Chapter 3
Visualization Techniques for Augmented Reality

Denis Kalkofen, Christian Sandor, Sean White, and Dieter Schmalstieg

Abstract Visualizations in real world environments benefit from the visual interaction between real and virtual imagery. However, compared to traditional visualizations, a number of problems have to be solved in order to achieve effective visualizations within Augmented Reality (AR). This chapter provides an overview of techniques to handle the main obstacles in AR visualizations. It discusses spatial integration of virtual objects within real world environments, techniques to rearrange objects within mixed environments, and visualizations which adapt to its environmental context.

1 Introduction

Augmented Reality (AR) applications enrich the real world environment with a certain amount of synthetic information, ideally just enough to overcome the limitations of the real world for a specific application. Azuma et al. [3] define three requirements of an Augmented Reality application. They claim that in addition to a mixture of real and virtual imagery, AR applications have to run in real time and virtual objects have to be aligned (registered) with real world structures. Figure 3.1 shows an example of a common AR system and the data which is acquired, computed, and presented. In order to register the virtual monster, the AR system derives tracking information from the video input. After rendering the registered 3D structure, its overlay allows to generate the impression of a virtual figure standing on a real world paper card.

This kind of visualization is a powerful tool for exploring real world structures along with additional contextual information. For example, by augmenting textual annotations, AR displays are able to provide semantics to real world objects or places. AR visualizations may also use the real world environment to provide

D. Kalkofen (✉)
Institutefor Computer Graphics and Vision, Graz University of Technology, Graz, Austria
e-mail: kalkofen@icg.tugraz.at

B. Furht (ed.), *Handbook of Augmented Reality*, DOI 10.1007/978-1-4614-0064-6_3,

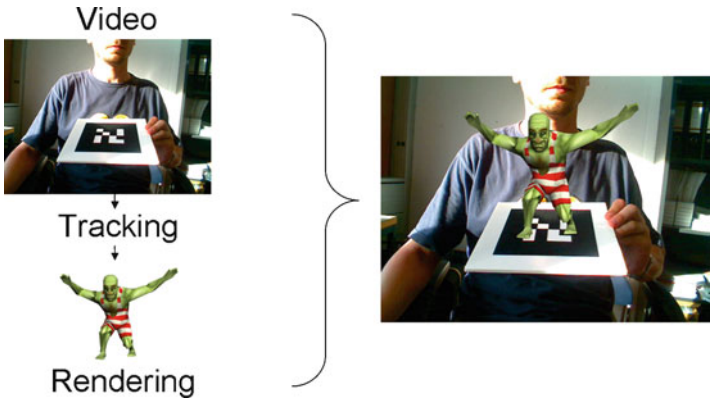

Fig. 3.1 Data flow in a common AR system. Real world imagery is delivered by the system's video feed and processed by vision based tracking algorithms. To align virtual and real data, the derived tracking data has been applied to transform the virtual content. Finally, the rendering is overlaid on top of the video feed

contextual information to computer generated graphics. For example, visualizations in AR are able to automatically select information relative to the current real world environment by using the user's location and orientation.

Since virtual information does not necessarily have to follow real world physical rules, AR displays are furthermore able to present a variety of nonnatural effects in a real world environment. This ranges from fictional characters living in a real world game environment [12] to a presentation of structures hidden in reality. By augmenting the virtual counterpart of a real world object, AR displays are able to uncover hidden objects, yielding the impression of seeing through formerly obscuring objects.

Azuma's definition of an AR application guarantees that the dynamics of real world environments remain unchanged after virtual renderings have been added. However, in order to comprehensibly fuse real and virtual information, both images have to be carefully combined rather than simply pasted together. If the computer graphics are generated independently from the information visible in the real environment, a successful visual interaction between both types of data may not be achieved. For example, without considering the information which is about to be removed by an augmentation, the resulting visualization may cause problems in depth perception. Our cognitive system interprets a set of depth cues (see Sect. 2.1) in order to pick up the spatial arrangements of the 3D objects in the environment. However, when carelessly adding computer graphics to the real-world imagery, the AR system may override some of those depth cues. This problem is illustrated in the simulated surgery depicted in Fig. 3.2. It demonstrates that spatial relationships between virtual and real data are lost after carelessly uncovering the hidden object. This effect happens due to the lack of occlusion cues. If no partial occlusion exists, judgment concerning the order of objects becomes difficult. Other existing depth cues (such as object size or object detail) are not strong enough to communicate the order of objects in such a visualization [48].

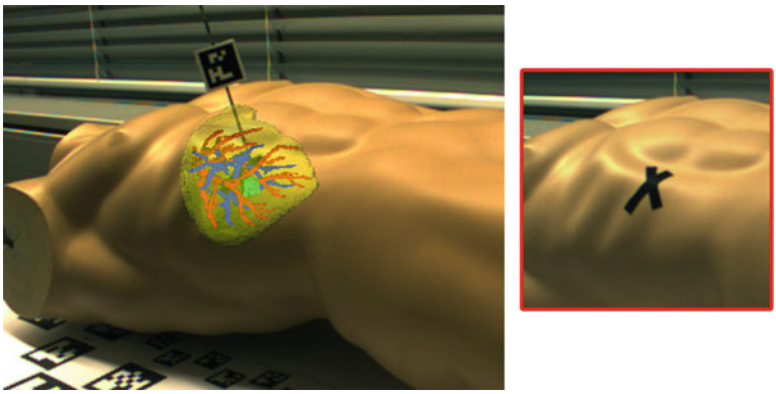

Fig. 3.2 Improper augmentation. In this example, careless augmentations of hidden structures cause two key problems: they override useful information (such as landmarks) and they lack depth cues. (*Right*) Original scene before augmentation, the *black cross* indicates the insertion point for the rfa-needle. (*Left*) Augmentation of the liver with its portal & hepatic vessel trees and a tumor

Furthermore, careless replacement of parts of the real world imagery may also hide important structures present in the real environment. Figure 3.2 shows how the computer-generated rendering of some of the inner anatomy obstructs the view of highly relevant landmarks which are present in the real-world imagery. In this example, the objective is to insert a needle into a patient's abdomen using a predefined entry point (black markings). By overlaying virtual organs on top of the real world imagery, the user is left unable to see the entry points which are marked on the skin of the patient.

In addition to the problems caused by overriding (and thus removing) real world imagery, careless generation of visualization in AR environments may lead to misleading interactions of colors and shades representing the real and the virtual objects. If the rendering of virtual objects does not take the prospective real world surroundings into account, the composition of both may fail to transport the intention of the visualization. Comprehensible visualizations in AR environments require that the appearance of the added 3D computer graphics fits into the real-world environment. Consequently, successful visualizations have to consist of easily distinguishable elements. For example, Fig. 3.3a shows an x-ray visualization with similar appearance of both hidden and occluding structures. Even though depth cues are preserved, it is difficult to make out the elements of the visualization. As a result the spatial relationships are difficult to perceive.

Quite a number of visual deficiencies may be caused by a naïve combination of virtual and real world imagery. To avoid this, the limitations of the AR systems itself have to be considered in order to generate comprehensible visualizations. For example, AR visualization have often to be generated from imperfect data. Incomplete descriptions of the environment as well as erroneous or unsynchronized tracking information may affect the impact of the resulting visualization [26].

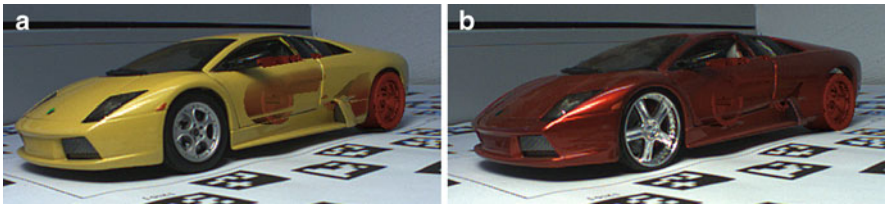

Fig. 3.3 Visual interaction between real and virtual renderings. The visualizations in (**a**) and (**b**) use the same parameters. However, while the visualization in (**a**) clearly presents all important objects, the augmented elements are hardly visible in (**b**)

Moreover, AR visualizations often have to deal with hardware restrictions such as small display sizes, limited field of view or restrictions caused by the egocentric nature of AR visualizations itself.

In the remainder of this chapter, we will first discuss techniques to enable a comprehensible integration of virtual objects into real world environments (Sect. 2). Then, we present techniques to manipulate mixed environments enabling to overcome limitations such as narrow fields of view (Sect. 3). Finally, we present context driven visualization techniques which allow to automatically adapt AR visualizations to their real world surrounding as well as to the uncertainty of the data involved in generating the rendering (Sect. 4).

2 Data Integration

A simple overlay of hidden structure on top of the system's video feed can cause a number of cognitive problems, caused by the processes involved in creating the impression of depth. Understanding these causes allows to develop rendering techniques which successfully add and preserve such information in AR visualizations.

2.1 Depth Perception

Our cognitive system takes approximately 15–20 different psychological stimuli into account in order to perceive spatial relationships between 3D objects [21]. These so called depth cues can be divided into monocular and binocular. While binocular depth cues require the use of two eyes, monocular cues appear even when one eye is closed.

Monocular depth cues can be further divided into pictorial cues, dynamic depth cues and oculomotor cues. Dynamic depth cues are caused by the fact that objects further away seem to move slower than objects close by when moving our viewpoint. Our cognitive system translates the difference in speed into an approximation of distances between the objects. Consequently, those depth cues appear if either the objects in the 3D environment or the observer move.

Oculomotor cues are caused by the feeling which occurs when the eyes converge and change shape in order to focus nearby objects. The feeling caused by the contraction of the muscles of the eye and the alteration of the shape of its lens is interpreted as distance from the object. While focusing on objects, which are very close, stresses the eyes more (causing more of the noted feeling), focusing on further away structures relaxes muscles and the lenses; this causes less of this particular feeling.

Pictorial depth cues are those that can be found in a single image including:

- Occlusion: if the 2D projections of two objects in the environment overlap, objects which are closer to the observer occlude objects which are further away.
- Relative size: more distant objects appear to be smaller than closer objects.
- Relative height: objects with bases higher in the image appear to be further away (compare the stakes of the bridge).
- Detail: objects which are closer offer more detail.
- Atmospheric perspective: due to dust in the atmosphere, objects which are further away appear more blurry than those which are nearby.
- Shadows: depending on the position of the light source, shadows can be cast from one object onto another.
- Linear perspective: parallel lines converge with increasing distance. Notice how the sidewalks seem to converge at some infinite place although in reality they appear to be approximately parallel.

Even though monocular depth cues enable one to make depth judgments, our typical impression of a 3D object is caused by binocular depth cues. This type of cue exploits the difference between the 2D projections of a point in 3D space on the retinas of the left and the right eye. Corresponding retinal points move further away the closer a 3D point gets. Nevertheless, in order to provide the user of an AR application with binocular depth cues, special stereoscopic display devices have to be used to provide the user with both renderings at the same time. Therefore, in the remainder of this chapter we will concentrate on monocular depth cues, in particular on those which can be found in a single static visualization.

2.2 Augmenting Pictorial Depth Cues

By rendering the virtual structure using a camera which uses parameters reflecting the characteristics of the real camera, the fusion of virtual and real world imagery will automatically provide pictorial depth cues which match to those present in the real world environment.

Figure 3.4 shows an example of an augmentation of a virtual Lego figure which stands (a) behind and (b) to the left of two real Lego figures. Among others factors, linear perspective, relative height (the virtual figure bases higher in the image) and relative size (the virtual figure is smaller) indicate the spatial relation of the virtual figure relative to the real ones. Since the parameters of the virtual camera have been

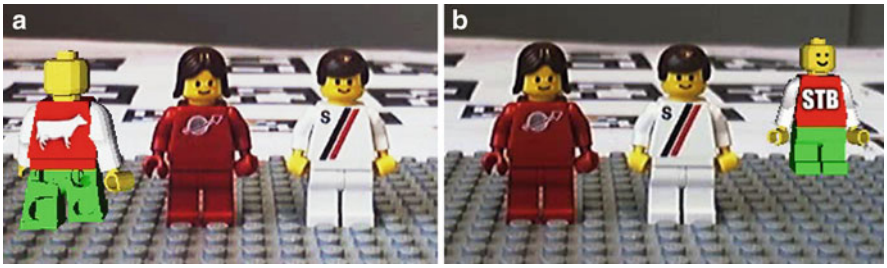

Fig. 3.4 Synchronizing the parameter of the virtual and the real camera allows to align real and virtual pictorial depth cues. The virtual Lego figure in (**a**) is correctly perceived next to the real figures, whereas the virtual one in (**b**) is correctly perceived behind both. This effect is achieved by aligning depth cues such as perspective distortion and relative size

aligned with those of the real one, the depth cues from both presentations line up. Even though a number of depth cues are missing (for example no shadows are cast and the virtual and the real world lighting do not match), the virtual Lego figures in (a) and (b) are perceived in its correct position in 3D space.

Synchronization of other parameters can further increase the perception of AR visualization. For example, Fischer et al. [17] as well as Okumura et al. [35] demonstrated the alignment of image noise and blur caused by defocusing and motion. Later Klein et al. [31] showed a more extensive set of synchronized parameters including color and lens distortion.

If the AR rendering framework is able to synchronize camera parameters, depth cues of 3D objects will line up with those present in the environment. Using this fact, additional virtual objects are able to enrich the perception of the real environment. The fact that renderings of registered 3D objects add a number of depth cues to the environment was used by Wither and Höllerer [47] as a means to increase the accuracy of depth judgments in real world environments. They additionally render carefully designed virtual objects, such as a chess board pattern, to encode additional depth cues.

2.3 Occlusion Handling

While renderings from synchronized real and virtual cameras are already able to align depth cues, as soon as occlusions between real and virtual objects appear, those depth cues are no longer sufficient to produce believable augmentations (Fig. 3.5a). Even though all other depth cues would have been added to the AR display, the virtual object will be perceived as floating in front of the video image. A believable integration of virtual structure into the real world environment becomes only possible if occlusions between real and virtual objects have been resolved (Fig. 3.5b).

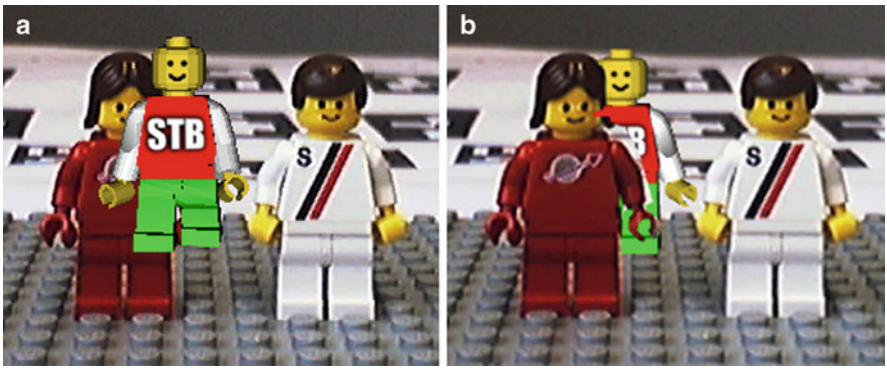

Fig. 3.5 Importance of occlusion cues (**a**) Even though a number of different depth cues exist, depth order is ambiguous and perception is wrong if occlusions have been ignored (**b**) The same rendering as in (**a**) with occlusion correctly resolved. This visualization is able to communicate the spatial relationship between its real and virtual content

Algorithm 1 Occlusion handling using phantom objects

1. Draw Video
2. Disable writing to color buffer
3. Render virtual representations of real objects (Phantoms)
4. Enable writing to color buffer
5. Draw virtual objects

2.3.1 Phantom Objects

To identify occlusions between virtual and real world structures, we have to compare the depth values of both types of objects per pixel in the screen space of the AR display. A common method to assign virtual and real world fragments has been presented by Breen et al. [7] and is called *Phantom Rendering* (outlined in Algorithm 3.1). Registered virtual counterparts of real world objects are included in the representation of the virtual environment, which subsequently enables to compare depth values in a common space. Phantom objects (which represent the virtual counterparts of real world objects) are rendered invisible (only to the z-buffer), before purely virtual objects are being rendered. Assuming that phantom objects are properly registered with their real world counterparts, the AR display becomes able to reject hidden virtual fragments using ordinary OpenGL depth testing. This strategy reveals occlusion by allowing only visible fragments of virtual structure to pass (Fig. 3.6).

Note that rendering of phantom objects does not only allow for occlusion handling but also enables for efficiently casting shadow from virtual to real world objects which can be used to provide further depth cues [23].

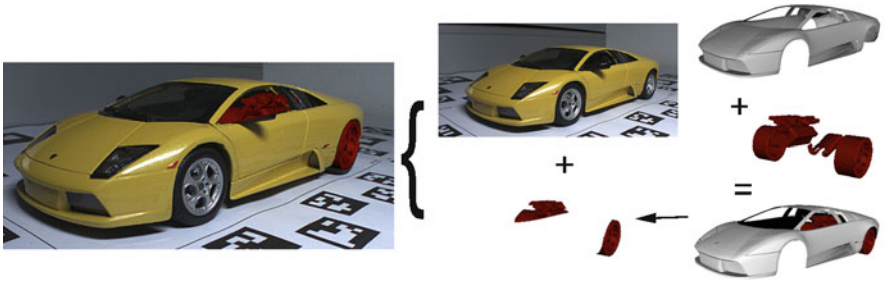

Fig. 3.6 Phantom Rendering in AR. The invisible phantom object prevents occluded virtual fragments from passing the render pipeline. The augmentation consists of only nonoccluded virtual fragments

Fig. 3.7 Uniform Transparency Modulation. Complex color arrangements will result in ambiguous presentations if uniform transparency modulations are applied

2.3.2 Advanced Occlusion Handling Using Video-Textured Phantom Objects

One strength of AR is the ability to uncover hidden structure and thus altering the original order of occluding and occluded elements. This *x-ray visualization* is a powerful tool in exploring hidden structures along with related real world information. However, as outlined before, without considering the information which is to be removed, the augmentation may lead to perceptual problems.

The easiest approach to present both hidden and occluding structure is to make the occluding one transparent so that hidden structure is visible. However, a simple uniform modification of the transparency values of occluding structures will most often result in ambiguous presentations. Figure 3.7 demonstrates the presence of clutter after blending the occluding structure uniformly with hidden objects. Even though very uniformly colored occluders covering high contrastive structures may allow one to perceive both types of data. The study of Buchmann et al. [8] showed that spatial relationships are lost if transparency is uniformly altered. The users in their experiment perceived hidden text on top of an occluding hand.

Rather than uniformly modulating transparency, comprehensible x-ray visualizations vary transparency values non-uniformly over the object. This results in

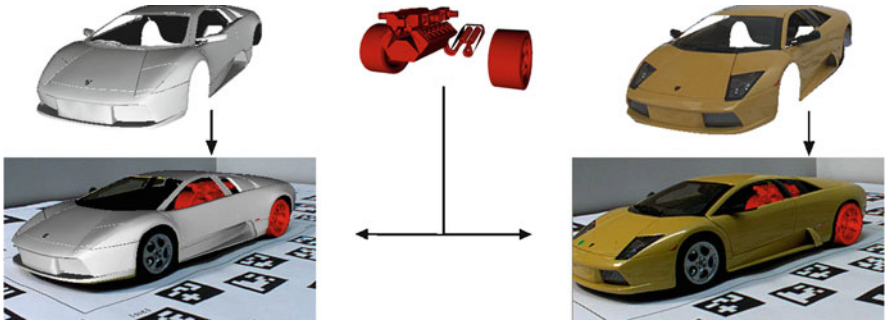

Fig. 3.8 Video Phantoms. Since ordinary phantom objects resolve occlusions by rejecting occluding virtual fragments occluding structure are not directly identified. Video phantoms (image on the *right hand side*) make use of current real world video information instead of virtual shades

a so called *ghost presentation* of occluding elements. In order to automatically generate ghost illustrations, a number of different approaches have been applied to control the means of modulation. Very early work from Crow proposes a function of the cosine [14] of the angle between a surface normal and the current viewing vector. To simulate different transparent media, Crow additionally uses the cosine to a modifiable power which can vary the drop-off function of his transparency modulation. Even though he didn't explicitly point it out, this allows for modulating transparency depending on the silhouette of the occluding structure. Notice that the silhouette of a 3D object is defined at those points where the viewing vector hits a surface point within an angle of 90 degree to its normal vector [22].

In order to render such effects in AR, we have to extend the idea of phantom rendering. Basic phantom rendering was only intended to resolve occlusions using a cheap process that can be executed on fixed function graphics hardware. The algorithm does not specifically identify real world pixels, which are suitable to be turned transparent. Instead of real world occluding fragments, only visible virtual fragments will be identified by this algorithm. Consequently, a system to generate ghost renderings has to use a slightly modified version of the original idea of Phantom Rendering. Instead of rendering the phantom invisible, the new approach renders a virtual and fully opaque presentation of real world structure (Fig. 3.8). Since this will obscure the real world object, the phantom object uses color information from the video feed. The implementation passes the current video frame to a fragment shader, which renders the phantom visible, resulting in a video textured phantom. Instead of using shades of virtual material (Fig. 3.8 left side), a texture lookup at the fragment's position after its projection to image space provides the output values (Fig. 3.8 right side).

By using a Video Phantom Object, we are able to apply any model based stylization technique to generate a ghost representation of occluding structure in AR. Figure 3.9 shows an example of two ghost renderings, which were generated from a video phantom. Their stylization uses principal curvature information [32] of the registered 3D mesh. The curvature values are defined per vertex and computed

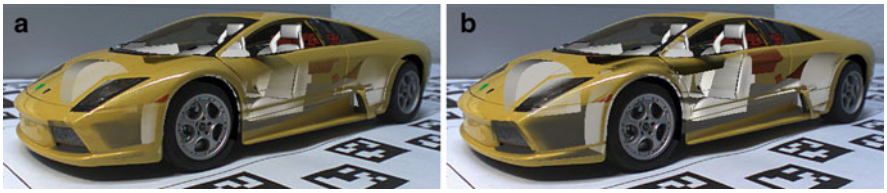

Fig. 3.9 Ghost presentations using video phantom objects (**a**) Dense ghost presentation (**b**) Sparse ghost presentation

in an offline process. During rasterization, the curvature values are interpolated between vertices, so that for each fragment (and thus for each pixel covered by the video phantom object) a curvature value is carried out. Finally, curvature values are linearly mapped to transparency values using (3.1). To control the object's transparency the parameters k and *tShift* have to be modified. While *tShift* linearly changes the mapping from curvature to opacity values, k changes the shape of the mapping function. High curvature values map to high opacity values while low curvature values map to low opacity values.

$$O_i = \frac{C_i}{k} + tShift;\ k \neq 0.0 \tag{3.1}$$

i:Current fragment; O: Opacity; C: Maximal curvature; k: Curvature weight; tShift: Transparency shift

2.4 Image Based X-Ray Visualization

Perfectly registered virtual models of real world objects enable to resolve occlusions between real and virtual objects. However, AR scenes commonly suffer from incomplete virtual representations. Often only the video in combination with the object of interest is available to the AR system. In such situations, the AR system requires knowledge about the organization of the scene in order to correctly sort their elements. While this information is often difficult to acquire for a general visualization, in case of applications using x-ray visualization to "see through" real world structure, the virtual data can often be assumed as being completely covered by real world objects. In this case, the depth order is known, and the AR system can analyze the video stream only in order to preserve important depth cues. In the following, we will review the extraction and preservation of image features which have been used to aid depth perception in x-ray visualizations in AR.

2.4.1 Edge Features

Several researcher have demonstrated the value of preserving or enhancing the result of an edge detector on the AR system's video feed [2, 28]. For example, Figs. 3.10 and 3.11 show edge preservations by applying an image based edge detector to the

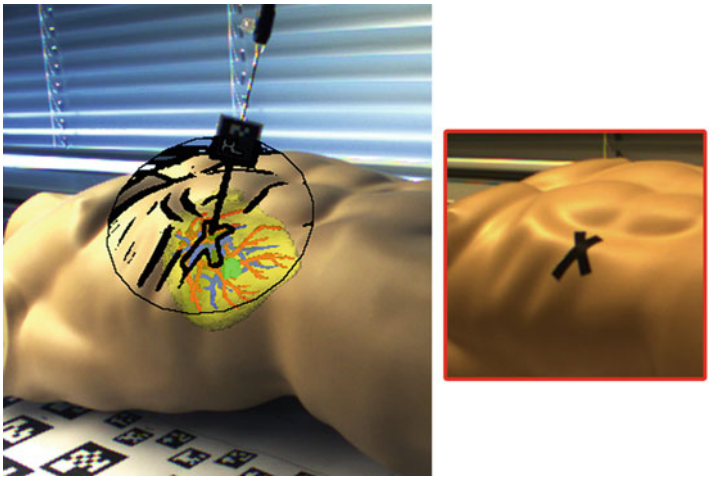

Fig. 3.10 Image based Ghosting using Edge Preservation. Preserving edges on the entire image clutters the presentation. An interactive Flat Magic Lens allows to control cluttering video ghosting. Edges have been preserved by operators in 2D image space only. Each video frame is processed and discrete edges are detected using and ordinary edge detector (such as the Canny operator [9])

Fig. 3.11 The Augmented Reality system uses a priorly acquired textured reconstruction of a remote scene (*center*) based on captured images and derived models. The textured reconstruction is overlaid on the user's view of the environment (*left*). In combination with highlighting the edges of occluding structure, only where hidden elements can be revealed, a comprehensible x-ray visualization is achieved while at the same time edge clutter is avoided (*right*)

current video feed. Note, by using only the video data as a source for preservation, other features than those which belong to the occluding object have been detected. In Fig. 3.10 the edges on the puppet, but also the edges of the ARToolkit marker and the needle (which is inserted into a tumor in this simulated surgery) have been detected and emphasized in black.

Even more confusing than edges from unwanted objects, is that edges extracted from video may clutter the whole view. As a remedy, a hybrid approach using features from video and tracked objects as stencil masks is able to reduce the image clutter. Figure 3.10 has been enhanced with edges from the video stream which were stenciled by the region covered by a Flat Magic Lens [43]. The Magic Lens

has been interactively positioned to augment only relevant edges from the video. However, the position of the magic lens has to be adjusted in each frame which may require an inappropriate amount of interaction. Therefore, if the manipulation of an additional filter mask is undesired or practically impossible, the visualization may filter depth cues automatically using the 2D footprint of the hidden elements (which describes the 2D area on the screen which is covered by the rendering of the hidden elements). Note that this area can easily be controlled by scaling the 2D footprint.

The implementation of such an information filter was demonstrated using GPU fragment shader in combination a multi-pass render-to-texture strategy [27] as well as by exploiting the capabilities of stencil buffering [2]. The video image captured from a user worn camera is rendered as a 2D background on the display before any other rendering occurs. Then, the occluded objects are rendered to the display and either to the stencil buffer or to a texture map. After this, the video image is re- rendered to the display while stencil tests (or fragment based texture lookups) ensure that edges are only drawn over the occluded objects. A fragment shader operates on each pixel of the video image and performs a 3×3 Sobel edge operator on surrounding pixels, and outputs opaque pixel colors for edges while preventing their rendering otherwise. Rendering of soft edges can be achieved by altering the alpha values near the edges. This can be achieved by applying a blur filter to the alpha channel of the edge texture.

2.4.2 Salient Features

Preserving edges from occluding elements helps to maintain context and improves spatial perception. However, visually important information about the occluding elements may still get lost. For example, contrasts in the visual features of the image, including color, luminosity, orientation and motion, determine *salient regions* and thus should be preserved as well. Salient regions can be understood as the regions in an image, which are most likely to attract the viewer's gaze [41]. In the following we discuss three salient features: hue, luminosity, and motion. The goal of this approach is to provide users with richer information about the occluding structure.

The presented saliency model is based on Walther's approach [44] (for a more thorough presentation, please refer to [38]). The sensory properties of the human eye are recognized to form a hierarchy of receptive cells that respond to contrast between different levels to identify regions that stand out from their surroundings. This hierarchy is modeled by subsampling an input image I into a dyadic pyramid of $\sigma = [0 \ldots 8]$, such that the resolution of level σ is $1/2^{\sigma}$ the resolution of the original image. From this image pyramid, P_{σ}, we extract the visual features of luminosity l, color hue opponency c, and motion t.

While motion is defined as observed changes in the luminosity channel over time, luminosity is the brightness of the color component, and is defined as:

$$M_l = \frac{r+g+b}{3}$$

Color hue opponency mimics the visual system's ability to distinguish opposing color hues. Illumination independent Red-Green and Blue-Yellow opponency maps are defined as following before the maps M_{rg} and M_{by} are combined into a single map M_c.

$$M_{rg} = \frac{r-g}{max(r,g,b)} \qquad M_{by} = \frac{b-min(r,g)}{max(r,g,b)}$$

Contrasts are modeled in the dyadic feature pyramids as across scale subtraction $\ominus$ between fine and coarse scaled levels of the pyramid. For each of the features, a set of feature maps are generated as:

$$F_{f,p,s} = P_p \ominus P_s$$

where f represents the visual feature $f \in \{l,c,m\}$. p and s refer to pyramid levels and are applied as $p \in \{2,3,4\}$, $s = p+S$, and $S \in \{3,4\}$. Features maps are combined using across-scale addition $\oplus$ to yield conspicuity maps C. Finally, all conspicuity maps are combined to form the saliency map S.

$$C = \bigoplus_{p=2}^{4} \bigoplus_{s=p+3}^{p+4} F_{p,s} \qquad S = \frac{1}{3} \sum_{k \in \{l,c,t\}} C_k$$

In the next stage, occluded and occluder regions have to be composed using their saliency information to create the final x-ray visualization. Figure 3.12 illustrates the composition of the different maps into the final x-ray visualization. Saliency maps S_o and S_d are generated for both the occluder I_o and occluded I_d images respectively. Further to this, we highlight edges in the occluder to emphasize structure. An edge map E is generated from the occluder region and weighted with the occluder saliency map.

$$E = \gamma(I_o) \times S_o \times \varepsilon$$

Where γ is a Sobel edge function and ε is a weighting constant. This edge map is combined with the occluder saliency map as an addition, $S_{o'} = S_o + E$. We combine $S_{o'}$ and S_d to create the final saliency map indicating the transparency of the occluder. We assume that salient regions of the occluder should take precedence over salient regions of the occluded. Additionally, a mask M and inverse mask M' is generated to reveal only the portion of the occluded region we are concerned with. Given this, the final image is composed as:

$$I_c = S_{o'} \times M + P_o \times M + P_d \times M'$$

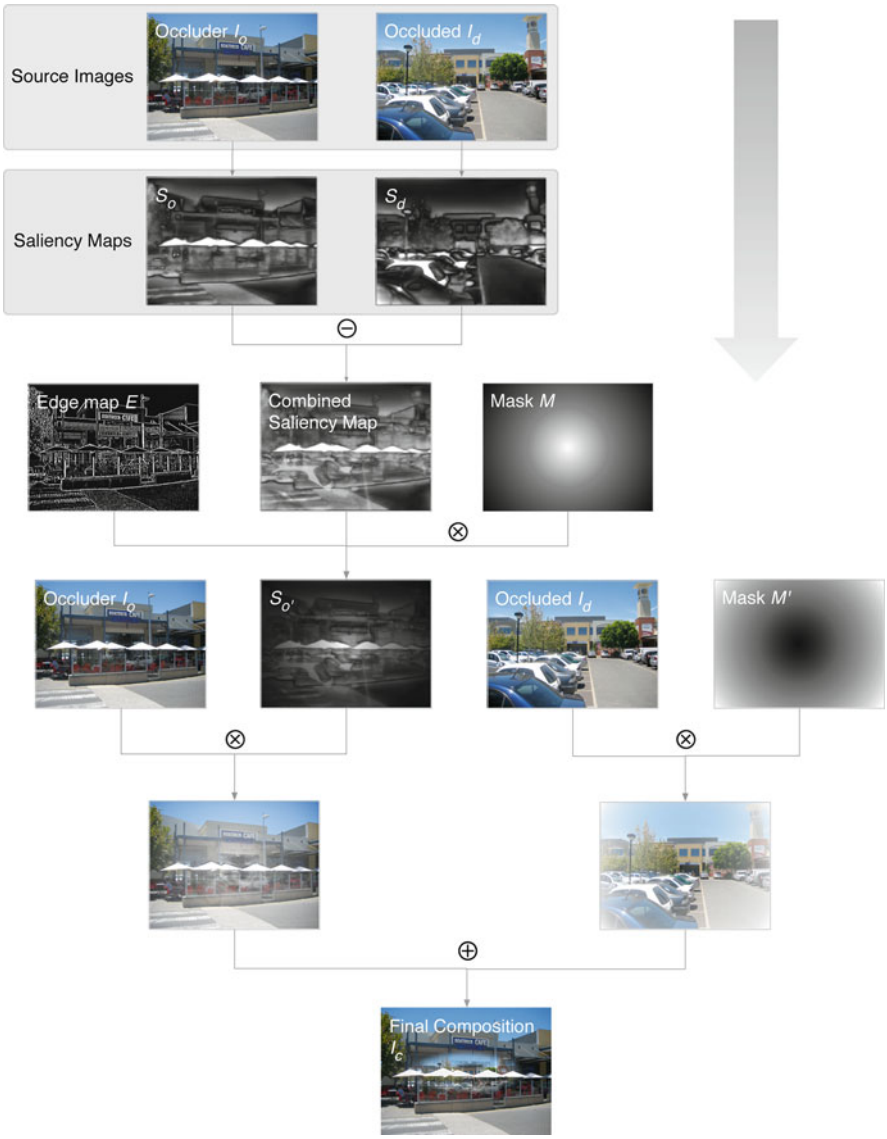

Fig. 3.12 Composition: source images are processed through a series of filters and combinations to produce the final output image. $\oplus$, $\ominus$, and $\otimes$ denote addition, subtraction, and multiplication of pixel values

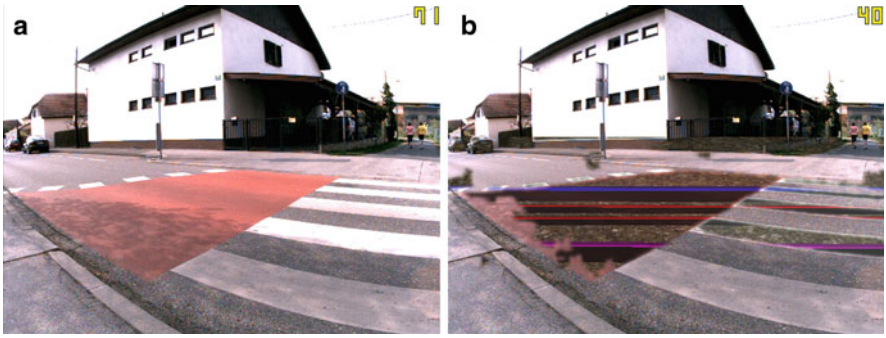

Fig. 3.13 Image based X-Ray Visualization using Feature Evaluation per Area. By using feature analyzes per area the AR system is able to assign the same stylization of video elements within an entire region of the image. Notice how the white crossings in (**a**) appear similar ghosted in (**b**), and how most of the pixels belonging to the concrete have been assigned to the same transparency modulation

2.4.3 Area Based Feature Preservation

While the algorithms discussed so far compute a transparency value for each pixel, Zollmann et al. [49] demonstrated occlusion handling per area. In addition to per pixel evaluations of image features, she computes saliency values on a superpixel representation of the image [36]. Such a hybrid approach enables for two advantages compared to an exclusive pixel based one. Firstly, it allows to evaluate further features such as the amount of texture in a region. If occluding elements do not provide a sufficient amount of detectable features, the resulting x-ray visualization will suffer from a lack of occlusion cues. However by inserting synthetic features (such as hatch marks or stipplings [22]) depth perception can be rectified in such cases. In order to decide when and where to augment additional depth cues, Zollmann computes the amount of texture as a local variation of illumination per superpixel.

An area based approach furthermore provides the data to apply similar transparency modulations on similar regions in the image. Since an image naturally consists of homogeneous areas, transparency modulations per area will introduces less noise than frequently changing values. For example, most of the street markings in Fig. 3.13 have been assigned to similar transparency values. The pixel in the red as well as those belonging to the white markings have been automatically grouped to form two classes of occluding structure. Finally, all pixel in a group have been assigned to the same transparency modulation.

3 Scene Manipulation

Whereas naïve combination of virtual and real world imagery may easily cause visual deficiencies, the limitations of the AR system itself has to be considered as well in order to generate comprehensible visualizations. In addition, hardware

restrictions such as small display sizes, narrow fields of view or the limitations caused by the egocentric nature of AR influence the comprehension of their visualization.

As a remedy, spatial rearrangements of the objects within the mixed environment have been demonstrated to be effective. For example, non linear distortions allow to increase the current field of view and rearrangements of real and virtual objects enable to increase their visibility.

This section provides techniques that deliberately modify real world imagery in order to increase the information content. First, we present techniques to relocate real-world objects (Sect. 3.1). Then, we present two space-distorting techniques (Sect. 3.2), that apply non-linear transformations to real-world objects.

3.1 Rearranging Real World Objects

Rearranged AR scenarios consist of real, virtual and relocated real information. To correctly compose an image out of all three types of information, the rendering algorithm has to fulfill three requirements. Firstly, it must be able to convincingly relocate real-world structures. Therefore, visual information has to be transferred from its original to the target location after the explosion was applied. Secondly, new imagery has to be generated to fill the original locations. Thirdly, the rendering algorithm has to correctly resolve occlusions between all used data.

To relocate real world information, we exploit the idea of video-textured phantom objects (see Sect. 2.3.2). To texture a phantom object with video information, we calculate the (u,v) coordinates for each fragment, as if the video background was applied using projective texture mapping from the camera's point of view. This is implemented by multiplying each vertex of a phantom with the model-view-projection (MVP) matrix before other transformations will be applied.

By using a vertex shader, relocation of video information can be achieved in two ways. One can either render the phantom geometry in their real location, look up the color from the video feed before the transformation to relocate the elements will be applied, or one can compute the MVP matrix for each phantom beforehand and use it in the shader which renders the phantom object in its new location. Since the second approach fits better into a scene graph framework, leveraging its ability to cascade transformations, we choose it in our implementation. We pass the matrix to transform a vertex from object to world space before the explosion's transformation is applied. The necessary interpolation of the calculated (u,v) coordinates is performed by passing the calculated values from the vertex shader to the pixel shader. Note that interpolation of (u,v) coordinates rather than color values is necessary to avoid artifacts.

Since all objects are rendered only once and no shading is used, the computation of pixel colors only consists of a calculation of texture coordinates and a lookup of the current video feed per fragment. Therefore, rendering of video-textured phantoms has negligible overhead compared to simple phantom rendering and also works if no relocation is applied.

3.1.1 Dual Phantom Rendering

With video-textured phantoms, we can relocate real world objects to another location in the image, thereby revealing the virtual objects behind the relocated objects. This assumes that the complete area of the relocated object is covered with virtual objects, which overrides the part of the image originally covered by the relocated object. However, frequently only a part of the uncovered area is occupied by a virtual object. Without special measures, the remaining area will still show the original video image (Fig. 3.14c). We must therefore extend the algorithm to invalidate any relocated real world information in its original location, to be able to either create a cut-out or to supplement incomplete hidden information (Fig. 3.14d).

To identify invalid pixels, we add a second render pass in which we project all fragments of a phantom onto their original real world location. This generates a 2D mask, consisting of only those pixels which will occur twice in a simple video-textured phantom rendering. This mask can then be used to remove redundant real world information, resulting in, e.g., a black background where no information is available. This algorithm is called *dual phantom rendering* [29], and can be described as follows:

1. *Enable and initialize framebuffer-object*
   (a) *Enable rendering to target 1 (T1)*
   (b) *Clear depth buffer and render target (T1 is cleared with 100% transparent pixels)*
2. *Render all video-textured phantoms (as described in Sect. 3.1) to T1*
3. *Render all virtual objects to T1*
4. *Switch rendering to target 2 (T2)*
5. *Render all phantoms in its original location to T2*
6. *Disable render-to-framebuffer-object and switch back to on-screen rendering*
7. *Fill the color-buffer with the current video feed*
8. *Cut out invalid real world information using T2*
9. *Superimpose T1*

Note that in step 5 of our algorithm, we are only interested in a binary 2D mask. This allows us to disable shading, thereby accelerating the rendering process. Furthermore, in cases where only a simple cut-out (and no restoration) of invalid real world information is desired, steps 7 and 8 can be combined by filling the color buffer depending on the 2D mask of the invalid video pixel (T2).

The algorithm as outlined only marks those fragments as invalid that will be visible in the final composition. This is controlled by the values in the depth buffer after virtual objects- and video-textured phantoms are rendered (after step 3). Not touching the depth buffer before rendering the phantom objects (step 5) allows us to reject all fragments which are hidden by either virtual or relocated real world information. This is an appropriate approach for those cases where a simple cut out of invalid information is desired. However, if the restoration of hidden information is requested, a 2D mask representing the entire phantom object produces better

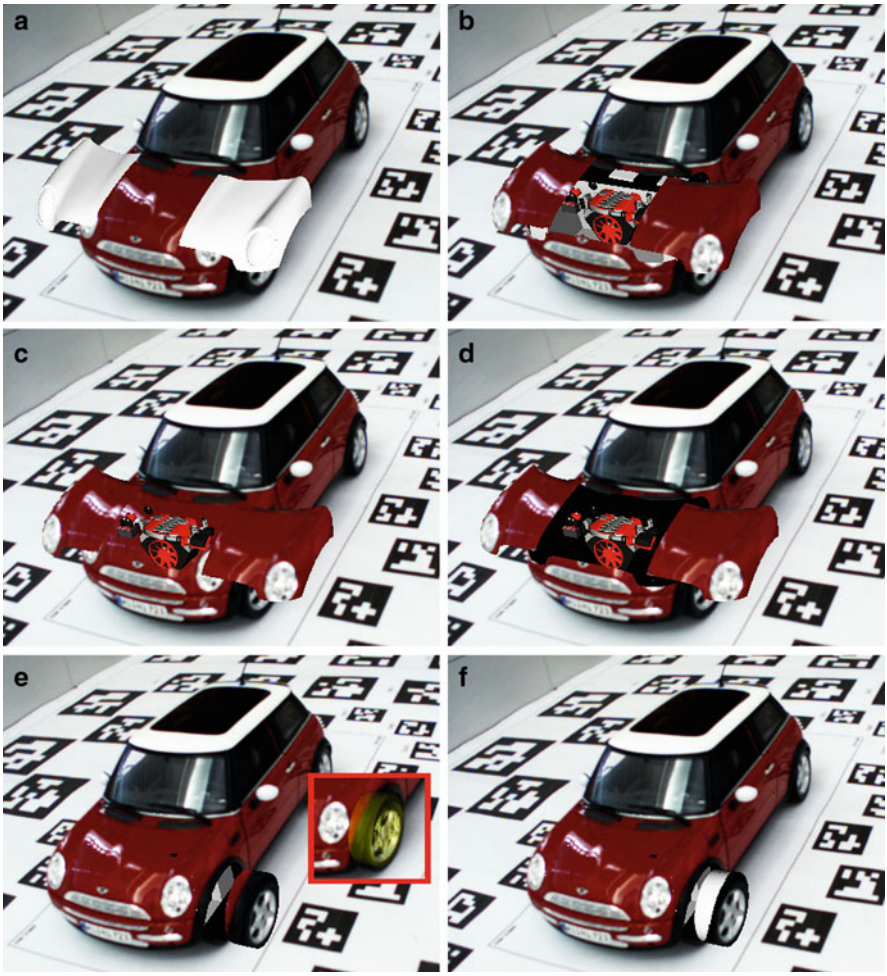

Fig. 3.14 Distributing real world information. (**a**) Exploded virtual phantom object (**b**) Transferred real world information to the phantom object (**c**) Incomplete virtual scene and phantom rendering (**d**) Dual phantom rendering is used to remove void information (**e**) Phantoms with overlapping 2D footprints using the same video information (**f**) Synchronized dual phantom rendering to control the usage of video information

results, because it presents the 2D footprint of the entire object and not only its visible portion. Such a 2D mask can be computed by clearing the depth buffer before phantom rendering is initiated (before step 5).

Even though dual phantom rendering can be accelerated in many cases, it still incurs a considerable performance overhead because of the required second rendering pass. Therefore, this approach is only recommended if required by the AR application.

3.1.2 Synchronized Dual Phantom Rendering

Labeling transferred video information in those places where a part of an explosion has been originally located enables us to remove redundant information. However, in those cases where phantoms overlap in screen-space, we will still transfer the same real world information to more than one object (Fig. 3.14e). To completely avoid duplicate usage of real world information, we have to further restrict the transfer of information to only those fragments of the phantom that are actually visible in its original location (Fig. 3.14f).

Therefore, instead of directly texturing a relocated phantom, we will first render the phantom object at its original location. However, instead of simply marking the information as invalid, it is labeled with the phantom's object ID. By using regular OpenGL depth tests we obtain an ID buffer of only visible fragments. This ID buffer allows us to restrict the transfer of video information to only those fragments which have been identified as visible in their original location. The algorithm to synchronize the transfer of real world information can be outlined as following:

1. *Enable and initialize FBO*
   (a) *Enable rendering to target 1 (T1 = ID-Buffer)*
   (b) *Clear depth buffer and render target*
2. *Render IDs of all phantoms in its original location to ID-Buffer (T1)*
3. *Disable FBO / Switch back to on-screen rendering / Clear depth buffer*
4. *Fill color-buffer with the current video feed*
5. *Cut out invalid real world information using the ID-Buffer as 2D mask (phantom ids >0)*
6. *Render all video-textured phantoms. Use ID-Buffer (T1) to control the usage of video information*
7. *Render all virtual objects*

While Synchronized Dual Phantom Rendering [29] requires both a second render pass and an ID buffer, we favor this approach over an unsynchronized Dual Phantom Rendering only in scenarios where the phantoms may overlap in screen space. Most of the real world scenarios consist of a set of objects which overlap in 2D, but some applications may only focus on a subset, allowing the use of the simpler unsynchronized dual phantom rendering.

3.1.3 Restoration

Since the video feed of an AR system delivers only information about visible real world objects, their rearrangement may introduce spots without any available information. Virtual objects are used to fill out these empty places, but often the virtual model fails to completely cover this area (Fig. 3.15).

We therefore utilize a restoration technique to fill in empty areas resulting from relocating real world objects. In our current implementation we identify the background information on the border of a mask, resulting from a relocation of parts

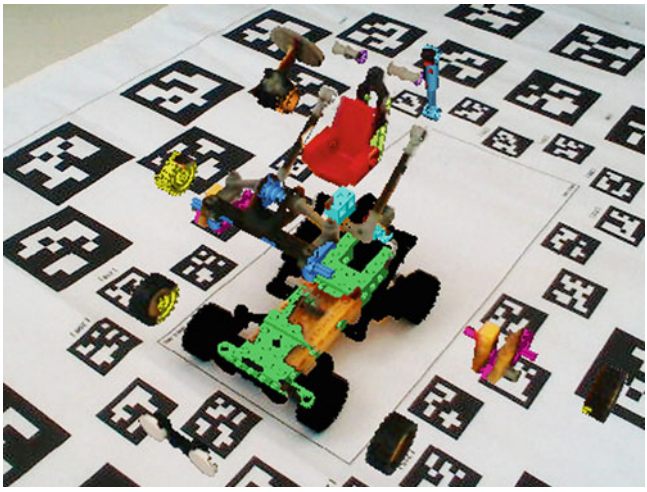

Fig. 3.15 Bad example of an explosion diagram in AR. No further shading of the transferred real world information is used. Notice the clutter of real and virtual information

of an object. The empty area is filled using the mean value of the all identified real world background information. This technique was chosen because it is simple and fast, and leads to acceptable results for a number of different applications. However, more advances techniques such as inpainting [6] exist and should be considered if this simple method fails.

The chosen strategy to visually unify the occurring material depends on the ratio of visible virtual to real world information. The visible pixels are counted with an occlusion query before pixel shading is applied. If the amount of available video pixel is too small (empirically set to less than 50%), we will only use the virtual color of an object (Fig. 3.16). However, if enough video information is present and only some virtually shaded fragments may disturb the perception of an object, we will re-shade the virtual information to visually fit to the used real world imagery. We have implemented this re-shade operator similar to the restoration of video background, by computing the mean value of real world color information on the border to the virtual fragments. This is implemented by computing the sum of all rgb values of all pixels at the border, which is divided by the amount of pixel at the border.

3.2 Space-Distorting Visualizations

Being able to perceive points of interest in detail within the user's current context is desirable, however, it is challenging to display off-screen or occluded points of interest. Sandor et al. [39] present a system which allows for space-distorting visualizations to address these situations (see Fig. 3.18). While this class of visualizations has been extensively studied in information visualization (for example [10]), this system presents the first instance of applying it to AR.

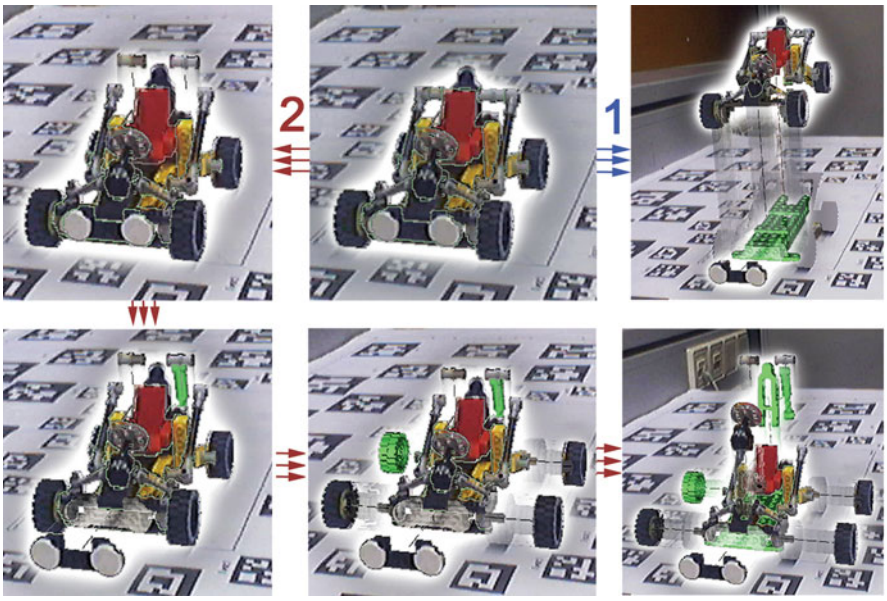

Fig. 3.16 Two animations (labeled 1 and 2 on the starting frame in the middle of the upper row) demonstrating reorganization of real world objects within an AR display

The system uses the approach of video-textured phantom objects (see Sect. 2.3.2), implemented by projecting the current video feed onto a 3D reconstruction of the environment. When the reconstructed model is distorted, the video image is distorted accordingly, minimizing the dissonance between the real-world and reconstructed model, and therefore reducing the cognitive load required to understand the distorted space.

To further reduce cognitive load, the system uses a *Ray* cue which acts as an overview and unifying signpost for the visualizations – a "cognitive anchor" for space distortion. Rays are rendered as wedges, emanating from the user towards the point of interest. Rays that pass through objects such as buildings become semi-transparent as an added depth cue. When distortions such as the Radial Distort occur, the Rays bend towards the distortion, indicating both their original direction in the near half of the ray and the distorted location in the far half. The amount of distortion is reinforced in the ray color, akin to stress visualization, where a ray under no distortion is green and a heavily distorted ray is rendered red. These ray cues were added from informal user feedback and are invaluable in providing a grounding for users who have no experience with space distortions.

This visualization technique builds upon the aforementioned cues to distort the space while keeping semblance with the real-world. To bring POIs outside the user's FOV into view, the system radially distorts the world around the user, compressing regions of the FOV that don't contain a POI. To reveal occluded POIs, it melts the occluding geometry. To reveal a POI that is both outside the FOV and occluded, it first radially distort the POI into view and then melts the occluding objects.

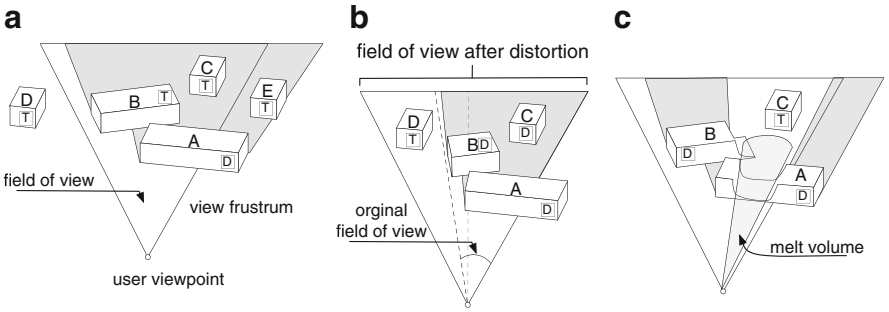

Fig. 3.17 The adaption of the Elmqvist and Tsigas occlusion model and how they apply to Melt and Radial Distort. (**a**) Elmqvist and Tsigas occlusion model to include targets outside the FOV. Target objects are flagged with "T" and distractors are flagged with "D". (**b**) Shows a schematic of the Radial Distort after distortion. (**c**) The melt volume with distractor objects flattened

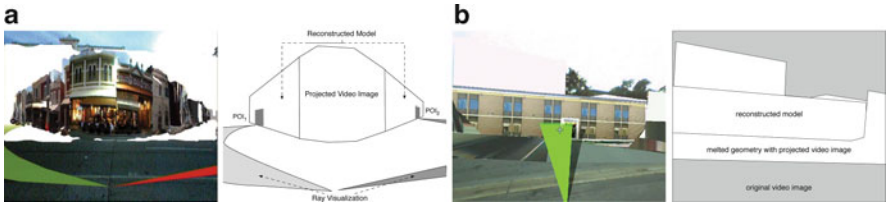

Fig. 3.18 The space distortion visualizations enable users to rapidly grasp the location of points of interest (POIs) that are either outside their field of view or otherwise occluded. (**a**) illustrates Radial Distort, revealing POIs that are outside the field of view of the user. (**b**) illustrates Melting, for discovering occluded POIs

3.2.1 Melt

The purpose of Melt is to reveal occluded targets. Occluded targets are revealed by virtually melting the distractor objects (Fig. 3.19d–f). A circle sector volume originating from the user in the direction of the POI defines the melt volume (Fig. 3.17c). The POI may be zoomed to gain more screen space and therefore present more information about its immediate context (Fig. 3.19g–h). Melt fits into the Elmqvist and Tsigas occlusion taxonomy [16] as a volumetric probe design pattern; however, it is passive/offline. The user selects the target from a list and the system animates the melting of distractors within the melt volume.

The melting metaphor replicates a common metaphor which we reinforce through animation. Like Virtual X-Ray, which is inspired by the 'superman'-like ability to see through buildings, Melt is inspired by the superman-like ability to melt buildings. Compared to Virtual X-Ray, Melt is suited to situations where there is more than one occluding object, such as POIs that are two or more streets over from the user. In this case, Virtual X-Ray loses most depth information besides the most

Fig. 3.19 The visualization applied in a city location. It enables users to rapidly grasp the location of points of interest that are outside their FOV by using our Radial Distort visualization. (**a–c**) show three frames of the animated Radial Distort visualization. (**d–f**) Show that occluded locations can be observed by our Melt visualization: the occluding geometry is flattened while projecting the video onto it. (**e–h**) POIs are zoomed closer to the user for visibility

immediate distractor. Virtual X-Ray also introduces high visual complexity when rendering transparency, decreasing depth perception and increasing cognitive load.

3.2.2 Radial Distort

The purpose of Radial Distort is to bring targets outside the FOV within the FOV, as illustrated in the Fig. 3.17b. Figure 3.19a–c show a video sequence of the visualization where one POI is outside the FOV. This POI is rotated inwards, so that it lies at the border of the FOV. The center of the FOV is compressed accordingly. The video image is projected in realtime onto the deforming geometry.

The benefit of such a technique is the ability to bring a POI to the attention of the user without forcing them to change their viewing direction, thereby keeping current context. Users can then turn their view towards the POI, with the distortion interactively unravelling as they do. In our experience, this interaction is more natural than using arrows to point to offscreen content.

Like Vallance [42], Radial Distort presents both immediate detail and continuous global context. Exact spatial relationship is degraded in favor of representing relative location to the user's current point of view. This follows the projection distorter design pattern. The user's current point of view is merged into n points of view for each point of interest, benefitting POI discovery at the cost of POI invariance.

3.2.3 Implementation

The video information is applied to the 3D reconstruction by using a video-textured phantom object (Sect. 2.3.2). A vertex shader projects texture coordinates for the current video image onto the front-most vertices of the undistorted scene.

Subsequently, vertex positions are distorted to show melting and radial distort. The space-distorting visualizations are implemented as GLSL 1.2 shaders. The pseudo code in 3.1 describes all POIs $p \in P$ as angles p_{angle_min} and p_{angle_max}, relative to the user's viewing direction, and a distance p_z, relative to the user's current position.

Algorithm 3.1: VERTEX SHADER(V, P, d)

```
for each v ∈ V
do { // calculate γ, the angle of v relative to the user's current
     // viewing direction d
     γ ← ANGLE(v, d)
     // radial distortion
     γ ← γ * RADIALDISTORT(γ)
     // melt any occluding objects
     for each p ∈ P
     do { if p_angle_min ≥ γ ≤ p_angle_max and v_z ≤ poi_z
          then v_z ← 0
```

Where V is the set of all vertices and RADIALDISTORT returns an angular coefficient that radially distorts the vertex into view. Our RADIALDISTORT implementation linearly compresses the FOV until all POIs are visible, and is defined by the following equation:

$$r = \begin{cases} \dfrac{1/2\, FOV}{\arg\min(p_{angle\_min} | p \in P)} & \text{for } \gamma \leq 0, \\ \dfrac{1/2\, FOV}{\arg\max(p_{angle\_max} | p \in P)} & \text{for } \gamma \geq 0. \end{cases}$$

Where r is the angular coefficient returned by RADIALDISTORT, such that it is the ratio of half the FOV to the greatest angle of a POI outside the FOV. At this stage, all visualizations have been calculated and are rendered by the fragment shader.

4 Context Driven Visualization

In previous sections, we have discussed techniques for integrating virtual structures into real world scenes through depth cues and occlusion handling (Sect. 2) and rendering techniques for manipulation of physical objects and scenes (Sect. 3). In this section, we focus on visualizations that alter appearance based on context by considering visualizations that are relevant to the physical context in which they are displayed (Sect. 4.1).

Virtual data presents additional contextual information to real world objects (Sect. 4.2), can represent invisible aspects of a scene (Sect. 4.3), or it gains from showing the object of interest within a specific real world context (Sect. 4.4). In addition, information about the quality of the data of the AR system provides the context of the application from a system's point of view. In order to improve the comprehensibility of AR displays their visualizations have to furthermore be able to adapt to those information (Sect. 4.5).

4.1 Situated Visualization

AR visualizations differ from more traditional computer visualization in that the virtual information is displayed in a physical context. Visualization typically appears on a computer screen, mobile device, or virtual world where the background is under the complete control of the visualization designer. However, the visualization itself often has no relevance to the surrounding environment. For example, a visualization of molecular structures in a virtual world might be presented with 3D data floating in front of a black background. AR visualization inherently has some form of background from the physical world but the background may have no relevance to the displayed data. For example, Fuhrmann et al. present an AR visualization of dynastic cycles of ancient China [18]. In this case, the background for the visualization is a white room. Moving to a different room would not change the meaning of the visualization. White et al. [46] use the term *situated visualization* to describe visualizations that are displayed in the context in which they are relevant. For example, carbon monoxide data presented on the street where the sensors collect the data.

The goal of a situated visualization is often to make meaning from the combination of the physical world and the virtual representation. In this case the visual data needs to clearly be associated with the physical location or object and take into account the user, task, and physical context. In the case of information visualization, this can change the representation or spatial layout of the visualization.

Here we focus on the challenges of information visualization situated in the physical world. Several challenges arise in data visualization in AR: the individual data points must be clearly associated with their proper physical world relationships, visualizations must take into account the background and user to assure legibility of the data, appropriate mapping from data to representation and meaningful representations must be chosen, and spatial layout of the visualization must reflect any existing spatial constraints in the data.

Some of these issues have been addressed in previous sections of this chapter. Here we focus on changes driven by context. Sources of context include objects in the scene, the scene itself, type of data being displayed, changes in live sensor data, user, type of display, and even the task. Changes to the visualization include the mapping and representation, spatial layout and coordinate system,

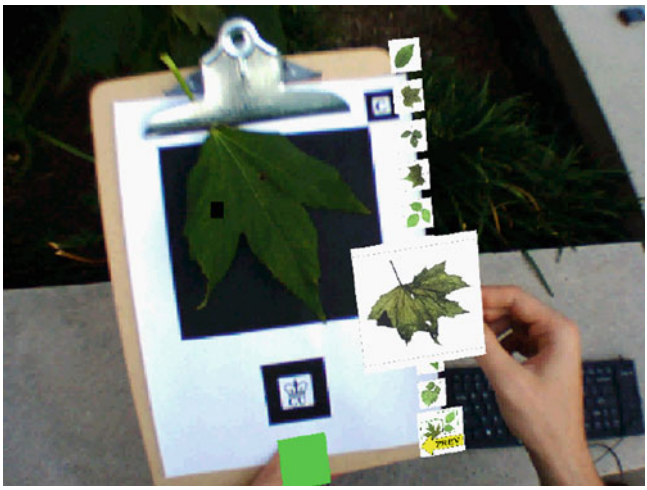

Fig. 3.20 The physical leaf species is identified using computer vision techniques. The visualization changes based on semantic context, in this case based on the specific species identified

4.2 Object as Context

Some of the most meaningful visualizations use an object in the scene as the focus. If the object is already known, a static visualization can be crafted to convey the appropriate information. However, if the focus of the visualization is unknown, the representation and layout made need to change based on the object. The approach taken by White et al. [46] uses computer vision to identify the object of focus and changes the visualization based on object classification. In the Tangible Electronic Field Guide (Fig. 3.20), a leaf is placed at the center of a clipboard to begin the task. Moving a handheld fiducial marker under the leaf triggers the matching algorithm and a best match for species is found using the inner distance shape context developed by collaborators Ling and Jacobs [34]. As the list of matching species is completed, the visualization forms proximal to the physical leaf. The spatial layout of the visualization can be flexible here because proximity is the primary constraint. Thus, the visualization can be arranged linearly or oriented in a circle around the leaf, depending on the size of the leaf. Each leaf node can be picked up and examined by the user and compared with the physical leaf. The images are dependent on the physical leaf itself and the task of the user. If the user changes modes by flipping the handheld marker, the leaf images change to full tree image or bark images to aid in the identification process.

4.2.1 Implementation

The architecture of the EFG incorporates aspects from context-aware computing, image identification, and visualization. In particular, our architecture borrows from Chi's Information Visualization Data State Reference Model (or Data State Model) [11] and Abowd et al.'s Cyberguide [1].

In the Data State Model, Chi describes four states of data for visualization (value, analytical abstraction, visualization abstraction, and view) and the transformations between states (data transformation, visualization transformation, and visual mapping transformation). The data transformation converts data values to an analytical abstraction. For example, data transformation could involve taking a set of ranked matching species data and transforming them into an ordered list. The visualization transformation converts the analytical abstraction into a visualization abstraction. For example, it could transform an ordered list into a set of images displayed in row-major order. The visual mapping transformation then provides a view onto the visual abstraction. For instance, it might render an overview of the entire set of images or a zoomed view of a single image. Heer and Agrawala suggest a design pattern based on this model that they refer to as the Reference Model pattern [25].

The advantage of this model is that we can separate out both analytical models and views with view transformations from the original data set. We find this useful in that we can take the original data from both the entire dataset or matching results, convert to hierarchies or lists, and then provide appropriate views such as quantum tree maps [4] or row-major layouts.

The Cyberguide tour guide is a mobile system that provides information based on the relevant context of the user (e.g. the physical location of the user). It has four service components: map, librarian, navigator, and messenger. The map component provides a set of maps for the system. The librarian maintains information about a tourist site. The navigator maintains the position of the user in the system. The messenger handles communication between the user and other users or systems (see Fig. 3.21). In our case, we are interested in context services that represent location, time, explicit meta-information from the user, and object matching. By using this context service, we can change the underlying values based on object context or visualization abstraction.

4.3 Sensor Data as Context

Visualizing sensor data can also provide a challenge because the specific data coming from the sensor will change and the location of that data may change. In the Sitelens system, White et al. [45] use situated visualization to visualize carbon monoxide (CO) data in the Manhattanville area of New York City. CO levels are measured using a CO sensor together with a combined DRM+GPS system. Each measurement is represented as a sphere. For the red spheres, location of the measurement is mapped to sphere location and CO level for each sample location

Fig. 3.21 Sitelens. Colored spheres represent geocoded sensor data that is invisible to the eye

is mapped to the altitude of each sphere. For the green spheres, the measurements come from a static sensor source provided by the Environmental Protection Agency, which is meant to represent CO levels in this area. The location is mapped to the same location as the local CO measurements to provide a comparison of two sensors meant to represent a given location.

Note that care must be taken in mapping sensor data to visual representations in situated visualizations. In a pilot study, White et al. [45] found that mapping data to size provided visual cues that were confused with depth cues. In a typical 2D visualization, size can be used and even in an AR visualization that is not 3D, size can be used. There is no control over the background in the situated visualization case so visual mapping that might be mistaken for depth cues are avoided and redundant encoding of data using multiple channels (e.g. color and shape) can be helpful.

4.3.1 Implementation

The SiteLens architecture borrows from our Electronic Field Guide architecture in the previous sections, extending it in several areas. First, we now need to know the orientation and location of the display relative to the physical world. A new tracking component, which gathers orientation, GPS, and fiducial marker orientation and position, manages this information to represent spatial context. Second, we use the same concepts for visualization management but maintain a collection of visualizations that are currently active, either display-referenced or world-referenced. Third, we incorporate a component for importing and loading

georeferenced data. Fourth, we incorporate a component within the context service for gathering live sensor data via Bluetooth.

The visualization manager is similar to the EFG visualization manager. However, here each visualization is initialized with specific abstract representations for each data element in a data set and these representations can be changed to represent different colors, shapes, and mappings. Each visual representation is a subclass of a data node in our architecture, so we can easily create new visual representations and data mappings.

4.4 Scene as Context

The background scene provides both semantic and optical context for the visualization. This becomes important in avoiding placing information over relevant aspects of the scene (as discussed in the previous section on saliency), reflecting relevant parts of the physical scene through outlining, highlighting, or lowlighting, and insuring that the visualization itself is legible.

In terms of legibility, Gabbard et al. [19,20] conducted a set of studies to compare the readability of different text styles on different textured backgrounds, such as brick or pavement, under different illuminance conditions. Their studies focused on optical see-through displays and found that billboarding and fully-saturated green text provided best results while fully-saturated red text styles were problematic. They also suggest that the right active text style will result in better performance than static text drawing styles although they were unable to show this conclusively.

Placement of virtual information must also take into account the background scene. Leykin and Tuceryan used a machine learning approach to address this problem [33]. In their experiment, they use text overlaid on a variety of textured backgrounds. Each example was ranked by a human for readability and this was then used as training data for an SVM. The feature extracted from the training data was the response from a Gabor Filter across the 90×90 pixel sample and was meant to represent texture. Their approach then subdivides the scene into 90×90 blocks which are fed to the SVM. The block with the highest readability rank is then used for displaying text.

Tanaka et al. [40] also address this problem by focusing on color space characteristics. Their approach subdivided the image based on a grid and analyzes averages in RGB and HSV color spaces and variances in RGB, YCbCr, and HSV color spaces. Each grid subdivision is then ranked and information is placed in the location with the maximum viewability.

These techniques are related to view management developed by Bell et al. [5] in that we look for the best location to place information. However, Bell takes advantage of knowledge of the spatial layout of a model of the scene and does not take into account the actual image of the scene.

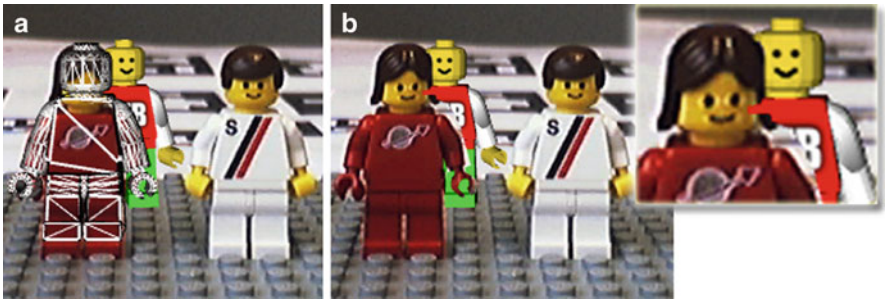

Fig. 3.22 Erroneous Phantom Rendering. (**a**) The phantom object does not perfectly reflect its real world counterpart. (**b**) The error causes wrong depth sorting at some pixels. Notice the error close to the shoulder of the *red* Lego figure

4.5 Uncertainty as Context

To allow visual interaction between virtual and real world data, the virtual information has to be integrated into the real environment instead of simply being layered on top of real imagery. To be able to resolve occlusions, both types of data have to be analyzed and compared in a common space, using e.g. a phantom or a video phantom object. The corresponding algorithms transfer real world information into the coordinate system of the virtual camera, followed by an analysis of all depth values using ordinary 3D computer graphics hardware.

However, to perfectly map a virtual object to its real world counterpart, the virtual model has to exactly reflect the real object's shape, and its 3D registration has to transform the virtual model perfectly to fit to its real location. While both steps are prone to errors, both also influence the quality of the resulting visualization in AR. Figure 3.22 shows the effect of a virtual counterpart which does not perfectly reflect the real world object. The classification falsely identifies parts of the background as being in front of the virtual figure. To avoid ambiguous AR visualizations we have to consider the existence of erroneous data and thus have to support error-friendly visualizations which are robust enough to either overcome or lower the problems resulting from errors.

A pragmatic approach to handle imperfect data is to either refine the data using fusions of multi data sources or to refine the augmentation itself. For example, while Klein et al. [30] use additional sensors to adjust an object's registration, Diverdi and Höllerer [15] refine the prospective overlay in image space, so that it fits to its real world counterpart. Both algorithm search for edges in the augmentation which correspond to edges detected from the video feed. However, while Klein adjusts the pose of the virtual camera, Diverdi alters the resulting rendering in favor of a more precise augmentation.

An approach to deal with registration error is to incorporate the error estimate as a parameter of the visualization technique. As demonstrated by Coelho and his colleagues [13], information about the current uncertainty of the data can be used

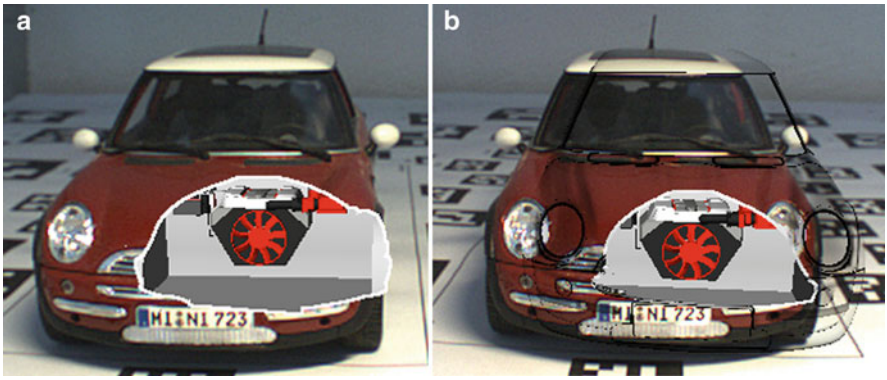

Fig. 3.23 Error Communication (**a**) The error is not communicated (**b**) An additional presentation of contextual structure is able to communicate the error. The additional augmentation is augmented as a ghost stylization of the structure in the context of the object of interest

as contextual information to the application in order to change the way textual annotations will be arranged. Coelho shows how a switching from an internal to an external placement strategy (see [24] for an overview of annotation strategies) can be used to resolve disambiguations in case of registration errors.

As outlined by Robertson et al. [37]), by additionally presenting easily perceivable augmentations of virtual counterparts of real scene elements helps in understanding the current error. This technique is able to visually communicates the current error to the user and thus enables them to ameliorate scene understanding in AR. For example, compared to Fig. 3.23a, the AR visualization in Fig. 3.23b has been enriched with a sparse representation of elements in the surrounding of the object of interest. By providing such an additional augmentation, the user of the AR system becomes able to mentally correct the error.

5 Closing Remarks

Augmented reality displays extend the viewer's perception via the addition of computer generated information. The visual interaction between virtual and real world imagery is the main advantage of AR visualizations compared to traditional visualizations. AR visualizations have a high potential, however, their success is dependent on their comprehensibility. If heedlessly implemented AR visualizations easily fail to visually communicate their information.

The complex character of AR environments requires complex visualization techniques to neither isolate certain structures nor to generate ambiguous presentations. In this chapter we have provided an overview of the fundamental techniques to comprehensibly fuse virtual and real world imagery. We have provided techniques to spatially integrate virtual objects within real world environments. In

addition, we have presented techniques to use contextual information to increase the comprehension of AR visualizations and we have explained techniques to enable spatial manipulations of objects within AR environments.

Acknowledgments For valuable discussions on the topics presented in this chapter we would like thank Markus Tatzgern, Stefanie Zollmann, Steve Feiner, Peter Belhumeur, David Jacobs, John Kress, Sarah Williams, Petia Morozov, Andrew Cunningham and Arindam Dey. This research was in part funded by Nokia Research Center, a grant from the Boston Society of Architects, NSF Grant IIS-03-25867, and a gift from Microsoft.

References

1. Gregory D. Abowd, Christopher G. Atkeson, Jason Hong, Sue Long, Rob Kooper, and Mike Pinkerton. Cyberguide: a mobile context-aware tour guide. *Wireless Networks*, 3:421–433, 1997.
2. Ben Avery, Christian Sandor, and Bruce H. Thomas. Improving spatial perception for augmented reality x-ray vision. In *Proceedings of the IEEE Conference on Virtual Reality*, pages 79–82, 2009.
3. Ronald Azuma. A survey of augmented reality. *Presence: Teleoperators and Virtual Environments*, 6(4):355–385, 1997.
4. Benjamin B. Bederson, Ben Shneiderman, and Martin Wattenberg. Ordered and quantum treemaps: Making effective use of 2d space to display hierarchies. *ACM Transactions on Graphics*, 21:833–854, 2002.
5. Blaine Bell, Steven Feiner, and Tobias Höllerer. View management for virtual and augmented reality. In *Proceedings of the ACM symposium on User interface software and technology*, pages 101–110, 2001.
6. Marcelo Bertalmio, Guillermo Sapiro, Vincent Caselles, and Coloma Ballester. Image inpainting. In *Proceedings of ACM SIGGRAPH*, pages 417–424, 2000.
7. David E. Breen, Ross T. Whitaker, Eric Rose, and Mihran Tuceryan. Interactive occlusion and automatic object placement for augmented reality. *Computer Graphics Forum*, 15(3):11–22, 1996.
8. Volkert Buchmann, Trond Nilsen, and Mark Billinghurst. Interaction with partially transparent hands and objects. In *Proceedings of the Australian User Interface Conference*, pages 17–2, 2005.
9. John F. Canny. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.
10. M. Sheelagh T. Carpendale, David J. Cowperthwaite, and F. David Fracchia. Extending distortion viewing from 2D to 3D. *IEEE Computer Graphics and Applications*, 17(4):42–51, 1997.
11. Ed H. Chi. A taxonomy of visualization techniques using the data state reference model. In *Proceedings of the IEEE Symposium on Information Vizualization*, pages 69–75, 2000.
12. Ben Close, John Donoghue, John Squires, Phillip De Bondi, Michael Morris, Wayne Piekarski, Bruce Thomas, Bruce Thomas, and Unisa Edu Au. ARQuake: An outdoor/indoor Augmented Reality first person application. In *Proceedings of the IEEE International Symposium on Wearable Computers*, pages 139–146, 2000.
13. Enylton Machado Coelho, Blair MacIntyre, and Simon J. Julier. Osgar: A scene graph with uncertain transformations. In *Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 6–15, 2004.
14. Franklin C. Crow. Shaded computer graphics in the entertainment industry. *Computer*, 11(3):11–22, 1978.

15. Stephen DiVerdi and Tobias Hollerer. Image-space correction of ar registration errors using graphics hardware. In *Proceedings of the IEEE conference on Virtual Reality*, pages 241–244, 2006.
16. Niklas Elmqvist and Philippas Tsigas. A taxonomy of 3d occlusion management for visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14:1095–1109, 2008.
17. Jan Fischer, Dirk Bartz, and W. Straßer. Enhanced Visual Realism by Incorporating Camera Image Effects. In *Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 205–208, 2006.
18. Anton Fuhrmann, Helwig Löffelmann, Dieter Schmalstieg, and Michael Gervautz. Collaborative visualization in augmented reality. *IEEE Computer Graphics and Applications*, 18:54–59, 1998.
19. Joseph Gabbard, Edward Swan, II, and Deborah Hix. The effects of text drawing styles, background textures, and natural lighting on text legibility in outdoor augmented reality. *Presence*, 15:16–32, 2006.
20. Joseph L. Gabbard, J. Edward Swan, II, Deborah Hix, Robert S. Schulman, John Lucas, and Divya Gupta. An empirical user-based study of text drawing styles and outdoor background textures for augmented reality. In *Proceedings of the IEEE Conference on Virtual Reality*, pages 11–18, 2005.
21. Eugen Bruce Goldstein. *Sensation and Perception*. Brooks/Cole, Pacific Grove, CA, 2001.
22. Amy A. Gooch and Bruce Gooch. *Non-Photorealistic Rendering*. AK Peters, Ltd., 2001.
23. Michael Haller, Stephan Drab, and Werner Hartmann. A real-time shadow approach for an augmented reality application using shadow volumes. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, pages 56–65, 2003.
24. Knut Hartmann, Timo Götzelmann, Kamran Ali, and Thomas Strothotte. Metrics for functional and aesthetic label layouts. In *Proceedings of International Symposium on Smart Graphics*, pages 115–126, 2005.
25. Jeffrey Heer and Maneesh Agrawala. Software design patterns for information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12:853–860, 2006.
26. Richard L. Holloway. Registration error analysis for augmented reality. *Presence*, 6(4):413–432, 1997.
27. Denis Kalkofen, Erick Mendez, and Dieter Schmalstieg. Interactive focus and context visualization for augmented reality. In *Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 191–200, 2007.
28. Denis Kalkofen, Erick Mendez, and Dieter Schmalstieg. Comprehensible visualization for augmented reality. *IEEE Transactions on Visualization and Computer Graphics*, 15(2):193–204, 2009.
29. Denis Kalkofen, Markus Tatzgern, and Dieter Schmalstieg. Explosion diagrams in augmented reality. In *Proceedings of the IEEE Conference on Virtual Reality*, pages 71–78, 2009.
30. Georg Klein and Tom Drummond. Sensor fusion and occlusion refinement for tablet-based AR. In *Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 38–47, 2004.
31. Georg Klein and David W. Murray. Simulating low-cost cameras for augmented reality compositing. *IEEE Transactions on Visualization and Computer Graphics*, 16:369–380, 2010.
32. Shoshichi Kobayashi and Katsumi Nomizu. *Foundations of Differential Geometry*. Wiley-Interscience, 1996.
33. Alex Leykin and Mihran Tuceryan. Determining text readability over textured backgrounds in augmented reality systems. In *Proceedings of the ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry*, pages 436–439, 2004.
34. Haibin Ling and David W. Jacobs. Shape classification using the inner-distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29:286–299, 2007.
35. Bunyo Okumura, Masayuki Kanbara, and Naokazu Yokoya. Augmented reality based on estimation of defocusing and motion blurring from captured images. In *Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 219–225, 2006.

36. Xiaofeng Ren and Jitendra Malik. Learning a Classification Model for Segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10–17, 2003.
37. Cindy M. Robertson, Blair MacIntyre, and Bruce N. Walker. An evaluation of graphical context as a means for ameliorating the effects of registration error. *IEEE Transactions on Visualization and Computer Graphics*, 15(2):179–192, 2009.
38. Christian Sandor, Andrew Cunningham, Arindam Dey, and Ville-Veikko Mattila. An augmented reality x-ray system based on visual saliency. In *Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 27–36, 2010.
39. Christian Sandor, Andrew Cunningham, Ulrich Eck, Donald Urquhart, Graeme Jarvis, Arindam Dey, Sebastien Barbier, Michael Marner, and Sang Rhee. Egocentric space-distorting visualizations for rapid environment exploration in mobile mixed reality. In *Proceedings of the IEEE Conference on Virtual Reality*, pages 47–50, 2010.
40. Kohei Tanaka, Y. Kishino, M. Miyamae, T. Terada, and S. Nishio. An information layout method for an optical see-through head mounted display focusing on the viewability. In *Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 139–142, 2008.
41. Anne M. Treisman and Garry Gelade. A feature-integration theory of attention. *Cognitive psychology*, 12(1):97–136, 1980.
42. Scott Vallance and P. Paul Calder. Context in 3D planar navigation. *Australian Computer Science Communications*, 23(5):93–99, 2001.
43. John Viega, M. Conway, G. Williams, and R. Pausch. 3d magic lenses. In *Proceedings of the ACM symposium on User interface software and technology*, pages 51–58, 1996.
44. Dirk Walther. *Interactions of visual attention and object recognition : computational modeling, algorithms, and psychophysics*. PhD thesis, California Institute of Technology, 2006.
45. Sean White and Steven Feiner. Sitelens: situated visualization techniques for urban site visits. In *Proceedings of the international conference on human factors in computing systems*, pages 1117–1120, 2009.
46. Sean White, Steven Feiner, and Jason Kopylec. Virtual vouchers: Prototyping a mobile augmented reality user interface for botanical species identification. In *Proceedings of the 3D User Interfaces*, pages 119–126, 2006.
47. Jason Wither and Tobias Höllerer. Pictorial depth cues for outdoor augmented reality. In *Proceedings of the IEEE International Symposium on Wearable Computers*, pages 92–99, 2005.
48. Shumin Zhai, William Buxton, and Paul Milgram. The partial-occlusion effect: utilizing semi-transparency in 3d human-computer interaction. *ACM Transactions on Computer-Human Interaction*, 3:254–284, 1996.
49. Stefanie Zollmann, Denis Kalkofen, Erick Mendez, and Gerhard Reitmayr. Image-based ghostings for single layer occlusions in augmented reality. In *Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 19–26, 2010.