



**HAL**  
open science

# Interactive Generation and Modification of Cutaway Illustrations for Polygonal Models

Sebastian Knödel, Martin Hachet, Pascal Guitton

► **To cite this version:**

Sebastian Knödel, Martin Hachet, Pascal Guitton. Interactive Generation and Modification of Cutaway Illustrations for Polygonal Models. 2009. inria-00392710

**HAL Id: inria-00392710**

**<https://inria.hal.science/inria-00392710>**

Preprint submitted on 8 Jun 2009

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

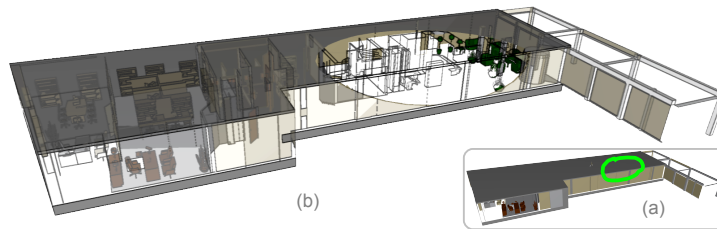
# Interactive Generation and Modification of Cutaway Illustrations for Polygonal Models

Sebastian Knödel\*, Martin Hachet\*\*, and Pascal Guitton\*\*\*

INRIA - Université de Bordeaux - France

**Abstract.** We present a system for creating appealing illustrative cutaway renderings. This system bases on simple sketch-based interfaces and stylized rendering techniques for the study of elaborate 3D models. Since interactive visualization technology found its way to the general public, there is a demand for novel interaction techniques that allow easy exploration of the displayed illustrations. Hence, our system lets users create individual cutaway views to focus on hidden objects. At the same time, important contextual information is emphasized by illustrative rendering techniques.

**Key words:** Cutaway, Sketch-based Interaction, Illustrative Visualization



**Fig. 1.** Exploration of an office building (a) sketching a circular cutaway (b) final composite illustrative rendering including cutaways, transparency and ghostviews.

## 1 Introduction

The advances of 3D interactive technology as well as novel rendering techniques have enhanced the real time creation of detailed and appealing imagery of elaborate 3D models for a large audience. At the same time, the emergence of direct touch-based interface technology has led to the creation of new interaction metaphors. Consequently, new perspectives appear that involve users in the process of content creation and modification. However, there is still a lack of suitable user interfaces that make these technological achievements accessible to a general public with little experience in 3D interaction.

\* e-mail: knoedel@labri.fr

\*\* e-mail: hachet@labri.fr

\*\*\* e-mail: guitton@labri.fr

In recent years a wide variety of complex 3D models have become available. At the same time grew a demand for interactive techniques that let users inspect these 3D models in an interactive manner, in order to understand their structure. In this case, exploration represents the opportunity to have a closer look into the interior of the model. From our point of view, one major key element to accomplish such an exploration is the possibility for users to cut away parts of the 3D model. Secondly, the system must also be able to present and preserve contextual information that underlines special characteristics and structures.

Such 3D models often consist of so many parts that the exploration can become cumbersome. In order to examine the interior of a 3D model, users need be able to expose hidden parts. One example is the study of a 3D model of an office building that contains extensive details, like detailed pieces of furniture, as illustrated in Figure 1(a). Without having further information about the inner structure, users need intuitive and simple interfaces to be able to "open" the 3D model and to modify the resulting cutaway.

The purpose of our approach is to develop a system that allows regular users to create and interact with appealing illustrations of diverse virtual 3D models. Our focus lies on the development of cutaway techniques that allow users to explore the interior of a 3D model in a simple and efficient way. To achieve this goal, we introduce novel sketch-based interfaces and smart widgets. In particular, users generate such cutaways using simple gestures and refine the shape of the cut afterward by means of repositioning and resizing operations.

To better convey, the resulting imagery is displayed using stylized rendering techniques to permit users to focus on important properties, while preserving the surrounding context information, as shown in the final rendering in Figure 1(b). To create such a vivid rendering we use transparent representations and highlight important contours of the modified 3D model.

In the following, we present in section 2 related work. In section 3 we explain our general approach, whereas in section 4 we show in detail the generation of the cutaway and we present the widget that is used to modify them. After that, in section 5 we describe visualization techniques and we conclude in section 6.

## 2 Related Work

### 2.1 Illustrative Visualization and Sketch-based Interaction

Various research has been conducted on the creation of interactive technical illustrations of complex 3D models. Gooch et al. [1] created appealing technical illustrations by enhancing functional characteristics of shape and structure through drawing of silhouette curves. Silhouette curves and contours play an important role in shape recognition as Isenberg et al. [2] pointed out. We share this argument and perform a contour extraction as well. Another important role

in technical illustration plays transparency. Diepstraten et al. [3] created technical illustrations containing opaque and non-opaque objects using depth ordered transparency. Nienhaus et al. [4] went further and created non-photorealistic blueprint renderings using the so called depth peeling technique. Our depth-ordered transparency rendering approach has similarities with the depth peeling technique. In the contrary, we do not slice the object but rather reorder the structure of the 3D model in a view dependent visibility graph, which we present in section 5.1.

Sketch-based interfaces became popular in the area of freeform modeling. Zeleznik et al. [5] and Igarashi et al. [6] showed that simple gestures and curves are useful tools for creating and manipulating 3D geometry. We use a similar approach to let the user specify regions on the surface that are meant to be cut away. Apart from freeform modeling research, Schmidt et al. [7] proposed a sketch-based interface that allows the user to compose widgets for 3D object manipulation. We propose a widget with essential manipulation functionalities as a graphical user interface for users during the modification of the cutaway. This widget is located in proximity to the mouse cursor adopting the Marking Menus approach by Kurtenbach [8].

## 2.2 Cutaways in 3D Data

A cutaway illustration is a popular and powerful method that allows users to expose hidden objects of a complex geometry. There exist several different approaches that allow the generation of cutaways with polygonal and volumetric models. However, a lot of existing research proposes interface solutions for interactive cutaways in volumetric data sets. In the contrary, there exist only a few works that propose user interfaces for cutaways in polygonal 3D models

**Cutaways in Volumetric Data** McGuffin et al. [9] presented an interactive system for volumetric objects, where users can create and manipulate cuts inside a volumetric object using additional interfaces and widgets. We use in our approach similar cut shapes as proposed in their system, but unlike to our approach, they positioned the interface widget inside the 3D model. In another approach McInerney et al. [10] used 3D slice plane widgets as a 3D interaction model for exploring volume image data. Chen et al. [11] proposes a variety of complex volume manipulation tools that allow drilling, lasering, peeling, as well as cut and past operations. However, we want to provide simple interfaces, whose functionalities are easy to understand.

Brückner et al.[12][13], Viola et al. [14] and Wang et al. [15] discuss related approaches to enhance the focus and guard context of an illustration. They combined cutaways and non-photorealistic visualization techniques to smartly uncover the most important information and to let the user interactively inspect the interior of a volumetric data set. In our system, the aspect of context preservation plays an important role as well as the use of transparency and contour rendering techniques.

**Cutaways in Polygonal Data** Basically, the creation of cutaways in volumetric data sets is different from polygonal ones, since voxels have to be removed. There exists several approaches that allow cutaway creation in polygonal data sets. Often they use Constructive Solid Geometry (CSG) algorithms to create such cutaways. One early approach of cutaway creation by Viega et al. [16], the *3D Magic Lenses*, proposed a multi pass algorithm using clipping planes with *flat* lenses embedded in a 3D environment. Another work by Coffin et al. [17] proposes so called perspective *cut-away views* that define a cutout shape or choose a standard shape and sweep it over the occluding wall segments to reveal what lies behind the surface. Diepstraten et al. [18] presented different methods to create cutaways in polygonal data, including the *Cutout* and the *Breakout* techniques. However, we use image-based CSG rendering techniques to create cutaways in polygonal data, as Li et al. [19] and Burns et al. [20] presented in their approaches. We present our implementation in more detail in section 4.3. In their work, Li et al. [19] presented a system for authoring and viewing interactive cutaway illustrations of complex 3D models that uses sophisticated data structures and animations. They propose to create the cutaways in a automatic or semi-automatic interactive process and provide very convincing results that limits users to predefined viewpoints, which we want to avoid. Another method for automatically generating cutaway renderings of polygonal scenes at interactive frame rates was presented by Burns et al. [20]. They use illustrative and non-photorealistic rendering cues as well, but do not propose direct interfaces to users.

### 3 General Approach

The level of interactivity in traditional cutaway applications is usually reduced. The possibilities to modify the resulting cutaway are often restricted. Most of the time, users choose the part to be revealed from a list. Then, the system creates automatically the cutaway from a meaningful point of view. In contrast our approach enables users to proceed and modify cutaways by themselves. We decided to use a sketch-based approach as an interface for the creation of a cut.

Performing a sketch allows users to cut the surface of an object. We think that there is a strong semantic link between the drawing of a sketch and the cutting of an object. We propose four different essential gestures, which we present in detail in section 4.1.

After the initial cutting phase, the modification of a cutaway can be performed via a smart widget that provides an essential variety of functionalities. The tools, provided by the widget, follow the mouse cursor and are consequently easily accessible. Users choose whether such a modification may influence only a local cutaway or all the cutaways in the 3D model. If necessary, size and structure of the global cutaway is adapted to avoid obstructions during editing.

In our approach, we focus on the creation of cutaway illustrations for polygonal data, since it is the most common data representation of 3D information. A lot of existing research about cutaways tend to work with volumetric data. Nowadays powerful tools that allow the creation of highly complex 3D models are available to the general public. Consequently, we need a sophisticated data management to be able to create and modify cutaways in such 3D models, while maintaining interactive frame rates. The reader will find more details about our implementation in section 4.3.

To explore, to analyze, and to understand the functionality of a complex 3D model, it is not sufficient to only allow users to create a cutaway. At the same time, illustrative and instructive visualization techniques are necessary to preserve the general context, especially if whole parts are cut away or other parts are still occluded. Therefore, we additionally use visualization techniques, like transparency, contour shading, and ghostlines to emphasize invisible structures.

## 4 Interactive cutaways

### 4.1 Sketch Gestures

Users create a cutaway using a simple sketch gesture. The system evaluates the gesture and generates the corresponding cut. To conduct the cutting interaction, users only need to know four basic gestures, which are the *line*, *circle*, *corner* and the *ribbon* gesture. In the following, we present the different gestures and the corresponding cutaways.

***Line Gesture*** The most simple gesture to cut through the surface of a 3D model is a stroke of a straight line. It is similar to cutting a surface with a knife, which is a common everyday life task. In that case the system creates a narrow window cut aligned to the length and direction of the stroke, as shown in Figure 2(a). Since the resulting cutaway might not reveal enough underlying parts, users may edit it afterward using the tools presented in section 4.4.

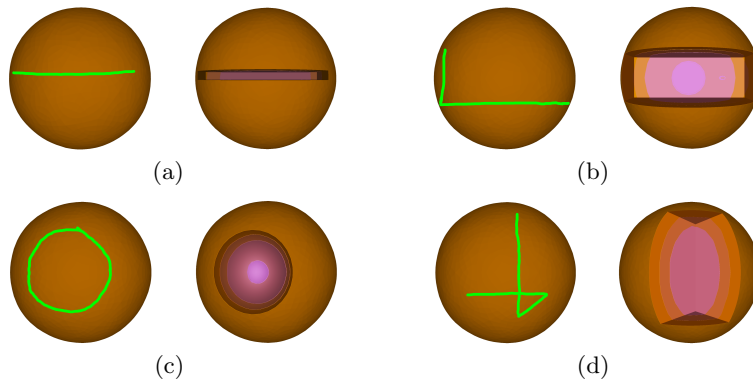
***Corner Gesture*** Certainly users might want to create bigger cutaways right away. To do so, they draw a vertical stroke followed by a horizontal stroke. As a result the system creates a window cut, whose dimensions are relative to the length of the vertical and horizontal stroke. The cutaway is positioned at the center of the approximated rectangle, which is defined by the vertical and horizontal stroke length, as we show in Figure 2(b).

***Circular Gesture*** Another simple gesture consists of spherical-shaped sketch line. The sketch can also be elliptic- or egg-shaped. The position of the spherical cutaway is derived from the center of the circle sketch. The size of the circular cut is determined by the vertical and horizontal dimensions of the spherical-sketch, which we illustrate in Figure 2(c).

***Ribbon Gesture*** We want to let users also create wedge type cutaways. The main characteristic of the wedge cut is its sharp edge that breaks into the object

and creates an opening with a certain angle. Therefore, we decided to represent this cutaway as a ribbon shaped sketch. Furthermore the ribbon sketch unifies the two parameters in one coherent movement, as one can observe in Figure 2(d). The vertical sketch line determines position and size of the edge, whereas the horizontal one is used to calculate the opening angle of the wedge.

Additional cutaways could be imagined. However, the multiplication of possible gestures would make the interface more complex and would require a longer learning phase for users.



**Fig. 2.** (a) Line sketch with resulting narrow window cut, (b) corner sketch with resulting window cut, (c) spherical sketch with resulting spherical cut, (d) ribbon sketch with resulting wedge cut.

**Automatic Cutaways** We offer also the possibility to create automatic cutaways, as presented by Li et al. [19]. Of course, users may manipulate these cutaways afterward. The system provides a preview list of all parts of the 3D model. Users select the part they are interested in, thus the system creates automatically all necessary cuts to expose the object.

## 4.2 Cut Shapes

Depending on the detected gesture and their position as well as their dimensions, our system determines the resulting shape of the cutaway. Due to traditional geometric illustration conventions, as denoted by Li et al. [19] and McGuffin et al. [9], we focus on four basic shapes to create a cut; the spherical shaped cut, the cubic shaped cut, the wedge shaped cut, and the tube shaped cut. Figure 3 presents the different cut shapes with the resulting cutaway. All four cut shapes can be effected using simple sketch-based gestures that are connatural to the resulting cutaway. The tube shaped cut, presented in Figure 3(d), is used for automatic cutaways.

The previously described cuts for polygonal geometry, are generated using image-based *Constructive Solid Geometry* algorithms, provided by the OpenCSG li-

brary [21]. These algorithms basically combine two shapes by taking the union of them, by intersecting them, or by subtracting one shape of the other.

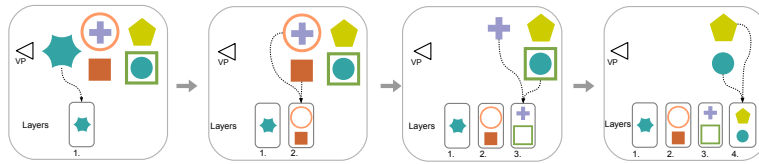
Initially we have to adjust the depth of the cutaway to a default value that corresponds to the depth of the bounding box. This might be considered as an arbitrary choice, but it turned out that the resulting size represents an good compromise. Users may change the depth of the cut at anytime using the widget, which is presented in section 4.4. Furthermore, we provide continuous and smooth animations to visualize the opening of a cutaway, which makes it easier for users to follow the cutting procedure and to understand the spatial relations.



**Fig. 3.** Cut shapes and their resulting cutaway, (a) cubic cut shape, (b) spherical cut shape, (c) wedge cut shape, (d) tube cut shape.

### 4.3 Cutaway Generation

To be able to create correct cutaways including geometric constraints that allow a modification, the data of the 3D model has to be reorganized in a so called visibility graph. The view dependent visibility graph, introduced by Li et al. [19], organizes the parts of the 3D model in structured layers. This is realized by determining their visibility from the current viewpoint using multiple occlusion tests.



**Fig. 4.** Example of the creation of a Visibility Graph.

To create the graph, the relative and absolute visibility of an object is compared in an iterative process. If one object is completely visible, it is added to the current layer. If parts are occluded by another object and the relative visibility is consequently lower than the absolute one, the object is not added to the current layer. The complete procedure is presented in scheme 4.

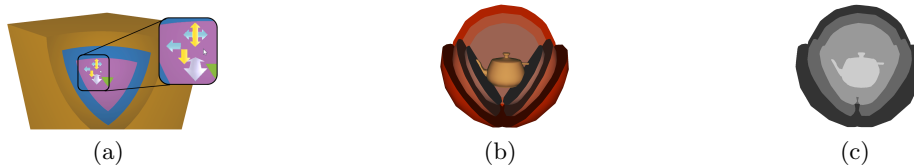
### 4.4 Modifying the Cutaway

The possibility to manipulate the cutaway is important since solely automatic creation leads often to unsatisfying results. Indeed, the system can hardly esti-



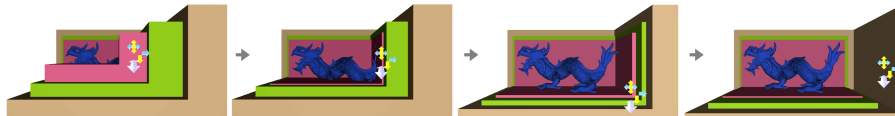
mate what users exactly want to see, since various kinds of cutaways are imaginable. However, we think that giving more control to the user may promote a better understanding of the structure of the 3D model. Thus, good interaction metaphors and interfaces are very important, such that users are able to easily customize the cutaway to meet their needs. In our approach, we want to let users directly modify the cutaways. More precisely, users should be able to reposition and to resize the cutaways freely using simple interfaces.

We propose a smart widget as the graphical user interface (GUI). It appears next to the cursor like the *Marking Menus*, which were introduced by Kurtenbach et al.[8]. This way it is always easily accessible. This widget provides three basic selectable functionalities, which are illustrated in Figure 5(a). The first function allows translation of the cutaway in the x-y plane. The second functionality resizes the cut in the x-y plane and the third option increases the size in the z axis. To guaranty that the cut remains always visible to users, we do not provide translation in the z axis. Instead users can increase the scale of the cutaway in direction of the z axis.



**Fig. 5.** (a) Widget representation in cutaway, (b) Multiple cutaways of a complex 3D model containing various parts, (c) Corresponding ID buffer.

We allow the global modification of multiple cutaways as well as the local modification of a selected one. Since for every part of the 3D model one particular cutaway object is created, every cut for every part can be modified independently. Users select the part they want to modify with a simple mouse click. Subsequently, they can modify the cutaway of this part using the widget as previously explained.



**Fig. 6.** From left to right, resizing the cutaway of the red box consequently modifies the size of the cutaway of the green and brown box.

We want to provide simple selection mechanisms, so that every visible part of the 3D model is selectable. Hence users modify the cutaway of one part independently from other cutaways. To allow the cutaway selection we assign to every part of the 3D model one unique ID. After that, the 3D model is rendered using its assigned ID as a color, as shown in Figure 5(b) and (c). After picking on one part with the mouse, the system can retrieve the color at the pixel and conse-

quently the ID of the selected part. With that ID the system can immediately refer to the proper data in the visibility graph and adjust position and size of the selected cutaway.

When users modify one local cutaway the position and size of surrounding cutaways are adapted due to constraints given by the visibility graph, which is illustrated in Figure 6. For example, if one local cutaway is resized by users other cutaways will also be resized, but only in the case where the boundaries of the local cutaway intersect boundaries of another cutaway. As already stated, the necessary information about surrounding parts, which have to be affected by the modification, is provided by the layered visibility graph. The resulting transformations for surrounding cuts can be calculated easily, since our cut shapes are represented through basic geometrical objects, as spheres, cubes and cylinders.

## 5 Illustrative Context Visualization

In the previous section we have shown that users can create and modify cutaways of elaborate 3D models. However, they often need more information about invisible parts, to better understand the relationships between them. We divide such occluded parts into two groups. The first group contains parts that are not visible since they were removed by the cutaway. Parts of the second group are still hidden by other geometry. Since these parts represent important contextual information we want to visualize them in a way that the objects of interest in the cutaway remain in focus. We use two basic rendering techniques. First of all, the ghostview visualization displays parts that were cut away. Secondly, a transparent rendering with contour enhancement depicts parts that are still covered. Both techniques rely on the visibility graph structure to perform layer dependent blending of all scene parts.

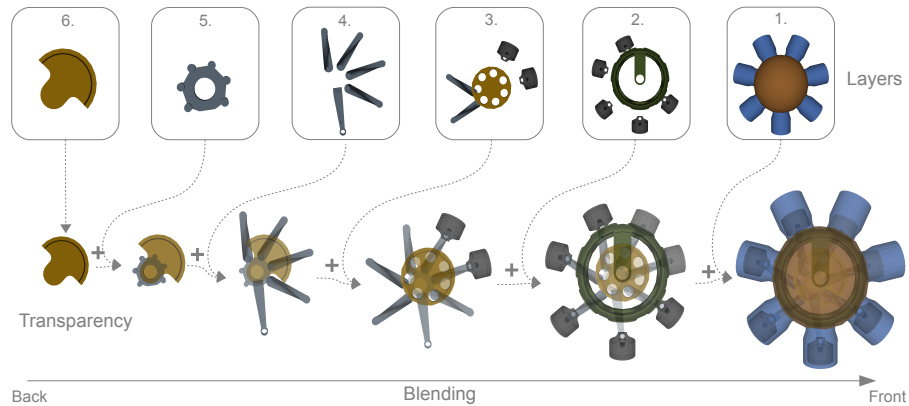
### 5.1 Layered Transparency and Contours

To create a transparent visualization of the 3D model we make use of the view dependent visibility graph. Since all parts are organized in layers, depending on their distance to the viewpoint, we consequently have information about the visibility and the depth. We adapt the blending procedure to create a transparent visualization. This is done by rendering each layer of the graph from back to front and blending it with the previously accumulated layers. As a result, farther parts appear more transparent than parts in layers closer to the viewpoint. This procedure is illustrated in Figure 7. This process also increases rendering times depending on the number of layers. However, the advantage is that all parts in one layer appear similar, since they obtain the same level of transparency. We illustrate this effect in Figure 8(a).

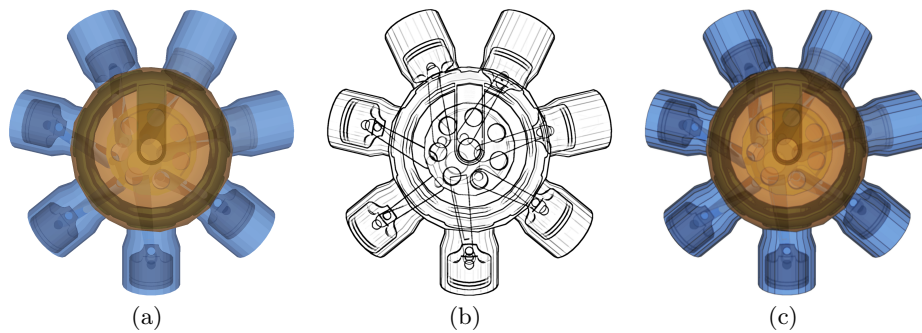
One purpose of our approach is to create blueprint-like renderings that illustrate parts of a 3D model in such a way that users can infer their function and relation

to other parts. Therefore, we benefit again from the visibility graph structure and apply edge detection on each layer separately. To detect the contours we use a standard Sobel filter on the diffuse Lambertian visualization. After that, the system blends the resulting contours from back to front into the final image. This blending approach is similar to the procedure we use to blend the transparent layers (see Figure 7). Since we use the visibility graph, we are able to obtain correct contours for all parts in each layer. Furthermore, the parts in one layer do not occlude themselves, hence they do not hide important information. As a result, the contours of closer parts appear stronger in the final image than the contours of parts that are located more distant to the viewpoint. Parts from the same layer appear with equal contours.

To enhance rendering quality, we add contour information to the final blended transparent image, which makes it easier for the user to differentiate between different parts, as shown in Figure 8(c).



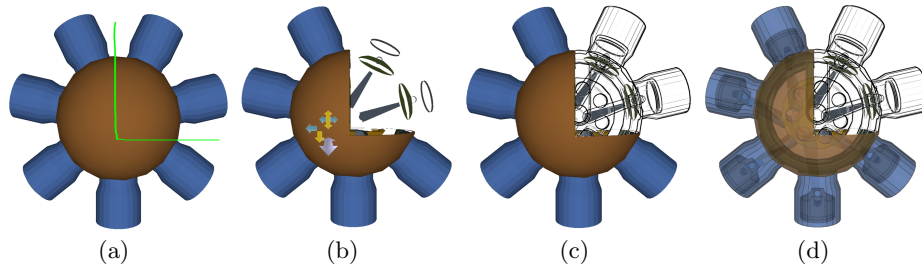
**Fig. 7.** Rendering pipeline to create transparent illustrations using the layered visibility graph.



**Fig. 8.** Final renderings with (a) transparency, (b) illustration only with contours (blueprint) and (c) adding contours information to the transparent rendering.

## 5.2 Composite Visualization

After users created a cutaway to reveal hidden parts, as shown in the Figures 9(a) and (b), our system provides additional information about the context inside and around the cutaway. Our system combines the result with a ghostview rendering that reveals the structures inside the cut, as illustrated in Figure 9(c). Then, we use transparent rendering to display parts of the 3D model that are still covered. The final technical illustration presents multiple information about different parts of the 3D model. It also provides at the same time contextual information about the overall structure, as illustrated in in Figure 9(d).



**Fig. 9.** (a) Sketching the rectangular cutaway, (b) modifying the resulting cutaway, (c) visualizing cut parts with ghostview, (d) final composite illustrative rendering of an engine including cutaways, with transparent and ghostview rendering.

## 6 Conclusion and Future Work

We presented new interaction and rendering techniques to create appealing illustrative renderings for elaborate polygonal 3D models. First, users create cutaways in a novel and simple way using our sketch-based interaction technique. Then, they modify the resulting cutaway using the smart widget. Finally, the system enhances rendering to show more contextual information by representing invisible parts with transparency effects and ghostviews. With our approach users easily explore a complex 3D model with simple interaction techniques and obtain rich illustrative visualizations at the same time.

In the future we want to identify other simple gestures to provide more distinct manipulation of a cutaway. Furthermore we have to evaluate the usability of our approach in a comprehensive user-study. We consider also to examine different visualization techniques that enhance the visibility of a bigger variety of characteristics of the 3D model, as for example the depth cue.

Such a system provides rich visualizations via simple interaction and facilitates the access to elaborate 3D content especially for non experienced users. Regarding the long-term goal, our system opens new perspectives for 3D interaction by providing simple interface solutions for the general public.

## References

1. Gooch, B., Peter, Gooch, A., Shirley, P., Riesenfeld, R.: Interactive technical illustration. In: Proceedings of symposium on Interactive 3D graphics, ACM (1999)
2. Isenberg, T., Freudenberg, B., Halper, N., Schlechtweg, S., Strothotte, T.: A developer's guide to silhouette algorithms for polygonal models. *IEEE Computer Graphics and Applications* **23**(4) (2003) 28–37
3. Diepstraten, J., Weiskopf, D., Ertl, T.: Transparency in interactive technical illustrations. *Computer Graphics Forum* **21** (2002)
4. Nienhaus, M., Döllner, J.: Blueprints: illustrating architecture and technical parts using hardware-accelerated non-photorealistic rendering. In: Proc. GI. (2004)
5. Zeleznik, R.C., Herndon, K.P., Hughes, J.F.: Sketch: an interface for sketching 3d scenes. In: SIGGRAPH '96, ACM (1996) 163–170
6. Igarashi, T., Matsuoka, S., Tanaka, H.: Teddy: a sketching interface for 3d freeform design. In: SIGGRAPH '99. (1999)
7. Schmidt, Ryan, Singh, Karan, Balakrishnan, Ravin: Sketching and composing widgets for 3d manipulation. *Computer Graphics Forum* **27**(2) (April 2008)
8. Kurtenbach, G.P.: The design and evaluation of marking menus. PhD thesis, Toronto, Ont., Canada, Canada (1993)
9. McGuffin, M.J., Tancou, L., Balakrishnan, R.: Using deformations for browsing volumetric data. In: VIS '03: IEEE Visualization, IEEE Computer Society (2003)
10. McInerney, T., Broughton, S.: Hingeslicer: interactive exploration of volume images using extended 3d slice plane widgets. In: GI '06: Graphics Interface 2006. (2006)
11. Chen, H.L., Samavati, F., Sousa, M.: Gpu-based point radiation for interactive volume sculpting and segmentation. *The Visual Computer* **24** (July 2008)
12. Brückner, S., Gröller, E.M.: Volumeshop: An interactive system for direct volume illustration. (2005)
13. Brückner, S., Grimm, S., Kanitsar, A., Gröller, E.M.: Illustrative context-preserving volume rendering. In: Proceedings of EuroVis 2005. (2005) 69–76
14. Viola, I., Sousa, M.C.: Focus of attention+context and smart visibility in visualization. In: SIGGRAPH '06: ACM SIGGRAPH 2006 Courses, ACM (2006)
15. Wang, L., Zhao, Y., Mueller, K., Kaufman, A.: The magic volume lens: an interactive focus+context technique for volume rendering. In: IEEE Visualization. (2005)
16. Viega, J., Conway, M.J., Williams, G., Pausch, R.: 3d magic lenses. In: UIST '96: Proceedings of the 9th annual ACM symposium on User interface software and technology, New York, NY, USA, ACM (1996) 51–58
17. Coffin, C., Hollerer, T.: Interactive perspective cut-away views for general 3d scenes. In: IEEE Symposium on 3D User Interfaces. (2006) 25–28
18. Diepstraten, J., Weiskopf, D., Ertl, T.: Interactive cutaway illustrations. In: *Computer Graphics Forum*. (2003) 523–532
19. Li, W., Ritter, L., Agrawala, M., Curless, B., Salesin, D.: Interactive cutaway illustrations of complex 3d models. *ACM Trans. Graph.* **26**(3) (July 2007)
20. Burns, M., Finkelstein, A.: Adaptive cutaways for comprehensible rendering of polygonal scenes. In: SIGGRAPH Asia '08, ACM (2008)
21. Kirsch, F., Döllner, J.: Opencsg: a library for image-based csg rendering. In: ATEC '05: Proceedings of the annual conference on USENIX Annual Technical Conference, Berkeley, CA, USA, USENIX Association (2005) 49