# Comprehensible Visualization for Augmented Reality

Denis Kalkofen, *Student Member*, *IEEE*, Erick Mendez, *Student Member*, *IEEE*, and Dieter Schmalstieg, *Member*, *IEEE*

**Abstract**—This article presents interactive visualizations to support the comprehension of spatial relationships between virtual and real-world objects for Augmented Reality (AR) applications. To enhance the clarity of such relationships, we discuss visualization techniques and their suitability for AR. We apply them on different AR applications with different goals, e.g., in x-ray vision or in applications that draw a user's attention to an object of interest. We demonstrate how Focus and Context (F+C) visualizations are used to affect the user's perception of hidden or nearby objects by presenting contextual information in the area of augmentation. The organization and possible sources of data for visualizations in AR are presented, as well as cascaded multilevel F+C visualizations to address complex cluttered scenes that are inevitable in real environments. This article furthermore shows filter operations and tools to interactively control the amount of augmentation. It compares the impact of real-world context preservations to a pure virtual and uniform enhancement of occluding structures for augmentations of real-world imagery. Finally, this paper discusses the stylization of sparse object representations to improve the comprehension of x-ray visualizations in AR.

**Index Terms**—Data structures, graphs and networks, methodology and techniques, interaction techniques, graphics data structures and data types, multimedia information systems, artificial, augmented, and virtual realities, user interfaces, style guides.

## 1 Introduction

Augmented Reality (AR) displays extend the user's perception with additional computer-generated information. This information is usually registered in 3D space and related to objects and places in the real world. If we consider AR as a visualization technique, the relationship of real and virtual objects is one of focus and context: Either we want to provide additional virtual context to an important object in the real world or we want the user to focus on a virtual object embedded in a real context.

In both cases, AR generates the final image by overriding parts of the real-world imagery with synthetic images. However, heedless replacement of portions of the real world can easily cause a number of cognitive problems. For example, "x-ray" visualizations render hidden structures over visible objects, but careless augmentation with synthetic imagery may obscure important real-world information, as illustrated in the simulated surgery depicted in Fig. 2. The computer-generated augmentation occludes highly relevant information present in the real-world imagery. In this case, the objective is to insert a needle into a patient's abdomen using a predefined entry point (black crossing). While visualizing the internal anatomy using AR, the user is unable to see the entry marks for the needle placed on the skin of the patient. The visual augmentation furthermore leads to problems of depth perception, caused by the heedless removal of depth cues (Fig. 2a). Fig. 3b shows a better augmentation that considers edges extracted from the real video stream with our framework. The extracted features provide additional depth cues, and since they come from the real imagery, they are also able to preserve important landmarks (such as the entry points), in addition to being perfectly registered.

The problem of correct depth perception has been a common subject of research in the past and has been addressed by a number of strategies, mainly based on adding monocular depth cues to the scene [3]. For example, [9] adds a window-shaped restriction to the rendering area in order to enhance depth perception through partial occlusion. However, none of the existing approaches take into account what is actually removed from the real-world imagery.

Another class of problems is caused by the amount of information added to reality. For example, if too much information is added in areas that were unimportant in the original image, the impression of a cluttered display may be caused. Furthermore, if important information from both real and virtual is present in the same area, the visualization algorithm should aim for an optimal combination of all features, rather than always giving preference to the virtual features over the real ones. Otherwise, this can cause grave problems as, for example, the one illustrated in Fig. 1, where the system tries to draw the attention of the user to the middle part of the robot by overlaying a red box on top of it. However, it completely occludes the object it wants to draw attention to.

In this paper, we address the task of sensibly blending virtual and real-world imagery. We take into account the information that is about to be occluded by our augmentations, as well as the visual complexity of the computer-generated augmentations added to the view. Our work is

• *The authors are with the Institute for Computer Graphics and Vision, Graz University of Technology, Inffeldgasse 16a, A-8042 Graz, Austria. E-mail: {kalkofen, mendez, schmalstieg}@icg.tugraz.at.*

*Manuscript received 4 Mar. 2008; revised 29 May 2008; accepted 30 June 2008; published online 10 July 2008.*
*Recommended for acceptance by T. Ertl and M.A. Livingston.*
*For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCGSI-2008-03-0034.*
*Digital Object Identifier no. 10.1109/TVCG.2008.96.*

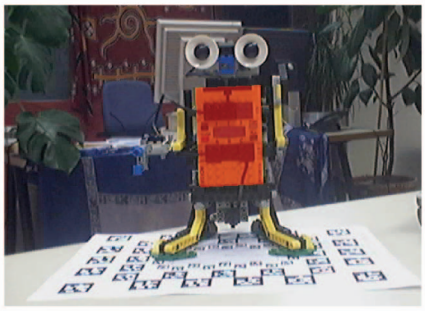
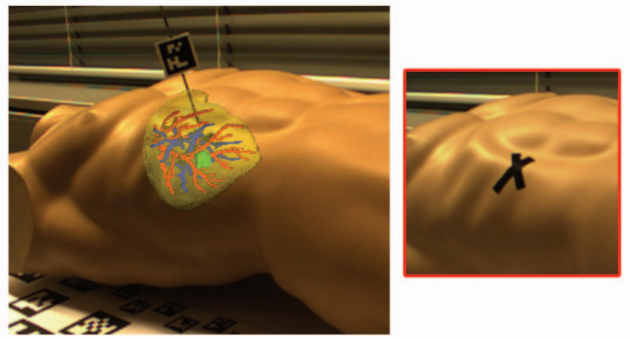


Fig. 1. Occluding augmentations. A typical AR visualization to draw the user's attention with overlaid semitransparent geometry, occluding the object of interest.

inspired by research on focus and context (F+C) visualization techniques and specifically by importance-driven rendering [22]. Rather than using a single scalar importance value, we distinguish scene objects by a vector of attributes describing the object's importance at different levels, evaluated per pixel in a programmable shader. This process can incorporate multiple levels of importance through the use of a user-scripted shader tree. Our AR visualization framework allows the controlled removal of real-world information. Moreover, rather than merely removing information, objects can be styled according to their importance relative to other objects in its vicinity. This process can be manipulated interactively, for example, with Magic Lens tools for a selective clutter removal.

In our previous work [12], we have applied basic F+C techniques to AR, especially to x-ray vision. This paper extends it to a generalized framework and discusses the advantages and disadvantages of various filter operations (Section 4). In particular, it introduces cascaded and multilevel F+C visualizations, which are better suited to address complex cluttered scenes that are inevitable in real environments.

We further extended our previous paper with a discussion about stylizations of sparse representations to improve context-preserved x-ray visualizations. Finally, this paper discusses possible virtual and real preservations in AR (Section 4.4). We compare sparse virtual overlays with sparse preservations of real-world imagery, coming from the AR system's video feed.



(a) (b)

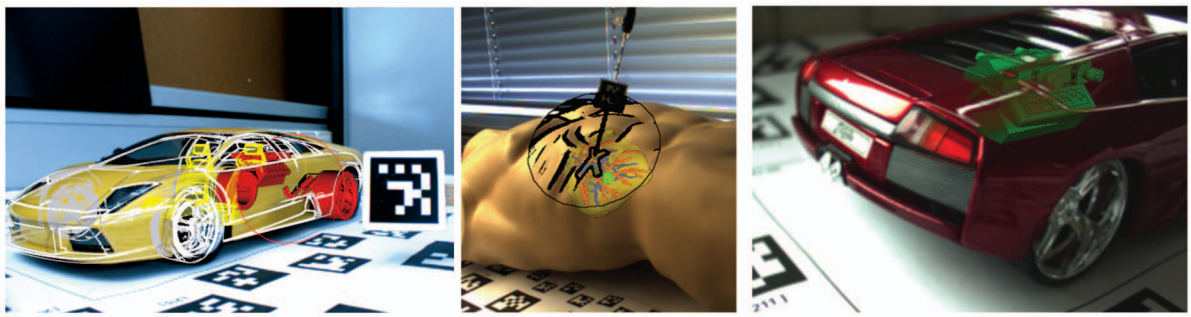
Fig. 2. Naïve augmentations. Careless augmentations of hidden structures suffer two key problems: they override useful information (such as landmarks) and they lack depth cues. (a) Augmentation of the liver with its portal and hepatic vessel trees (red and blue) and a tumor (green). (b) Original photo before augmentation, the black cross indicates the entry port for the rfa-needle.

## 2 Related Work

F+C techniques build the core of our framework. They are important topics across several disciplines such as human-computer interaction, volume visualization, and rendering. Following the overview of F+C techniques in [13], the creation of F+C visualizations can be roughly divided into two major steps: data classification and data visualization.

The objective of the first step, data classification, is to identify the roles in the data, i.e., what information should be focus and what should be context. The relationship between focus and context data may be either spatially driven or knowledge driven. Spatially driven techniques retain or enhance information in close proximity to the focus object. Knowledge-driven techniques do not depend on spatial proximities; instead, they use semantic interpretations of a given set of objects or scene parameters to distinguish between focus and context objects. Such interpretation can be directly controlled by the user, based on task knowledge [11] or inferred from the contextual information associated with the scene [17].

Interactive data separation is usually performed based on user input. There are a number of approaches on how the user can be involved in the definition of focus and context, for example, through direct pointing, widget



(a) (b) (c)

Fig. 3. Variable compositions and styles. (a) Interactively applying different compositing strategies to reduce the number of context information inside a Magic Lens. (b) Emphasized edges extracted from the video stream not only provide depth cues but also preserve important landmarks, in this example, the needle's entry point. (c) X-ray visualization by preserving real-world information.

manipulation, or tangible interfaces. A classical example is the Magic Lens metaphor [2], which interprets a particular application state and spatial region to infer the F+C roles. The selection of F+C roles is an important topic when confronted with visualization problems that require making hidden information visible. This problem has sometimes been called x-ray visualization. Examples of such approaches in volume rendering include the use of application-dependent importance values [22] or isovalues [15].

Once having discerned focus and context, the second step is to render the objects in visually distinctive styles in order to draw the user's attention to the focus. Most techniques for F+C rendering of 3D scenes rely on opacity modification [15], [22], [8], [4], [1]. Other styling techniques use color [19], depth of field [14], or hybrid rendering techniques [10] to catch the observer's attention. We draw inspiration from these works since our framework allows very general effects to be applied to AR scenes.

## 3 Focus and Context Rendering Framework

The core of our approach for expressive visualizations in AR is a general GPU-based rendering framework similar to [15]. In this section, we will first describe the basic algorithm, followed by an explanation how to use this method in combination with cascaded F+C separation, controlled by an F+C graph.

### 3.1 Rendering Technique

F+C visualizations demand that we visually discriminate objects depending on whether they are marked as focus or context. Recent work on volume visualization such as that of Krüger et al. [15] and Hauser et al. [10] share the idea of a two-pass rendering technique to achieve the desired visualization. The first pass renders the individual objects of the scene into a set of buffers, while the second pass composes the final result from the individual buffers by sweeping over the buffers in depth order. This is a very general and simple approach, which is not just applicable to volume data. Our rendering algorithm renders polygonal models in three steps as follows (implementation details are given in Section 5):

1. *Buffer rendering*. Rather than just making a binary decision between focus and context, we classify the objects in the scene by its context into multiple families, using a scene graph markup mechanism [18]. For every context family, a separate buffer is allocated, which is used to render all object families into distinct buffers. This enables us to apply different treatments such as image manipulation in the next step.
2. *Buffer processing*. Once the buffers have been rendered, we process each of them to visually discriminate focus from context. We apply the desired visual styling during this step, which may include edge enhancement and color or transparency manipulation. The actual modification is highly application dependent and can be scripted through a mechanism similar to shade trees [6].

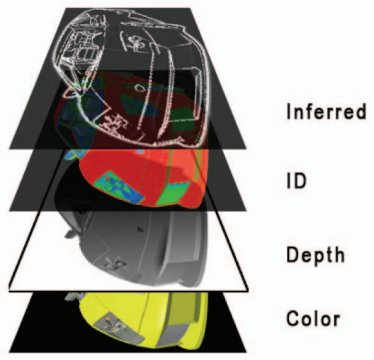



Fig. 4. A single G-Buffer. Conceptual representation of the image buffers contained by a single G-Buffer.

3. *Scene compositing*. This step combines the information contained in the processed buffers. They are composed in front to back order, which are evaluated by sorting depth values associated with every fragment. The compositing is not fixed but may also be scripted, which can vary on a per-pixel basis, enabling different compositing strategies for different areas of the image.

#### 3.1.1 Buffer Rendering

From the algorithm, it is clear that a single frame buffer does not fulfill our requirements. We need not only color+alpha but also depth for the scene compositing and, potentially, several additional buffers resulting from the Buffer Processing. To store this information, Geometric Buffers (G-Buffers) [20] are used. G-Buffers are a collection of image buffers storing color, transparency, depth values, object IDs, texture coordinates, normals, or other per-pixel information. A G-Buffer encodes data about objects, such as view-dependent information, and can be seen as a 2.5D scene approximation.

As many G-Buffer implementations do, we extend the information held in the G-Buffer to include inferred information from the processing step. This inferred information can later be used to change the visual appearance of objects belonging to a particular family. Fig. 4 provides an illustration of a G-Buffer containing four different bitmaps: color, depth, ID, and inferred edge information.

A single G-Buffer contains an approximation of those objects in the scene belonging to a particular context family. We can thereby isolate the styling applied to different families, while the collection of all G-Buffers approximates the whole scene. The objects in Fig. 5 have been bound to different buffers based on their family membership; in this case, the family membership is exemplified by shape.

During buffer rendering, we use the regular rendering pipeline to extract all the necessary information that will be used during the processing of the buffers. The scene is traversed in a single pass with multiple render targets. Every object is rendered to exactly one G-Buffer, which is determined by its family membership given as semantic markup.

#### 3.1.2 Buffer Processing

For every buffer, image processing techniques can be applied to compute additional information, for example,

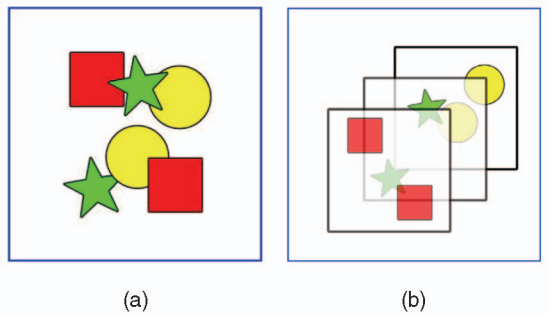



Fig. 5. Multiple G-Buffers. (a) An illustration of a scene. (b) One possible G-Buffer volume. Notice that a G-Buffer does not represent a depth layer.

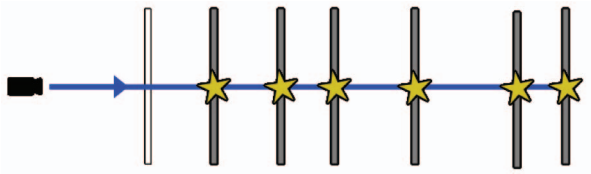


Fig. 6. Scene compositing. Nonuniform raycasting through the G-Buffer volume.

to detect edges or regions with high curvature, to extract regions with particular color or depth values, or to mark a particular region supplied interactively by the user. Some techniques consider more than just the fragment's value, for example, its neighborhood, fragment values of other buffers in the same G-Buffer, or fragment values from different G-Buffers. In this way, multiple additional image components containing auxiliary information can be added to the G-Buffer.

#### 3.1.3 Scene Compositing

In the final compositing step, the information from the set of G-Buffers is merged into a final image using a GPU raycasting algorithm with a nonuniform step length. Notice that simple blending of the G-Buffers is not enough since per-pixel occlusions are essential for the desired effects (as illustrated in Fig. 5). The ray traverses the scene approximation given by the G-Buffers in front to back order (Fig. 6). Since the G-Buffers are already available in view coordinates and in screen resolution, the problem of casting a ray is reduced to sorting the depth components of the G-Buffers.

Once this sorting has taken place, we proceed to compose all the fragments into one single output. Such composition, however, is based on compositing rules that can arbitrarily alter the contribution of a particular pixel from one of the G-Buffers. For example, the color or transparency of a particular pixel may be modified based on the importance of the pixel that was visited along the ray before the current one. Our system allows us to apply different compositing rules on different regions in the image (Fig. 3a). This is a particularly useful operation for complex AR scenes to interactively suppress pixels that are unimportant or confusing.

### 3.2 Cascaded Focus and Context Separation

Conventionally, F+C visualizations consider a single level of discrimination into focus and context. Some applications add weights [7] or importance values [14] to the object classification so that membership to focus or context is no longer binary but continuous. However, 3D AR demands a more complex differentiation to keep information about spatial relationships of objects. Moreover, in classical F+C illustrations such as for textbook figures, no disturbing background is present. While in a real-life AR scenario, we do not have the luxury of limiting the range of objects to the desired focus or context objects.

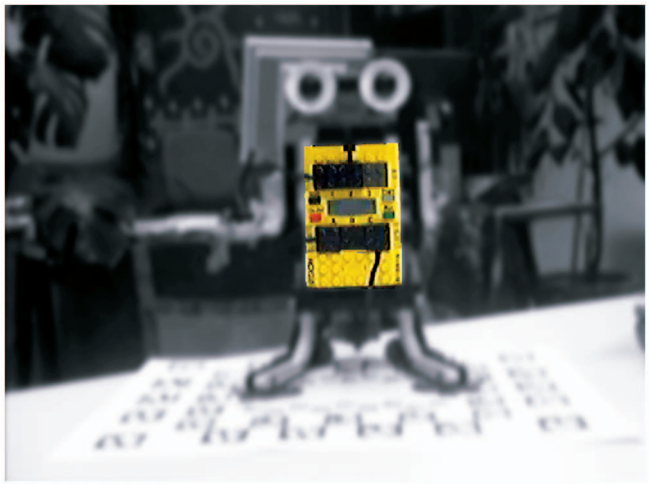


Fig. 7. Single-level F+C discrimination. Object and scene relations are lost. The object of interest seems to "fly" in the real world.

Fig. 7 shows a single level of F+C discrimination in AR similar to [14], used to direct the user's attention to an object of interest. It can be seen that the focus clearly stands out, but the remainder of the object cannot be discriminated from the background.

Rather than relying on the user to directly provide object classification and styling, our approach aims to preserve spatial relationships through heuristic rules. This approach can be described as an implicit multilevel F+C separation, where F+C separators are applied consecutively on the results of previous separation steps.

We control the authoring of complex F+C scenarios by cascading simple F+C separators, using a directed acyclic graph structure, the F+C graph. Such a graph is easy to create and to understand, and it allows the explicit modeling of the control flow and data flow of the rendering process. The graph separates focus and context discrimination in interior nodes, while the leaf nodes execute rendering operations. Graph nodes use G-Buffer content or image masks, created on the fly, for direct rendering or to control subsequent image processing.

Fig. 9 gives an example of an F+C graph, which was used to render the image in Fig. 8. In the given example, the scene is grouped into three families. One is presenting the robot's battery pack (the actual focus), one provides the information about the robot's extremities and its head (the focus' proximity), and one family contains the video background.

To outline the relationship between the focus and the objects in its close proximity, we first visually underline their affiliation to the same group, the foreground objects. Thus, we first divide all scene content into two groups, background and foreground objects, followed by adding a

Fig. 8. Multilevel F+C visualization. This conveys the spatial relationships between the focus and objects in its close proximity.

strong visual discriminator between them, a 2D halo in image space, similar to [5]. The halo is derived from all objects that belong to the same group as the focus object (all parts of the robot).

To be able to direct the user's attention to the actual object of interest, a second separation splits the focus from its proximity. If we now heedlessly reduce the conspicuity of the focus' surroundings, we may easily produce an incomprehensible presentation that renders them useless as contextual information of the focus. We therefore apply another F+C separator (an edge detector) to identify distinct features inside the focus' surroundings, which we emphasize instead. To moreover reduce the conspicuity with increasing distance to the focus, we simply connect a previously derived distance transform to the node, which shades the data, which was classified as less important in the surroundings of the focus object.

## 4 X-Ray Visualization

The F+C graph is a powerful tool, but it is independent of the current viewing direction and therefore does not take occlusions into account. However, a major goal of AR is displaying x-ray visualization to reveal hidden structures. To accomplish such augmentations, we face two main challenges:

1. The final image needs to have enough depth cues available to correctly communicate spatial relationships between hidden and occluding structures.
2. Significant information of occluding structures has to be preserved in order to retain the occluder's shape and to keep important landmarks visible in the real-world imagery (Figs. 2 and 3b).

Notice that partially occluding structures provide a strong monocular depth cue, which means that these two challenges are interdependent and can therefore be jointly addressed. In this section, we show how our framework can be used to create comprehensible augmentations of hidden structures. We will furthermore show how the same

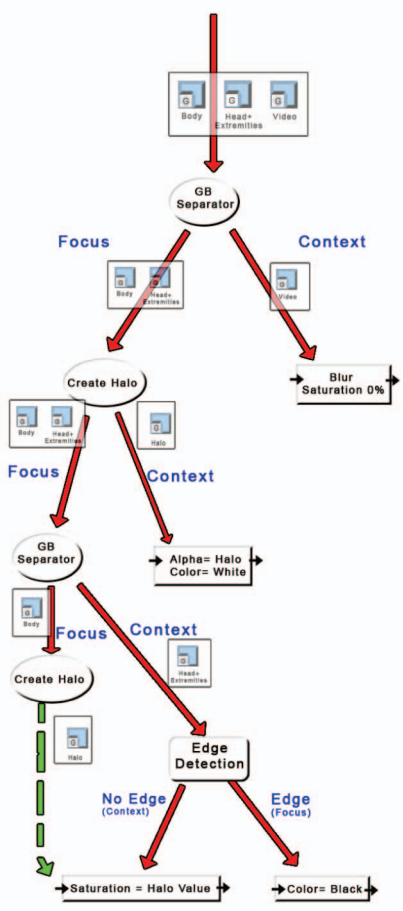



Fig. 9. F+C graph. The F+C graph used to create the visualization in Fig. 8.

techniques can also be used to keep important landmarks visible in the real-world imagery.

In case of x-ray visualization, hidden structures are the focus of attention, while occluding objects represent the context. Fig. 10a shows a naïve augmentation of hidden objects (the engine and the back wheels). Since this image contains only monocular depth cues [3], it is impossible to correctly perceive spatial arrangements of the hidden and occluding objects.

Conventional object occlusions provide better depth cues but may render focus objects completely hidden, thus defeating the goal of x-ray visualization. As a remedy, we render only the important structures of the context objects on top of the focus objects (Fig. 10b). In our case, the important structures are those that hint at the shape and the location of the context relative to its focus. By using only the important contextual information, care is taken that only a limited amount of context is used to occlude the focus. This approach combines the visibility of hidden structures with a sufficient amount of monocular depth cues to correctly interpret spatial object relations.

In order to identify important context information, we need filter operations to separate focus from context, followed by an identification of the important information in the previously identified context area. Notice once more that the problem of finding important information in a context area is again an F+C separation problem.

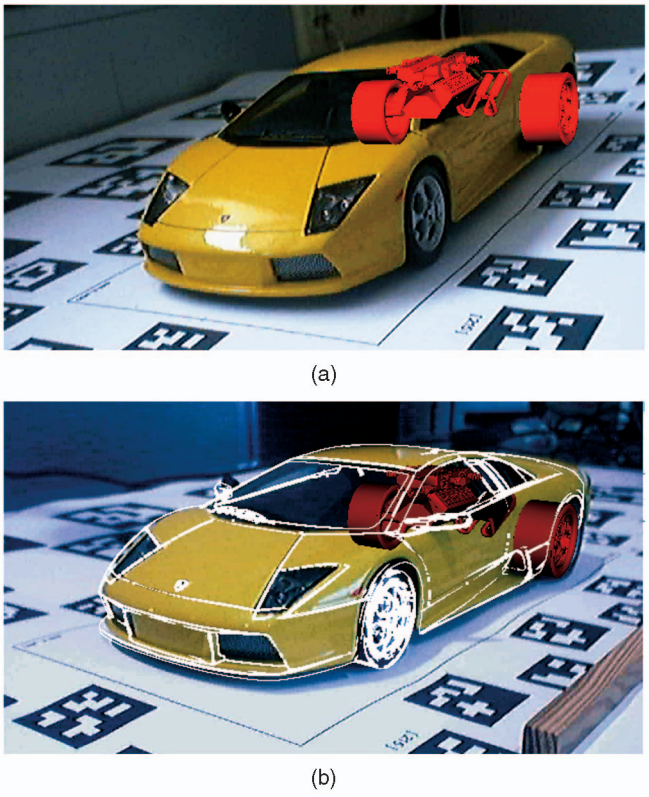


Fig. 10. Comparison of methods. (a) A naïve overlay of hidden information. Correct shading and registration of the engine and the wheels of the car does not provide sufficient depth cues. Occluded objects seem to be in front of the car rather than inside. (b) Enhancement of only important context information helps in communicating spatial relationships while the object of interest is still clearly visible.

Fig. 10b shows the augmentation of important context information of a single occluding object, the car's chassis. Unfortunately, real-world scenarios generally do not consist of just a single occluder; instead, we usually face a complex depth arrangement. Directly applying the aforementioned strategy of global F+C separation can easily lead to excessive context information and a cluttered image (Fig. 11a).

Our framework is able to not only extract key features from context but also control the amount of information visible in the final image. In this case, we are interested in filtering the amount of distracting information either in front of the focus or in uninteresting areas of the imagery. Such filter operations constitute one of the most important components in our F+C rendering pipeline. We therefore allow filtering in all three stages of our rendering pipeline.

While filtering during G-Buffer rendering and during scene compositing applies to all fragments per pixel, filtering during G-Buffer processing by using the F+C graph applies to groups of fragments depending on the G-Buffer affiliation, as well as on previously applied filter operations. In the next section, we discuss how filtering per pixel can be achieved, followed by a description of interactive filtering using Magic Lenses.

### 4.1 Controlling Complexity with Per-Pixel Filter

Depth complexity may increase if multiple objects, located at different depths in the scene, overlap in screen space. To control this problem, we filter fragments covering the same location on the screen, with the goal of a controlled reduction of the number of contributing fragments. This kind of filtering happens either during G-Buffer rendering or during scene compositing. We will start with filter operations during G-Buffer rendering.

If more than one fragment falls onto the same pixel, a test such as the depth buffer test is required to choose one remaining fragment. The way we group objects into families defines which fragments are tested against one another. The grouping, in combination with the choice of the G-Buffer's fragment tests, defines the result of per-pixel filtering. Fig. 11a shows a setup of four different G-Buffers, while Fig. 11d uses a different simplified grouping into only two G-Buffers. Filtering by first selecting a set of fragments depending on their G-Buffer affiliation, followed by regular OpenGL fragment tests, is a versatile tool to control the amount of visible information.

The amount of augmented information in the final image strongly depends on the number of G-Buffers—fewer G-Buffers are usually faster to process and create less clutter. However, in order to define appropriate filtering strategies to sensibly reduce the number of G-Buffers, we need detailed knowledge of the scene and its object structure, which may not always be available, in particular in dynamically changing scenes. Therefore, we have implemented another technique, filtering during scene compositing, which reduces the amount of visible information per pixel regardless of the number of G-Buffers. Any filter strategy that can be formulated using pixel values and sorted depth values can be applied to filtering during scene compositing.

We developed some simple heuristics that can successfully limit clutter in typical situations. Figs. 3a and 11c show one such strategy, where we use inside a Magic Lens only the first fragments (after sorting the depth values in front to back order) and those belonging to the focus objects (called first hit + focus strategy). While four different G-Buffers are used to visually discriminate the scene elements, such heuristic successfully reduces the amount of visible fragments inside the lens. Notice that a disadvantage of filtering during scene compositing is that it is generally more expensive since fragments discarded in this stage have already gone through the entire pipeline before they are finally eliminated.

### 4.2 Interactive G-Buffer Filtering

We control which parts of which G-Buffers get affected by a certain set of operations with our F+C graph structure during G-Buffer Processing, the second stage of our rendering pipeline. Recall that F+C separation is approached through a hierarchy of filter operations, similar to a shade tree (Section 3.2). Nodes in the graph represent filter or shade operations on G-Buffers. The data flow defines on which part a subsequent filter or shader is applied to. To be able to spatially control the operations, we integrated flat Magic Lenses into our F+C graph.

Magic Lenses implement filter operations that can be interactively applied on a user-controlled spatial area. They were first introduced as a user interface tool in 2D [2] and later extended to flat (2.5D) and 3D [21]. We use the Magic Lenses as interactive spatial filters to control the appearance of only distinctive G-Buffers. Fig. 11b shows how only the

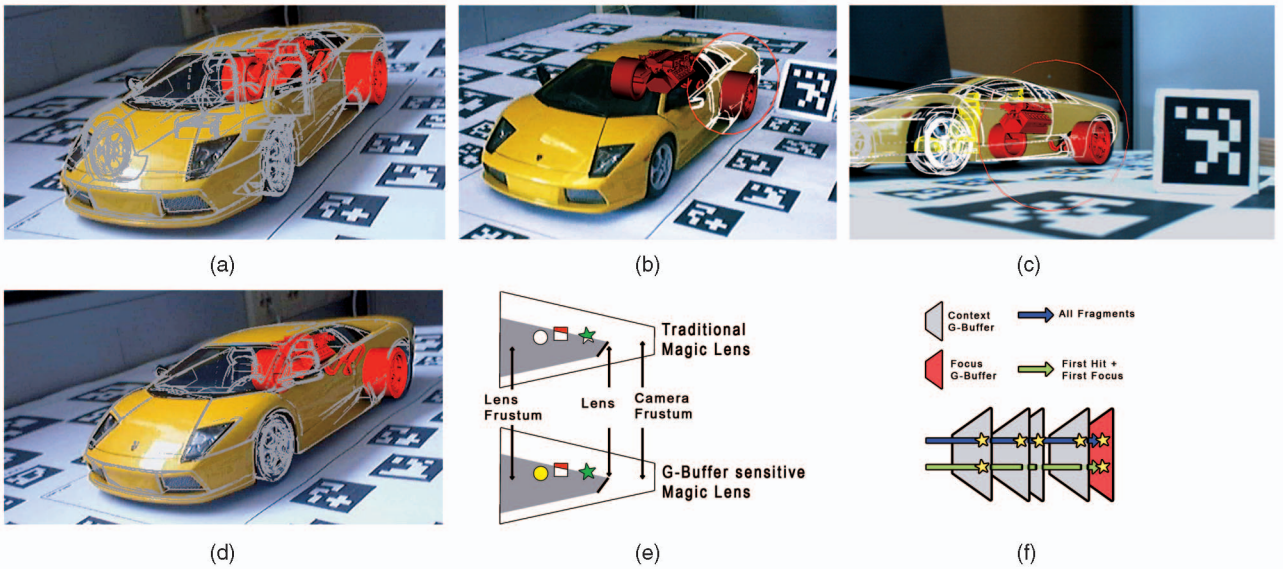



Fig. 11. Context filtering. (a) The cluttered display is caused by too many different G-Buffers. (b) Interactively reducing visible fragments by using our G-Buffer-sensitive Magic Lens. (c) Interactively applying different composting strategies to different regions in the same image. Inside the Magic Lens, we use a "First Hit + Focus" strategy to compose the fragments. (d) A grouping into fewer G-Buffer filters fragments by using regular OpenGL fragment tests. (e) Conceptual comparison of traditional and G-Buffer-sensitive Magic Lenses. Notice that the lens above affects the rendering of all objects in its lens' frustum, while our approach restricts styling to only certain G-Buffers. The inner color is removed only for those objects that are affected by the Magic Lens. (f) Conceptual comparison between two strategies during raycasting. The top ray uses all fragments, while the bottom ray only uses the first and the focus' fragments during compositing.

chassis of the car gets affected by our G-Buffer-sensitive Magic Lens, while the other objects remain unchanged. The difference between conventional Magic Lenses and our implementation is illustrated in Fig. 11e. The top image shows a traditional Magic Lens, affecting everything in its frustum, regardless of whether the effect is desired for a particular object. The bottom shows our implementation and how it affects only a certain set of G-Buffers (only the red square is stylized by the lens). Such a G-Buffer-sensitive Magic Lens is implemented as a mask in the F+C graph, by simply rendering it into a G-Buffer, which is later used in the graph traversal to distinguish focus (inside its footprint) from context (outside its footprint) information of other G-Buffers.

### 4.3 Stylization

If we reduce the amount of context to prevent clutter, we must be careful to choose the most valuable fragments to preserve. Thereafter, we have investigated a number of feature extraction techniques, mostly based on edge detection, which leads to a sparse 3D object representation. However, such sparse representations can exhibit depth ambiguities, which have been addressed in nonphotorealistic rendering. Some of these techniques are also used in our framework.

For example, edge halos provide the user with occlusion cues among edges (Fig. 12b). While edges at the front appear as straight lines, edges at the back become less dominant near foreground edges. Unfortunately, image-based edge detectors may compute many disconnected lines. By adding even more discontinuity with an edge halo, the impression of image clutter is worsened in such cases. We therefore have implemented faded edge halos, which avoid hard discontinuities, while still hinting at the arrangements between foreground and background edges (Fig. 12c).

Another technique to communicate spatial arrangements of sparse representations is edge stippling, which creates discontinuities directly on an edge, rather than on intersections among edges. Stippling is a widely known convention in illustration that conveys the spatial relationship among objects. Edges from back or inner faces are indicated by dotted edges, while front facing or occluding edges are continuous. Given a representation with reasonable straight lines, stippling provides additional depth cues without consuming additional image space. However, the use of stylized edges requires a limited amount of image clutter, as well as a limited amount of depth levels, to be effective.

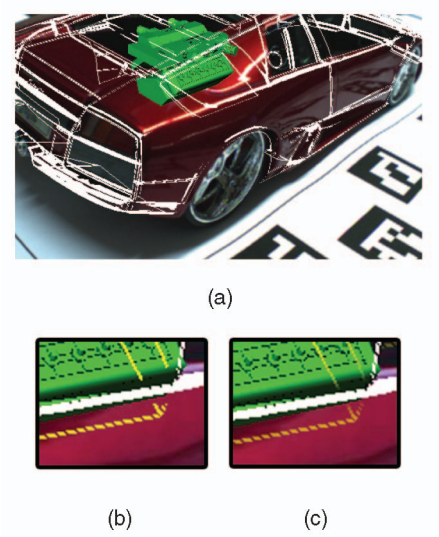



Fig. 12. Depth ambiguity of edges. (a) Sparse representations of foreground and background objects may lead to disambiguation in image space. (b) Edge halos have been used to prevent background edges from direct intersections with foreground edges. (c) Faded edge halos smoothly prevent intersection, which helps to reduce image clutter.

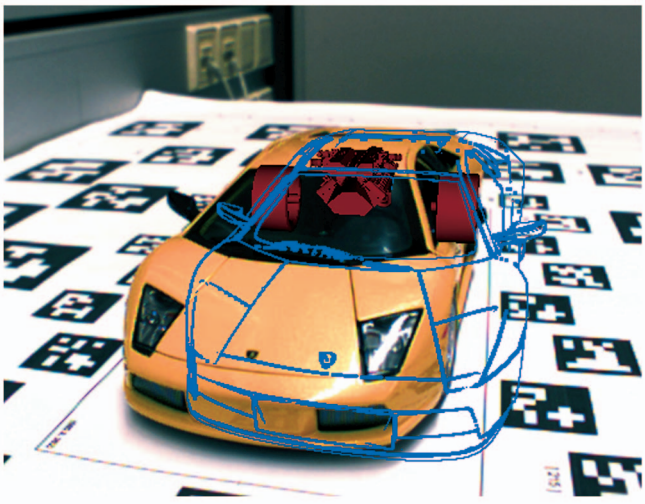


Fig. 13. Synthetic edges. An augmentation of edges extracted from a 3D model. The edges are thin and clear, but they suffer from poor registration.

### 4.4 Virtual versus Real Context Preserving

The rendering algorithm described in this paper is only dependent on the fact that all input data is available in a G-Buffer, while the sources of such data can vary. The most obvious source, which has been mentioned before, is the rendering of scene objects. Anther important source for AR scenes is the video stream used for video see-through augmentation. A video stream delivered into a G-Buffer can be subjected to all image-based operations described in this paper. The choice of which data source to use depends on the desired visualization and the conditions of the augmentation. For example, in the case of edge detection, better results may be obtained from G-Buffers generated from rendered objects. This is because rendered objects are not affected by image noise and are therefore easier processed. However, this implies that the resulting edges are also subject to tracking errors and poor registration. Then again, images from the video stream do not suffer from the registration problem, but they are subject to noise, which can cause artifacts in the computed visualization.

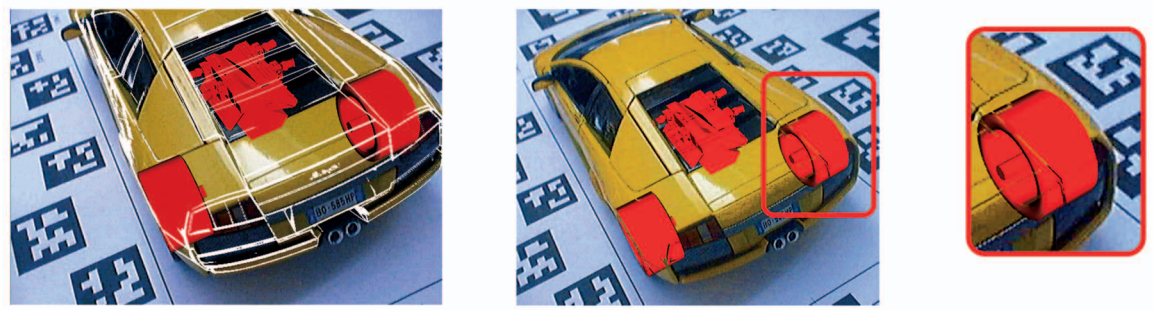
Fig. 13 shows an enhancement of edges of a modeled car. The edges are clearly distinguishable in thin lines. However, since they are model based, they are subject to tracking errors. Notice how the registration of the edges and the real model car is offset through intentionally sloppy registration. This can lead to confusing rather than helpful depth cueing.

This problem may be overcome if we rely on edge detection of the video stream. Fig. 3b shows an example of such augmentation. Notice that the edges are perfectly registered but rather thick and less detailed. The quality of the edges depends not only on the edge extraction technique itself but also on the input video image. Edges extracted from a rendered model will most of the time have superior quality than those extracted from video. Even more concerning is that edges extracted from video may clutter the whole view.

A hybrid approach using features from video and tracked objects as stencil masks is able to reduce the image clutter. Fig. 3b has been enhanced with edges from the video stream, which were stenciled by the region covered by a flat magic lens. This technique enables us to use the registration quality of edges from video without cluttering over the whole screen. While the hybrid approach inherits the disadvantages of its components (quality of the edges and occasional registration error of the mask), these are much less disturbing even in harsh conditions.

Adding extra artificial information to the image may nevertheless be distracting. A more subtle way to depict context is by using the original video information as context overlay. However, the limits of human perception impose a trade-off between the amount of preserved features and the clarity of revealed hidden structures. Therefore, if the visualization demands a high clarity of hidden structures, a very sparse representation has to be used for occluding objects. However, such very sparse representations necessarily provide only a small amount of features, making it difficult to mentally reconstruct the object's shape, especially if its appearance is similar to its surroundings, which is the case in most of the nonartificial scenarios. Fig. 14b shows such a very sparse preservation of real-world imagery. Notice the difficulty to perceive the preservations of the real-world imagery.

The visualization should aim to contribute features of optimal clarity. In case of very sparse preservations, such features are more comprehensive if they have a consistent tone or color and stand in strong contrast to their surroundings. Unfortunately, video-based AR does not guarantee that such features exist in the original video



(a) (b)

Fig. 14. Virtual emphasizing versus real-world preservation for sparse object representations. (a) A uniformly colored accentuation of sparse object preservation is able to provide a good impression of the occluding object. (b) In contrast, sparse real-world preservations may be hard to perceive.

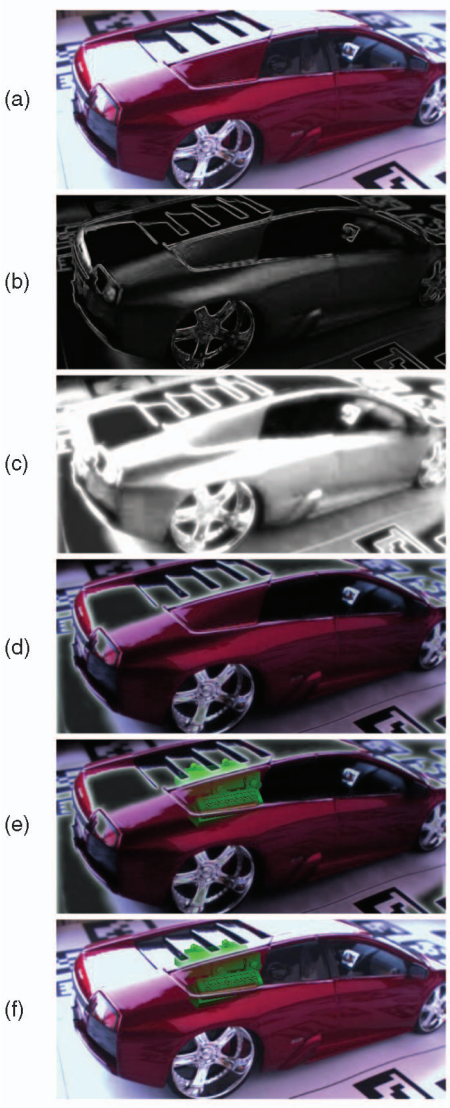



Fig. 15. Soft context preserving. Given the (a) input video, we (b) compute its features and (c) apply a haloing operation. We use this as (d) a mask for video fragments, which is finally composed with (e) virtual objects and (f) the original video.

image, as they are subject to noise, varying illumination, etc. We therefore emphasize very sparse representations, by using an artificial coloring (Fig. 14a). However, if the visualization's goal demands video preserving instead of artificial coloring, a certain amount of preserved features has to be guaranteed.

Fig. 15 shows an example of how we achieve this in image space. The example uses the features extracted from the video feed. To ensure a certain amount of context preservation, we distribute the detected features in its close proximity, by computing a 2D halo around them. The halo operator additionally weights the distributed features depending on their distance to the original features (Figs. 15a, 15b, and 15c). We use the derived halo as a mask to retrieve those fragments from the video feed, which are used for context augmentation (Fig. 15d). The halo values are finally used to gradually modify the opacity of the originally detected, as well as the distributed, features, which ensures a smooth composition of virtual and real objects (Figs. 15e and 15f).

Fig. 16. Smooth composition of real and virtual objects.

Figs. 3c and 16 show other examples of soft opacity modifications along detected features. The usage of smooth opacity changes guarantees that hidden objects will not be completely covered by the video information, even though a high amount of information is preserved.

### 4.5 Focus on Focus Visualization

Our rendering framework is based on a hierarchical approach of F+C separation. This causes a natural problem: some situations might have more than one focus object. In some scenarios, the relative importance between objects from different branches of the hierarchy cannot be clearly resolved. If such objects overlap in screen space, a strategy to select the contributing fragments has to be defined.

A naïve approach is to simply blend between such objects to keep both foreground and background object visible. However, blending fails to convey the spatial arrangement and is therefore mostly inappropriate for our purposes. A better solution is to use a sparse representation of the hidden focus object but only where it is occluded. Nonoccluded fragments of the focus object are treated using a dense representation (Fig. 17). We generate such combinations of sparse and dense visualizations of one object by simply using the partially occluding object as a runtime mask in our F+C graph.

## 5 Implementation

To visually discriminate focus from context, we need to render both using different styles. Assuming a scene graph,

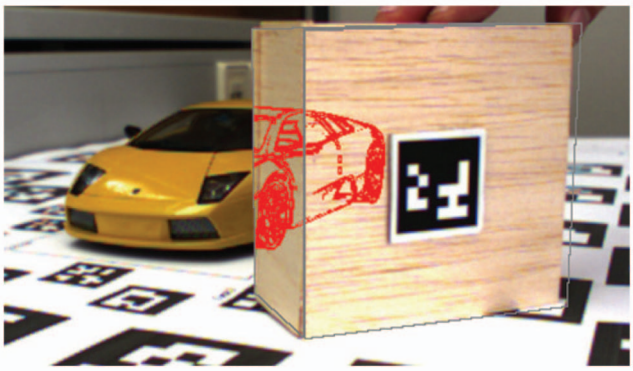


Fig. 17. Two objects with equal importance assignment. We render the partially hidden object by using a sparse representation for occluded fragments and a dense representation for nonoccluded fragments.

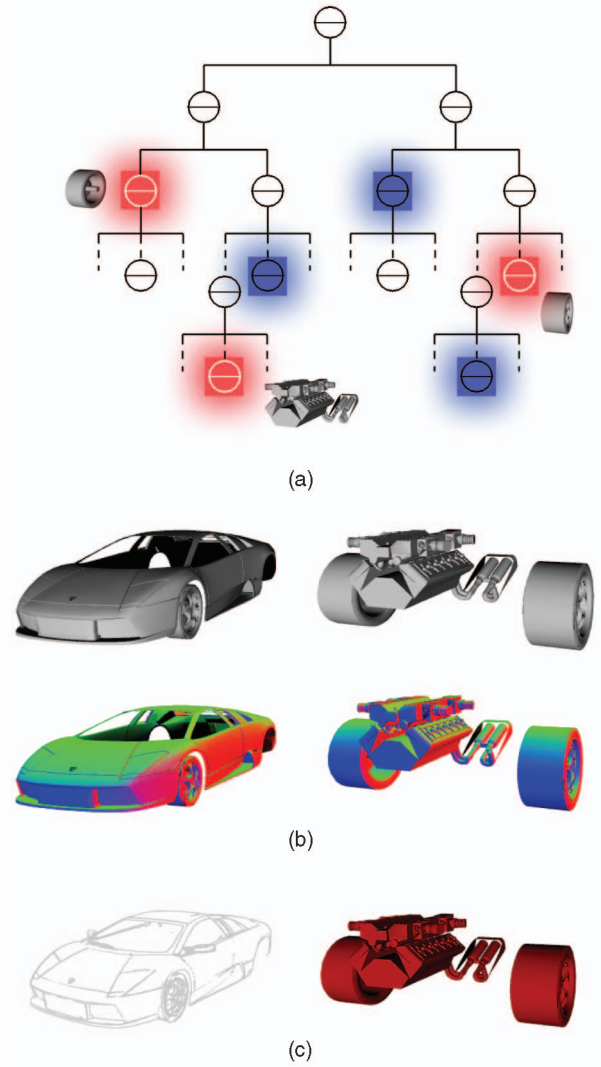



Fig. 18. Data handling before compositing. (a) Conceptual illustration of a context-sensitive scene graph. Context families of objects with the same contextual information (in blue and red) can be jointly referenced regardless of their position in the scene graph. (b) Result of the first step of our algorithm (G-Buffer rendering). (c) A computed sparse representation and the shaded focus objects. This is the result of the second stage of our rendering pipeline. The final composition is shown in Fig. 10b.

as it is commonly used in VR and AR applications, a simple approach places focus objects and context objects in separate branches of the scene graph. This, however, would limit the possible data sources to those with a specific separation of these two elements. Moreover, it would make interactive changes of F+C separation possible only if the application is tightly coupled with the scene graph data structure.

Instead of depending on a hierarchy that fulfills our requirements, objects are marked with contextual information and scattered throughout the scene graph in any naturally occurring order without any enforced grouping. Sorting objects by context family happens implicitly during the scene graph traversal, using the parameterized scene graph described in [18]. Fig. 18 shows such a conceptual grouping of objects in the scene graph (highlighted in red and blue), regardless of their occurrence in the graph. In our implementation, objects are marked up with the context family they belong to, and this property is inherited by the subgraph. The family in turn determines which G-Buffer to target in the subsequent rendering. The implementation of this marked-up scene graph is based on Coin3D (www.coin3d.org), but the mechanism itself is independent of any specific platform and is easy to duplicate on other systems.

The scene graph is also used as the software infrastructure for the F+C graph, which directly makes use of important scene graph capabilities: In-order traversal is used to visit the nodes of the F+C graph, while field connections enable out-of-order data flow dependencies. All elements of our visualization framework are implemented as node extensions and can be mixed freely with any kind of content available in Coin3D. This makes the system easily extensible, and furthermore, the file format of Coin3D allows convenient user scripting of all content, facilitating the development of authoring tools for end users.

Our implementation is based on the GPU programming language GLSL, the OpenGL Frame Buffer Object (FBO) extension, and multiple render targets. We have implemented a G-Buffer as a collection of 2D textures. Each of the texture's components is used to represent a specific value such as a color component or depth value. For example, a G-Buffer with RGBA color, depth, and object-ID information needs to have six scalar components. While a single 2D texture can only hold up to four components, we use up to eight textures for a maximum of 32 components, addressed simultaneously in a single rendering pass, resulting from a single scene graph traversal. While we have found this to be sufficient for our experimental implementation, the use of render target arrays can reduce this limitation even further.

We use a simple texture tiling technique, where each tile represents a G-Buffer. Switching G-Buffers from a different context family merely means looking up corresponding viewport parameters of a single large target texture. We use a Nvidia GeForce 8800 GPU graphics card, on which our examples typically exceed 30 fps without significant optimization. Notice that the images in this paper are bound to the quality achieved by the camera. In this case, we use a uEye camera with a 1/2'' CCD and an exchangeable Fujinon lens.

Edge halos used to discriminate sparse foreground and background representations can be efficiently computed from edges as Gaussian blur. However, this approach tends to remove contributions from small features. Bruckner and Gröller [5] describe a GPU implementation of halos, which preserves the contribution of small isolated features. We have added both techniques to our system to optimally handle video and rendered objects.

Another problem with halos is that their depth value is ill-defined. The three options that we have conceived are the following: 1) assign the depth value of the video feed (usually the value of the far plane), 2) assign the depth value of the near plane, and 3) assign the depth value of the closest contributing fragment. The choice of which strategy to use depends on the application.

## 6 Conclusion

F+C techniques are highly successful in scientific visualization and have the potential to become an essential tool for AR. They enable applications to draw the attention of users to objects in the focus while still perceiving contextual information. We make use of these ideas to correctly communicate spatial arrangements of hidden structures.

We have presented a framework for computing high-quality AR visualizations using multilevel F+C visualization. The input to the system is a standard scene graph containing the relevant objects, marked up with classification information, together with a live video stream. From a set of style descriptions given as an F+C graph, the system can automatically compute view-dependent augmented images. All rendering runs on the GPU at real-time frame rates and can directly benefit from enhancements in graphics hardware. Since styles are defined independent of content, existing applications can be visually improved at a later point in time.

The core algorithm of the framework is a two-pass technique consisting of rendering and compositing of G-Buffers. This approach only requires the capability to sort objects into families of similar context. It therefore fits into a standard rendering pipeline and can be easily integrated into almost any system.

Many techniques for stylized rendering can potentially be integrated into such a framework, leading to richer expressions. MacIntyre et al. [16] argue that every new medium needs time to establish a visual language and a set of conventions. Bichlmeier et al. [1] have tested the effectiveness of context information preservation on AR scenarios with very encouraging results. We believe that expressive visualization will be a key ingredient of such a language. A main goal for future work is therefore not just to implement new techniques for styling but also to establish visual conventions and possibly infer appropriate visualization styles automatically from metainformation of the rendered objects.

## ACKNOWLEDGMENTS

This work was sponsored in part by the Austrian Science Fund FWF under Contract Y193, the Austrian Forschungs-Förderunggesellschaft FFG under Contract BRIDGE 811000, and the European Union under Contract MRTN-CT-2004-512400. The authors would like to thank Meister Eduard Gröller and Stefan Bruckner for the useful discussions and suggestions.

## REFERENCES

[1] C. Bichlmeier, F. Wimmer, H.S. Michael, and N. Nassir, "Contextual Anatomic Mimesis: Hybrid In-Situ Visualization Method for Improving Multi-Sensory Depth Perception in Medical Augmented Reality," *Proc. Sixth IEEE and ACM Int'l Symp. Mixed and Augmented Reality (ISMAR '07),* pp. 129-138, Nov. 2007.

[2] E.A. Bier, M.C. Stone, K. Pier, W. Buxton, and T. DeRose, "Toolglass and Magic Lenses: The See-Through Interface," *ACM Trans. Graphics (Proc. ACM SIGGRAPH '93),* pp. 73-80, 1993.

[3] G. Bruce, *Sensation and Perception,* seventh ed. Wadsworth, 2006.

[4] S. Bruckner, S. Grimm, A. Kanitsar, and M.E. Gröller, "Illustrative Context-Preserving Volume Rendering," *Proc. Joint Eurographics-IEEE VGTC Symp. Visualization (EuroVis '05),* pp. 69-76, May 2005.

[5] S. Bruckner and M.E. Gröller, "Enhancing Depth-Perception with Flexible Volumetric Halos," *IEEE Trans. Visualization and Computer Graphics,* vol. 13, no. 6, 2007.

[6] R.L. Cook, "Shade Trees," *ACM Trans. Graphics (Proc. ACM SIGGRAPH '84),* vol. 18, no. 3, pp. 223-231, 1984.

[7] H. Doleisch and H. Hauser, "Smooth Brushing for Focus+Context Visualization of Simulation Data in 3D," *Proc. 10th Int'l Conf. Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG '02),* pp. 147-154, 2002.

[8] S.K. Feiner and D.D. Seligmann, "Cutaways and Ghosting: Satisfying Visibility Constraints in Dynamic 3D Illustrations," *Visual Computer,* vol. 8, pp. 292-302, 1992.

[9] C. Furmanski, R. Azuma, and M. Daily, "Augmented-Reality Visualizations Guided by Cognition: Perceptual Heuristics for Combining Visible and Obscured Information," *Proc. First Int'l Symp. Mixed and Augmented Reality (ISMAR '02),* p. 215, 2002.

[10] H. Hauser, L. Mroz, G.I. Bischi, and M.E. Gröller, "Two-Level Volume Rendering," *IEEE Trans. Visualization and Computer Graphics,* vol. 7, no. 3, pp. 242-252, July-Sept. 2001.

[11] S. Julier, Y. Baillot, D. Brown, and M. Lanzagorta, "Information Filtering for Mobile Augmented Reality," *IEEE Computer Graphics and Applications,* vol. 22, no. 5, pp. 12-15, 2002.

[12] D. Kalkofen, E. Mendez, and D. Schmalstieg, "Interactive Focus and Context Visualization for Augmented Reality," *Proc. Sixth IEEE and ACM Int'l Symp. Mixed and Augmented Reality (ISMAR '07),* pp. 191-200, Nov. 2007.

[13] R. Kosara, H. Hauser, and D. Gresh, *An Interaction View on Information Visualization.* Prentice Hall, 2003.

[14] R. Kosara, S. Miksch, and H. Hauser, *Focus and Context Taken Literally,* 2002.

[15] J. Krüger, J. Schneider, and R. Westermann, "Clearview: An Interactive Context Preserving Hotspot Visualization Technique," *IEEE Trans. Visualization and Computer Graphics,* vol. 12, no. 5, pp. 941-948, Sept./Oct. 2006.

[16] B. MacIntyre, J.D. Bolter, E. Moreno, and B. Hannigan, "Augmented Reality as a New Media Experience," *Proc. IEEE and ACM Int'l Symp. Augmented Reality (ISAR '01),* pp. 197-206, 2001.

[17] E. Méndez, D. Kalkofen, and D. Schmalstieg, "Interactive Context-Driven Visualization Tools for Augmented Reality," *Proc. Fifth IEEE and ACM Int'l Symp. Mixed and Augmented Reality (ISMAR '06),* pp. 191-200, 2006.

[18] G. Reitmayr and D. Schmalstieg, "Flexible Parameterization of Scene Graphs," *Proc. IEEE Conf. Virtual Reality (VR '05),* pp. 51-58, 2005.

[19] T. Ropinski, F. Steinicke, and K.H. Hinrichs, "Interactive Importance-Driven Visualization Techniques for Medical Volume Data," *Proc. 10th Int'l Fall Workshop Vision, Modeling, and Visualization (VMV '05),* pp. 273-280, 2005.

[20] T. Saito and T. Takahashi, "Comprehensible Rendering of 3-D Shapes," *ACM Trans. Graphics (Proc. ACM SIGGRAPH '90),* vol. 24, no. 4, pp. 197-206, 1990.

[21] J. Viega, M.J. Conway, G. Williams, and R. Pausch, "3D Magic Lenses," *Proc. Ninth Ann. ACM Symp. User Interface Software and Technology (UIST '96),* pp. 51-58, 1996.

[22] I. Viola, A. Kanitsar, and M.E. Gröller, "Importance-Driven Volume Rendering," to appear in *Proc. IEEE Conf. Visualization (VIS '04),* pp. 139-145, 2004.

**Denis Kalkofen** received the Dipl Ing degree in computational visualistics from the University of Magdeburg in 2004. He is currently a research assistant and a PhD candidate in the Institute of Computer Graphics and Vision, Graz University of Technology, Austria. His current research interests include visualization for augmented reality, illustrative visualization and medical visualization. He is a student member of the IEEE.

**Erick Mendez** received the BS degree from the Benemerita Universidad Autonoma de Puebla and the MS degree from George Washington University. He is currently a research assistant and a PhD candidate in the Institute of Computer Graphics and Vision, Graz University of Technology, Austria. His current research interests include context information, as well as visualization techniques for augmented reality systems. He is a student member of the IEEE.

**Dieter Schmalstieg** received the Dipl Ing, Dr techn, and Habilitation degrees from the Vienna University of Technology in 1993, 1997, and 2001, respectively. He is currently a full professor of virtual reality and computer graphics in the Institute for Computer Graphics and Vision, Graz University of Technology, Austria, where he directs the "Studierstube" research project. His current research interests include augmented reality, virtual reality, distributed graphics, 3D user interfaces, and ubiquitous computing. He is an editorial advisory board member of *Computers & Graphics*, a member of IEEE ISMAR's steering committee, the chair of the EUROGRAPHICS Working Group on Virtual Environments, and an advisor of the K-Plus Competence Center for Virtual Reality and Visualization in Vienna. He is a member of the IEEE.