Interactive Focus and Context Visualization for Augmented Reality

Denis Kalkofen*
Graz University of Technology
Institute for Computer Graphics and Vision, Inffeldgasse 16a
Graz, 8010, Austria

Erick Mendez†
Graz University of Technology
Institute for Computer Graphics and Vision, Inffeldgasse 16a
Graz, 8010, Austria

Dieter Schmalstieg‡
Graz University of Technology
Institute for Computer Graphics and Vision, Inffeldgasse 16a
Graz, 8010, Austria

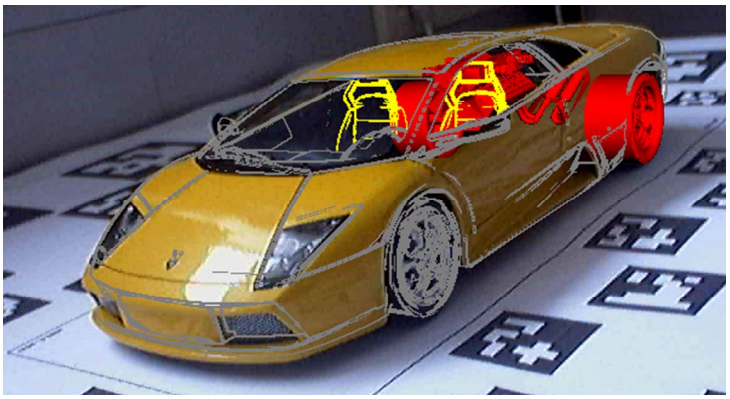


Figure 1 An example of an enhanced augmentation. Focus objects (in red) are not only overlaid on top of the video image, but they are partially occluded by key features from context objects. This provides object occlusion with key features of occluding objects. A second level context (yellow seats) further helps an understanding of the scene. Edges in this image are enhanced considering occlusions with other objects. This helps us to better control the depth complexity of the scene

ABSTRACT

In this article we present interactive Focus and Context (F+C) visualizations for Augmented Reality (AR) applications. We demonstrate how F+C visualizations are used to affect the user's perception of hidden objects by presenting contextual information in the area of augmentation. We carefully overlay synthetic data on top of the real world imagery by taking into account the information that is about to be occluded. Furthermore, we present operations to control the amount of augmented information. Additionally, we developed an interaction tool, based on the Magic Lens technique, which allows for interactive separation of focus from context. We integrated our work into a rendering framework developed on top of the Studierstube Augmented Reality system. We finally show examples to demonstrate how our work benefits AR.

Categories

H.5.1 [Multimedia Information Systems]: Artificial, augmented, and virtual realities; H.5.2 [User Interfaces]: Style guides; E.1.3 [Data Structures]: Graphs and Networks; I.3.6 [Methodology and Techniques]: Interaction techniques, Graphics data structures and data types.

Keywords

Object overlay and spatial layout techniques, real-time rendering, interaction techniques for MR/AR, mediated and diminished reality.

1 INTRODUCTION

Augmented Reality (AR) displays provide extra information to a user's perception by overriding parts of the real world with synthetic, computer generated images. However, heedless replacement of portions of the real world image can easily cause a number of cognitive problems. For example, if too much information is added in areas that were unimportant in the original image, the impression of a cluttered display is caused. Another example of problematic augmentations can be found in X-Ray visualizations, where hidden structures are rendered on top of visible objects. Careless augmentation with synthetic imagery may obscure important real-world information. This problem is illustrated in the simulated surgery depicted in Figure 2, it shows an attempt to insert a needle into a patient's abdomen while visualizing internal anatomy using AR. The computer-generated augmentations occlude extremely relevant information presented in the real world imagery. In this case the user is unable to see the entry marks for the needle placed on the skin of the patient.

Augmenting information on top of real imagery may also lead to problems of depth perception. This 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 [10]. For example [24][3] add a window-shaped restriction of the rendering area to enhance depth perception through partial occlusion, while [9] uses a box-shaped restriction that additionally provides linear perspective cues. However, none of the existing approaches take into account the removed information or try to use cues present in the real image as an aid for depth perception.

* kalkofen@icg.tugraz.at
† mendez@icg.tugraz.at
‡ schmalstieg@icg.tugraz.at

978-1-4244-1750-6/07/$25.00 ©2007 IEEE

In this paper, we address the task of carefully overlaying synthetic data on top of the real world imagery (Figure 1). 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 inspired by Focus and Context (F+C) visualization techniques, a powerful instrument for visually communicating relevant structures. To address the problem of augmentations occluding useful real imagery, we introduce the notion of context preserving X-Ray vision which controls the removal of real world information based on an importance measure of this information. Moreover, we address the problem of cluttered displays by providing solutions for controlling depth complexity with an interactive, user controlled filter which combines spatial and contextual information.

The techniques presented in this paper are embedded in a novel AR visualization framework which operates on an intermediate representation extracted from conventional 3D models. This representation is image based and can be described as a sparse, non-uniformly sampled, view-dependent volumetric model. We use recent GPU programming techniques for efficient rendering of this representation inspired by the G2 buffer framework [7].

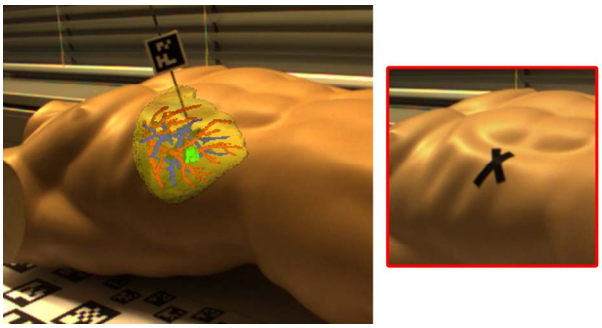


Figure 2 Careless augmentations of hidden structures suffer two key problems: they override useful information (such as landmarks) and they lack depth cues. (Right) Original photo before augmentation (Left) Augmentation of the liver with portal & hepatic vessel trees as well as a tumor (in green).

2 Related Work

Using Focus+Context (F+C) visualizations allows the application to draw the attention of the user to a portion of the data (Focus), while at the same time the overall spatial relationship of the neighboring information (Context) is presented in an abstracted way. Following the overview of F+C techniques in [14], the creation of F+C visualizations can be roughly divided into two major steps: data classification and rendering.

The objective of the first step, data classification, is to identify the F+C roles in the data, i.e. what information should be focus and what should be context. Data classification may be statically given for a particular application, or it may be determined procedurally by some kind of interpretation procedure.

Data separation is usually performed based on user input, so the choice of focus can be controlled by the user. A classical example of this is the Magic Lens metaphor [2]. There are a number of approaches how the user can be involved in the definition of context, for example through direct pointing, widget manipulation or tangible interfaces.

More indirect approaches interpret a particular application state and aim to infer the user's needs and intentions. There are several examples of this approach with applications in AR. The interpretation can be based on task knowledge [13] or contextual information associated with the spatial data [18] [20]. Another application of data separation is found in X-Ray visualization, which is relevant both in AR and medical visualization. Examples of the latter are techniques that use application-dependent importance values [26] [27] [22] or iso values [4] [12] for X-Ray vision in volume rendering.

Once having discerned these two categories, the second step is to render the categories in visually distinctive styles in order to separate and to draw the user's attention to the focus. This may be done by modifying geometry – e.g., visually distorting the data through a magnifying lens, or by modifying appearance, e.g., by manipulating saturation or opacity.

Most of the techniques for applying F+C rendering to 3D scenes rely on opacity modification [1] [17] [8] [6]. Few other works use interesting rendering changes to draw the attention of the user such as color or depth of field discrimination [15] or modifying shading parameters [22]. We draw inspiration from these works since our framework allows very general non-photorealistic effects to be applied to AR scenes.

Interaction usually works hand in hand with visual discrimination. It facilitates a localized region where the F+C effect should take place [2] [22] [18] [20] [1]. However, it should be emphasized that while the focus should always be locally bound, this does not mean that this boundary must be a particular display region. The focus can also be defined non-spatially, for example as a particular category of object the user is interested in. In particular, some projects have successfully applied it to the entirety of the image with very encouraging results [6] [17]. In the work in this paper, we also take a similar approach where the definition of focus and context is globally available and can be used to achieve a variety of effects in combination.

3 Focus+Context Rendering

Focus+Context 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. [17] and Hauser et al. [15], 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 we found suitable for our aims. Our rendering algorithm consists of the following steps:

Buffer Rendering: We classify the objects in the scene into multiple context families using a simple markup mechanism, rather than just making a binary decision of focus or context. For every context family, a separate buffer is allocated. During this step we render all object families into the corresponding buffers. This enables us to apply different treatments, such as image manipulation, in the next step.

```
For every object(i) {
  a) Determine context family F of object (i)
  b) Render object(i) into a Buffer (F)
}
```

Buffer processing: Once the buffers have been rendered, we process each of them in turn to visually discriminate focus from

context. We apply the desired visual styling during this step. This may include, for example, edge enhancement, color or transparency manipulation. The actual modification is highly application dependent and can be scripted through a mechanism similar to shader trees [5].

```
For every buffer(i) {
  Modify buffer(i) according to rule(i)
}
```

Scene Compositing: This step combines the information contained in the processed buffers. The compositing is not fixed, but may also be scripted. The buffers are composed in front to back order, evaluated by sorting the buffers based on the depth buffer values associated with every pixel. This allows us to maximize the control of compositing parameters.

```
For every pixel(x,y) {
  Sort all buffers by depth d(x,y)
  For all buffers(j) in order given by d
    r = compose(fragment(j,x,y),strategy(j),r)
  return r
}
```

3.1 Buffer Rendering

From the algorithm, it is clear that a single frame buffer does not fulfill our requirements. We need 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) [23] 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.

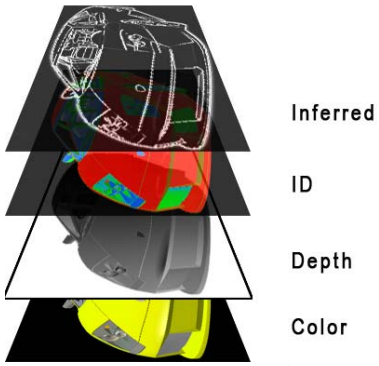


Figure 3 Conceptual representation of the image buffers contained by a single G-Buffer.

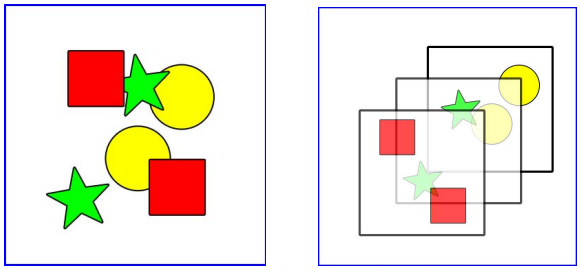


Figure 4 (Left) An illustration of a scene. (Right): One possible G-Buffer volume.

A key concept of our work is to 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. Figure 3 provides an illustration of a G-Buffer containing four different bitmaps: color, depth, normals 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 all G-Buffers combined approximate the whole scene. This is illustrated conceptually in Figure 4. Notice that objects from different families have been bound to different buffers, in this case the family membership is examplified 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, namely all G-Buffers. Every object is rendered to exactly one G-Buffer, which is determined by its family membership given as semantic markup.

3.2 Buffer Processing

For every buffer, a number of image processing techniques can be applied to compute additional information, for example 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. Many such operations will consider a pixel together with its neighborhood, and use information from multiple components of the G-Buffer, such as both depth and normal information. In this way multiple additional image components containing auxiliary information can be added to the G-Buffer.

3.3 Scene Composition

In the final compositing step, the information from the set of G-Buffers is merged into a final image using a non-uniform GPU raycasting algorithm. Simple blending of the G-Buffers is not enough since occlusions are given on a per pixel instead of a per G-Buffer basis. The ray traverses the scene approximation given by the G-Buffers in front to back order. 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 which can arbitrarily alter the contribution of a particular pixel from one of the G-Buffers. For example, 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. A particularly useful operation for complex AR scenes is to suppress pixels that are unimportant or confusing.

4 Context Preserving X-Ray Vision

X-Ray visualization is capable of displaying hidden structures behind real world imagery. In order to accomplish meaningful augmentations, we face two main challenges. First, the final image needs to have enough depth cues available to correctly communicate spatial relationships between hidden and occluding structures. Second, significant information of occluding structures has to be preserved in order to retain the occluder's shape or simply to keep important landmarks visible in the real world imagery (Figure 2). 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 we use F+C visualizations to create comprehensible augmentations of hidden structures. Later, we will show how the same techniques can be used to keep important landmarks visible in the real world imagery.

4.1 Focus and Context Separation

For X-Ray visualization, we consider hidden structures as information in the focus of attention, while occluding objects represent the context. To achieve comprehensible augmentations of hidden structures, we need to preserve important information of occluding objects. We derive from our previous considerations that important context information is the one which preserves the shape of the occluder or keeps important landmarks visible.

In order to identify important context information we need to find filter operations to first separate Focus from Context followed by an identification of important information in the context area. The problem of finding important information in a context area is again an F+C separation problem. However, this separation is applied to the already identified context from the first stage and therefore, the resulting Focus and Context may be seen as Second Level Focus and Second Level Context.

Filter operations build one of the most important components in our F+C rendering pipeline. We allow filtering to be applied in all three stages of our algorithm. While filtering during *G-Buffer Rendering* and during *Scene Compositing* applies to all fragments per pixel, filtering during *G-Buffer Processing* applies to groups of fragments depending on G-Buffer affiliation.

In the next section we discuss filter operations on fragments per G-Buffer. Subsequently, we explain how fragment filtering per pixel is achieved.

4.2 Filtering through G-Buffer Processing

We control which parts of which G-Buffers get affected by a certain shading operation during *G-Buffer Processing*. As already mentioned, our F+C separation problem for X-Ray visualizations builds up a hierarchy, where filters are applied on results of previous filter operations. Therefore, we have built our G-Buffer Processing architecture on top of a data flow graph, similar to a shader tree [5]. Nodes in the graph represent filter or shading operations on G-Buffers. Links are used to define on which part of the previous filtered data a subsequent filter or shader is applied to. Since all of our filter nodes implement a binary segmentation into focus and context elements, links select only between these two types.

Figure 5 shows a simple graph and its corresponding visualization is shown in Figure 6. This graph renders the main focus object in red (back wheels & engine) while the second level focus is presented in yellow. Furthermore, it segments the edges out of the second level context area and shades the extracted edges in grey while it turns the remaining fragments to fully transparent.

Two types of filter operation have been implemented to achieve the desired shadings. First, filter which separate G-Buffers. Second, filter which differentiate between fragments of the same G-Buffer by using image processing algorithms or simple 2D masks, which can even be derived from other G-Buffer's footprints. For example, one possible way to filter important information out of a specific G-Buffer is done by applying an edge detector on the color buffer of the G-Buffer itself.

4.3 Controlling Depth Complexity By using per pixel F+C filtering

Making hidden structures visible may cause information overflow. In particular, depth complexity may increase if multiple objects that are located at different depths in the scene overlap in screen space. To control this problem of depth complexity we filter fragments covering the same location on the screen, with the goal of a controlled reduction of the number of contributing fragments.

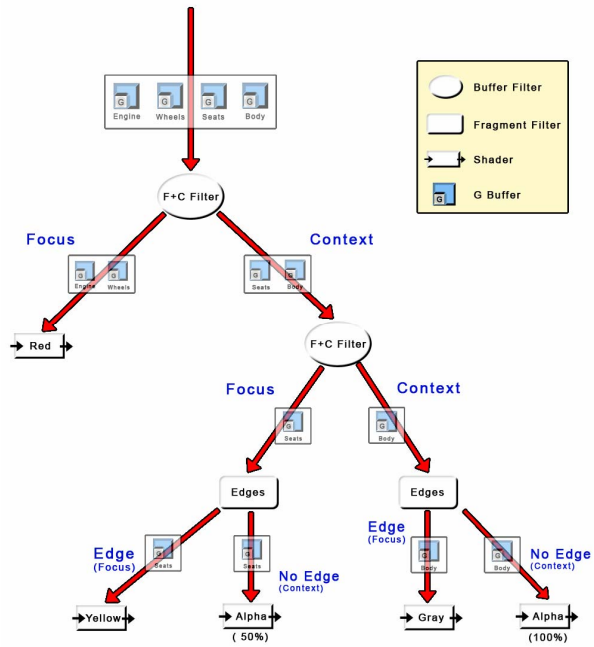


Figure 5 Conceptual data flow of the G-Buffers to obtain an AR visualization. Separation can occur in a per G-Buffer or per fragments basis. The leaves are shaders that process the filtered fragments.

This kind of filtering happens either during G-Buffer Rendering or during Scene Compositing. In this section we will explain both approaches, starting with filter operations while rendering the G-Buffer. This means that the rendering of the scene updates all values of a particular G-Buffer.

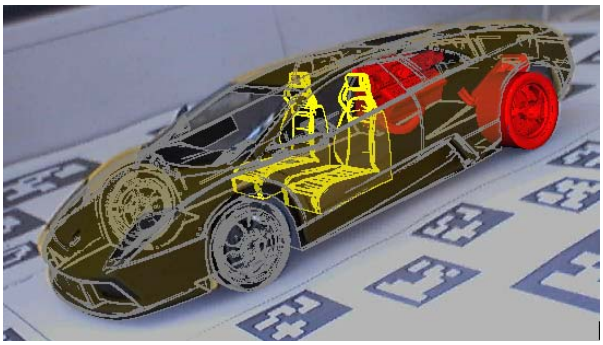


Figure 6 Corresponding visualization of the tree in the previous Figure

If more than one fragment falls on the same pixel of a G-Buffer, a test needs to be defined to choose only a single fragment. For this kind of filter operations we use traditional

OpenGL fragment testing. For example, a G-Buffer can be configured to only use values from fragments which are not occluded by other fragments of the same G-Buffer. Such a strategy is simply achieved by using OpenGL's depth test.

The way we group objects into families defines which fragments are tested against each other. This approach, together with the configuration of a G-Buffer's fragment tests defines the result of per pixel filtering during G-Buffer Rendering.

Filtering by first selecting a set of fragments depending on their G-Buffer affiliation, then followed by regular OpenGL fragment tests, implements an easy but powerful tool to control the amount of visible hidden information. The amount of augmented information in the final image depends on the number of G-Buffers and the strategy to discard fragments. Examples in section 6 show different ways to filter information during G-Buffer Rendering.

In order to reduce the depth complexity during G-Buffer Rendering, we need to have semantic information about the used data available to be able to define a satisfying strategy to group and discard fragments. With filtering during compositing we reduce the amount of visible information per pixel, regardless of the number of existing G-Buffers by filtering out fragments depending on their z-order after depth sorting was applied.

There is no universal rule describing how much information an observer can handle in front of an object, nor is it obvious which of this information should be best retained. Therefore, we enable the application to define raycasting strategies to identify those fragments which should contribute to the final pixel's value.

From our consideration in the beginning of this section, we know that a very important indication of spatial relationships comes from partial occlusions which indicate the shape of the occluding object. Consequently we have used a strategy which uses only the fragments of the first occluding object and those which belong to the focus – First Hit + Focus (Figure 7). Notice that filtering during the last stage is generally more expensive, because fragments discarded in this stage have already gone through the entire pipeline before they are finally eliminated.

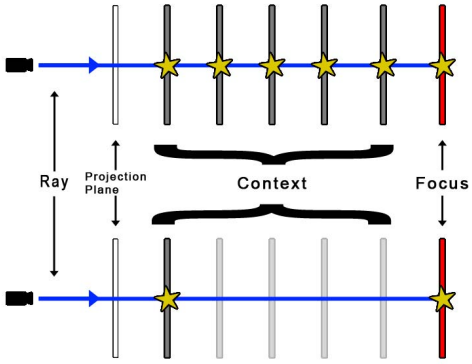


Figure 7 Comparison between two strategies during raycasting. Above: all fragments contribute to the final image. Below: the illustration shows our First Hit + Focus strategy where only the first context fragments and the focus' fragments are used to compute the final pixel's color value

4.4 Video Data in the G-Buffer

The information used for filtering in all the stages of the algorithm is only dependent on the fact that it is available in a G-Buffer. The sources of data that are stored in a G-Buffer are varied. The most obvious source, which has been mentioned before, is the rendering of scene objects. However, an important source of information in AR is the video stream used for video-see through augmentation. Since a video stream can also be represented in terms of a G-Buffer, it is also subject to the operations described throughout the 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, therefore easily processed. However, this implies that the resulting edges are also subject to tracking errors and poor registration. A video stream does not suffer the registration problem. However, it is subject to noise which can cause artifacts in the computed visualization. In section 6 we will illustrate some of these situations with practical examples. We will use different sources of data to compare and enhance our visualizations.

5 Implementation

To visually discriminate Focus from Context we need to render both with different styles. Assuming a scenegraph as is commonly used in VR and AR applications, a simple approach would place focus objects and context objects in separate branches of the scenegraph. 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 scenegraph data structure.

Instead of depending on a hierarchy that fulfils our requirements, objects are marked with contextual information and may be scattered throughout the scenegraph in any naturally occurring order without any enforced grouping. Sorting objects by context family happens implicitly during the scenegraph traversal, using the parameterized scenegraph described in [21]. Figure 8 shows a conceptual grouping of objects in the scenegraph regardless of where they are in the graph (highlighted in blue). In our implementation, objects are marked up with the context family they belong to, and this property is inherited to the subgraph below such a markup. The family in turn determines which G-Buffer to target in subsequent rendering.

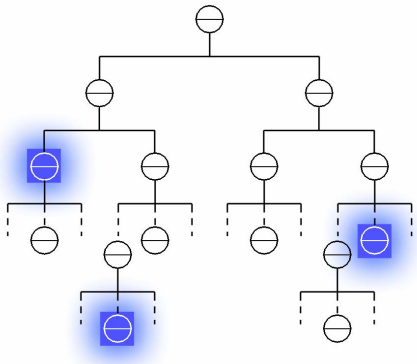


Figure 8 Context families of objects with the same contextual information (in blue) can be jointly referenced regardless of their position in the scenegraph

Our implementation is based on the GPU programming language Cg [19] and the OpenGL Frame Buffer Object (FBO) extension [11] as well as on multiple render targets. FBOs are collections of logical buffers such as color, stencil or depth. This

extension provides a mechanism for rendering to destinations other than those provided by the window system, and is therefore highly suitable for our purposes.

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 the color buffer's red component or the fragments depth value. For example, a G-Buffer with color, depth and object-id information needs to have six fields available: Four for RGBA color, one for depth and one for the id. Those six fields are represented by two 2D textures with three components each, and rendered using an FBO with multiple render targets and a fragment shader. The usage of multiple-render targets makes it possible to extract a large number of information from a single rendering pass. Latest graphics hardware is capable to render up to eight different targets. By using textures with four components, we are able to store 32 values per G-Buffer, a value we have found to be sufficient for our experimental implementation. We use a simple texture tiling technique, where each tile represents a G-Buffer. Switching G-Buffers when objects from a different context family are encountered during the traversal thereby merely means looking up corresponding viewport parameters to address the appropriate tile within the single target texture. We easily get an average of about 25fps without optimization on the examples shown in this paper.

6 Examples

In this section we demonstrate how our visualizations benefits Augmented Reality. We show an example of preserving context information to better communicate spatial relationships. We exemplify how the problem of depth complexity can be controlled and we cover different sources of context data.

6.1 Context preserving visualization with control of complexity

We will now illustrate the problem of depth perception in X-Ray vision and how our system addresses it by preserving important context information. We also demonstrate how our developed filters can be used to control the complexity of the created visualizations.

Figure 9 shows a naïve augmentation of objects in the focus of attention. Because the image is monocular, it is impossible to perceive the correct spatial arrangements of the hidden objects (in this case the engine and the back wheels).

Figure 9 A naïve overlay of hidden information. The engine and the wheels of the car do not provide sufficient depth cues. Occluded objects seem to be in front of the car rather than inside

A better approach for X-Ray visualization is to include key information of the context layers that occlude the focus. This information may be in the form of texture detail, half transparent material or edges. This technique adds a monocular depth cue called Object Occlusion [10]. Figure 10 shows an example of an enhancement of depth cues of the focus object. However, this comes at the expense of image clutter. The resulting image has an excess of important context information leading to a reduced visual perception of the focus.

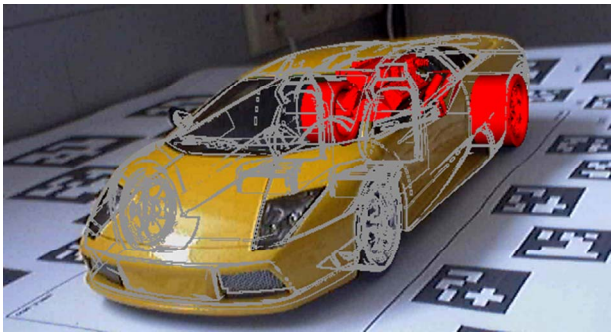


Figure 10 Adding key features from the context area. In this example we overlay visual key features such as edges. This enhances the depth perception of the focus objects. However, image clutter is caused due to excess of key features

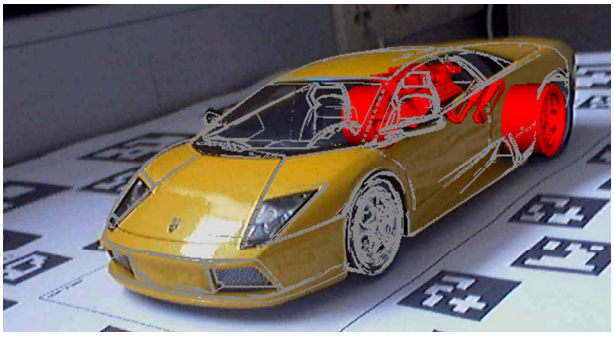


Figure 11 An example of the removal of fragments during Buffer Rendering. This is supported by grouping objects and using OpenGL fragment testing. In the image we combine all the car's faces and the seats. Extracted key features are no longer distracting. This is achieved by displaying only those key features which are visible in the real world's imagery. This yields to a more pleasant and better informative augmentation

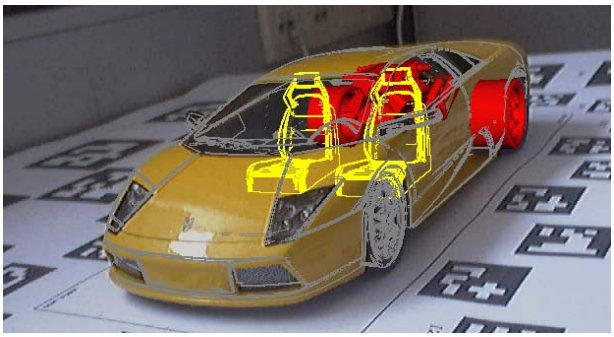


Figure 12 Including several level of context allows us to further discriminate the presented key features. In this image we present second level context objects in yellow outline. This further helps in scene perception

Our framework enables us not only to extract key features from context layers, but to also to control the amount of information visible in the final image. In this case we are interested in reducing the amount of distracting information.

Figure 11 shows filter results during G-Buffer Rendering achieved by simply grouping different objects into fewer G-Buffers and using regular OpenGL depth testing. Compared to Figure 10, we group the car's exterior back faces with its front faces and the seats in a single G-Buffer. This results in a display of less information by presenting only those key feature fragments of the context area which are visible in the real world imagery.

Color coding objects gives further cues to the user, not only relying on object occlusion (Figure 11). Shading objects differently can be achieved by using the objects id or by rendering objects to different G-Buffers. Figure 12 shows the image resulting from rendering the seats of the car in a new G-Buffer which is then shaded with yellow edges.

By introducing more G-Buffers we also introduce more information in the final image. By using per pixel filtering during raycasting we are able to control the amount of fragments depending on a chosen strategy. Figure 13 shows different colored G-Buffers and the 'First Hit + Focus' strategy (described in section 4.3) during scene compositing. Notice with per pixel filtering during scene compositing we are able to interactively change the strategy to filter fragments in different regions of the display which would be a complex and time consuming task by using filtering during Buffer filling. An example of using a magic lens tool to interactively control the filter region is given in Figure 22a.

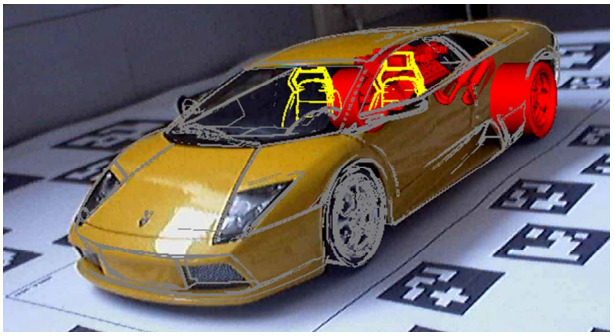


Figure 13 By using a First Hit + Focus strategy we are able to remove hidden fragments in the context area (regardless whether they are marked as important information or not).

6.2 Interactive Magic Lens Filtering

Magic Lenses implement filter operation which 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 3D [25]. Viega et al. also define flat lenses in 3D space which create another frustum, inside the cameras frustum in which they affect the rendering style of objects.

We use the concept of flat lenses to add another way of spatial filtering to interactively control the appearance of augmentations. We therefore apply magic lens filtering during the second step of our rendering algorithm (G-Buffer Processing) by using a lens' footprint as mask for F+C separation. This is done by rendering the lens to a G-Buffer which is used to mark fragments of another G-Buffer as either focus or context information. Figure 14 illustrates the G-Buffer Processing graph using a magic lens to generate the results shown in Figure 15 and Figure 21. This illustrates how we enhance key information in the context area, only in regions intersected by the lens. We do not apply magic lens filtering on all G-Buffers in a magic lenses frustum. In this example you can see a magic lens that adds the edges of context information only in regions that fall inside the lens frustum. However, focus objects are augmented regardless of whether they fall inside or outside the lens.

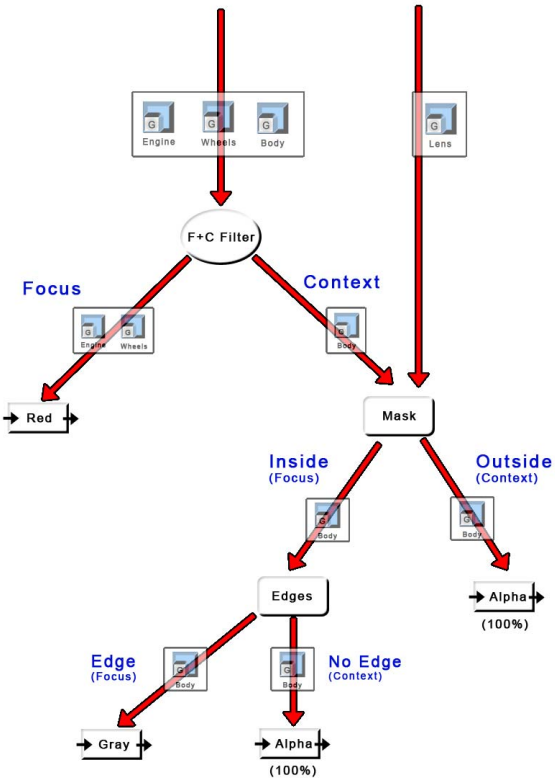


Figure 14 We can combine interaction techniques such as magic lenses in our framework. For example, lenses can serve as fragment filters. Furthermore, they can be restricted to act only in certain G-Buffers

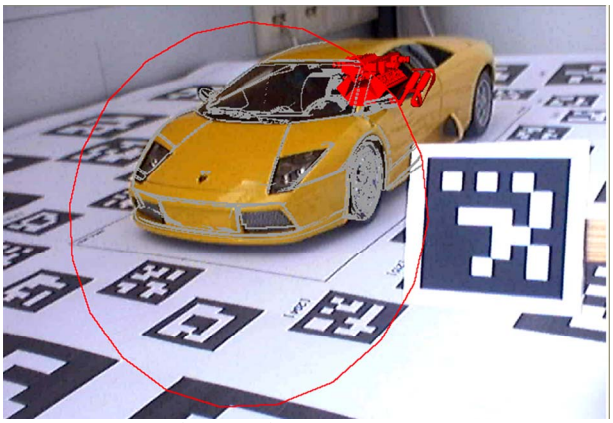


Figure 15 Result of the tree from the previous Figure. Notice how in this image the edges are only shown inside the lens. However, the focus object (the engine) is displayed regardless of whether it is inside the lens or not. This can be achieved by using the lens mask to control *G-Buffer Processing*.

Figure 16 illustrates the difference between traditional flat lenses and our lens implementation. The top image shows how a traditional magic lens affects everything in its frustum regardless of whether the effect is desired in all levels of information or not. The bottom shows our implementation and how it can affect differently the G-Buffers.

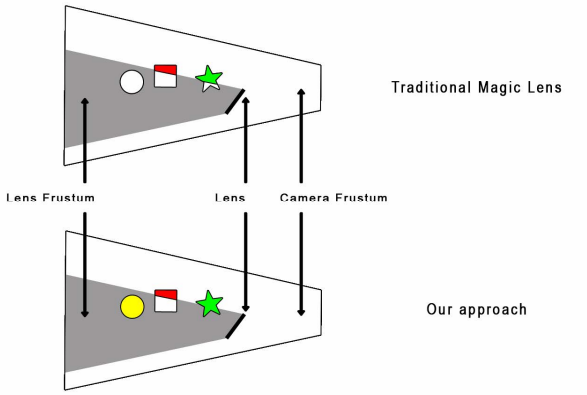


Figure 16 A conceptual comparison of traditional magic lenses and our work. Notice that the lens above affects the rendering of all the objects in its viewing frustum. Our approach, however, may be restricted to only act in certain G-Buffers

6.3 Source of Context Features: Real Vs. Virtual

Our framework enables us to process G-Buffers regardless of their source of information. Whether this information comes from rendered objects, pre-tailored masks or video streaming we are able to process them and take advantages of the best options. For example, adding key features, such as edges, to our augmentations can be a powerful depth cue, but it can also be distracting if used carelessly.

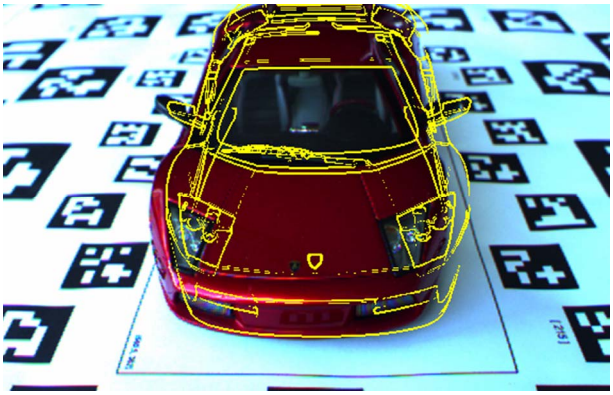


Figure 17 An image showing an augmentation of edges extracted from a 3D model. The edges are thin and clear, however, they suffer from poor registration

Figure 17, for example, 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. This can lead to confusing rather than helpful depth cueing.

This problem may be solved, however, if we rely on edge detection of the video stream. Figure 18 shows an example of this. Notice that the edges on top of the car are perfectly registered. However, these edges are thick and less detailed than those of Figure 17. This is a natural result of edge detection techniques on real imagery, which may be improved but will always be below model edge detection quality. Additionally, edges in this mode span the whole of the image and may lead to image clutter, since unnecessary areas are being enhanced.

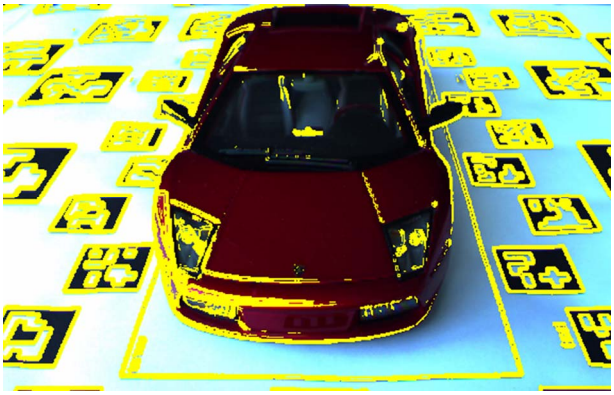


Figure 18 Edges that are extracted from the video stream are naturally registered to the image. However they are less detailed than those of the previous Figure. They also lead to image clutter since they are applied to the whole video image

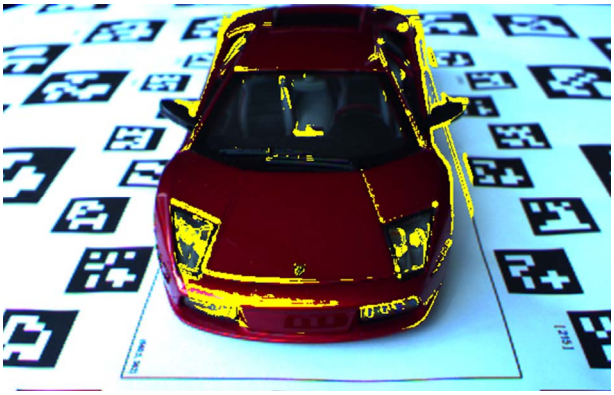


Figure 19 Using the footprint of the model as a mask leads to a less cluttered augmentation of edges from the video stream

A hybrid approach provides better results. Figure 19 shows an image that has been enhanced with edges detected from the video stream. The resulting edges, however, have not been augmented in the whole of the image, but a virtual mask has been used. This mask can be taken from the regions where the modeled object was rendered. This technique enables us to take advantage of the good registration quality of the video edge detection. Additionally it allows us to filter the undesired artifacts that produced image clutter in Figure 18 while at the same time allows us to solve the problem of poor registration of Figure 17. Nevertheless, it also suffers the problem of thick blurred edges and the mask may occasionally allow the augmentation of unimportant regions. In the next example we show how we use a user controlled flat magic lens to define the region of edge detection from the video.

Figure 20 shows how the problem presented in Figure 2 may be solved with edges extracted from the real video stream. The extracted edges provide an additional depth cue and since they

come from the real imagery, they are also able to preserve important landmarks (such as the entry points).

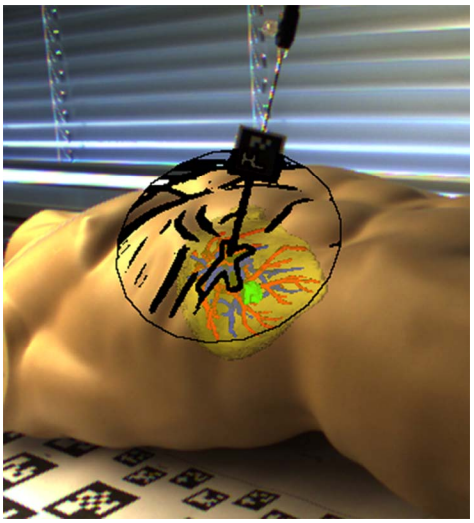


Figure 20 By enhancing with edges, extracted from the video stream we provide not only important occlusions but also we preserve important landmarks. The area where edges are being detected is interactively controlled by a flat Magic Lens which locally restricts augmentations with edges and prevents from cluttered displays.

7 Conclusions

Focus and Context techniques are highly successful in scientific visualization and have the potential to become an essential tool for Augmented Reality. 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 while the focus objects are not completely hidden with respect to correct occlusion handling.

We have presented a framework for computing augmentations using multi level F+C visualizations based on G-Buffer rendering and compositing, and we have outlined a variety of filter operations to separate the components for compelling visualizations.

The control of visual information presented to the user does not depend only on data structure management, nor is it only a function of image processing. We have shown how a combination of both enables us to better control the information augmented in the real world imagery. This includes context preserving X-Ray vision, and the ability to control the amount of overlaid visual information.

Context preserving X-Ray vision has a high potential for depth cueing, a fundamental problem of visual communication in AR. We have shown how important features from occluding objects may be used to enhance the depth perception to the scene rather than being discarded by careless augmentation. We have also addressed the problem of visual information overflow of visible hidden structures. Based on our framework we are able to control the amount of obscuring information in a scene, effectively reducing image clutter.

The development of AR applications can be simplified through the F+C techniques described in this paper, because they enable simultaneous interaction with multiple layers of information including the video stream, without requiring complex separation of cases. The scriptable framework we have developed enables the rapid prototyping of X-Ray visualization techniques. The core of the visualization framework is simple and only relies only on the capability to sort scene objects into families of similar context, so that it can be easily added to existing rendering frameworks as a post-processing step.

Acknowledgements

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. We would like to thank M. Eduard Gröller and Stefan Bruckner for useful discussions and suggestions.

References

[1] Bane R. and Höllerer T., "Interactive tools for virtual X-Ray vision in mobile augmented reality," In proceedings International Symposium on Mixed and Augmented Reality, 2004, pp. 231-239

[2] Bier E., Stone M., Pier K., Buxton W., DeRose T., "Toolglass and Magic Lenses: the see-through interface," In proceedings SIGGRAPH, 1993, pp. 73-80

[3] Bichlmeier C., Navab N, "Virtual Window for Improved Depth Perception in Medical AR", International Workshop on Augmented Reality environments for Medical Imaging and Computer-aided Surgery (AMI-ARCS), 2006

[4] Bruckner S., Grimm S., Kanitsar A., Gröller M. E., "Illustrative Context-Preserving Volume Rendering," In Proceedings of EUROVIS 2005, pp. 69-76

[5] Cook R. L., "Shade trees," Proceedings of SIGGRAPH, 1984, pp. 223 - 231

[6] Diepstraten J., Weiskopf D., and Ertl T., "Interactive cutaway illustrations," In Procceedings of EUROGRAPHICS, 2003, pp. 523-532

[7] Eissele M., Weiskopf D., and T. Ertl. "The G2-Buffer Framework", In proceedings of SimVis, 2004, pp. 287-298,

[8] Feiner S., Seligmann D., "Cutaways and ghosting: satisfying visibility constraints in dynamic 3D illustrations", In The Visual Computer 8, 1992,pp. 292-302

[9] Furmanski C., Azuma R., Daily M., "Augmented-reality visualizations guided by cognition: Perceptual heuristics for combining visible and obscured information", In proceedings of ISMAR, 2002, pp. 215-226

[10] Goldstein B.,"Sensation and Perception", Wadsworth Publishing; 7 edition, 2006

[11] Green S., "The OpenGL Framebuffer Object Extension", http://download.nvidia.com/developer/presentations/2005/GDC/OpenGL_Day/OpenGL_FrameBuffer_Object.pdf

[12] Interrante V., Fuchs H., Pizer S., "Illustrating Transparent Surfaces with Curvature-Directed Strokes," In IEEE Visualization, 1996, pp. 211-218

[13] Julier S., Lanzagorta M., Baillot Y., Rosenblum L., Feiner S., and Hollerer T., "Information filtering for mobile augmented reality", In proceedings of ISAR, 2000, pp. 3-11

[14] Kosara R., Hauser H., Gresh D., "An Interaction View on Information Visualization," In proceedings of EUROGRAPHICS, 2003, pp 123-137

[15] Kosara, R.; Miksch, S.; Hauser, H.: Focus and Context Taken Literally, in IEEE Computer Graphics and its Applications, Special Issue: Information Visualization, 2002, pp. 22-29

[16] Kosara Robert, Hauser Helwig, Gresh Donna L., "An Interaction View on Information Visualization," In State-of-the-Art proceedings of EUROGRAPHICS, 2003, pp. 123-137

[17] Krüger J., Schneider J., Westermann R., "ClearView: An Interactive Context Preserving Hotspot Visualization Technique," In IEEE Transactions on Visualization and Computer Graphics (12-5), 2006, pp. 941-948

[18] Looser J., Billinghurst M., Cockburn A., "Through the looking glass: the use of lenses as an interface tool for Augmented Reality interfaces," In proceedings of the 2nd international conference on

Computer graphics and interactive techniques in Australasia and SouthEast Asia, 2004, pp. 204-211
[19] Mark W., Glanville R. S., Akeley K., Kilgard M. "Cg: A system for programming graphics hardware in a C-like language." In ACM Transactions on Graphics, 2003,
[20] Mendez E., Kalkofen D., Schmalstieg D., "Interactive Context-Driven Visualisation Tools for Augmented Reality," In Proceedings of ISMAR 2006, pp. 209-218
[21] Reitmayr G., Schmalstieg D., "Flexible Parameterization of Scene Graphs," In proceedings IEEE VR, 2005, pp. 51-58
[22] Ropinski T., Steinicke F., Hinrichs K. H., "Interactive Importance-Driven Visualization Techniques for Medical Volume Data," In proceedings of the 10th International Fall Workshop on Vision, Modeling, and Visualization, 2005, pp. 273-280
[23] Saito Takafumi, Takahashi Tokiichiro "Comprehensible rendering of 3-D shapes," In proceedings of SIGGRAPH 1990, pp. 197 – 206
[24] Sielhorst T.,Bichlmeier C.,Heining S.,Navab N.,"Depth perception a major issue in medical AR: Evaluation study by twenty surgeons", Proceedings of Medical Image Computing and Computer-Assisted Intervention, 2006, pp. 364-372
[25] Viega J., Conway M., Williams G., Pausch R., "3D Magic Lenses," In proceedings of ACM Symposium on User Interface Software and Technology, 1996, pp. 51-58
[26] Viola I., Kanitsar A., Gröller M. E., "Importance-Driven Volume Rendering", In proceedings of IEEE Visualization, 2004, pages 139-145
[27] Viola I., Kanitsar A., Gröller M. E., "Importance-Driven Feature Enhancement in Volume Visualization," In IEEE Transactions on Visualization and Computer Graphics, (11-4), 2005. pp. 408-418

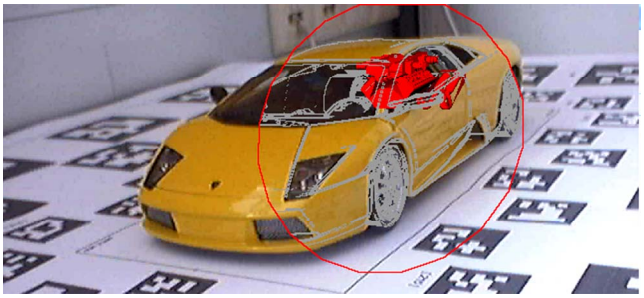
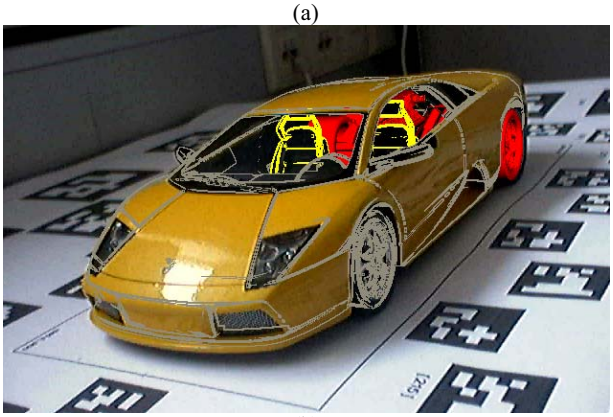
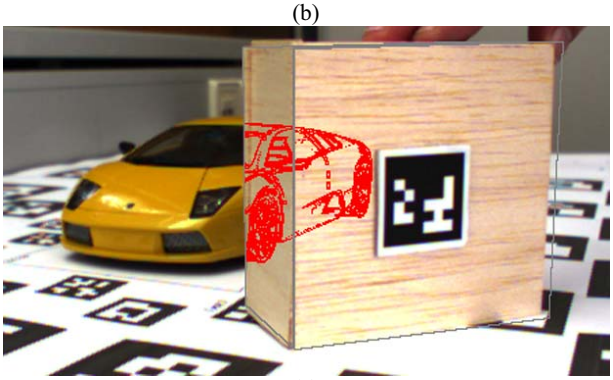


Figure 21 A Magic Lens is used to interactively control the augmentation of important context information

(a)



(b)



(c)

Figure 22 (a) A Magic Lens tool is used to interactively control the region where the 'First Hit + Focus' strategy is applied during scene compositing. The interaction allows for spatial driven adjustments of the amount of presented information. (b) As a side effect of our algorithm, we are able to render correct occlusions (c) In this example, we are not interested in a complete augmentation of hidden structures. Instead the visible parts of the box and the car are used as focus while the hidden parts of the car represent context information (for example the length of the car). We use the box as a Magic Lens tool to augment important context information only where occlusion occur.