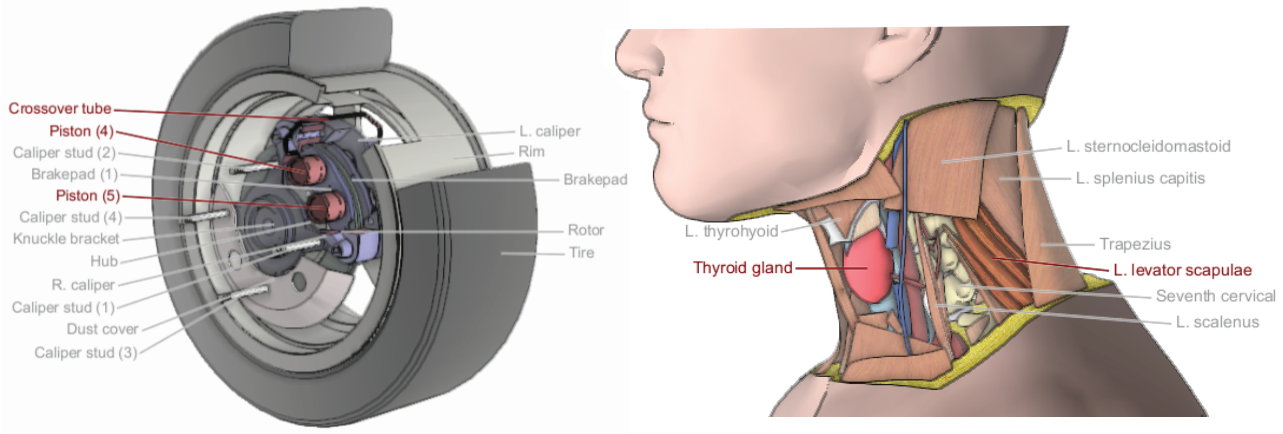
Interactive Cutaway Illustrations of Complex 3D Models

Wilmot Li[1] Lincoln Ritter[1] Maneesh Agrawala[2] Brian Curless[1] David Salesin[1,3]

[1]University of Washington [2]University of California, Berkeley [3]Adobe Systems



(a) Disk brake (b) Neck

Figure 1: *Cutaway views generated by our system. To create these illustrations, we "rigged" 3D models of a disk brake and human neck using the authoring component of our system. The illustrations were then generated automatically from the rigged models to expose user-selected target structures (shown in red).*

Abstract

We present a system for authoring and viewing interactive cutaway illustrations of complex 3D models using conventions of traditional scientific and technical illustration. Our approach is based on the two key ideas that 1) cuts should respect the geometry of the parts being cut, and 2) cutaway illustrations should support interactive exploration. In our approach, an author instruments a 3D model with auxiliary parameters, which we call "rigging," that define how cutaways of that structure are formed. We provide an authoring interface that automates most of the rigging process. We also provide a viewing interface that allows viewers to explore rigged models using high-level interactions. In particular, the viewer can just select a set of target structures, and the system will automatically generate a cutaway illustration that exposes those parts. We have tested our system on a variety of CAD and anatomical models, and our results demonstrate that our approach can be used to create and view effective interactive cutaway illustrations for a variety of complex objects with little user effort.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry & Object Modeling; I.3.8 [Computer Graphics]: Applications

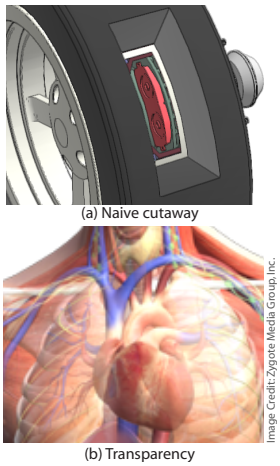
Keywords: cutaway illustration, interactive, visualization

ACM Reference Format
Li, W., Ritter, L., Agrawala, M., Curless, B., Salesin, D. 2007. Interactive Cutaway Illustrations of Complex 3D Models. *ACM Trans. Graph. 26*, 3, Article 31 (July 2007), 11 pages. DOI = 10.1145/1239451.1239482 http://doi.acm.org/10.1145/1239451.1239482.

1 Introduction

Complex geometric models composed of many distinct parts and structures arise in a number of domains, including medicine, engineering, and industrial manufacturing. Illustrations of such complex models are often essential for helping viewers understand the spatial relationships between the constituent parts that make up these datasets. Well-designed illustrations reveal not only the shape and appearance of important parts, but also the position and orientation of these parts in the context of the surrounding structures.

Yet creating illustrations that clearly depict the spatial relationships between parts is not an easy task. The primary problem is occlusion. Most complex 3D objects contain many tightly connected and intertwined parts that occlude one another. Thus, illustrators often use cutaways to reduce occlusions and expose important internal parts. However, naively cutting a hole through the occluding parts usually does not reveal the context of the surrounding structures. Although the cutaway in Figure 2a exposes the brakepad (in red), it does not show how the brakepad is situated with respect to nearby parts. Another approach is to increase the transparency of the occluding parts as in Figure 2b. While the resulting image reveals the complexity of the chest, it is extremely difficult to distinguish the layering relationships between the transparent parts, especially because there is more than one layer of transparency. The best cutaways clearly expose the target parts but also preserve some context from occluding structures so that viewers can better understand the spatial relationships between all parts (see Figure 1).



(a) Naive cutaway (b) Transparency

Figure 2: *Techniques for reducing occlusions.*

In this paper, we present a system for authoring and viewing interactive cutaway illustrations of complex 3D models. Our system is based on two key ideas:

Cuts should respect geometry of occluding parts. After analyzing a variety of well-designed cutaway illustrations [Hodges 1989; Netter 1989; Agur and Lee 2003; Biesty and Platt 1992], we have found that the most effective cuts are carefully designed to partially remove occluding parts so that viewers can mentally reconstruct the missing geometry. Thus, the shape and location of cuts depend as much on the geometry of the occluding parts as they do on the position of the target internal parts that are to be exposed. As we will show, illustrators use different cutting conventions depending on the geometry of the occluding structures. For example tubular structures are cut differently than rectangular parts. In addition, occluding structures that are farther away from the viewer are cut away less than parts that are closer to the viewer. Such insetting reveals the layers of occluders that hide the target parts. We instantiate these conventions in a set of cutting tools that can be used in combination to produce effective cutaway illustrations like those shown in Figure 1.

Cutaway illustrations should support interactive exploration. Static cutaway illustrations can reveal only a limited amount of information and may not always depict the structures of interest. Interactive control over the viewpoint and cutting parameters make it easier for viewers to understand the complex spatial relationships between parts. Yet low-level controls that force viewers to precisely position individual cutting tools (e.g., cutting planes) are tedious to use and assume viewers have the expertise to place the cuts in a way that reveals structures of interest. At the other end of the spectrum are pre-rendered atlases of anatomy [A.D.A.M. Inc. 2005; Höhne et al. 2003] that allow viewers to interactively control a few cutting parameters (e.g., sliding a cutting plane along a fixed axis), but do not allow users to freely navigate the model. We provide an interactive viewing interface that falls between these two extremes. Viewers are free to set any viewpoint and directly manipulate the model with the mouse to create, resize, and move cuts dynamically. However, these cuts are parameterized such that they always adhere to the conventions of traditional illustration, making it easy to produce useful cutaways. Alternatively, viewers can select a set of target parts, and the system will automatically generate a set of cuts from an appropriate viewpoint to reveal the specified targets.

Our work makes several new contributions. We identify a set of conventions for creating effective cutaway illustrations. In addition, we describe an approach for algorithmically encoding these conventions in a set of interactive cutaway tools. Finally, we demonstrate how our system can be used to create effective cutaways for a variety of complex 3D objects including human anatomy and mechanical assemblies.

2 Related work

There is a vast body of work on illustrative visualization algorithms. A recent report by Viola et al. [2005a] is an excellent overview of current techniques. We touch on previous techniques designed to expose internal structure.

Interactive cutting. Interactive techniques for cutting 3D models allow users to directly specify the cuts they would like to make. Höhne et al. [1992] limit users to placing cutting planes oriented along principal axes. Bruyns et al. [2002] survey a number of mesh cutting algorithms that allow users to directly draw the cut they would like to make. The goal of these algorithms is to simulate surgical tools such as scalpels and scissors. More recently, Igarashi et al. [1999] and Owada et al. [2003; 2004] have presented complete sketch-based systems for creating and cutting solid models. A drawback of such techniques is that there is little support for placing the cutting strokes. To expose a specific internal structure, users may have to cut the object multiple times just to find it. Moreover, the strokes have to be drawn very carefully to produce precise cuts.

Another approach is to interactively deform parts of the model to expose the internal structures. LaMar et al. [2001] and Wang et al. [2005] extend Magic Lenses [Bier et al. 1993; Viega et al. 1996] to perform non-linear deformations that push away outer structures and increase the relative size of important inner structures. However, such deformations can significantly distort the spatial relationships in the model. McGuffin et al. [2003], Correa et al. [2006], and Bruckner and Gröller [2006] have developed techniques for interactively cutting and deforming outer surfaces of volumetric models. However, these algorithms are designed primarily to expose perfectly layered structures in the volume, and as a result they cannot separate the intertwined structures often found in complex 3D datasets (e.g., anatomical models).

Automatic cutting. Many automatic techniques for cutting 3D models ask users to specify the important internal structures of interest and then automatically design the appropriate cuts to reveal them. Feiner and Seligmann [1992] and Diepstraten et al. [2003] have demonstrated such importance-based automatic cutting for surface models, while Viola et al. [2005b] apply a similar approach to volumetric data. VolumeShop [Bruckner and Gröller 2005] extends the latter approach into a complete volume illustration system with support for insets and labeling. In all of these systems the cuts are completely based on the geometric shape of the important parts. Because the occluding parts and structures have no influence on the cutting geometry, it can be difficult for viewers to reconstruct the spatial relationships between the structures that are removed and the remaining parts.

Rendering and shading. Many algorithms have focused on simulating illustrative rendering styles [Dooley and Cohen 1990; Ebert and Rheingans 2000; Lum and Ma 2002; Burns et al. 2005]. We borrow the approach of rendering silhouette lines [Gooch et al. 1998] to make it easier to differentiate structures from one another. Tietjen et al. [2005] and, more recently, Cole et al. [2006], highlight important structures in illustration by rendering them in a different style from the less important structures. We similarly use brighter colors to emphasize selected parts. Illustrators often use inconsistent lighting to emphasize shape and surface features [Akers et al. 2003]. Based on this idea researchers have developed synthetic shading techniques [Lee et al. 2004; Cignoni et al. 2005; Rusinkiewicz et al. 2006] that enhance depth discontinuities and surface orientation, We borrow the technique of Luft et al. [2006] to emphasize depth discontinuities and better reveal the layering of parts. We also develop a new shading technique to emphasize the orientation of the surfaces that are exposed after a cut.

3 Overview

Our system consists of two components. The *authoring interface* allows an author to equip a 3D geometric model with additional information (or *rigging*) that enables the formation of dynamic cutaways. The *viewing interface* takes a rigged model as input and enables viewers to explore the dataset with high-level cutaway tools. Although we do not make any assumptions about the viewer's familiarity with the object, we assume the author knows the structure of the object well enough to specify the geometric type of each part (e.g., tube, rectangular parallelepiped etc.) and to identify good viewpoints for generating cutaway illustrations.

The remainder of this paper is organized as follows. We summarize common illustration conventions that characterize effective cutaway visualizations (Section 4), before presenting the techniques and algorithms we developed to incorporate these conventions into our interactive visualization system (Section 5). We then outline the typical workflow required for a human author to specify this rigging using our authoring interface (Section 6) and describe the interaction modes provided by our viewing interface for exploring rigged models (Section 7).

4 Conventions from traditional illustration

Effective cutaway illustrations exhibit a variety of conventions that help emphasize the shape and position of structures relative to one another. However, books on scientific and technical illustration techniques [Hodges 1989; Loechel 1964; Wood 1979; Zweifel 1961] mainly present low-level drawing and rendering methods. Thus, to identify cutaway conventions, we worked with scientific illustrators and analyzed illustrations from well-known anatomy atlases [Netter 1989; Agur and Lee 2003], technical manuals [Hoyt 1981; Elliott et al. 1924], and books on visualizing complex buildings and machinery [Biesty and Platt 1992; Biesty and Platt 1993]. Despite differences in artistic style, we noted a number of similar cutting techniques across these sources.

4.1 Geometry-based conventions

The geometric shape of a part often determines the most appropriate cutting strategy. We consider several common shapes and the cutting conventions appropriate to each of them.

Object-aligned box cuts. Illustrators often use box cuts that are aligned with the principal Cartesian coordinate frame of a part. For man-made objects the shape of the part usually implies the orientation of this frame. Such objects are typically designed with respect to three orthogonal principal axes and in many cases they resemble rectangular solids. Aligning a box cut to the principal axes of a part helps to accentuate its geometric structure (Figure 3) and makes it easier to infer the shape of the missing geometry.

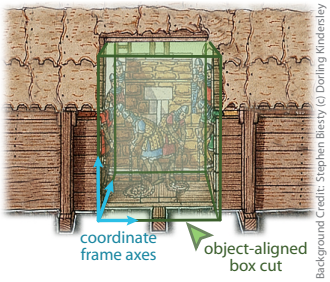


Figure 3: *Object-aligned box cut.*

In some domains, models are oriented with respect to a canonical coordinate frame. For example, in anatomy the canonical frame is defined by the saggital, coronal and transverse viewing directions. Box cuts are often aligned with these canonical axes to produce cross-sections that are familiar to viewers who have some experience with the domain.

Tube cuts. 3D models of both biological and man-made objects contain many structures that resemble tubes, either because they exhibit radial symmetry (e.g., pipes, gears), or because they are long and narrow (e.g., long muscles and bones, plumbing). In cutting such parts, illustrators usually align the cut with the primary axis running along the length of the part. Often, illustrators will remove a section of the structure using a *transverse* cutting plane that is perpendicular to the primary axis (Figure 4). The orientation of the cutting plane provides a visual cue about the direction in which the tube extends and thereby helps viewers mentally complete the missing part of the tube.

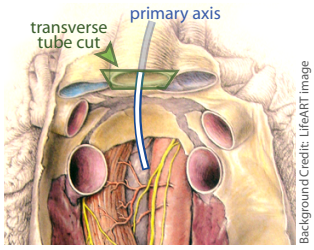


Figure 4: *Transverse tube cut.*

A second variant of the tube cut removes a *wedge* from the object (Figure 5) to expose interior parts while removing less of the tube than the transverse tube cut. Since more of the exterior structure remains visible, the viewer has more context for understanding the tube's position and orientation in relation to the exposed internal parts. In addition, the wedge shape of the cut emphasizes the cylindrical structure of the tube and makes it easier for the viewer to mentally reconstruct the missing geometry. Wedge cuts are typically used for radially symmetric (or nearly radially symmetric) parts.

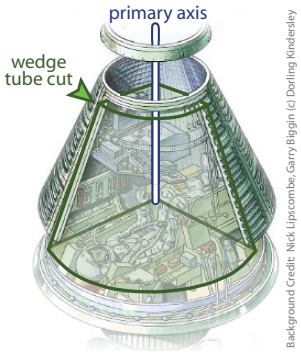


Figure 5: *Wedge tube cut.*

Window cuts. Many complex 3D models include thin extended enclosing structures (e.g., skin, the chassis of a car) that occlude much of the model's internal detail. To expose internal parts, illustrators often cut windows out of these structures.

The window boundaries can also provide a useful cue about the shape of the enclosing structure. Boundary edges near silhouettes of the object help emphasize these contours (Figure 6). Similarly, boundary edges that are further from silhouettes often bend according to surface curvature. Another convention, usually reserved for technical illustrations, is to make the window jagged. This approach emphasizes that a cut was made and distinguishes the boundaries of the cut from other edges in the scene. All three of these boundary conventions help viewers mentally reconstruct the shape of the enclosing structure.

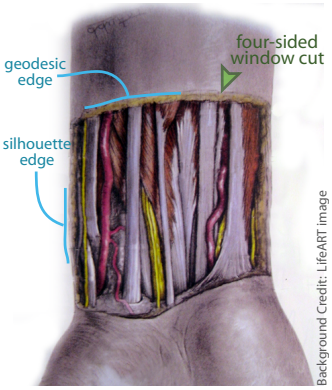


Figure 6: *Four-sided window cut.*

4.2 Viewpoint conventions

Illustrators carefully choose viewpoints that help viewers see the spatial relationships between the internal target parts they are interested in and the occluding parts. Typically, the viewpoint not only centers the target parts in the illustration, but also minimizes the number of occluding structures. This strategy makes it possible to expose the parts of interest with relatively few cuts, leaving more of the surrounding structures intact for context. In addition, illustrators often choose domain-specific canonical views, such as the saggital, coronal and transverse views used for human anatomy.

4.3 Insetting cuts based on visibility

Some models contain many layers of occluding structures between the internal target part and the exterior of the model. Illustrators often reveal such layering by *insetting* cuts. Occluded parts are cut to a lesser extent than (i.e., inset from) occluding parts, thus organizing the structures into a series of terraces that accentuate the layering (see Figure 7). Since the layering of the parts depends on viewpoint, such inset cuts are viewpoint dependent.

4.4 Rendering conventions

Shading is a strong cue for conveying surface orientation and depth relationships between structures. Illustrators often exaggerate shading to emphasize object shape. We describe two such illustrative shading techniques that are shown in Figure 7.

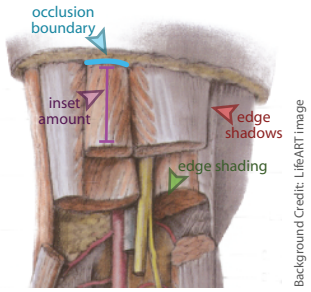


Figure 7: *Inset cuts and illustrative shading techniques.*

Edge shadows. Cast shadows provide strong depth cues at discontinuity edges. However, physically accurate shadows may also darken and obscure important parts. Instead, illustrators often darken a thin region of the far surface along the discontinuity edge. Such edge shadows [Francis 1987] pull the near surface closer to the viewer while pushing back the far surface. The width and softness of edge shadows usually vary with the discrepancy in depth at the discontinuity edge; overlapping structures that are close to one another have a tighter, darker shadow than structures that are farther apart.

Edge shading. While simple diffuse shading provides information about surface orientation, it can also cause planar surfaces that face away from the light source to be rendered in a relatively dark shadow, making it difficult to see surface detail. Some illustrators solve this problem by darkening only the edges of the cut surface to suggest diffuse shading [Hodges 1989]. As with edge shadows, the width and softness of the darkened region may vary, but in general, the overall darkness depends on the surface orientation.

5 Implementation of illustration conventions

In this section, we introduce a parametric representation for cutaways of individual parts that makes it easy to create the different types of cuts outlined in Section 4.1. We then describe how our system determines good views for exposing user-selected target structures, based on the conventions discussed in Section 4.2. Next, we explain our constraint-based approach for generating view-dependent inset cuts, as described in Section 4.3. Finally, we present simple rendering algorithms for generating the effects discussed in Section 4.4.

5.1 Cutaway representation

The input to our system is a 3D solid model in which each part or structure is a separate geometric object. Our system cuts the model by removing volumetric regions – *cutting volumes* – using CSG (constructive solid geometry) subtraction. Each volume cuts exactly one part. As shown in Figure 8, we parameterize each of the conventional cuts described in Section 4.1 using a 1D, 2D, or 3D parameter space that is mapped to a volume in model space (i.e., the local coordinates of the structure to be cut). Under these mappings, cutting volumes are specified as simple parametric ranges that get transformed to model space cutting volumes, which we represent as polyhedral solids.

Object-aligned box cutting volumes

For object-aligned box cuts, we map a 3D parameter space u, v, w to three orthogonal model space axes, u', v', w' (Figure 8a). Under this mapping an axis-aligned box in parameter space transforms to an axis-aligned box in model space.

As noted in Section 4.1, object-aligned cuts are often oriented along the principal axes of a part. To compute these axes, the system samples the convex hull of the object and then performs PCA on the resulting point cloud to determine the three principal directions of variance. We also allow users to directly specify three orthogonal model space axes as the principal axes.

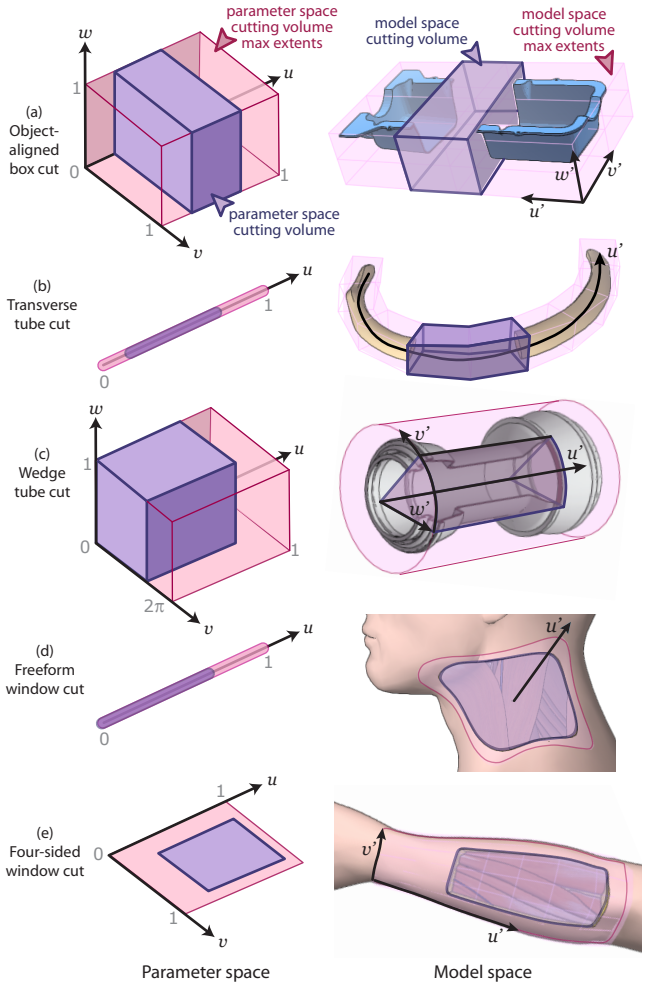


Figure 8: *Cutting volumes. In our system, a cut is represented as a cutting volume that is defined in a one, two, or three dimensional parameter space (left column) and then mapped to the model's local coordinate system (right column). For each type of cut, the cutting volume (purple) and its maximum extents (pink) are shown in both parameter and model space. The model space mappings u', v', w' of the parametric dimensions u, v, w are also illustrated on the right.*

Tubular cutting volumes

Tube cuts are generally defined using a 3D parameter space. The u axis in parameter space maps to the primary axis of the tube. The v and w axes then map to the angular and radial components of a polar coordinate system defined on the normal plane of the Frenet frame of the primary axis (Figure 8c). Under this mapping, an axis-aligned u, v, w box corresponds to a cutting volume that removes a wedge from the structure. To create transverse cutting planes, the v and w ranges are set to their maximum extents. Thus, transverse tube cuts are fully parameterized using a 1D parameter space (Figure 8b).

To compute tubular cutting volumes efficiently from their parametric representation, the system precomputes the primary axis for each tubular part. At runtime, the system constructs the cutting volume for a transverse cut by extruding a rectangle at uniformly

spaced intervals along the specified portion of the axis. At each point along the axis, the rectangle is placed in the normal plane and made large enough to enclose the tube's cross-section. To form a polyhedral solid, the system creates faces connecting the edges of consecutive rectangles and caps the cutting volume at both ends. Wedge cutting volumes are computed in a similar fashion by extruding a planar wedge along the primary axis.

Our system computes the primary axis for long, narrow parts using the *skeletal curve* construction of Verroust and Lazarus [2000]. They define a skeletal curve as the locus of centroids of the level sets of surface points computed with respect to their geodesic distance from an extremal point on the surface. To find an extremal point, the algorithm picks an arbitrary query point on the surface and then selects the point that is furthest away from it, in terms of geodesic distance. The surface vertices are grouped into level set intervals of increasing geodesic distance from the extremal point, and the centroids of the vertices within each interval are connected to form a piecewise linear skeletal curve, which runs along the length of the tube and roughly passes through its center. The system computes Frenet frames at the vertices of this curve using discrete derivatives.

For short, radially symmetric tubes, the skeletal curve is usually not a good approximation of the primary axis (i.e., the axis of radial symmetry). For such structures, the system samples the convex hull of the part and then computes the cylinder that best fits (in a least-squares sense) the resulting point cloud. The user can also override the automatic computation and directly specify the primary axis. In this case the user orients the model so that the view direction coincides with the desired axis, and then clicks the point where the axis should be positioned. In this mode, the author can tell the system to snap the orientation of the specified axis to run parallel to a nearby canonical axis. Users can also snap the position of the axis so that it intersects the centroid of the entire model or the centroid of any individual part.

Window cutting volumes

The mapping for window cuts is defined with respect to a closed *bounding curve* on the surface of the part we wish to cut. The bounding curve represents the maximum extents of the cutting volume. Our system supports two variants of window cuts.

Freeform window cuts are defined by a bounding curve and a single parameter u that represents the size of the window (Figure 8d). Each value of u defines a window boundary that is computed by eroding the bounding curve along the surface of the part toward the curve's centroid, using u as the erosion parameter. The erosion is performed by moving curve points inwards along the curve's normal direction and then smoothing the curve (by averaging nearby points) to help eliminate self-intersections. At $u = 0$, the bounding curve is fully eroded and the window is fully closed. At $u = 1$, the window is fully open. To accelerate the computation of this mapping, the system precomputes bounding curves at uniformly spaced u values (Figure 9) and then interpolates between them at runtime. From a given window boundary, the system forms a polyhedral cutting volume by first triangulating the region R of the surface enclosed by the window boundary, and then creating a second copy of these triangles. The two sheets of triangles are then offset a small amount (we use the average triangle edge length of the surface) in opposite directions normal to the surface so that they lie on either side of R. Finally, the sheets are sewn together with triangles to form a closed cutting volume that completely encloses R.

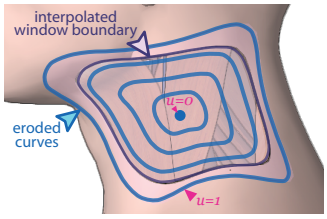


Figure 9: *Precomputed bounding curves.*

Four-sided window cuts are defined by a bounding curve that is partitioned into four segments, as shown in Figure 8e. Each segment is an arc-length parameterized curve (normalized to a total length of one) corresponding to one of the four surface curves along $u = 0$, $u = 1$, $v = 0$, and $v = 1$, as illustrated in Figure 10. For a given u,v point, the corresponding model space point is the intersection of two geodesic paths, one connecting the model space points at $(u,0)$ and $(u,1)$, the other connecting points at $(0,v)$ and $(1,v)$. Thus, a rectangle in parameter space is mapped to a window boundary in model space by transforming the four corners into model space points which are then connected using geodesics. As with freeform window cuts, the system uses the four-sided boundary to construct a closed polyhedral cutting volume.

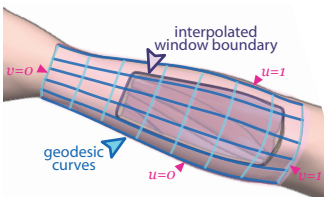


Figure 10: *Precomputed grid of geodesic paths.*

To accelerate the mapping for four-sided window cuts, the system precomputes a grid of geodesic paths that connect the two pairs of opposing segments at uniformly spaced values of u and v, as shown in Figure 10. By interpolating within this grid, the system can quickly compute a model space window boundary from its parametric representation without having to compute geodesics at runtime.

For both freeform and window cuts, the user can draw the initial, outermost bounding curve directly on the surface of the part. We also provide more automated interfaces for specifying these curves. The system can automatically compute a freeform window boundary by first determining the regions of the surface that occlude underlying parts (with respect to the current viewpoint), and then computing a closed window boundary that completely encloses all of these occluding regions. For four-sided window boundaries, the user can sketch two curves that roughly denote the extent of the window. The system projects the sketched curves onto the structure (again using the current viewpoint) and discards the portions that do not project onto the surface. The system then computes the four bounding curve segments by connecting the endpoints of the projected curves using paths that either follow geodesics or hug the silhouette of the surface, as shown in Figure 11. A silhouette curve is chosen only if the corresponding endpoints are near the silhouette and the length of the silhouette curve is within a threshold of the geodesic distance between the endpoints. Otherwise, the endpoints are connected with the geodesic path.

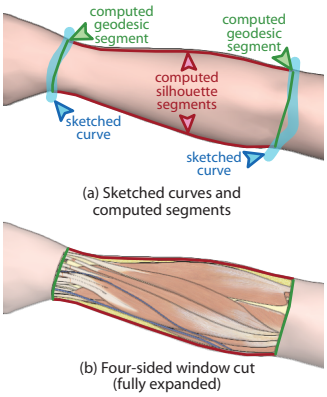


Figure 11: *Sketch-based bounding curve construction.*

5.2 Viewpoints

To generate effective cutaway illustrations, our system asks the author to specify a set of good candidate views during the rigging process. When the model is examined using our viewing interface, the system chooses amongst the candidates to determine the best view for exposing the user-selected set of target structures. The

view selection algorithm takes into account the layering relationships between parts, which we encode as an occlusion graph that is computed for each viewpoint. The system also uses the occlusion graph to create inset cuts (Section 5.3).

5.2.1 Occlusion graph

To compute an occlusion graph from a given viewpoint, the system first assigns each part to a visibility layer based on the minimum number of occluding structures that must be removed before the part in question becomes sufficiently visible. This assignment is performed using an iterative procedure. We begin by initializing the current set of parts S to include all the parts in the model. We then repeat three steps until all the parts have been placed in a layer:

1. Render all parts in S
2. Create new layer containing visible parts
3. Remove visible parts from S

In step two, we use an approximate visibility test whereby a part is considered visible if its *visibility ratio* (i.e., the ratio of its visible screen space to its maximum unoccluded screen space) is greater than an author-specified threshold. For most of our examples, we set this threshold to 0.9, which ensures that parts that are mostly, but not entirely visible are placed in the same layer as parts that are fully visible. If no parts are sufficiently visible to be added to the current visibility layer, the algorithm adds the part with the highest visibility ratio. Figure 12c illustrates the visibility layers for an example model.

After computing the layering, the system computes an occlusion graph. For each part P, the algorithm steps through the visibility layers starting from the layer containing the part and moving towards the viewer. In each layer, the system identifies parts that directly occlude P with no intervening occluding structures. As with the visibility test described above, we use an approximate notion of occlusion whereby a part is identified as an occluder if it occludes more than 20% of P's visible screen space area. For each occluder, a directed edge from the occluder to P is added to the graph. As shown in Figure 12d, this construction guarantees that edges flow only from shallower to deeper visibility layers resulting in a directed acyclic graph (DAG).

5.2.2 View selection

For a given set of target structures, the system evaluates each saved viewpoint v using a simple metric that considers the projected screen space pixel area a_v and the average number of occluders o_v for the targets. To compute a_v, the system renders just the target parts from the given viewpoint and counts the number of corresponding pixels. To compute o_v, the system considers each target and traverses the occlusion graph upwards (i.e., against the flow of the edges) starting at the node corresponding to the target part. Since the graph is a DAG, this traversal will not encounter any cycles. The system counts the number of traversed nodes and then computes the average number of occluders for all the targets.

Given these definitions, we define the energy E_v of a viewpoint v as

$$E_v = 2o_v + \frac{A_{max} - a_v}{A_{max}}, \tag{1}$$

where the normalization constant A_{max} is the size of the application window. In accordance with the conventions described earlier, our energy function penalizes views that result in more occluders. The o_v term is scaled by two so that the number of occluders always takes precedence over the projected screen space area of the targets. If o_v is the same for two views, the area term breaks the tie. After computing E_v for each view, the system chooses the minimum energy viewpoint.

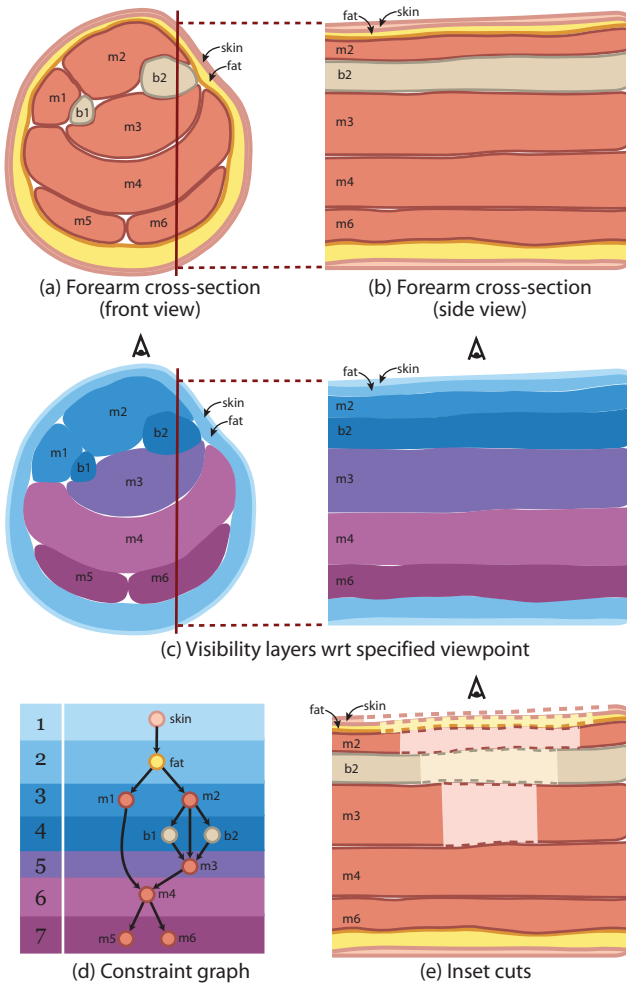
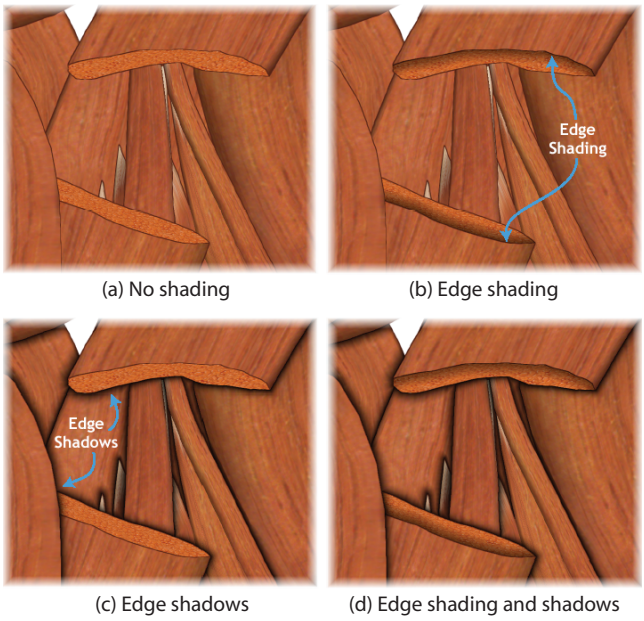


Figure 12: *Computing constraint graph. (a)-(b) Forearm cross-section viewed from front and side. (c) Structures sorted into visibility layers based on depth complexity from specified viewpoint. (d) Directed edges from occluders to occluded parts are added to graph. Edges only flow from lower to higher layers. (e) Nodes are sorted in topological order to produce a set of inset cuts.*

5.3 Inset constraints

To generate inset cuts from a given viewpoint, the system applies an inset constraint at each edge of the occlusion graph. Given the model space cutting volume C of a part, the cutting volume of each occluded part is constrained to be smaller than C; conversely, the cutting volume of each occluder is constrained to be larger than C. Our system supports the propagation of constraints both up and down the graph to enable the constrained manipulation of cuts, as described later in Section 7.1. Propagating constraints in both directions allows the system to maintain insets for both the occluded and occluding structures of the part being manipulated. Based on the structure of the graph, inset constraints can produce a cascade of inset cuts, as shown in Figure 12e.

Our system evaluates an inset constraint as follows. Let $b2$ and $m3$ be two structures, where $m3$ is constrained to be inset from $b2$ (see Figure 12e). To update $m3$'s cutting volume given $b2$'s cutting volume, the system determines the unoccluded region of $m3$ and computes the smallest model space cutting volume (i.e., the tightest cutting parameter ranges) for $m3$ that encloses this region. These cut-



(a) No shading (b) Edge shading

(c) Edge shadows (d) Edge shading and shadows

Figure 13: *Illustrative shading techniques. Some depth relationships are difficult to interpret when the scene is rendered with a standard lighting model (a). Edge shading helps convey the orientation of cutting surfaces (b). Edge shadows disambiguate depth relationships at occlusion boundaries (c). Combining these techniques produces a more comprehensible depiction (d).*

ting parameters are then shifted inwards by a specified inset amount to ensure that a portion of $m3$ is visible behind $b2$. By default, we set the inset amount to be 5% of the largest dimension of the cutting volume's maximum model space extents. This amount is mapped to parameter space in order to compute the inset. To evaluate the constraint in the opposite direction, the system uses a similar method to ensure that $b2$'s cutting volume is expanded enough to reveal a portion of $m3$.

5.4 Rendering algorithms

Our system renders cutaways using OpenCSG [Kirsch and Döllner 2005], a freely available hardware-accelerated CSG library. If the input dataset includes 3D textures that model the appearance inside of parts (e.g., muscle fibre), these textures are mapped onto cut surfaces to add visual detail to cross-sectional faces.

In addition, we have implemented several stylized rendering techniques that can make it easier for viewers to understand the shape and orientation of parts as well as the layering between the parts. The first two techniques, edge shadows and edge shading, take several seconds to compute and therefore are only produced when the users asks for a high-quality rendering. Real-time depth-cueing and label layouts are computed at interactive rates in our viewer.

Edge shadows. To render edge shadows, our system scans the depth buffer to identify pixels that lie on the near side of depth discontinuities (i.e., the side that is closer to the viewer). For each of these discontinuity pixels, the system determines the set of neighboring pixels that lie on the far side of the discontinuity and identifies them as being in the edge shadow. Our system then darkens each edge shadow pixel based on its image space distance d_{img} and depth discrepancy d_{depth} from the near side depth discontinuity pixel. These values are clamped to $[0, D_{img}]$ and $[0, D_{depth}]$ respectively. Typically, we set D_{img} to 10 pixels and D_{depth} to 0.5% of the entire model's average bounding box dimension. For each shadow pixel, the system computes a scale factor γ that is multiplied with the pixel's current color.

$$\gamma = s * \left(\frac{|D_{img} - d_{img}|}{D_{img}} \right)^t \quad \text{where} \quad \begin{array}{rcl} s & = & (1-\alpha) * s_{max} - \alpha * s_{min} \\ t & = & (1-\alpha) * t_{max} - \alpha * t_{min} \\ \alpha & = & \frac{d_{depth}}{D_{depth}} \end{array}$$

The parameter s scales the overall amount of darkening linearly, and t controls how quickly the shadow falls off with respect to d_{img}. Both s and t are restricted to user-specified ranges ($[s_{min}, s_{max}]$ and $[t_{min}, t_{max}]$ respectively), and they both get smaller as d_{depth} gets larger. By default, we set the range for s to $[0.6, 0.9]$ and for t to $[1, 3]$. As a result, edge shadows are darker and tighter where the depth discontinuity is small and are more diffuse where the depth discontinuity is large.

Edge shading. To generate edge shading, our system identifies pixels that correspond to creases (in model space) where two faces of the same cutting volume meet, or where a cutting surface meets the exterior surface of a part. For each of these crease pixels, the system determines the set of neighboring pixels that correspond to a cutting surface and identifies them as edge shading pixels. The system computes diffuse shading for each such edge shading pixel by looking up surface normals that are stored in a corresponding per-pixel normal map. The pixel's color is then determined by interpolating between its unshaded (i.e., fully lit) color and its diffuse shaded value. The interpolation is performed with respect to d_{img} so that the edge shading fades farther away from the edge.

Real-time depth-cueing. Our system supports two real-time depth-cueing effects that make it easier to distinguish parts from one another and to see how they layer in depth. To emphasize silhouettes and sharp creases, the system first detects these contours by finding edges in the depth buffer (for silhouettes) and a normal map (for creases). These edges are then darkened in the final shaded rendering of the scene. To emphasize the relative depth of parts in the model, the system dims and desaturates objects proportional to their distance from the viewer. Both effects are implemented as fragment shaders on the GPU.

Label layout. To generate label layouts in real-time we have implemented the labeling algorithm of Ali et al. [2005]. Their approach organizes labels into two vertical columns on the left and right of the illustration and then stacks them in a way that avoids intersections between leader lines. To make the layout more compact, the system positions labels near the silhouette of the object. Figures 1,15, and 16 show label layouts produced by our system.

6 Authoring workflow

To create an interactive cutaway illustration, the author rigs the input model using our authoring interface. The rigging process has two stages. First, each structure is instrumented with the appropriate mapping. Second, the author identifies good viewpoints for generating cutaway views of the model.

To specify the mappings, the author first categorizes each structure as either a rectangular parallelepiped, a long narrow tube, a short radially symmetric tube, or an extended shell. Based on these categories, the system automatically constructs the appropriate mapping for every part in the dataset using the automatic techniques described in Section 5.1. This operation usually takes just a few seconds per part.

After examining the results of the automatic computation, the author decides which mappings he wants to edit. These modifications can be performed quickly using the interactive tools described in

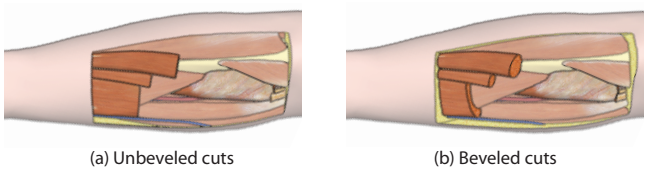


Figure 14: *Beveled cuts. The viewer adjusts the amount of beveling to examine the cut surfaces of three tubular muscles and the surrounding fat.*

Section 5.1. In practice, however, this type of manual editing is rarely necessary. As reported in Section 8, the automatically constructed mapping was used for 90% of the parts for the datasets shown in this paper.

Next, the author specifies good views for generating cutaway illustrations. To save a view, the author simply manipulates the model to the desired orientation and then clicks a button. The system automatically constructs an occlusion graph, as described in Section 5.2.1, and associates the graph with the specified viewpoint. In many cases, we envision the author would have enough domain knowledge to identify good viewpoints *a priori*. However, the author can also evaluate a viewpoint by generating cutaways of a few parts from that view using the automatic cutaway generation interface described in the next section. Typically, two or three views are sufficient to produce effective cutaway illustrations of all the parts in the model.

7 Viewing interactive cutaway illustrations

Our viewing interface allows the viewer to interactively explore a rigged model using both direct manipulation and higher-level interaction modes.

7.1 Direct manipulation

Our system allows users to directly resize and move cuts. Dragging with the left mouse button resizes the cut by snapping the nearest cutting face to the surface point directly below the mouse position. Dragging with the middle mouse button slides the cut without changing its extent, and holding down a control key constrains the direction of motion to just one of the free parameter dimensions.

For complicated models, directly manipulating cuts one at a time can become tedious. Activating inset constraints allows users to make multiple cuts at once. In this mode when the viewer resizes or moves a cutting volume, the system updates all the cuts by enforcing the inset constraints in both directions throughout the occlusion graph, starting from the part being manipulated. This two-way propagation of constraints ensures that the insets (and thus, the layering relationships) both above and below the manipulated part remain clear during the interaction.

The user can also expand or collapse cuts just by clicking on a structure. In one mode, the system smoothly opens or closes the cutting volume of the clicked structure as much as possible with respect to the inset constraints. In another mode, the system cuts away or closes entire layers of structures either above or below the clicked part in the occlusion graph. Expanding cuts in this manner allows the viewers to quickly expose underlying parts. Fast, animated transitions help emphasize the layering relationships between structures.

Our system also allows the viewer to *bevel* (i.e., adjust the angle of) cuts to expose cross-sectional surfaces, as shown in Figure 14. The viewer controls the amount of beveling by adjusting a single slider. In response, the system rotates the faces of the selected cutting volumes towards the viewer.

7.2 Automatic cutaway generation

Although direct interaction enables an immediate exploration experience, in some situations a higher-level interface may be preferable. For instance, a mechanic might want to see all of the bolts that fasten two parts together, or an anatomy student might want to examine a particular muscle without knowing where it is located. To support such exploration, we developed an automated algorithm for generating cutaways that expose user-specified target structures.

Once the viewer has selected a set of target parts from a list, the system chooses a viewpoint using the view selection algorithm described in Section 5.2.2 and then solves for a set of cutting parameters in the following way. The model space cutting volume for each part above a target structure in the occlusion graph is fully expanded subject to the inset constraints. Conversely, the cuts for the target structures and all of their descendants are completely closed. This initialization step exposes the target structures as much as possible. Next, to provide some context from occluding structures, the algorithm works upwards from the leaves of the occlusion graph, visiting each non-target part and closing its corresponding cutting volume as much as possible subject to the inset constraints and without occluding any portion of any target structure.

To make it easier for the viewer to see which parts must be cut to form the final solved cutaway, the system smoothly animates cuts by interpolating cutting parameters from their current values. In the current implementation, the system animates all of the cuts simultaneously. As future work, we plan to investigate techniques for animating the cuts in sequence based on the visibility layers of the parts in order to emphasize layering relationships.

Once the cuts have been updated, the system partitions the model into connected components of touching parts and then detects small components that have been separated from the rest of the model by the cuts. Since it is difficult for a viewer to interpret how such components attach to the rest of the model, they are removed from the illustration. For the examples shown, we used a threshold of 10% of the volume of the entire model's bounding box to identify "small" components. Finally, the system highlights the target structures by desaturating the other objects and labels the illustration.

8 Results

We have tested our system on a variety of 3D models as shown in Figures 1 and 15. All of these results are rendered with edge shadows, edge shading, depth-cueing and labels. In each cutaway visualization, the orientation and insetting of the cuts help convey the geometry and layering of the parts that have been removed. For example, the wedge cuts in Figure 15c that remove the occluding portions of the turbine's nose cone and outer shell also accentuate the radial symmetry of these parts. In Figure 1b, the insetting of the long tubular muscles emphasizes the fact that there are multiple layers of muscle in the neck.

To evaluate the operating range of our system, we automatically generated cutaway illustrations that expose each part of several models. Figure 16 shows four of the 27 results for the disk brake model. Out of the 27 illustrations, we found artifacts in only two, one of which is shown on the right of Figure 16. In these two cases, on close inspection we found that the cuts generated by our system to expose the target part also leave it disconnected from the surrounding parts, making it difficult for viewers to understand how the target relates spatially to its neighbors.

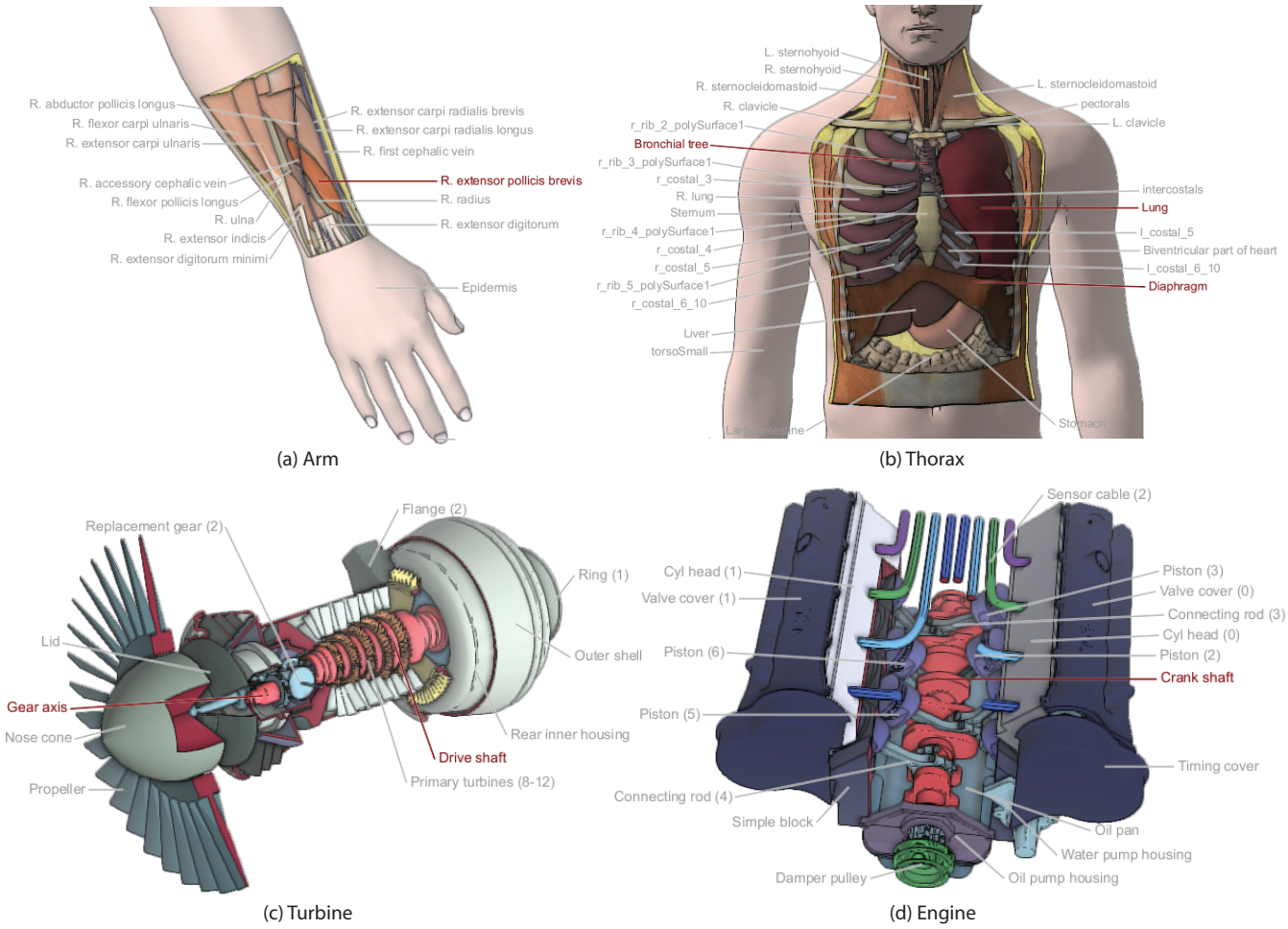


Figure 15: *Illustrations generated using our automatic cutaway generation interface. A combination of window and transverse tube cuts removes the layers of skin, muscle and bone that occlude the target muscle in (a) and the internal organs of the thorax in (b). Extruded angle tube cuts expose target parts within the turbine (c). Axis-aligned and transverse tube cuts reveal the engine's crank shaft (d). In (c) and (d), cut surfaces are highlighted in red.*

Model	N_{long}	N_{short}	N_{rect}	N_{shell}	N_{auto}	T_{rig}
Disk brake	1	24	2	0	24	3m
Turbine	0	37	3	0	36	5m
Engine	22	84	28	0	122	20m
Thorax	39	0	6	4	37	10m
Arm	20	0	0	1	20	3m
Neck	58	0	0	1	58	8m

Table 1: *Rigging statistics. For each model, we report the number of long tubes N_{long}, short radially symmetric tubes N_{short}, rectangular parallelepipeds N_{rect}, and shells N_{shell}. We also indicate the number of parts for which the automatically computed mapping was used in the final rigged model N_{auto}. Finally, we report the user time (in minutes) required to rig each model T_{rig}.*

In terms of authoring effort, none of the models took more than 20 minutes to rig (see Table 1). The most time-consuming step was identifying the type of cut to use for each part in the dataset. However, even for the engine model, which contains well over one hundred parts, this entire process took less than 5 minutes.

As shown in Table 1, the system automatically computes good mappings for 90% of the parts. For the remaining 10% of parts, we were able to specify a suitable mapping interactively in just a few seconds. In most cases, this manual rigging was used to create wedge cut mappings for parts that are clearly derived from cylinders but exhibit a low degree of radial symmetry (e.g., parts that resemble incomplete cylinders). In a few cases, we manually oriented the box cut mappings for parts whose principal axes do not correspond to the desired box cut axes. Finally, for some parts, we overrode the automatically computed window cut mappings using our system's sketch-based rigging tools, which give the author more control over the shape of the window boundary.

Although we have not conducted a formal user study, we have demonstrated our system to medical educators and illustrators, architects, and airplane engineers. All of the people we spoke with were excited about the system's ability to effectively expose the internal structure of complex 3D datasets. Furthermore, the medical educators we interviewed said they would be willing to invest some of the effort that they typically spend preparing class materials to creating interactive cutaway illustrations that could be used as teaching aids and study guides. This informal feedback suggests that our system could be a useful and practical tool for creating and viewing 3D datasets in a variety of scenarios. As future work, we would like to further validate our system with a more formal user evaluation.

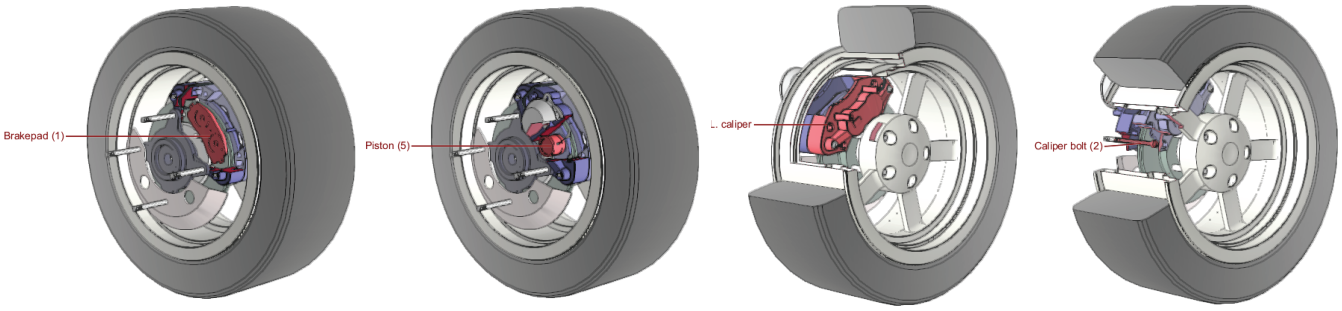


Figure 16: *Automatically generated cutaway views of disk brake model. We generated an illustration for each of the 27 parts in the model. Here, we show four representative images that use both saved viewpoints. The image on the right shows an artifact; the cuts that expose the target part also leave it disconnected from the surrounding parts. Out of the 27 illustrations, one other exhibited a similar artifact.*

9 Conclusions and future work

In this paper, we have introduced an approach for incorporating cutting conventions from traditional illustration into an interactive visualization system. The authoring and viewing techniques we have developed allow interactive cutaway illustrations of complex models to be authored with a small amount of user effort, and to be explored in a simple and natural way. We believe our interactive cutaway illustrations would be invaluable in educational material for medical students, repair documentation for airplanes and cars, technical manuals for complex machinery, and in fact, any scenario where static illustrations are currently used.

We see several opportunities for future work:

Experimenting with other types of 3D models. We are eager to extend our techniques to work for complex 3D models in more specialized domains, including mathematics and architecture. Although in some cases our cutting tools can already be used to create reasonable illustrations of such datasets (see Figure 17), we believe more domain-specific techniques could be developed to generate better visualizations. For instance, visualizations of mathematical surfaces exhibit a variety of interesting cutting and shading conventions for conveying surface orientation and self-intersections.

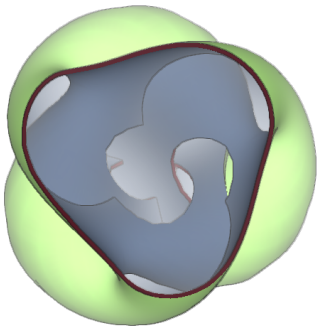


Figure 17: *Illustration of Boy's surface with window cuts.*

Extending authoring tools. Although our authoring interface makes it possible to create interactive cutaway illustrations with little user effort, we believe the rigging process could be further streamlined. We would like to explore more automatic techniques for determining the geometric type of parts. In addition, we believe we could leverage existing work on view selection algorithms to suggest good viewpoints to the author. We would also like to investigate geometric filtering techniques that would allow our system to handle a wider range of datasets, including those that contains cracks.

Exploring other illustration modalities. Finally, we would like to experiment with and integrate other illustration modalities, such as peeled-away structures, exploded views, and so on. In many illustrations, these other techniques are combined with cuts to produce effective visualizations.

Acknowledgments

The authors would like to thank Dr. Jim Brinkley for many useful discussions about potential applications of our technology in medicine and biology. We also want to thank Eric Haines at Autodesk, David Kasik at Boeing, and Bahman Dara and François Chrétien at Adobe Systems for their help in acquiring 3D models. This work was supported by Adobe, Microsoft, the University of Washington Animation Research Labs, and the Washington Research Foundation.

References

A.D.A.M. INC., 2005. A.D.A.M. Interactive Anatomy.

AGUR, A. M. R., AND LEE, M. J. 2003. *Grant's Atlas of Anatomy*. Lippincott Williams and Wilkins.

AKERS, D., LOSASSO, F., KLINGNER, J., AGRAWALA, M., RICK, J., AND HANRAHAN, P. 2003. Conveying shape and features with image-based relighting. In *Proceedings of IEEE Visualization 2003*, 349–354.

ALI, K., HARTMANN, K., AND STROTHOTTE, T. 2005. Label layout for interactive 3d illustrations. In *WSCG (Journal Papers)*, 1–8.

BIER, E. A., STONE, M. C., PIER, K., BUXTON, W., AND DEROSE, T. 1993. Toolglass and magic lenses: The see-through interface. In *Proceedings of ACM SIGGRAPH 93*, 73–80.

BIESTY, S., AND PLATT, R. 1992. *Stephen Biesty's Incredible Cross-sections*. Dorling Kindersley.

BIESTY, S., AND PLATT, R. 1993. *Man-of-war*. Dorling Kindersley.

BRUCKNER, S., AND GRÖLLER, M. E. 2005. Volumeshop: An interactive system for direct volume illustration. In *Proceedings of IEEE Visualization 2005*, 671–678.

BRUCKNER, S., AND GROLLER, M. 2006. Exploded views for volume data. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (Sept./Oct.), 1077–1084.

BRUYNS, C., SENGER, S., MENON, A., MONTGOMERY, K., WILDERMUTH, S., AND BOYLE, R. 2002. A survey of interactive mesh-cutting techniques and a new method for implementing generalized interactive mesh cutting using virtual tools. *Journal of Visualization and Computer Animation 13*, 1, 21–42.

BURNS, M., KLAWE, J., RUSINKIEWICZ, S., FINKELSTEIN, A., AND DECARLO, D. 2005. Line drawings from volume data. *ACM Transactions on Graphics 24*, 3 (Aug.), 512–518.

CIGNONI, P., SCOPIGNO, R., AND TARINI, M. 2005. A simple normal enhancement technique for interactive non-photorealistic renderings. *Computer & Graphics 29*, 1 (Feb), 125–133.

COLE, F., DECARLO, D., FINKELSTEIN, A., KIN, K., MORLEY, K., AND SANTELLA, A. 2006. Directing gaze in 3d models with stylized focus. In *Rendering Techniques 2006: 17th Eurographics Workshop on Rendering*, 377–388.

CORREA, C., SILVER, D., AND CHEN, M. 2006. Feature aligned volume manipulation for illustration and visualization. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (Sept./Oct.), 1069–1076.

DIEPSTRATEN, J., WEISKOPF, D., AND ERTL, T. 2003. Interactive cutaway illustrations. *Computer Graphics Forum 22*, 3 (Sept.), 523–532.

DOOLEY, D., AND COHEN, M. 1990. Automatic illustration of 3d geometric models: Lines. In *1990 Symposium on Interactive 3D Graphics*, 77–82.

EBERT, D., AND RHEINGANS, P. 2000. Volume illustration: Non-photorealistic rendering of volume models. In *Proceedings of IEEE Visualization 2000*, 195–202.

ELLIOTT, B. G., CONSOLIVER, E. L., AND HOBBS, G. W. 1924. *The Gasoline Automobile*. Mcgraw-Hill.

FEINER, S., AND SELIGMANN, D. D. 1992. Cutaways and ghosting: Satisfying visibility constraints in dynamic 3d illustrations. *The Visual Computer*, 292–302.

FRANCIS, G. 1987. *A Topological Picturebook*. Springer.

GOOCH, A., GOOCH, B., SHIRLEY, P. S., AND COHEN, E. 1998. A non-photorealistic lighting model for automatic technical illustration. In *Proceedings of ACM SIGGRAPH 98*, Computer Graphics Proceedings, Annual Conference Series, 447–452.

HODGES, E. R. S. 1989. *The Guild Handbook of Scientific Illustration*. Van Nostrand Reinhold.

HÖHNE, K. H., BOMANS, M., RIEMER, M., SCHUBERT, R., TIEDE, U., AND LIERSE, W. 1992. A 3d anatomical atlas based on a volume model. *IEEE Computer Graphics and Applications 12*, 4, 72–78.

HÖHNE, K. H., PFLESSER, B., POMMERT, A., PRIESMEYER, K., RIEMER, M., SCHIEMANN, T., SCHUBERT, R., TIEDE, U., FREDERKING, H., GEHRMANN, S., NOSTER, S., AND SCHUMACHER, U., 2003. VOXEL-MAN 3D Navigator: Inner Organs. Regional, Systemic and Radiological Anatomy.

HOYT, W. A. 1981. *Complete Car Care Manual*. Reader's Digest Association.

IGARASHI, T., MATSUOKA, S., AND TANAKA, H. 1999. Teddy: A sketching interface for 3d freeform design. In *Proceedings of ACM SIGGRAPH 99*, 409–416.

KIRSCH, F., AND DÖLLNER, J. 2005. OpenCSG: A library for image-based CSG rendering. In *Proceedings of USENIX 05*, 129–140.

LAMAR, E., HAMANN, B., AND JOY, K. I. 2001. A magnification lens for interactive volume visualization. In *9th Pacific Conference on Computer Graphics and Applications*, 223–232.

LEE, C. H., HAO, X., AND VARSHNEY, A. 2004. Light collages: Lighting design for effective visualization. In *Proceedings of IEEE Visualization 2004*, 281–288.

LOECHEL, W. E. 1964. *Medical Illustration: A Guide for the Doctor-Author and Exhibitor*. C. C. Thomas.

LUFT, T., COLDITZ, C., AND DEUSSEN, O. 2006. Image enhancement by unsharp masking the depth buffer. *ACM Transactions on Graphics 25*, 3 (July), 1206–1213.

LUM, E. B., AND MA, K.-L. 2002. Hardware-accelerated parallel non-photorealistic volume rendering. In *Proceedings of NPAR 02*, 67–74.

MCGUFFIN, M. J., TANCAU, L., AND BALAKRISHNAN, R. 2003. Using deformations for browsing volumetric data. In *Proceedings of IEEE Visualization 2003*, 401–408.

NETTER, F. H. 1989. *Atlas of Human Anatomy*, 3rd ed. Icon Learning Systems.

OWADA, S., NIELSEN, F., NAKAZAWA, K., AND IGARASHI, T. 2003. A sketching interface for modeling the internal structures of 3d shapes. In *Smart Graphics 2003, Lecture Notes in Computer Science (LNCS)*, vol. 2733, 49–57.

OWADA, S., NIELSEN, F., OKABE, M., AND IGARASHI, T. 2004. Volumetric illustration: designing 3d models with internal textures. *ACM Transactions on Graphics 23*, 3 (Aug.), 322–328.

RUSINKIEWICZ, S., BURNS, M., AND DECARLO, D. 2006. Exaggerated shading for depicting shape and detail. *ACM Transactions on Graphics 25*, 3 (July), 1199–1205.

TIETJEN, C., ISENBERG, T., AND PREIM, B. 2005. Combining silhouettes, shading, and volume rendering for surgery education and planning. In *Proceedings of IEEE/Eurgraphics Symposium on Visualization 2005*, 303–310.

VERROUST, A., AND LAZARUS, F. 2000. Extracting skeletal curves from 3d scattered data. *The Visual Computer 16*, 1, 15–25.

VIEGA, J., CONWAY, M. J., WILLIAMS, G., AND PAUSCH, R. 1996. 3d magic lenses. In *Proceedings of UIST 96*, 51–58.

VIOLA, I., GRÖLLER, E., BÜHLER, K., HADWIGER, M., PREIM, B., AND EBERT, D., 2005. Eurographics tutorial on illustrative visualization.

VIOLA, I., KANITSAR, A., AND GRÖLLER, E. 2005. Importance-driven feature enhancement in volume visualization. *IEEE Transactions on Visualization and Computer Graphics 11*, 4, 408–418.

WANG, L., ZHAO, Y., MUELLER, K., AND KAUFMAN, A. E. 2005. The magic volume lens: An interactive focus+context technique for volume rendering. In *Proceedings of IEEE Visualization 2005*, 367–374.

WOOD, P. 1979. *Scientific Illustration: A Guide to Biological, Zoological, and Medical Rendering Techniques, Design, Printing, and Display*. Van Nostrand Reinhold.

ZWEIFEL, F. W. 1961. *A Handbook of Biological Illustration*. University of Chicago Press.