Drilling into Complex 3D Models with Gimlenses

Cyprien Pindat, Emmanuel Pietriga, Olivier Chapuis, Claude Puech

Drilling into Complex 3D Models with Gimlenses

Cyprien Pindat*
Univ Paris-Sud, CNRS & INRIA

Emmanuel Pietriga†
INRIA & INRIA Chile

Olivier Chapuis‡
CNRS, Univ Paris-Sud & INRIA

Claude Puech§
INRIA Chile

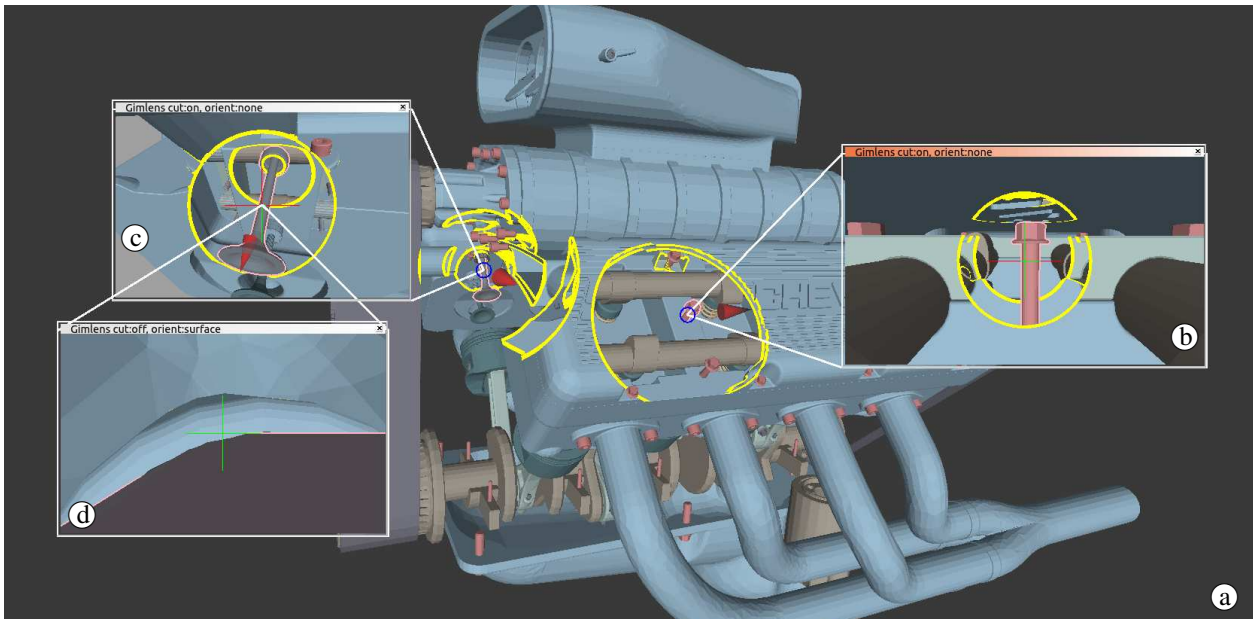

Figure 1: *Exploring the CAD drawing of a car engine. The three Gimlenses provide detailed views of different constituent parts of the engine, at different magnification levels and with varying orientation, while revealing their location inside the global 3D model. a) Context view. b) Magnified side view of a knot behind, and thus originally hidden by, the cylinder head cover. c) View fully revealing a poppet valve in-context from a different angle than the main view, with d) another Gimlens configured so as to provide a low-angled point of view on the valve.*

Abstract

Complex 3D virtual scenes such as CAD models of airplanes and representations of the human body are notoriously hard to visualize. Those models are made of many parts, pieces and layers of varying size, that partially occlude or even fully surround one another. We introduce Gimlenses, a multi-view, detail-in-context visualization technique that enables users to navigate complex 3D models by interactively drilling holes into their outer layers to reveal objects that are buried, possibly deep, into the scene. Those holes get constantly adjusted so as to guarantee the visibility of objects of interest from the parent view. Gimlenses can be cascaded and constrained with respect to one another, providing synchronized, complementary viewpoints on the scene. Gimlenses enable users to quickly identify elements of interest, get detailed views of those elements, relate them, and put them in a broader spatial context.

CR Categories: H.5.2 [Information Interfaces and Presentation]: User Interfaces - Graphical user interfaces— [I.3.5]: Computer Graphics—Computational Geometry and Object Modeling.

Keywords: Detail-in-Context, Lenses, Multi-scale Visualization

*e-mail:pindat@lri.fr
†e-mail:emmanuel.pietriga@inria.fr
‡e-mail:chapuis@lri.fr
§e-mail:claude.puech@inria.cl

1 Introduction

Three-dimensional computer modeling has become an essential activity in many domains: the movie industry, medicine, scientific visualization at large, and the automotive, aerospace and electronics industries, which sometimes rely on 3D modeling throughout the design and production chain (Computer-Aided Design and Manufacturing). Processing and graphics rendering capabilities now make it possible to model and visualize extremely complex datasets on widely varying types of displays: desktop workstations featuring one or multiple screens, ultra-high-resolution wall-sized displays [Nancel et al. 2011], and virtual reality environments such as CAVEs [Cruz-Neira et al. 1992]. For instance, Boeing's 777 airliner was entirely modeled using CAD software, resulting in a very complex and dense 3D scene featuring more than 350 million polygons [Dietrich et al. 2007]. Highly-detailed models of the entire human anatomy, of complex mechanical components and even entire cities are now available, usually for a price given the difficulty of creating such datasets.

As their real-world counterparts, complex 3D models are difficult to inspect, because they are made of numerous distinct parts assem-

Cyprien Pindat, Emmanuel Pietriga, Oliver Chapuis, and Claude Puech. 2013. Drilling into complex 3D models with gimlenses. In Proceedings of the 19th ACM Symposium on Virtual Reality Software and Technology (VRST '13). ACM. 223-230.

bled together into dense scenes that generate a lot of visual occlusion. Constituent elements will often partially occlude or even fully surround one another, and can also have widely varying sizes (e.g., an airplane's wings compared to a bolt). Quickly identifying elements of interest, getting detailed views of those elements from different perspectives, putting them in a broader spatial context, and relating them to other elements are all challenging tasks. Often, several designers will be working collaboratively on these datasets, requiring support for merging their work into a single assembly: detecting and understanding conflicts such as, e.g., intersecting parts due to problems of scaling and positioning.

These different tasks involve setting up multiple cameras and manipulating them to inspect elements from different angles and at different scales, as well as changing the visibility settings of elements that lie in the line of sight between the observer and the objects of interest. Those interactions are tedious to perform and cause significant distraction.

We introduce Gimlenses (Figure 1), a multi-view, detail-in-context visualization technique that enables users to navigate complex 3D models by drilling holes into the outer layers of a model to reveal objects that are buried, possibly deep, into the scene. The geometry of those holes gets constantly adjusted so as to guarantee the visibility of objects of interest from the parent view. Gimlenses can be cascaded and constrained with respect to one another. They provide synchronized, complementary viewpoints on the scene at different scales and from different angles, enabling users to inspect distinct parts simultaneously, facilitating their understanding, inspection and comparison [Plumlee and Ware 2006].

2 Related Work

Gimlenses build upon work in multi-scale visualization, 3D navigation and smart visibility techniques for CAD.

2.1 Multi-scale Visualization

Gimlenses are essentially detail-in-context representations, providing multiple simultaneous points of view on the 3D model at different scales. Gimlenses are categorized as an overview+detail technique [Cockburn et al. 2008]: they relate one or more detailed views, called focus regions, into a larger, less detailed contextual view of the dataset. Multi-scale techniques apply either to 2D visualization, e.g., [Carpendale and Montagnese 2001], 3D visualization, e.g., [Carpendale et al. 1997], or both [Pietriga et al. 2010]. Some techniques use spatial distortion to achieve a smooth integration of the focus in the surrounding context [Carpendale and Montagnese 2001], possibly adapting the focus' shape to the underlying geometry [Pindat et al. 2012], while other techniques achieve this integration through a combination of spatial distortion, alpha blending and dynamic parameter adjustments according to user input [Pietriga et al. 2010]. Gimlenses are more specifically drawing upon the DragMag [Ware et al. 1995], a technique that offsets the magnified view and visually relates it to the corresponding region in the context view using simple line segments.

In the realm of 3D multi-scale visualization, Carpendale *et al.* [Carpendale et al. 1997] transposed the general focus+context approach [Cockburn et al. 2008] based on spatial distortion to 3D data cubes. The technique guarantees the visibility of particular cells in a cube by moving and resizing the surrounding ones. The concept was generalized to arbitrary meshes using an energy grid optimization model [Wang et al. 2011]. The technique only magnifies objects in-place, though, and does not address the problem of visibility in dense scenes. Magic Volume Lenses [Wang et al. 2005] provide in-place magnification and can reveal some details hidden behind outer layers, but the detailed view is necessarily oriented according to the context view, and magnification factors are severely limited by both the spatially-distorted transition and problems of quantization [Appert et al. 2010] due to the single focus region. The DragMag technique mentioned earlier was also transposed to 3D in [Plumlee and Ware 2003], providing users with a multi-window detail-in-context visualization technique that somewhat resembles ours. However, the technique was designed for navigating in oceanic data, i.e., relatively flat 3D scenes compared to car engines or layered models of the human anatomy. Thus the technique does not deal with problems of visual occlusion typically encountered in complex, dense 3D scenes.

Magic Lenses [Bier et al. 1993] are lenses that are used to modify the rendering of objects seen through them. The concept is very generic and powerful. Examples more closely related to our work are lenses that render, e.g., a wireframe version of the original object. Originally designed for 2D graphics, Magic Lenses were later extended to work with 3D graphics [Viega et al. 1996; Ropinski and Hinrichs 2004].

2.2 3D Navigation

Numerous techniques have been designed to make navigation into 3D environments more efficient than when using the usual pan, zoom and rotate functions found in most 3D user interfaces, using different metaphors such as that of a flying vehicle. [Tan et al. 2001] combine speed-coupled flying with orbiting. McCrae *et al.* [McCrae et al. 2009] adapt the travel speed of the flying vehicle in a space-scale-aware manner using a cubemap technique to enable navigation in multi-scale environments. Navidget [Hachet et al. 2008] is a gesture-based interactive widget that enables users to get a preview from a different perspective on the scene before actually repositioning and reorienting the main camera to match this new point of view. Another way to ease navigation is to constrain camera movements to object dependent surfaces [Khan et al. 2005] or authored surfaces [Burtnyk et al. 2002].

2.3 Smart Visibility

Smart Visibility originates from static technical illustrations [Viola and Gröller 2005]. It aims at conveying the most information possible using a single, often static illustration by unveiling its most important parts. Smart visibility techniques are based on cutaway views [Feiner and Seligmann 1992; Li et al. 2007], ghosted-views [Bruckner et al. 2005; Viola et al. 2005; Kruger et al. 2006; Correa and Ma 2009; Correa and Ma 2011] or spatial rearrangement [Li et al. 2008; McGuffin et al. 2003]. Burns *et al.* [Burns and Finkelstein 2008] propose adaptive cutaways that adapt their geometry to guarantee the visibility of predefined parts of interest. While some of these techniques do make it possible to interactively select the main point of interest to be revealed [Li et al. 2007; Li et al. 2008], these are still aimed at producing static illustrations. They do not provide a true solution for exploring complex 3D scenes.

Gimlenses are conceptually related to the 3D occlusion management techniques surveyed in [Elmqvist and Tsigas 2008], and more specifically to the x-ray category of tools: perspective cutout [Coffin and Hollerer 2006], x-ray tunnels [Bane and Hollerer 2004], and the looking glass from [Looser et al. 2004]. Such techniques also provide solutions for revealing hidden parts of a 3D scene, as do Gimlenses. However, those techniques are used *in addition* to a 3D navigation technique, which increases the burden put on users, who have to handle both navigation and visual occlusion management conjointly but sequentially. On the contrary, Gimlenses integrate occlusion management with 3D navigation, automatically revealing the focused part of the scene to the observer. This integration

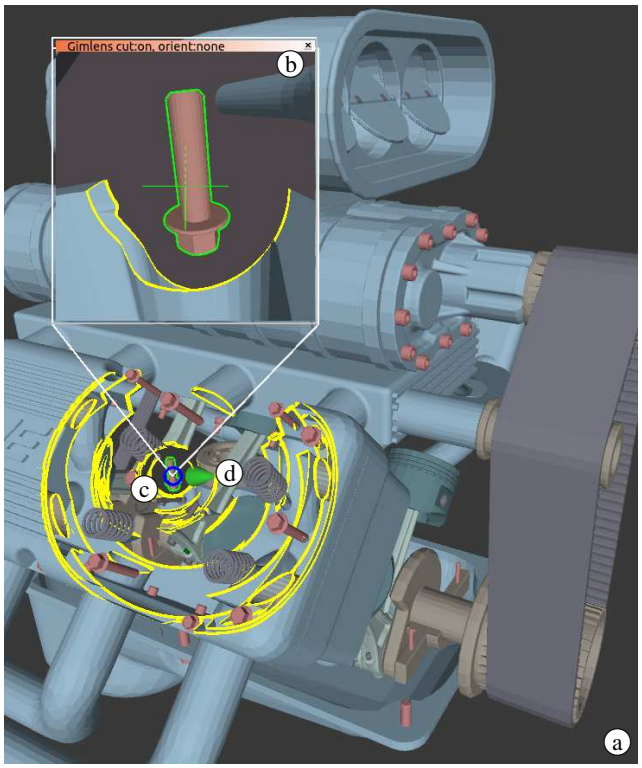

Figure 2: *A Gimlens magnifying a knot deep inside a car engine. Context view (a); Gimlens view window (b), object selector (c), proxy (d).*

facilitates exploration no matter the type of display, but is especially useful in contexts where input capabilities are limited or where interaction is more challenging than on a desktop workstation such as, e.g., wall-sized displays operated from a distance or via a touch-sensitive surface. Another distinctive feature of Gimlenses compared to earlier work are their capacity to get coordinated and cascaded, as detailed in Section 3.6.

3 Gimlens

Gimlenses provide users with multiple, possibly coordinated, detail-in-context views on 3D scenes that provide support for the deep exploration of complex layered models at different scales. We first describe the interface's main features. Then we explain how holes are drilled into the model to guarantee the visibility of objects of interest while preserving context. The following section introduces more advanced features: automatic orientation constraints and lens cascading. We conclude with implementation details and performance figures, followed by a discussion of current limitations and future work.

3.1 Main Interface Components

As in all focus+context techniques, the purpose of the main viewport is to provide users with an overview of the scene. This viewport, which we call the *context view* (Figures 1-a and 2-a), can be freely rotated, panned and zoomed using conventional techniques adapted to the display platform.

From this context view, users may instantiate multiple Gimlenses that can then be used to drill into the model (Figure 1). Each Gimlens is composed of four main elements: the *view window*, the *object selector*, the *lens proxy* and the *cone-shaped cut*. The view window (Figure 2-b) is an independent viewport that users can freely move and resize. It contains the detailed view that corresponds to the lens' focus region. The object selector, symbolized by a blue ring (Figure 2-c), identifies the current object of interest in the model, focused on by the lens. The object selector and view window are visually linked together by two line segments, called tethers in [Ware et al. 1995]. While this set of elements is sufficient to relate the detail and context views in 2D visualizations, additional information has to be conveyed to users to achieve full 3D linking [Plumlee and Ware 2003] of views. This is especially important in the case of scenes that contain numerous layers and closely clustered parts and pieces, as those are prone to occluding one another, making it even more challenging for users to get a good sense of spatial orientation.

The lens proxy (green cone in Figure-2-d) indicates the orientation of the Gimlens. The cone is oriented such that its axis is aligned with the line of sight of the corresponding Gimlens, its tip oriented towards the observer. The cone is rendered as an opaque, smoothly shaded object with lighting effects to help users evaluate its orientation in the scene. To help them evaluate its position in the scene, the cone is blended with the elements of the model that should actually occlude it, as illustrated in Figure-1-c (red cone).

Finally, as users inspect objects in the scene by repositioning the Gimlens (Section 3.2), we perform a cone-shaped cut (Section 3.5) to guarantee the visibility of the object of interest from the context view by drilling into the model all the way down to this object (Figure 2, outlined yellow cuts into the cylinder head cover and beyond). We further enhance the readability of this representation by outlining the object of interest both in the view window and in the context view (knot with green outline, Figures 2-b and 2-c).

3.2 Repositioning a Gimlens

There are multiple ways to reconfigure the point of view of a Gimlens on the model. Users can grab the object selector (blue ring, Figure 2-c) and drag it in the context view. As the Gimlens' focus point slides on the surface of the model, following the object selector, the viewing frustum gets adjusted accordingly, depending on some properties detailed later (automatic orientation constraints). The object of interest gets updated as soon as the focus point falls on another object.

Users can also make the lens orbit around the object of interest by dragging the lens proxy (green cone, Figure 2-d) around it. Finally, the Gimlens' viewing frustum can be reconfigured by freely rotating, panning and zooming the lens' viewport using dragging, as when reconfiguring the context view. The controls should be mapped to input device buttons in a consistent manner between the two types of viewports.

3.3 Drilling into the Model

The set of interactions described above make it possible to inspect the outside of complex 3D models, but are of little use when inspecting the inside of those models. Many models, including CAD drawings of engines or planes, 3D maps of buildings and plants, or layered representations of the human anatomy are composed of large amounts of objects, many of which are inside other objects.

A foundational feature of Gimlenses is that they let users drill into the model supporting its exploration and allowing for fast access to objects that are buried, possibly deep, into it. Users can iteratively

The technique's name is inspired by the word *gimlet*, a tool for drilling small holes, mainly in wood.

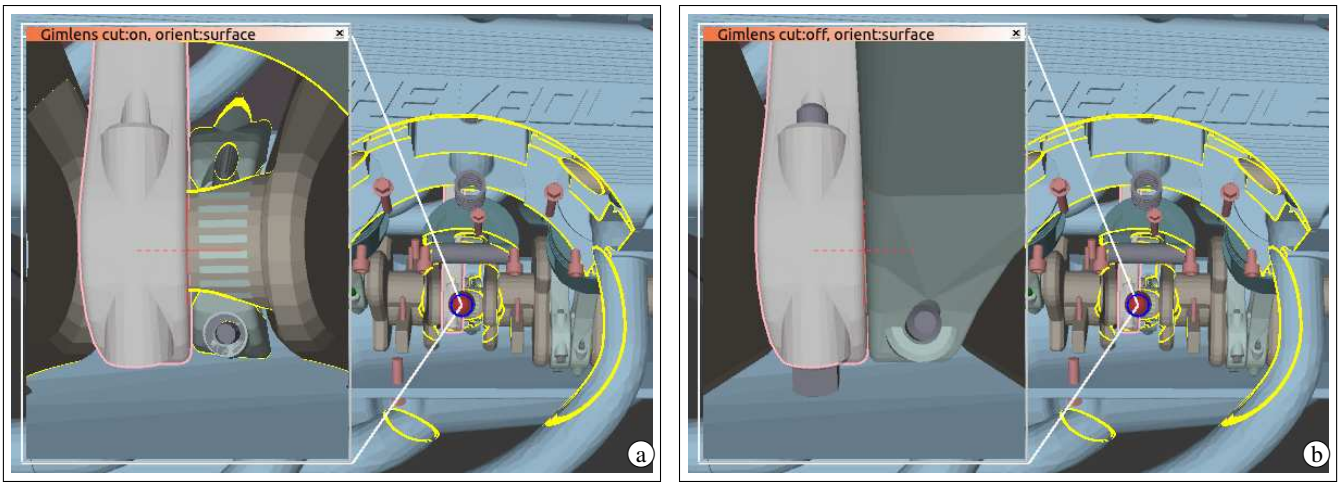

Figure 3: *A Gimlens verifying the surface intersection between a connecting-rod and the camshaft. (a) Enabling the cut reveals the object of interest during exploration but hides its neighbors. (b) Disabling the cut enables users to see both parts.*

cut holes into the layers of the model. The holes get automatically adjusted to reveal the new object of interest and clear a line of sight to the point of view set as the context view. The holes guarantee that this object remains visible in the context view even if the latter's viewing frustum gets reconfigured.

Cuts are shaped as truncated cones whose axis is oriented from the object of interest to the observer's eye. The cone itself is invisible, but hides every object that intersects it. If an object only partially intersects the cone, the geometrical intersection between the two shapes is computed and only the part that falls within the cone is made invisible.

The cone's geometry depends on the size of the object of interest, on the position of the focus point on this object (determined by the position of the selector – blue ring – on the object), and on the viewing frustums of both the context view and Gimlens. Being truncated, the cone has a top and a base, which we call the near and far bases in reference to the near and far clipping planes of a viewing frustum. The near and far bases are located close to the observer and close to the object of interest, respectively.

An interesting property of the cone is that the size of cuts decreases as the parts being cut are located deeper into the model. No matter the projection considered (perspective or orthogonal), this helps users get a better sense of how deep the focus point is, as they can always see how many layers got cut to get to the object of interest. To further emphasize this, we outline the surfaces that resulted from intersecting the cone with the parts that originally overlapped it (yellow strokes in all figures).

Cuts are also performed in the Gimlens viewport, coupling the near base to the apex of the lens' viewing frustum so that the object of interest always remains visible no matter how the former gets positioned. However, this behavior is not always desirable. For instance, when instantiating a Gimlens to inspect the intersection between the surfaces of two distinct parts in the model, the cut cone would hide one of those surfaces, hence hiding an important piece of information, as illustrated in Figure 3. Cuts performed inside the lens' view window can thus be toggled on/off.

3.4 Adjusting Depth

We designed an interaction technique based on the above drilling method, that enables users to navigate inside a 3D model by successively drilling holes into it starting from the outer layer.

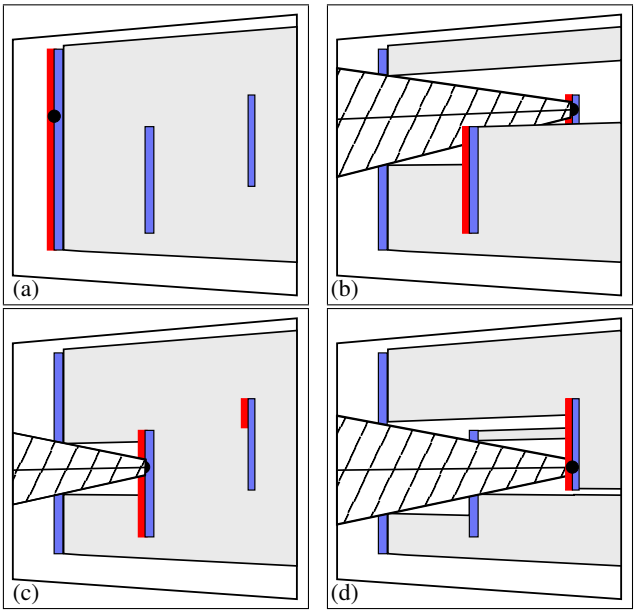

Figure 4: *Illustration of how the drilling technique behaves. (a) The focus point lies on the frontmost part. This part gets drilled into, revealing the other parts behind it. (c) The focus point gets slid on the parts visible from the observer's perspective (colored red) and, finally, (d) cuts through the second layer (d).*

On a desktop workstation, the technique is operated with the mouse wheel. In other environments, such as virtual reality platforms or wall displays, the corresponding actions have to be mapped to a linear continuous input control channel on one of the available interaction devices. Invoking the technique on the current part of interest (on which the selector lies) cuts out a hole in it and moves the focus point on the surface below, that was just revealed by the cut. Users can also climb back, filling the holes previously cut in reverse order, with the focus point and object of interest getting updated accordingly. The cut cone automatically adjust its geometry, following the focus point controlled with the cursor. Figure 4 illustrates this process.

Gimlenses keep a history of what parts got cut, allowing the user to drag the focus point from one part of interest to the next and drill

down iteratively. This navigation method can be seen as an easy way of toggling the visibility of the successive parts that the focus point falls on during the drilling process, except that each part's visibility is only affected locally around the line of sight so as to better preserve context.

3.5 Optimizing Cuts

Cuts are an essential features of Gimlens. They enable navigating inside models and act as visual cues that help situate objects of interest within the model. A good navigation experience depends on guaranteeing a good visibility of the focus point from the lens, on the readability of the depth cues and on the predictability of the interface's behavior, and on the performance in terms of frame rates. All of these factors depend on the shape of the cut.

Before opting for a truncated cone shape, we experimented with a content-aware dynamically adapting cut shape similar to the technique presented in [Burns and Finkelstein 2008]. The cut was adapting to the geometry of the Gimlens's current object of interest. Our intuition was that by adapting the shape of the cut to the shape of the content, we would optimize the portion of the scene to be cut, thus preserving more context information, in the same spirit as JellyLenses [Pindat et al. 2012], which dynamically adapt the shape of a fisheye to optimize the focus, context and distorted regions to provide more relevant focus+context representations. But an informal evaluation performed in our laboratory revealed that users preferred to interact with cuts that always had a cone-based shape. The cut shape radically changing its geometry depending on the geometric characteristics of the various objects of interest was found to be disturbing, as it was both unstable and unpredictable.

Good rendering performance is also essential to achieve a good user experience. Our prototype is based on a rasterization rendering method. The cut is implicitly defined by a piece-wise function that we compute for each fragment of primitives resulting from the rasterization. We note p the position of the focus point in eye-space coordinates, and m the position of the fragment being processed. We define x, the projection of the fragment on the cone's axis, and y, the distance of the fragment to the cone's axis as follows:

$$x = \frac{dot(m,p)}{\|p\|},\ y = \frac{\|m \times p\|}{\|p\|}$$

The near base, with a radius of S, controls the wideness of the aperture. A large base favors the visibility of what is inside the cut, but does so at the expense of the context since more of the model gets hidden. Users can control this parameter, but we have found a base radius that corresponds to roughly 25% of the main viewport's size to yield a good compromise between cut visibility and context preservation. We note n the distance from the near base to the camera's position. If a fragment is situated closer to the camera than the near base, i.e., if $x < n$, it gets discarded if: $y < S * x/n$.

The far base is positioned at the focus point. f, its distance from the observer in eye space coordinates, is equal to $\|p\|$. Its radius, s in eye-space coordinates, is computed automatically, so that the object of interest can fully fit within the cone, with some tolerance.

Additional constraints are enforced, so as to make sure that the far base does not get larger than the near base (which would not make sense) and that the cone's angle does not get larger than $20°$. Fragments between the near base and the far base, i.e., for which $n < x < f$, get discarded if:

$$y < (x-n) * \frac{s}{f-n} + (f-x) * \frac{S}{f-n}$$

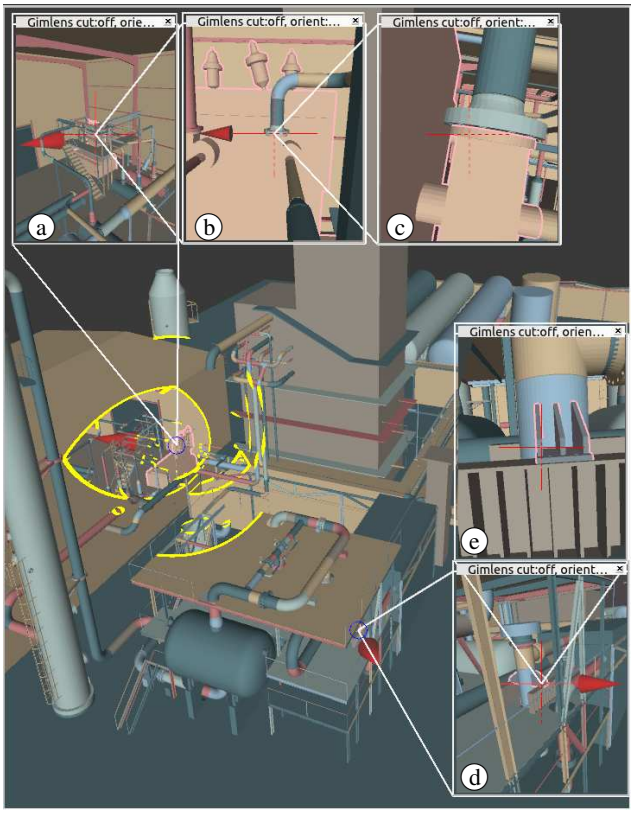

Figure 5: *Exploring an air distillation plant. Three Gimlenses are cascaded to magnify a steal pipe. (a) shows a view from the left to better situate the focused part within the shed, (b) and (c) apply magnification successively. (d) and (e) magnify a distinct part of the plant.*

To avoid sharp edges at the far base, we make the far base round by extending it with a spherical shape that gets merged with the truncated cone. Fragments behind the far base ($f < x$) get discarded if:

$$\|(x - (f - s * \frac{(S-s)}{f-n}), y)\| < s * \|(\frac{(S-s)}{f-n}, 1)\|$$

Finally, small parts that lie within the cut cone but are not causing significant obstruction should not be hidden, as they will often provide interesting contextual information. Our technique takes the size of a part that falls in the cone into consideration before deciding whether to hide it or not. If the part is big enough to fully occlude the object of interest, we apply the full cut. Otherwise we just make a small hole into it to reveal the focus point when it lies precisely on the line of sight, as illustrated in Figure 2.

3.6 Combining and Cascading Lenses

Multiple Gimlenses can be instantiated simultaneously to provide users with different perspectives on the 3D model: different locations, different scales, and different angles. This is useful, e.g., when comparing multiple parts of a model, such as two knots inside an engine. Gimlenses can be created for both, the user then drilling towards each knot, adjusting the view and then orbiting these two objects of interest.

Gimlenses can also be cascaded, in a way somewhat similar to what PolyZoom does for 2D maps [Javed et al. 2012]. Cascaded lenses

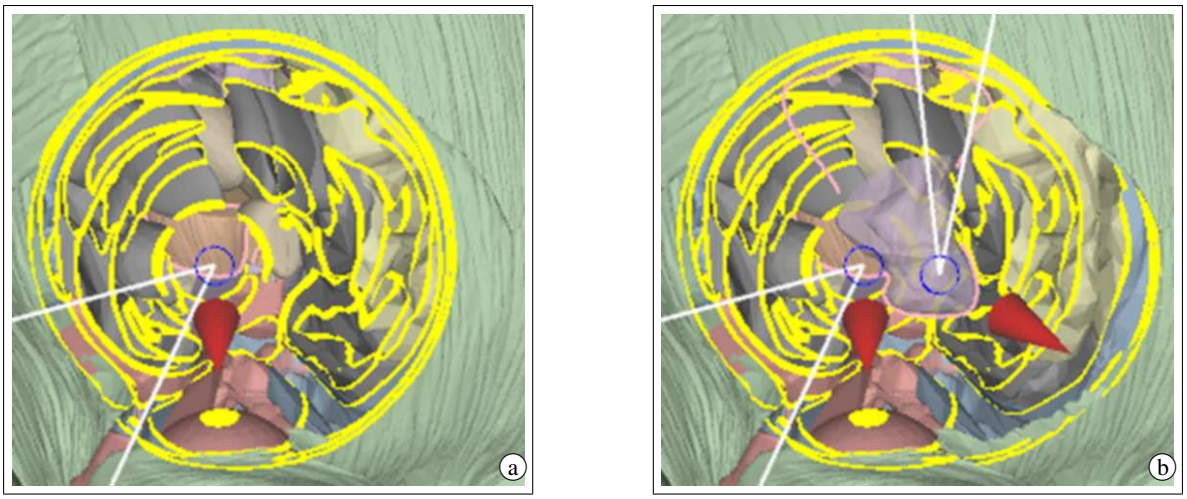

Figure 6: *Exploring a brain. (a) A first Gimlens gets instantiated and delves deep inside the head, hiding several objects in the process. (b) A second Gimlens gets instantiated to look at an object that lies in the cut cone of the first lens, causing a conflict. The portion of this object that lies near the focus point of the second lens gets rendered translucently.*

share the same focus point. They enable users to easily get multiple views on the same object of interest at varying scales and from different angles. Figure 5 gives an example. A user wishes to inspect a steal pipe in an air distillation plant. She cascades several Gimlenses, that give her points of view on the region of interest at different scales.

Instantiating multiple lenses can lead to conflicts if the focus point of one lens lies on a part of the model that got cut while drilling with another lens. To resolve this sort of conflict, our technique renders the portion, centered on the first focus point, of the object that is the source of the conflict using alpha blending to make it translucent. The part is thus revealed, without hiding the second focus point. See Figure 6.

3.7 Automatic Lens Orientation and Coordination

Gimlenses let users define orientation constraints between cascaded lenses. Lenses constrained in this way automatically update their orientation as the focus point is updated. This spares users the burden of having to reorient each camera to get the desired angle on the object of interest as the focus point gets moved, a process that is quite tedious to perform manually. We define three orientation constraints that enable different behaviors.

Figure 7 illustrates the potential of this feature. The first lens is set up with a **Pivot** constraint, that maintains the camera position in-place as it pivots around to follow the focus point. This is particularly useful when exploring the inside of cavities. The two other lenses are set up with a **Parent** orientation constraint. This means that their orientations are defined relative to the one of their parent lens (the first lens). With these constraints set up, as we slide the focus point along the teeth from left to right, the lenses automatically move along and rotate, maintaining views from both inside and outside the patient's mouth without the need for users to make any adjustment themselves.

When inspecting the interface between two surfaces, users can select the **Surface** constraint, that will preserve the view orientation relative to the orientation of the surface's normal at the focus point. During relocation of the focus point, or when changing the context view's viewing frustum, Gimlenses will automatically adjust the camera to preserve the view orientation on the surface, again sparing users the burden of manually adjusting the lens' cameras. As the focus point jumps from one polygon to the next, **Surface** orientation constraints might cause the view in the lens to change abruptly. To avoid this, we smoothly animate the transition over 100ms.

4 Implementation

We implemented our prototype in C++ using OpenGL. The implicit definition of the cone cut is written with GLSL in a fragment shader. Fragments falling inside the cut cone are discarded. Those intersecting its surface are outlined in yellow, and those falling outside are kept in the pipeline.

Changing the focus point or adjusting the viewing frustum of the context view requires casting rays into the scene to find all intersections with model parts. To ensure an interactive frame rate even with complex models, we make use of a space-partitioning data structure to optimize the ray casting. In our implementation, we currently use an AABB (axis-aligned bounding box) tree data structure.

As summarized in Table 1, we achieve interactive frame rates for all models demonstrated in this article. Performance figures were gathered on a HP Z-series 800 PC equipped with an nVidia Quadro 4000, and an Intel Xeon E5-1650 CPU running at 3.20GHz. The framebuffer size was set to 1920x1080 (running fullscreen on a Full HD monitor). The technique could be made to scale further by coupling it with state-of-the-art computer graphics techniques that can render extremely complex 3D polygonal scenes.

Model	Million triangles	1 Gim. / 2 Gim. (fps)
Air Plant	1600	15 / 10
Car Engine	266	57 / 40
Head	58	63 / 45

Table 1: *Performance (Frame-rates)*

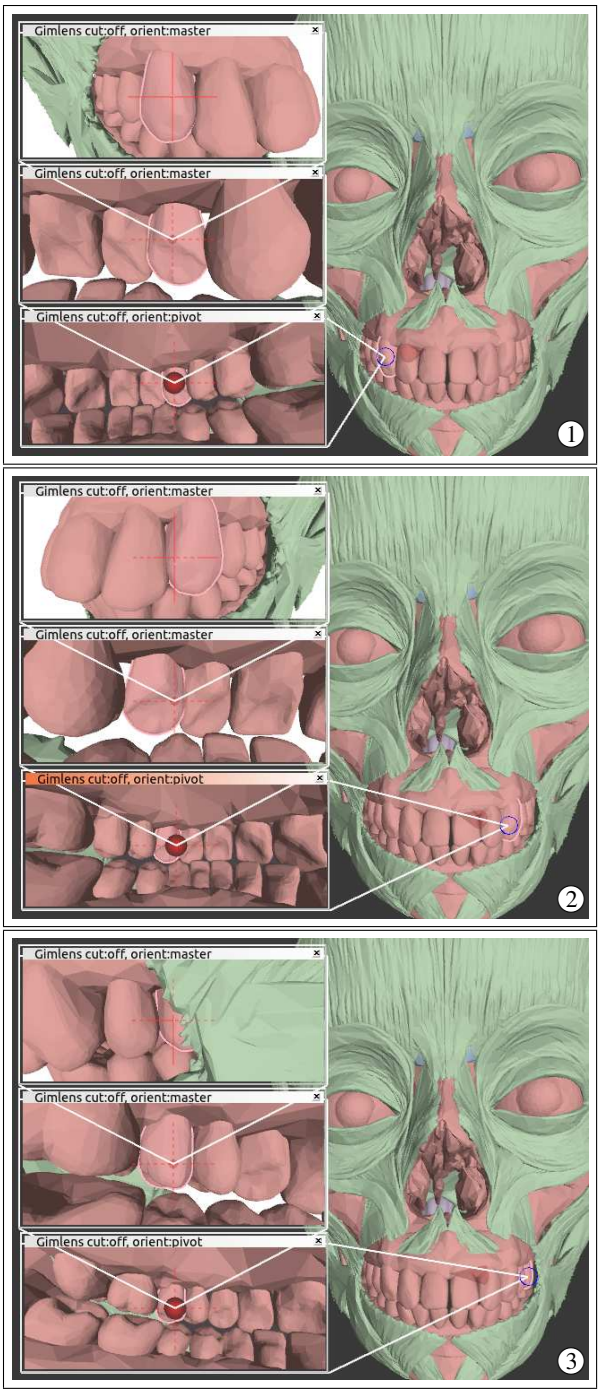

Figure 7: *Three Gimlenses combined to explore the inside of the maxillary dental arcade of a patient using different orientation constraints. Dragging the single focus point that controls all three lenses updates all views automatically, providing complementary perspectives on the successive teeth (1,2,3).*

5 Discussion and Future Work

We presented a multi-view, detail-in-context technique for navigating complex 3D scenes. As demonstrated throughout the article, the technique can prove useful in a variety of domains: for inspecting an industrial CAD model of a car engine, for exploring an anatomical representation of the human head, or walking through a 3D map of an air distillation plant.

The datasets we experimented with provided separate geometrical descriptions of each element in the model. Gimlenses can be applied in a straightforward manner in such cases, as the scene is already segmented into meaningful objects. Unfortunately, this is not the case for all datasets. For instance, if we consider the polygonal meshes generated from 3D scans of scenes such as art pieces or even entire cities, those can reach a high degree of complexity due to the very nature of the scene that was scanned and the precision of the technology now available to achieve such scans. These are good candidates for exploration with Gimlenses. However, those datasets are not segmented into meaningful components, and it would be necessary to pre-process them before they could be explored with our technique. Another option would be to design interaction techniques that would let users define their own segmentation in an easy way, either declaratively or by demonstration. The resulting segmentation could then be used as input by Gimlenses. We are faced with the same segmentation issue when considering volumetric datasets. Moreover, in this case, users might want to explore the fundamental, primitive elements of the data: the voxel. This would require extending Gimlenses to support the per-voxel inspection of the dataset. All considerations related to such segmentation of the scene into meaningful objects, however, are beyond the scope of this work.

Another aspect that requires further work is the mapping of Gimlens parameters to user interface controls. While the mapping to a wheel-equipped mouse or even a gesture-enabled trackpad for a laptop or desktop workstation is straightforward, the mapping to the more exotic input devices typically used in CAVEs or when interacting with wall-sized displays (motion tracking, mid-air pointing device, etc.) requires more thinking. This, however, highly depends on the available devices and their capabilities as well as, to some extent, on the application-dependent mappings already in place. In the particular case of CAVEs, additional challenges will arise, as the lens windows' position should be adapted automatically to match observers' position and orientation in the scene.

Finally, another potential area for improvements is that of graphics rendering. Currently we apply the same rendering technique to all lenses and to the context view. But as views might serve different purposes, enabling different rendering strategies, as Magic Lenses do [Wang et al. 2005], or even simply enabling users to easily change lighting conditions to emphasize some features of the data, would be useful improvements.

References

APPERT, C., CHAPUIS, O., AND PIETRIGA, E. 2010. High-precision magnification lenses. In *Proc. CHI '10*, ACM, 273–282.

BANE, R., AND HOLLERER, T. 2004. Interactive tools for virtual x-ray vision in mobile augmented reality. In *Proc. ISMAR '04*, ACM IEEE, 231–239.

BIER, E. A., STONE, M. C., PIER, K., BUXTON, W., AND DEROSE, T. D. 1993. Toolglass and magic lenses: the see-through interface. In *Proc. SIGGRAPH '93*, ACM, 7380.

BRUCKNER, S., GRIMM, S., KANITSAR, A., AND GRÖLLER, M. E. 2005. Illustrative context-preserving volume rendering. In *Proc. EUROVIS'05*, Eurographics Association, 69–76.

BURNS, M., AND FINKELSTEIN, A. 2008. Adaptive cutaways for comprehensible rendering of polygonal scenes. *ACM Transactions on Graphics 27*, 5, 154:1–154:7.

BURTNYK, N., KHAN, A., FITZMAURICE, G., BALAKRISHNAN, R., AND KURTENBACH, G. 2002. Stylecam: interactive stylized 3d navigation using integrated spatial & temporal controls. In *Proc. UIST '02*, ACM, 101–110.

CARPENDALE, M. S. T., AND MONTAGNESE, C. 2001. A framework for unifying presentation space. In *Proc. UIST '01*, ACM, 61–70.

CARPENDALE, M. S. T., COWPERTHWAITE, D. J., AND FRACCHIA, F. D. 1997. Extending distortion viewing from 2D to 3D. *IEEE Comput. Graph. Appl. 17*, 4 (July), 42–51.

COCKBURN, A., KARLSON, A., AND BEDERSON, B. B. 2008. A review of overview+detail, zooming, and focus+context interfaces. *ACM Computing Surveys 41*, 1 (Dec.), 131.

COFFIN, C., AND HOLLERER, T. 2006. Interactive perspective cut-away views for general 3d scenes. In *Proc. 3DUI '06*, IEEE, 25–28.

CORREA, C., AND MA, K.-L. 2009. The occlusion spectrum for volume classification and visualization. *IEEE Transactions on Visualization and Computer Graphics 15*, 6, 1465–1472.

CORREA, C. D., AND MA, K.-L. 2011. Visibility histograms and visibility-driven transfer functions. *IEEE Transactions on Visualization and Computer Graphics 17*, 192–204.

CRUZ-NEIRA, C., SANDIN, D. J., DEFANTI, T. A., KENYON, R. V., AND HART, J. C. 1992. The cave: audio visual experience automatic virtual environment. *Communications of the ACM 35*, 6 (June), 64–72.

DIETRICH, A., STEPHENS, A., AND WALD, I. 2007. Exploring a boeing 777: Ray tracing large-scale cad data. *IEEE Computer Graphics and Applications 27*, 6, 36–46.

ELMQVIST, N., AND TSIGAS, P. 2008. A taxonomy of 3D occlusion management for visualization. *IEEE Transactions on Visualization and Computer Graphics 14*, 5.

FEINER, S. K., AND SELIGMANN, D. D. 1992. Cutaways and ghosting: satisfying visibility constraints in dynamic 3D illustrations. *The Visual Computer 8*, 5-6 (Sept.), 292–302.

HACHET, M., DECLE, F., KNODEL, S., AND GUITTON, P. 2008. Navidget for easy 3D camera positioning from 2D inputs. In *Proc. 3DUI '08*, IEEE, 83–89.

JAVED, W., GHANI, S., AND ELMQVIST, N. 2012. Polyzoom: multiscale and multifocus exploration in 2d visual spaces. In *Proc. CHI '12*, ACM, 287–296.

KHAN, A., KOMALO, B., STAM, J., FITZMAURICE, G., AND KURTENBACH, G. 2005. HoverCam: interactive 3D navigation for proximal object inspection. In *Proc. I3D '05*, ACM, 73–80.

KRUGER, J., SCHNEIDER, J., AND WESTERMANN, R. 2006. Clearview: An interactive context preserving hotspot visualization technique. *IEEE Transactions on Visualization and Computer Graphics 12*, 5, 941–948.

LI, W., RITTER, L., AGRAWALA, M., CURLESS, B., AND SALESIN, D. 2007. Interactive cutaway illustrations of complex 3D models. *ACM Transactions on Graphics 26*, 3, 31:1–31:12.

LI, W., AGRAWALA, M., CURLESS, B., AND SALESIN, D. 2008. Automated generation of interactive 3D exploded view diagrams. *ACM Transactions on Graphics 27*, 3, 101:1–101:7.

LOOSER, J., BILLINGHURST, M., AND COCKBURN, A. 2004. Through the looking glass: the use of lenses as an interface tool for augmented reality interfaces. In *Proc. GRAPHITE '04*, ACM, 204–211.

MCCRAE, J., MORDATCH, I., GLUECK, M., AND KHAN, A. 2009. Multiscale 3D navigation. In *Proc. I3D '09*, ACM, 7–14.

MCGUFFIN, M. J., TANCAU, L., AND BALAKRISHNAN, R. 2003. Using deformations for browsing volumetric data. In *Proc. VIS '03*, IEEE, 401–409.

NANCEL, M., WAGNER, J., PIETRIGA, E., CHAPUIS, O., AND MACKAY, W. 2011. Mid-air pan-and-zoom on wall-sized displays. In *Proc. CHI '11*, ACM, 177–186.

PIETRIGA, E., BAU, O., AND APPERT, C. 2010. Representation-independent in-place magnification with sigma lenses. *IEEE Transactions on Visualization and Computer Graphics 16*, 03, 455–467.

PINDAT, C., PIETRIGA, E., CHAPUIS, O., AND PUECH, C. 2012. Jellylens: content-aware adaptive lenses. In *Proc. UIST '12*, ACM, 261–270.

PLUMLEE, M., AND WARE, C. 2003. Integrating multiple 3d views through frame-of-reference interaction. In *Proc. CMV '03*, IEEE, 34–43.

PLUMLEE, M. D., AND WARE, C. 2006. Zooming versus multiple window interfaces: Cognitive costs of visual comparisons. *ACM Transactions on Computer-Human Interaction 13*, 2 (jun), 179–209.

ROPINSKI, T., AND HINRICHS, K. 2004. Real-time rendering of 3D magic lenses having arbitrary convex shapes. In *Journal of the International Winter School of Computer Graphics (WSCG04)*, Science Press, 379–386.

TAN, D. S., ROBERTSON, G. G., AND CZERWINSKI, M. 2001. Exploring 3D navigation: combining speed-coupled flying with orbiting. In *Proc. CHI '01*, ACM, 418425.

VIEGA, J., CONWAY, M. J., WILLIAMS, G., AND PAUSCH, R. 1996. 3D magic lenses. In *Proc. UIST '96*, ACM, 5158.

VIOLA, I., AND GRÖLLER, M. E. 2005. Smart visibility in visualization. In *Proc. Computational Aesthetics'05*, Eurographics Association, 209–216.

VIOLA, I., KANITSAR, A., GRLLER, M. E., AND GROLLER, M. E. 2005. Importance-driven feature enhancement in volume visualization. *IEEE Transactions on Visualization and Computer Graphics 11*, 4, 40818.

WANG, L., ZHAO, Y., MUELLER, K., AND KAUFMAN, A. 2005. The magic volume lens: An interactive focus+context technique for volume rendering. In *Proc. VIS '05*, IEEE, 367–374.

WANG, Y.-S., WANG, C., LEE, T.-Y., AND MA, K.-L. 2011. Feature-preserving volume data reduction and Focus+Context visualization. *IEEE Transactions on Visualization and Computer Graphics 17*, 2 (Feb.), 171 –181.

WARE, C., LEWIS, M., AND CREATIVITY, O. F. 1995. The DragMag image magnifier. In *Proc. CHI '95 EA*, ACM, 407–408.