Graz University of Technology

Institute for Computer Graphics and Vision

Dissertation

Illustrative X-Ray Visualization in Augmented Reality Environments

---

Denis Kalkofen

Graz, Austria, 2009

Thesis supervisor

Prof. Dr. Dieter Schmalstieg

To My Family

To see is to forget the name of the thing one sees.

Paul Valéry

Abstract

Human visual perception is dependent on the physical behavior of light. Depending on the characteristics of the material it strikes, light is absorbed, transmitted and/or reflected. Although it is impossible to change the physics of real world environments, Augmented Reality (AR) displays are able to create renderings which seem to suspend some of the rules which apply in the real world. They achieve this effect by extending real world imagery using artificial, computer-generated information which is registered in 3D space and related to objects and places in the real world.

By augmenting the virtual counterpart of a real world object, AR displays are even able to uncover hidden objects yielding the impression of seeing through formerly obscuring objects. Such x-ray vision is a powerful tool in exploring hidden structures along with related contextual real world information. However, it may also easily confuse its user if it is implemented carelessly.

In order to improve the comprehensibility of x-ray visualizations, this dissertation introduces illustrative visualization techniques to Augmented Reality applications. It presents the real-time integration of the three major illustrative x-ray visualization techniques, cutaways, ghostings and explosion diagrams into interactive and complex Augmented Reality environments. It shows their rendering under perfect conditions as well as visualization strategies to handle the difficulties posed by imperfect virtual scene description or poor real-world data. Moreover, this thesis presents the approach of multi-level focus and context visualization templates, which enable effective visual communication of illustrative renderings in complex AR environments.

Contents

Chapter 1

Introduction

Human visual perception is dependent on the physical behavior of light. Depending on the characteristics of the material it strikes, light is absorbed, transmitted and/or reflected. Although it is impossible to change the physics of real world environments, Augmented Reality (AR) displays are able to create renderings which seem to suspend some of the rules which apply in the real world. They achieve this effect by extending real world imagery using artificial, computer-generated information which is registered in 3D space and related to objects and places in the real world.

Since virtual information does not necessarily have to follow real world physical rules, AR displays can present a variety of realistic and unrealistic objects in a real world environment. This ranges from photo-realistic renderings of virtual objects in different real world surroundings [56] to fictional characters living in a real world game environment [26]. Moreover, by augmenting the virtual counterpart of a real world object, AR displays are even able to uncover hidden objects yielding the impression of seeing through formerly obscuring objects.

1.1 Problem Statement

Such x-ray visualization is a powerful tool in exploring hidden structures along with related contextual real world information. However, it may also easily confuse the user if it is implemented carelessly. This section reviews a set of x-ray visualizations which disturb user perception hindering their communication of spatial arrangements between hidden and occluding structures. While quite a number of factors can cause visual deficiencies of naïve x-ray visualizations in AR environments, we can identify three categories of prob-

lems in the relaying of spatial arrangements. The first group consists of problems which are caused by the fact that AR applications override real-world imagery with computer generated renderings. 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 (section 1.1.1).

To allow visual interaction between both types of 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 do this, both types of data have to be analyzed and compared in a common space. For example, to resolve 3D occlusions between real and virtual structures, the depth values of both have to be compared. A common method to implement depth comparisons between three-dimensional real and virtual objects is known as phantom rendering [13]. The algorithm transfers real world information into the coordinate system of the virtual data and is followed by an analysis of all depth values using an ordinary 3D computer graphics algorithm, e.g. z-buffering [24]. To transfer real-world depth values to the coordinate system used by the virtual data, virtual counterparts of the real objects are generated and referenced to their real-world locations in exactly the same way that the virtual objects have been referenced. However, since this data has to be generated and correctly registered in 3D, this method is prone to errors; this leads to the second group of possible problems. If the virtual data or its three-dimensional registration contains errors, their use might be counterproductive (section 1.1.2).

The last group of possible problems in rendering easily comprehensible x-ray visualizations in AR environments is related to the amount of extra information which appears after hidden structures have been revealed. For example, if multiple structures occlude an object of interest, the integration of all occluding elements may result in a cluttered display. This results from the fact that the projection of all occluding structures falls into the same two-dimensional presentation space (section 1.1.3). While problems originating from imperfect data or those caused by a poor integration of virtual data are common to AR applications, an information overflow is characteristic of the x-ray visualization itself and independent of its presentation in a real or virtual environment.

The following subsections discuss all the possible problems in detail, beginning with those originating from carelessly adding virtual information to real world environments followed by those caused by imperfect data and concluding with a discussion of the possible problems inherent in x-ray visualizations itself.

1.1.1 Data Integration

Augmented Reality displays extend viewer perception via the addition of computer-generated information. The AR system generates the final image by overriding parts of the real-world image using renderings of virtual objects. However, careless replacement of parts of the real world imagery can easily cause a number of problems. For example, if x-ray visualizations render hidden structures on top of occluding objects, the augmentation may obscure important real-world information. This problem is illustrated in the simulated surgery depicted in Figure 1.1. 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 pre-defined entry point (black crossing). 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.

Without considering the information which is to be removed, the augmentation may lead to a number of perceptual problems. Our cognitive system interprets a set of depth cues (see section 2.2) in order to pick up the spatial arrangements of 3D objects. However, when carelessly adding computer graphics to the real-world imagery, the AR system may override some of those depth cues. For example, Figure 1.1 not only shows that important landmarks are hidden by the computer graphics - it also demonstrates that spatial relationships between virtual and real data are lost after carelessly uncovering the hidden objects. This effect happens due to the lack of occlusion cues. If no partial occlusion

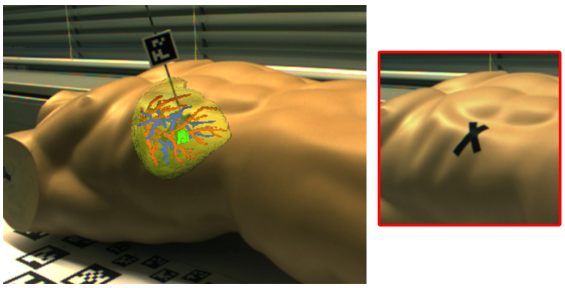
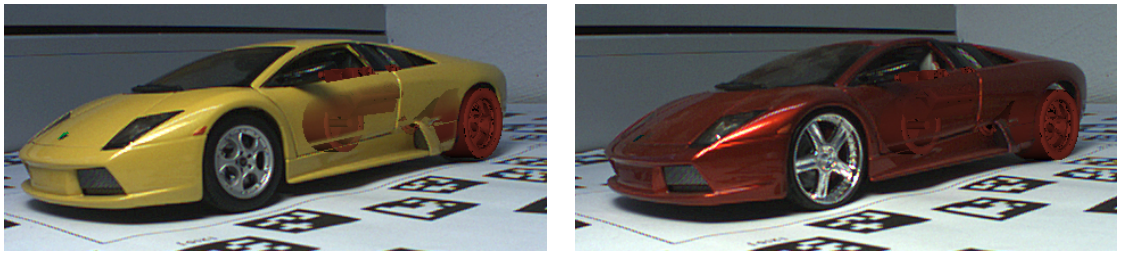


Figure 1.1: Spatial Integration of Virtual Data. Improper augmentations. In this example, careless augmentations of hidden structures cause two key problems: they override useful information (such as landmarks) and they lack depth cues. (Right) Original photo before augmentation, the black cross indicates the insertion point for the rfa-needle. (Left) Augmentation of the liver with its portal & hepatic vessel trees (red & blue) and a tumor (green).



(a) (b)

Figure 1.2: Visual interaction between real and virtual renderings in X-Ray Visualization. Both visualizations use the same parameter, however, the red overlay is hardly visible over the red car. The visualization parameter are more affective on the yellow car.

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 [133] in an x-ray visualization.

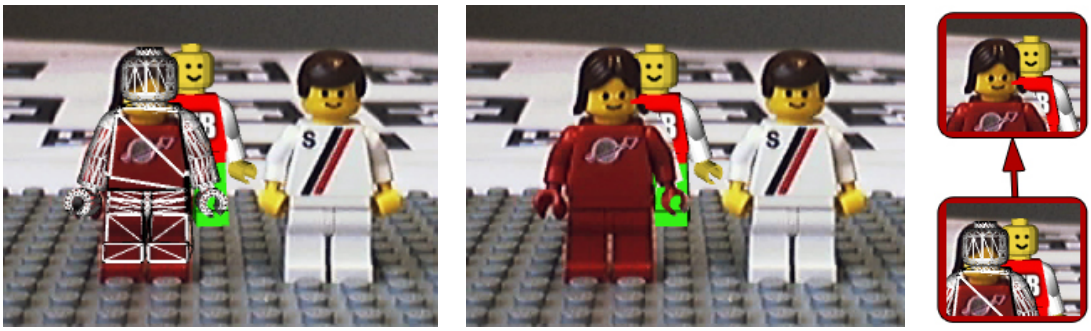
To successfully communicate the spatial relationships between 3D objects, more than just their occlusions must be communicated. The shapes of the objects themselves have to be visible to allow the viewer to understand the objects in the scene before their relationship can be perceived. If only an application's landmarks are preserved (such as the entry port in Figure 1.1), the final visualization may still suffer from a lack of shape cues. Even though depth cues can communicate the order of structures, if the preserved features do not line up to a single shape the obscured object may not be perceived properly and consequently no relationships between it and other objects can be communicated.

In addition to the problems caused by overriding (and thus removing) real world imagery, careless generation of x-ray 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. Consequently, to be able to generate understandable x-ray visualizations in AR environments, the appearance of the added 3D computer graphics has to fit into the real-world environment. Following this line of thought, both the coloring and the shading of the virtual and the real world objects have to be analyzed and adapted to correspond to each other.

Accessible x-ray visualizations consist of the uncovered hidden structure, preserved shape features and sometimes preserved landmarks. As these are the most important elements in communicating the spatial relationships in an x-ray visualization, they have to be easily distinguishable from their surroundings. For example, Figure 1.2(a) shows an x-ray visualization with a similar appearance of both the hidden and occluding structures. Even though shape and depth cues are preserved, it is difficult to make out the elements of the x-ray visualization and as a result the spatial relationships are difficult to perceive.

1.1.2 Imperfect Data

To handle the problems originating from carelessly overriding real world information or those which come from independent shading of the data, the virtual information has to be integrated with the real world environment instead of being independently rendered and simply overlaid. This means that to be able to comprehensively visualize hidden structures or to preserve the shape cues of real world objects, the AR system has to correctly identify real world structures first and then analyze the relationships between virtual and real data and subsequently adapt the output to rendering parameters. This strategy allows the presentation of virtual information relative to the identified real world information, enabling proper visual interplay between all types of data in the AR environment.



(a) (b) (c)

Figure 1.3: Occlusion Handling and Sources of Error

To identify real world structure, a virtual counterpart of the real object is commonly registered in real-world space. If this registration perfectly maps the virtual object to its real world counterpart, subsequent analysis can be completed using only the virtual counterpart. For example, to identify occlusions between virtual and real structures, we

have to compare the depth values of both types of objects per pixel in the screen space of the AR display. By using a registered virtual counterpart of a real 3D object, we are able to compare depth values in a common virtual space. For example, Figure 1.3 shows a rendering of a virtual Lego figure which is behind a real one. By using a virtual model of the real figure in front, the fragments of the virtual figure which are behind the real one have been suppressed and only those which are not occluded have been overlaid on top of the AR system's video feed.

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 1.3 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 while parts of the face of the real figure have been classified as background information causing the rendering of the virtual figure to falsely override them.

However, even though a virtual model perfectly mirrors its real world counterpart, in order to precisely match it, the virtual model has to be placed exactly where the real object is located. The location and orientation of the real world object can be computed using either direct vision-based tracking of structures that are visible in the video image or indirectly by using another type of tracking system (such as a tracking by an analysis of an induced magnetic field at the location of the real object). In case of direct visual tracking, often simple and easily detectable objects are added to the scene to reduce the computational complexity of the algorithm. In addition, an offset between the tracked object and its more easily detectable tracking target has to be specified to allow the transformation of the virtual object based on the information from the additional tracking target. While this strategy reduces the complexity of the tracking algorithm, it introduces another source of errors.

If the tracking data is not derived from the same image which is used for augmentation, a problem of synchronization between tracking, rendering and real world data may occur. Since all of the mentioned factors have to be estimated precisely to classify real world information, a visualization in AR environments will most likely never be free from errors. Consequently, the visualization techniques have to consider the existence of erroneous data and support error-friendly visualizations which are robust enough to communicate the spatial relationships even if they are based on erroneous or missing data.

1.1.3 Multiple Occlusions

If multiple objects occlude the object of interest, preservation of landmarks or shape cues may result in an overflow of information. In the worst case, all preserved shape cues and landmarks of all objects which lie in between the viewer and the object of interest add up to a complete obscuration of the object of interest. If multiple occlusions must be addressed, a choice must be made concerning which object's shape to preserve in order to communicate the visualization's intention best.

Figure 1.4 shows an example in which too many objects occlude the object of interest. The resulting display is cluttered and the object of interest (the engine and the back wheels) is hardly visible. Notice, that the previous sections address problems which are specific to x-ray visualizations in AR environments, this problem is independent from the presentation space of the x-ray visualizations and also occurs in virtual reality environments

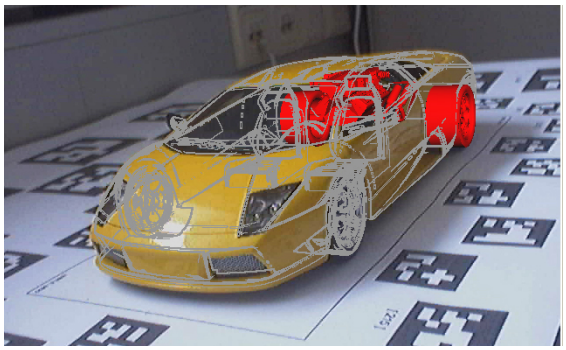


Figure 1.4: Multiple Occlusions

1.2 Contribution

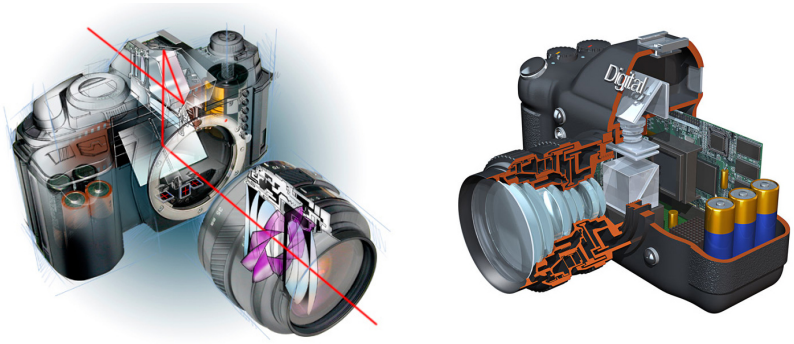
This thesis presents comprehensible x-ray visualizations in Augmented Reality environments. The key element to achieve this goal is the ability of the resulting visualization to communicate spatial relationships between hidden and occluding objects. As described in the previous section, we have to pay attention to a number of different problems influencing the intelligibility (see Table 1.1).

All of the problems outlined in Table 1.1. have been addressed in this thesis. The work utilizes presentation techniques which are inspired by artistic illustrations to handle shape and depth deficiencies (Figure 1.5). It furthermore demonstrates combinations of

(A) Overlay of virtual on real information	**(B) Multiple Occlusion**	**(C) Imperfect Data**
Landmarks and shape cues may get lost	Clutter due to an increased depth complexity	False classification due to erroneous data
Depth perception may become ambiguous		Incomplete classification due to incomplete data
Contrast and attention deficiencies		Misplaced overlay

Table 1.1: Main problems to create comprehensible x-ray visualizations in Augmented Reality environments, categorized by its origin

these techniques and shows when to safely use them to preserve real world landmarks. It discusses the capabilities of the visualizations and presents filter techniques to handle multiple occlusions in front of the object of interest. Since AR environments are dynamic and interactive, this thesis also presents a discussion of visualization strategies to handle erroneous data. The thesis is completed by a technique to control the visual combination of the real and virtual elements to ensure a comprehensible fusion in complex and dynamic environments.



(a) (b)

Figure 1.5: Illustrative x-ray visualization a) Ghost-, explosion- and b) cutaway presentations are able to simultaneously show hidden and occluding structures (Image courtesy of a)http://www.cutaway-illustration.com and b) www.digitalcamerareviews.us

The following summary lists the main contributions of this thesis:

- Integration of illustrative x-ray visualization techniques in AR views. Furthermore, this thesis discusses visualization parameters to comprehensively fuse visible real world and otherwise hidden virtual data.
- A general framework for stylized x-ray visualization in Augmented Reality.
- A discussion of the generation and presentation of x-ray visualizations in imperfect situations. This includes the demonstration of presentation parameters to handle erroneous data, a discussion of the pros and cons of renderings and presentations from registered phantom objects compared to those generated from 2d video data and visualization strategies to achieve comprehensible x-ray visualizations in those cases where the virtual representation of the environment is incomplete.
- The approach of Multi-Level Focus and Context Visualization Templates, which enable effective visual communication of illustrative renderings in complex AR environments.

While parts of this thesis are still unpublished, others contain previously published material presented at conferences, workshops and journals. The following list gives an overview of the papers and the collaborating researchers:

Kalkofen Denis, Tatzgern Markus, Schmalstieg Dieter, *Explosion Diagrams in Augmented Reality*, Proceedings of IEEE Virtual Reality, 2009

Kalkofen Denis, Mendez Erick, Schmalstieg Dieter, *Comprehensible Visualization for Augmented Reality*, IEEE Transactions on Visualization and Computer Graphics, 2009

Kalkofen Denis, Mendez Erick, Schmalstieg Dieter, *Interactive Focus and Context Visualization for Augmented Reality*, In Proceedings of 6th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR 2007), 2007

Kalkofen Denis, Reitinger Bernhard, Petter Risholm, Bornik Alexander, Beichel Reinhard, Schmalstieg Dieter, Eigil Samset, *Integrated Medical Workflow for Augmented Reality Applications*, International Workshop on Augmented environments for Medical Imaging and Computer-aided Surgery (AMI-ARCS), 2006

1.3 Organization

This thesis presents rendering techniques to handle the main problems in generating comprehensible x-ray visualizations in AR environments (see Table 1.1). It demonstrates the applicability of the three major x-ray visualization techniques in AR environments and it presents the approach of Multi-Level Focus and Context Visualization Templates.

Each of the x-ray visualization techniques are presented in a separate chapter. The problems presented in table 1.1 which originate from adding virtual information to real world environments (A) and those from presenting a set of formerly hidden information (B) are discussed in each of these chapters. Furthermore, each of these chapters presents rendering techniques utilizing virtual counterparts of real world occluding structures as well as rendering techniques which are able to cope without this information (Table 1.1-C2). This thesis is structured as follows:

Chapter 2 presents background information on the different fields that influence the creation of comprehensible x-ray visualizations in AR environments. This includes illustrative visualization, cognitive psychology and augmented reality.

Chapter 3 presents the rendering system developed in this thesis which allows the accurate integration of virtual structures into a real world environment. This chapter also presents Multi-Level Focus and Context Visualization Templates.

Chapter 4 presents ghost visualizations in AR environments as the first of three implemented illustrative visualization techniques. The chapter also presents rendering techniques for the use of image data as well as 3D objects as sources for an illustrative x-ray stylization in AR. It discuses the robustness of the illustrative x-ray visualization technique in AR environments and it proposes an error friendly visualization which is able to visually communicate the errors present in the application. The chapter concludes with a presentation of interactive information filters which are used to control the amount of shape cues presented in the case of multiple occluding objects.

Chapter 5 presents cutaway visualizations in AR environments as the second of three implemented illustrative visualization techniques. It shows how to use video as well as 3D CAD data as a source for stylization. It demonstrates the applicability of the error friendly visualization carried out in Chapter 4. This chapter closes with a

discussion of different cutaway styles used to control the amount of information if multiple occlusions hide the object of interest.

Chapter 6 presents explosion diagrams in AR environments as the third out of three implemented illustrative visualization techniques. Like the two previous chapters, this chapter presents different rendering techniques based on video as well as 3D CAD data as a source for stylization and it demonstrates the applicability of the Relative Rendering approach presented in Chapter 3 on Explosion Views. It furthermore shows the results of the error friendly visualizations carried out in chapter 4. This chapter closes with a discussion of different illustration styles, or to be more specific, of different explosion layouts used to control the amount of information in case multiple occlusion occur.

Chapter 7 discusses the techniques presented and closes the thesis with a list of possible directions for future work.

Chapter 2

Background

Content

Work from a number of different research areas must be combined in order to display comprehensible x-ray visualizations in an AR environment. First and foremost, an Augmented Reality system has to provide a presentation platform enabling the fusing of virtual content with real world imagery (2.1). Furthermore, the rendering has to attend to the fundamentals of depth and shape perception (2.2) in order to produce a useful presentation of occluding and hidden structure simultaneously. To effectively communicate both types of data, Focus and Context (F+C) visualization techniques have been applied in this

thesis. They are introduced in section 2.3. This chapter concludes with a presentation of the state of the art of a set of special F+C visualization techniques which are often used in x-ray illustrations (2.4).

2.1 Augmented Reality

The sensing of our environment is guided by the cognitive interpretation of the signals received from our sensory organs. These organs are stimulated by objects or events present in the environment. Since these stimuli can be replaced with a synthetic equivalent, virtual environments can be generated which simulate real world experiences.

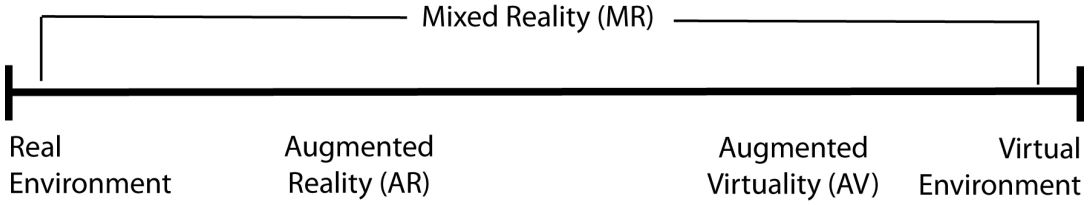
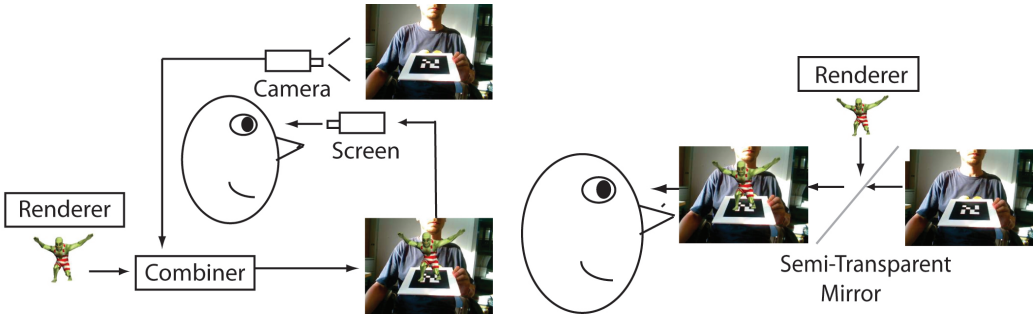


Figure 2.1: The Reality-Virtuality (RV) Continuum (Adapted from [94])

However as it is very difficult to precisely simulate any real world stimuli, Augmented Reality applications make use of those stimuli which are already present in the real world environment. They additionally enriched them with a certain amount of synthetic information (ideally just enough to overcome the limitations of the real world environment for a specific application). Nevertheless, virtual environments may capture real world stimuli in order to achieve a similar affect (e.g. by applying real world textures to virtual objects). Paul Milgram and his colleagues [94] called this type fusion Augmented Virtuality because of the incorporation with the virtual environment. However, both Augmented Reality and Augmented Virtuality present a mixture of real and synthetic stimuli. As Milgram et al. [94] outlined, they smoothly translate from one into the other via the ratio of virtual and real content. Based on this insight they created a taxonomy consisting of four types of environments, the Reality-Virtuality (RV) Continuum (Figure 2.1). Notice that no specific boundary exists between the types so sometimes a precise classification of an environment is not possible.

Since the Reality-Virtuality continuum defines an AR application only by the ratio of its content and the presentation space, Azuma and his colleagues extend this definition with two more requirements [4]. They claim that in addition to a mixture of real and virtual information, an AR application has to run in real time and its virtual objects have



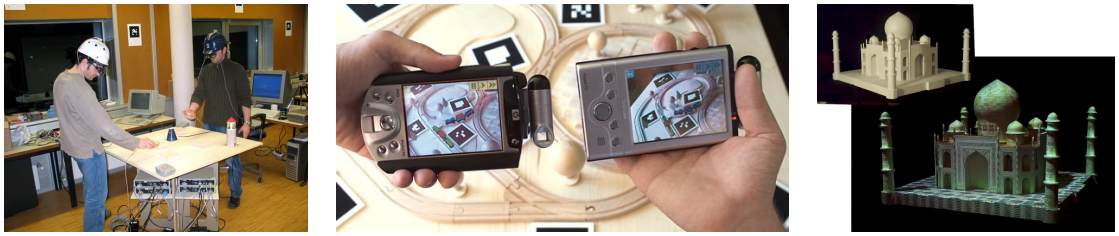
(a) (b)

Figure 2.2: Techniques for Combining Visual Information(a) Video See Through (b) Optical See Through

to be aligned (registered) with real world structures. Both of these extra requirements guarantee that the dynamics of real world environments remain after virtual renderings have been added. The real time capability of the application itself usually depends on the programmer and the hardware of the render system. Crucially, the renderings have to be well combined with the real world information. In order to both register the data and fuse virtual and real imagery in real time, special devices implementing a variety techniques are used by today's AR systems.

To combine visual information in real time, one of the following three techniques is usually used in AR display devices (Figure 2.2). Optical See Through devices project the current rendering of the virtual data onto a semi-transparent mirror. This special mirror allows the user to perceive the real world through it while at the same time it passes on the virtual content to the eyes of its user. In contrast, Video See Through devices capture the real world information with a video camera. Before the final result is presented to the user, the captured video will be blend with the rendering of the virtual content by the device. Direct Augmentations use projectors and the surfaces of the environment to present the virtual information directly in the 3D real world environment.

Notice that direct augmentation and optical see through add information to the user's normal vision. In contrast, video see through devices present a copy of the real world information; this has a number of disadvantages. First of all, the user will not see anything if there is a system failure (which can be very problematic, for example in surgical AR applications). In addition, an image coming from a video stream represents only an approximation of the real world information and may suffers from factors such as camera



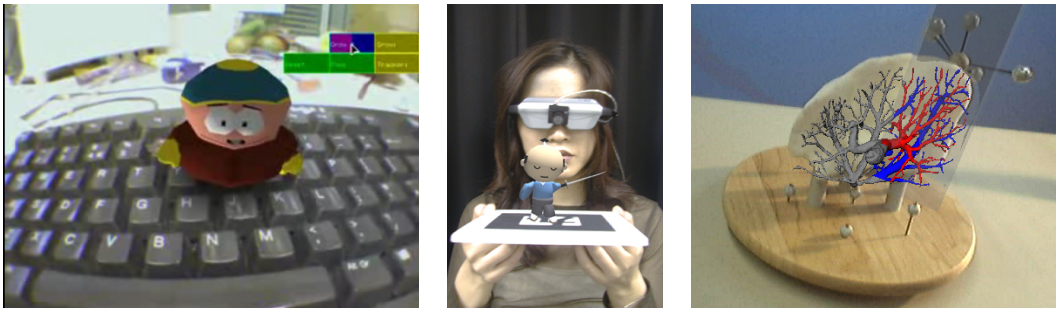
(a) (b) (c)

Figure 2.3: Display technologies used in Augmented Reality systems. (a) Head-mounted displays [4] (b) Handhelds [126] (c) Direct augmentation using projections of real word surfaces. The upper left corner shows a wooden object before projecting texture information [102].

resolution, lens distortion, camera noise, color modifications and so forth. On the other hand, using video see through devices allows one to more easily control the blending of virtual data into the real environment.

Display devises can also be distinguished by where they are installed in 3D space (see Figure 2.3). Displays worn on the head (Head Mounted Display or HMD) usually integrate video or optical see through technology into a helmet [20]. Small carried displays (or Handheld Displays) are equipped with a video camera and a 2D screen supporting video see through systems [126]. Nevertheless, both head mounted and handheld displays depend on additional hardware which is usually either heavy or uncomfortable to wear, or in case of handheld displays suffer from a small physical screen size. As an alternative, Spatial Displays may be used which are pre-installed in the 3D environment and support direct augmentation. They usually consist of either a set of stationary projectors [49] or an installation of a projector in combination with a semi-transparent mirror [102]. However, even though spatial displays allow one to perceiving the visual augmentations without dependence on a carried device, they are only applicable in prepared environments.

Azuma's second additional requirement for an AR application is a three dimensional registration of its virtual content relative to the real world objects. Therefore, AR systems estimate the position of either the virtual objects relative to the video camera or of both the video camera and the virtual objects relative to a common coordinate system (e.g. to also allow registered augmentations using optical see through devices). Up to now a number of different technologies have been developed to support the registration of 3D objects (Figure 2.4). For example, computer vision algorithms are able to compute the

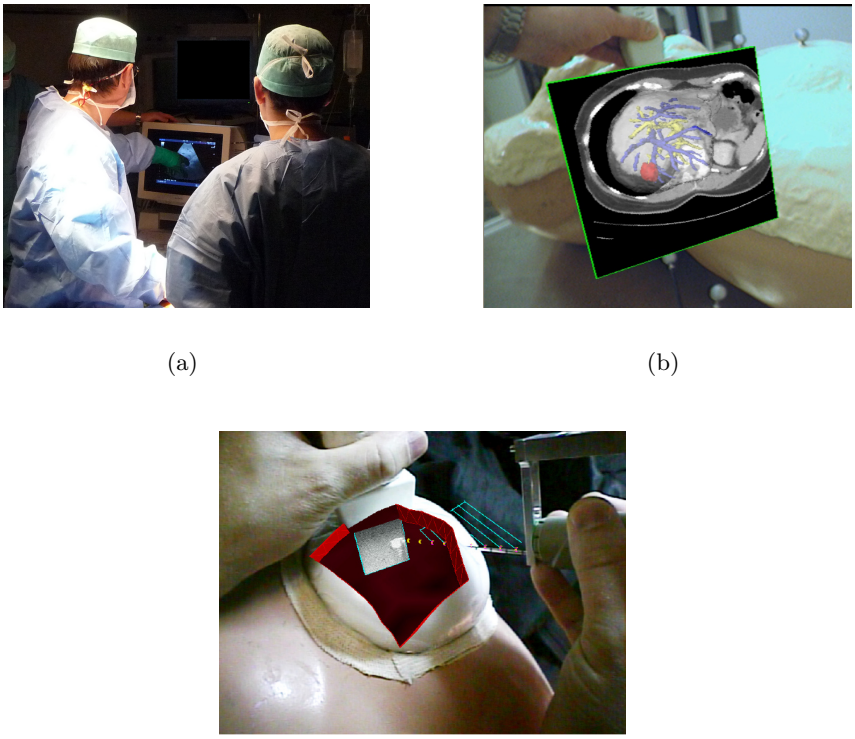


(a) (b) (c)

Figure 2.4: Examples of AR Tracking Technology (a) Natural feature tracking (NFT); [79] (b) Fiducial Tracking using ARToolkit marker; [75] (c) Tracking using retro-reflective markers [93].

position of a video camera directly from the images it delivers to the system [95]. To be able to compute a 3D position, the algorithms have to select feature points directly out of the images and analyze them. Since feature detection and its subsequent processing is a computationally expensive and often noisy operation, some other technologies can help by adding artificial landmarks into the 3D environment [74]. Those aids (called fiducials) add a specific stimuli to the envirorefnment which is easier and faster to detect and evaluate. A number of different types of stimuli have already been used as sources for fiducal tracking. For example, retro-reflective spheres allow for quick and precise visual identification (see (c) in Figure 2.4). Even audio sources [41] or magnetic fields (natural [47] as well as syntactically induced [101]) have been used). Nevertheless, the downside of adding fiducials to the environment is the artificial modification of the environment as well as the preparation required along with the need for specialized receivers.

Even though AR is still not commercially established, applications making use of AR technology can already be found in a number of fields. One of the main strengths of AR is its ability to overcome difficulties related to hand-eye coordination [72], specifically the problem that appears when operating somewhere other than where the operator's eyes look directly. Since AR displays are able to present registered virtual objects within real world environments, they are able to present the information exactly where the hands have to act. Figure 2.5 shows examples of this problem. While the doctor in (a) has to mentally transfer the data he sees to the real world location, the AR applications in (b) and (c) present the information directly where it ought to appear.



(a) (b)

(c)

Figure 2.5: Overcoming the problem of hand-eye coordination using Augmented Reality (a) Typical examination using sonography; (b) The AR display enables the user to see the data at the same location where his/her hands operate.(c) AR guided needle biopsies in which the ultrasound image is displayed in place and the biopsy needle is visualized using its virtual counterpart; [109].

2.2 Cognitive Perception

A simple overlay of hidden structure on top of the system's video feed can cause a number of cognitive problems. Figure 2.6 shows an example of a careless augmentation of the engine and the back wheels of a sports car. Even though the virtual objects are perfectly registered in 3D space, they seam to fly in front of the car. This section gives an overview of the processes involved in giving the impression of depth. Subsequent chapters of this thesis demonstrate the effect of retaining depth cues in the augmentation. However, since quite a number of fragments of the objects can retain the same information about depth, only a subset has to be preserved in order to communicate the message. To be able to retain the most effective cues, the second part of this section presents the Gestalt laws

Figure 2.6: A Problem of Depth PerceptionEven though the virtual objects are correctly registered, they are perceived of as being in front of instead of inside the car.

relating to perception.

2.2.1 Depth Perception

Our cognitive system takes approximately 15 to 20 different psychological stimuli into account in order to perceive spatial relationships between 3D objects [50]. These so called depth cues can be divided into the 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. The latter is caused by the feeling that occurs when the eyes converge and change shape in order to focus nearby objects. The feeling which is 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. Since 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. Figure 2.7 shows a photograph of a bridge which includes most of the following depth cues:

- 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. The sun in figure 2.7 shines from behind the bridge (relative to the position of the camera) which causes the shadow of an object in the environment to be cast on the objects which are in front of it.
- 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.

Figure 2.7: Static Depth Cues

Another powerful monocular source for depth judgments are dynamic depth cues. They appear if either the objects in the 3D environment or its observer move. Since further away objects seem to move slower than objects close by, our cognitive system translates the difference in speed into an approximation of distances between the objects.

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, binocular depth cues fail to approximate the distances between

objects which are far away because at a certain distance, all points appear to have very similar disparities values. To provide the user of an AR application with binocular depth cues, special stereoscopic display devices have to be used in order to deliver two different renderings of the virtual objects into each eye.

(a) (b) (c)

Figure 2.8: Importance of Occlusion Cues (a) Believable integrations of virtual objects can be achieved by simply overlaying their 3D rendering. (b) Even though a number of different depth cues exist, depth order is ambiguous and perception is wrong. (c) The same rendering with correct occlusion handling is able to communicate the order of real and virtual content.

Renderings of virtual objects, which are registered in 3D space, add a number of monocular or more specifically pictorial depth cues. In Figure 2.8 below, (a) shows an example of an augmentation of a virtual Lego figure which stands behind and to the right 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 distance of the virtual figure relative to the real ones. Since the virtual camera is also aligned with the real one, the depth cues from both presentations line up which delivers the fusion of virtual and real content. Even though a number of depth cues are missing (for example no shadow is cast and the setup of the virtual and the real world lighting do not match which results in an inconsistent shading of real and virtual objects), the virtual Lego figure in (a) is perceived to be in its correct position in 3D space. Since in AR the depth cues of 3D renderings align with those present in the environment, additional virtual objects can enrich the perception of the real environment. The fact that ordinary renderings of registered 3D objects add a number of depth cues to the environment was posited by J. Wither and his colleagues [130] as a means to increase the accuracy of depth judgments in real environments.

However as soon as occlusions between real and virtual objects appear, the depth cues

introduced by overlaying an ordinary rendering of registered virtual objects is no longer sufficient to produce believable augmentations (see (b) in Figure 2.8). Even if all other depth cues would be added to the AR display, the virtual object will be perceived as floating in front of the video image. However, by resolving occlusions between real and virtual objects, the rendering of a believable integration of virtual structure into the real world environment becomes possible (c).

2.2.2 Gestalt Laws

X-ray visualizations have to present both the hidden and occluding structure. To communicate their spatial relationships, the x-ray visualizations can either preserve important depth cues or spatially rearrange the objects in a way that our brain is able to approximate their spatial relationships even though they are not directly visible (see section 2.4). The latter are topics of research into perceptual organization. The area of Gestalt psychology originates from psychological research conducted in Germany during the 19th century (which explains the German name) and a number of different Gestalt laws have so far been identified. The following list introduces most of them while the subsequent chapters of this thesis demonstrate its application to x-ray visualization.

- Proximity: similar objects will be mentally combined with objects in their close proximity (see (a) below in Figure 2.9).
- Closure: items are mentally closed if they seem to complete a pattern (b)
- Similarity: objects are mentally grouped if they are similarly shaped or colored; i.e. in terms of similar values of hue, saturation or brightness (c).
- Common faith: objects moving in similar directions are perceived of as belonging together.
- Simplicity & good continuation: our cognitive system tends to organize items in a way that the resulting figures are more simple and regular rather than complex (d).
- Symmetry: symmetrical items are grouped together even if they located far away from each other (f).
- Figure-ground: to pay attention to parts of the scene, our cognitive system has to ignore the rest of it. In (e) below an example is shown in which we either pay attention to the white vase or to the black faces. Only one is perceived at a time

due to the lack of attention to important objects which are not the immediate focus of attention.

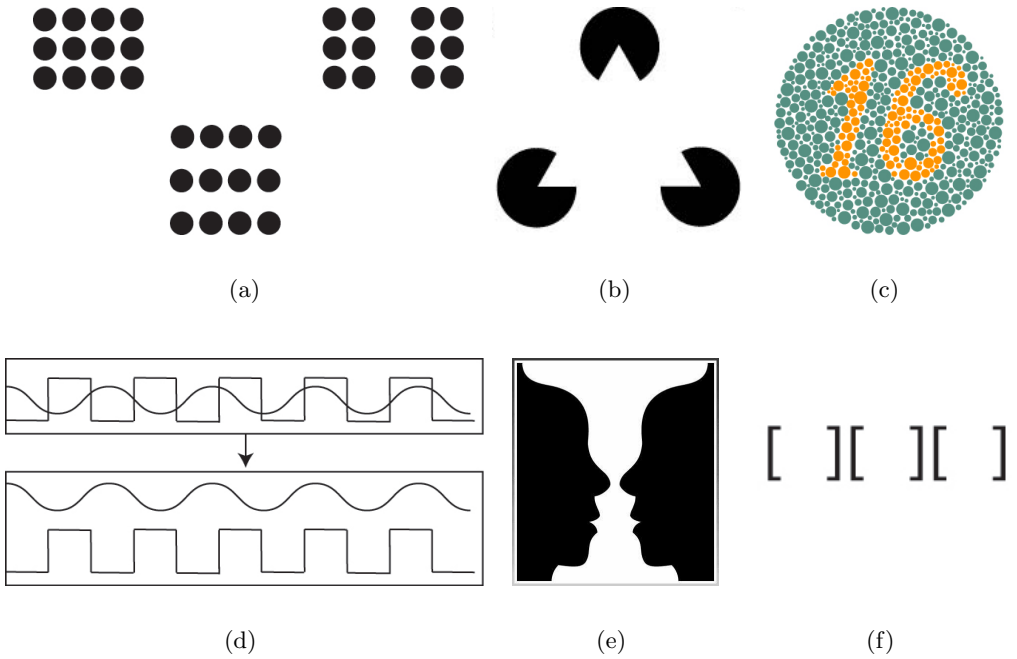


Figure 2.9: . Gestalt Laws (a) Proximity: similar items in a close proximity tend to form new units. (b) Closure: shapes are mentally closed if they indicate disruption by occlusion; both shapes, the occluding and the occluded one, are mentally reconstructed. (c) Similarity: the set of orange dots form the number 12 due to their similar color. (d) Good continuation: we perceive complex structure as an overlapping of simpler, smoothly continuing structures. (e) Figure-ground: the classical vase-face illusion in which our cognitive system either perceives a vase or two face but not both at the same time. (f) Symmetry: symmetric structures are mentally grouped together.

2.3 Focus and Context Visualization

To be able to communicate the spatial relationships in x-ray visualization, depth and shape cues have to be preserved. However, the main intention of x-ray visualization is still the presentation of hidden information, therefore the visualization should encode hidden structure in such a way that it is perceived of as being the most important one, while depth and shape cues of occluding structures represent the contextual information. Focus and Context (F+C) visualizations seem to be able to meet these requirements well. They allow the drawing of 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 unobtrusive way.

The creation of F+C visualizations can roughly be divided into two main steps: data classification and data visualization. The objective of the first step, data classification, is to identify the roles in the data, i.e., what information should be focus and what should be context. The second step is to render the categories in visually distinctive styles in order to guide user attention while presenting depth and shape cues of the contextual structures.

2.3.1 Classification

The relationship between focus and context data can either be spatially or knowledge driven. Spatially driven techniques retain or enhance information in close proximity to the focus object. While some applications select the focus' proximity directly in its presentation space [82], others first transform the data into a new space to allow for easier identification of the desired contextual information [122].

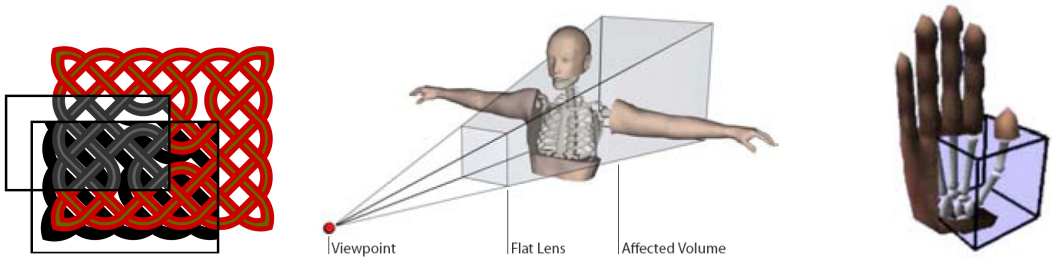
Knowledge-driven techniques do not depend on spatial proximities; instead, they use semantic interpretations of a given set of objects or scene parameters to distinguish between focus and context objects. Such interpretation can be directly controlled by the user and are based on task knowledge [46] or inferred from the contextual information associated with the objects in the scene [73]. Regardless whether the F+C classification is spatially driven or knowledge driven, a number of approaches exist for specifying how interactions can help to control the identification of the roles. For example, widget manipulation, direct selection or brushing and linking techniques [127] [22] allow one to interactively identifying portions of the data. If combined with a portable or desktop based Magic Lens tool [12], spatial classifications can easily be changed in any space.

Combinations of spatial and knowledge based classifications are often implemented using interactive tools. Hybrid classifications allow the interpreting of certain object semantics based on spatial arrangements. For example, S. Julier and his colleagues [73] identified the different roles of data depending on their 3D distance from the user. Furthermore the importance of an object is used to weight the current distance resulting in a hybrid classification. A similar approach is used by Ivan Viola in a volume rendering application. He identifies a voxel's visibility depending on its importance relative to the importance values of the voxel in front and behind [124]. This leads to a classification depending on the importance of the entire voxel per pixel.

The Magic Lens metaphor offers a tool for interactive selection in three different spaces

(figure 2.10). The classical 2D Magic Lens identifies certain parts of the data depending on its coverage in 2D space (a). Focus and context roles are assigned either to everything inside the 2D area, or for the smooth brushing technique [40], depend on such things as the distance to its center or its distance to the boundary of any other shape inside the area of the Magic Lens. Flat (or 2-1/2D) Magic Lenses operate similarly but in 3D space (b). They select a 3D volume based on a flat 2D mask which is positioned in 3D space. The 3D volume behind this mask, called the lens frustum, identifies the data which is to be processed by the Flat Magic Lens tool. While 2-1/2D Magic Lenses are not able to control the depth of the selection, 3D Magic Lenses have been designed to be used in 3D space independent of the depth of the viewing frustum. They allow one to use any 3D volumetric shape, but to reduce computational expenses often only convex shaped lenses are applied [108].

Magic Lenses facilitate the localized region where the F+C classification should take place. However, it is notable this does not mean that the boundary must be a particular display region. The Magic Lens metaphor may only identify certain parts of the data depending on its spatial location. If combined with a knowledge driven classificatory system, the data which is identified by the Magic Lens may only act as input to compute the actual F+C classification instead of directly identifying the data in the focus of attention. Also note that the interactive classification only works hand in hand with a real time visualization of its result.



(a) (b) (c)

Figure 2.10: Interactive F+C Classification Using the Magic Lenses Metaphor (a) 2D Magic Lens (Image courtesy of E. Bier [12].) (b)2-1/2D (Flat) Magic Lens (Image courtesy of J. Looser [90].) (c) 3D Magic Lens (Image courtesy of Viega.)

2.3.2 Presentation

Focus and Context visualizations demand the visual discrimination of objects depending on whether they are marked as focus or context. In addition, F+C visualizations have to encode different degrees of importance in order to communicate the prominence of the focus element relative to its contextual information. This is usually implemented by visually directing the user's attention to the object of interest while presenting its contextual information in a less salient style.

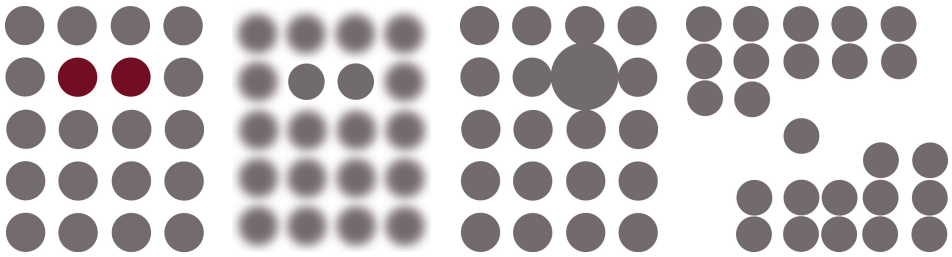
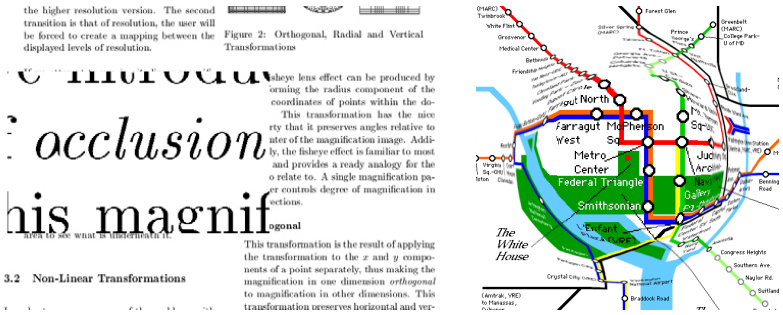


Figure 2.11: Sources of attention. (a) Saturation (b) Blur (c) Size (d) Objects per Group

Psychological research has identified a number of features which can influence the mental process of visual search [127]. Figure 2.11 demonstrates the effect of using blur, size, saturation and spatial grouping of objects to guide the user's attention (see [131] for a more complete list of sources of visual attention). F+C visualizations can be seen as an application which exploits the results of research into the properties of visual search. For example, F+C visualization may use the feature 'size' or 'saturation' as a guide for attention (see section 2.3.2.2) only since both implicitly communicate the F+C roles. This chapter shows implementations of different features of visual search in F+C visualizations starting with those using the features size and spatial distortion followed by those which work without spatial modifications. While this chapter presents a general overview of F+C visualization, section 2.4 summarizes the discussion on implementations of attributes known from research on visual search to communicate the roles in x-ray visualization.

2.3.2.1 Spatial Distortion

From psychological research we know about the effectiveness of the features 'size' (see (c) above) and 'number of objects per group' (d) [123] in directing user attention. F+C visualizations often exploit both features by spatially distorting either the presentation space or the objects themselves [85].



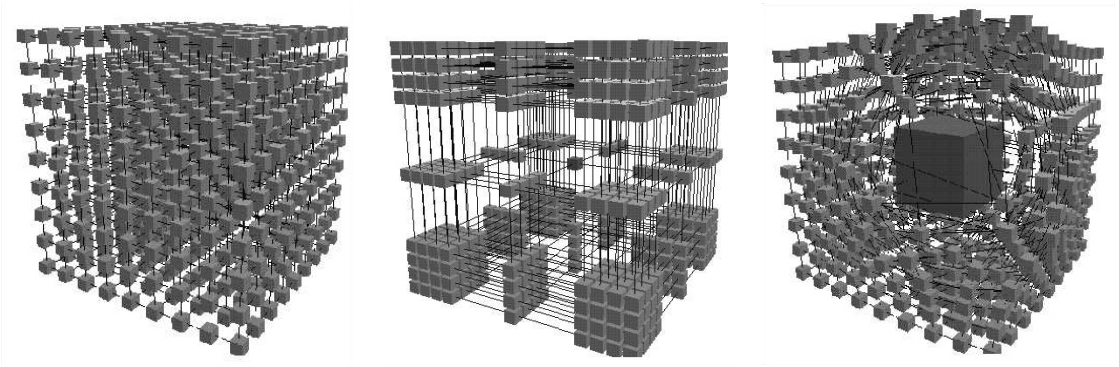
(a) (b)

Figure 2.12: F+C Visualization using Spatial Distortion (a) Binary magnification results in unconnected Focus and Context data (b) Using smooth degree-of-interest functions enable the mental connection of Focus and Context structures (images courtesy of T. Keahey [76])

A simple magnification of the presentation space often results in unconnected items (see Figure 2.12(b)). This problem is caused by the magnification of the objects of interest in their original position which overrides none magnified contextual information. Even if the contextual information is preserved by moving them out of the area of the magnification, their rendering will be difficult to mentally reconnect with the focus and context elements. Thus, the magnification of an object of interest is often implemented in combination with a spatial distortion of contextual information in its close proximity. In figure 2.12, (b) shows an example which uses a fisheye presentation technique. Rather than only magnifying the object of interest, the presentation uses a degree-of-interest (DOI) function which smoothly classifies the data to preserve contextual information which is already visually connected to the focus elements. This example shows that to comprehensibly magnify parts of rather continuous data (like the content of a 2D map), a more smooth classification has to be used (if a single presentation is required).

However, binary F+C classifications may comprehensibly presented if the data is rather discrete(e.g. using the features 'size' and 'number of objects per group'). Figure 2.13

shows an application of spatial distortions using a binary F+C classification only. While (b) exploits the attentive character of spatially isolating an item, (c) magnifies the object of interest in combination with a rearrangement of the items in its context. In chapter six, "Exploded Views", spatial rearrangements of real world objects to communicate the roles of the data in x-ray visualizations, are presented.



(a) (b) (c)

Figure 2.13: F+C visualization by magnification and rearranging 3D objects. (a) A set of 3D cubes before F+C visualization was applied. (b) Isolation of the focus element can comprehensibly encode the F+C roles. (c) Magnification of the object of interest only in combination with a rearrangement of contextual data. (Images courtesy of S.Carpendale [23])

2.3.2.2 In-Place Encoding

If spatial distortion of the objects in the environment is either impossible or due to the application requirements are undesired, other visual communicators have to be used to draw user attention. By changing visual properties only, visual encodings of the F+C roles do not make use of any spatial alteration (see figure 2.3.2.2). Instead, color properties (like saturation, brightness or hue [39]) can be directly modulated depending on the F+C classification, or more indirectly based on other properties in their proximity (for example by exploiting the gradient [82]). An even more indirect modulation of in-place encodings was shown by Kim et al. [78] through using the saliency of the image before modulation; this method was applied to control the amount of visual adjustments.

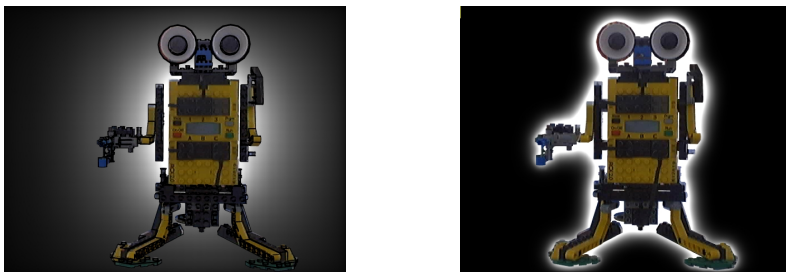
Another often used strategy to guide the user's attention without introducing distortions is to use additional visual communicators like a 2D frame (see (c) in figure 2.3.2.2).

(a) (b) (c)

Figure 2.14: In-Place F+C Visualization The F+C roles are communicated by (a) blurring contextual information [82], (b) de-saturating contextual information, or (c) facial detection drawing attention to the faces in the picture by framing them. (Images courtesy of Nikon.)

Schwertfeger et al. recently exploited a framing strategy in an industrial AR guidance system [112]. Like many others, the system uses the framing as a 2D rectangular overlay (similar to (c) below). However, other strategies may use a 3D outline of the box enclosing the object of interest or a halo visualization.

A halo emphasis has been used in a wide variety in virtual renderings, ranging from a spot of light with decreasing intensity as the distance to the center increases (see (a) in figure 2.3.2.2) to those which smoothly follow the outline of the object of interest (b). As (a) demonstrates, widely spread halos may cover a large amount of information in the background by smoothly fading from the object of interest to contextual information.



(a) (b)

Figure 2.15: Non Rectangular Framing Techniques. (a) Highlighting using a wide spread halo emphasis. (b) Emphasis using a halo which approximately follows the object's contour.

This makes them often impractical in AR if the object of interest is, for example, already covering a high amount of display space. However, rectangular framings (figure 2.14(c)) are the most obtrusive which is why they can easily result in cluttered displays.

F. Cole and colleagues [29] present a combination of different in place encodings to direct a user's attention to a selected area (Figure 2.16). This example uses a smooth DOI function in 2D image space which makes use of blur and fog rendering to suppress the conspicuousness of contextual information. In addition the visualization modifies the saturation in order to decrease focus towards context information as it uses a less obtrusive line stylization.

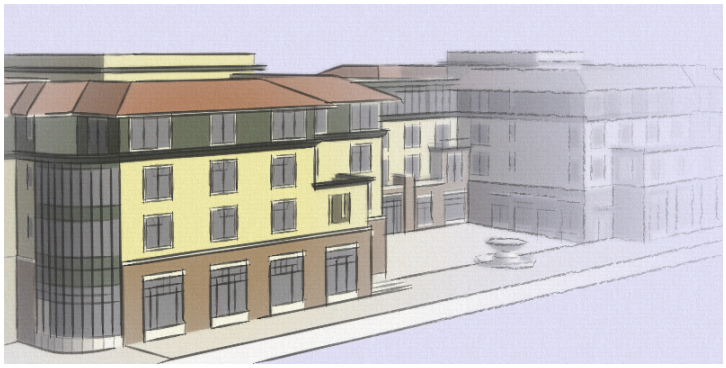
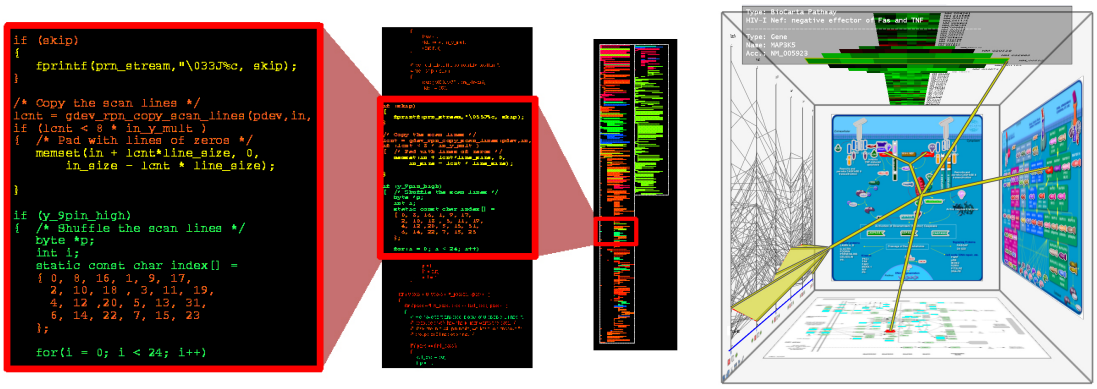


Figure 2.16: Combination of styles to direct the user's attention to the area of interest. Contextual information is de-emphasized by applying blur, fog, de-saturation and different line stylization. (Image courtesy of F. Cole and colleagues [29])

Since visual encodings in place do not modify the structure itself, they comply with the nature of real world presentations. Chapter 3.2 demonstrates how a combination of in-place F+C presentations may be used in AR in order to guide user attention to the hidden element in x-ray visualizations.

2.3.2.3 Multiple Windows

If an application requires an overview of data in combination with a detailed presentation of some selected parts, spatial distortions can be applied as demonstrated previously in this chapter. Such spatial distortions require smooth DOI functions to visually connect focus and context elements (see (b) in figure 2.12). However, if the magnification becomes very large (thus requiring a huge amount of presentation space) the visualization of contextual information becomes difficult if one of the previously introduced techniques is being applied.



(a) (b)

Figure 2.17: F+C Visualization with Multiple Windows. (a) A 2D layout. More detail is presented in the windows to the right (Image courtesy of Thomas Ball [6]) (b) A 3D layout. This setup trades distortion artifacts with spatial proximity, which enables less interfering linking (image courtesy of M. Streit & A. Lex [116])

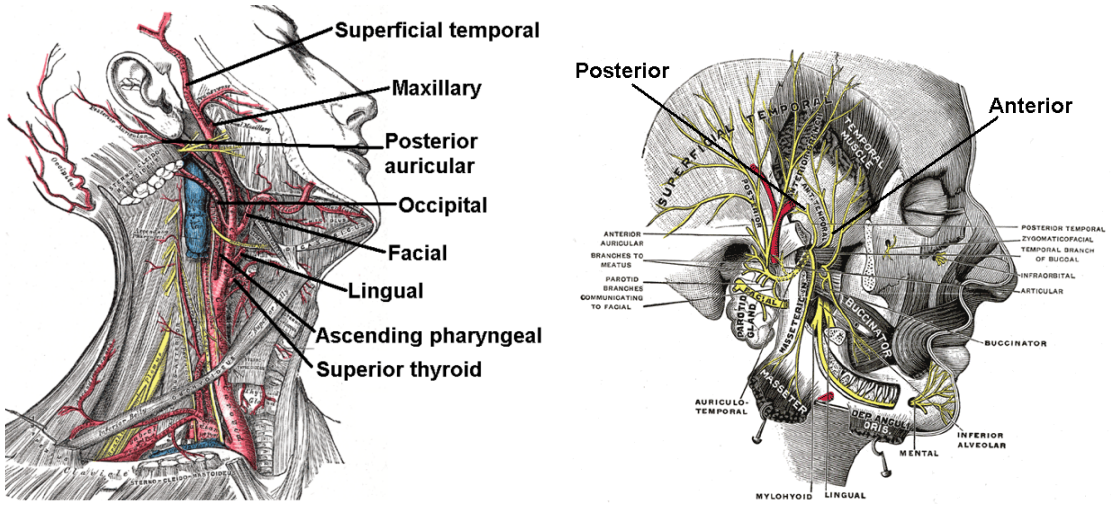
As a remedy, F+C presentations can distribute the data over multiple windows. Such presentations, where the data is visualized in multiple windows, enable one to optimize the available presentation space (figure 2.17(a)). However, an optimization of the presentation space by using multiple windows comes at the price of visually disconnecting the data. While spatial proximity can no longer be used to communicate the relationship between focus and corresponding context elements, multiple view presentations have to use other visual communicators to bring out their connection. Even though a number of visual attributes can transport this message, visual linking between the data elements is most often realized using lines or other 2D shapes (as demonstrated in figures 2.17).

Depending on the content, either 2D or 3D window configurations may be used. Figure 2.17(a) shows a 2D setup with three windows which present the source code of an application. However, if visually linking between items in different windows is desired (instead of linking the window itself) the resulting visualization may suffer from a high amount of clutter in 2D window placement. In comparison, windows in 3D enable one to reduce clutter by reducing the amount of information which a link covers [116] by placing objects closer to each other. To allow to understand the necessary rearrangements the elements are being perspectively distorted suggesting a 3d plane. Figure 2.17(b) demonstrates a configuration in 3D which presents the data in a way that complies with perspective distortions. Notice, including distortions may reduces the readability of the items.

Even though Elmquist's taxonomy of techniques to resolve occlusions [43] includes multiple view visualizations, and even though applications in AR exist which demonstrate its usefulness [129], this thesis considers single view techniques exclusively. Nevertheless, multiple view strategies for x-ray visualizations are to be considered in future work

2.4 Illustrative X-Ray Visualization

Graphics may be used for a number of reasons. For example a photographer tries to share a moment with the viewer of a photo while a pie chart is usually intended to present some kind of distribution. Of the various motivations, illustrations aim to communicate a specific meaning, and they do this very well. Since meaning is often better conveyed using an abstraction or distortion of reality, illustrations often do not even try to perfectly mimic real world conditions. Instead, they trade reality for comprehensibility. Consequently, illustrations include many kinds of Focus and Context techniques in the presentation of information wherein the goal is to emphasize aspects which are needed to communicate their message while at the same time reducing the visual impact of things which are less



(a) (b)

Figure 2.18: Illustrative Visualization of Human Anatomy Facial arteries (a) and facial nerves (b) are presented along with contextual information of the head. While the nerves and arteries are emphasized, the contextual information is presented in a less dominant way using only hatching and line rendering to preserve shape and shading. (Image courtesy of Gray's "Anatomy of the Human Body" [55])

important. Sometimes distortion techniques are applied, multiple windows are used or or in-place visual encodings use different styles on different structures (figure 2.18).

To stylize their data, illustrations often include Non-Photorealistic Rendering (NPR) techniques. Notice that while research on NPR looks into many kinds of rendering techniques to mimic artistic drawings, illustrative renderings only exploit a subset of them. They apply the methods which can "convey meaning" [117] and those which "clarify relationships between language and pictures" [117], for example using algorithms to automatically place labels [59] or automatically generate page layouts [54] [2].

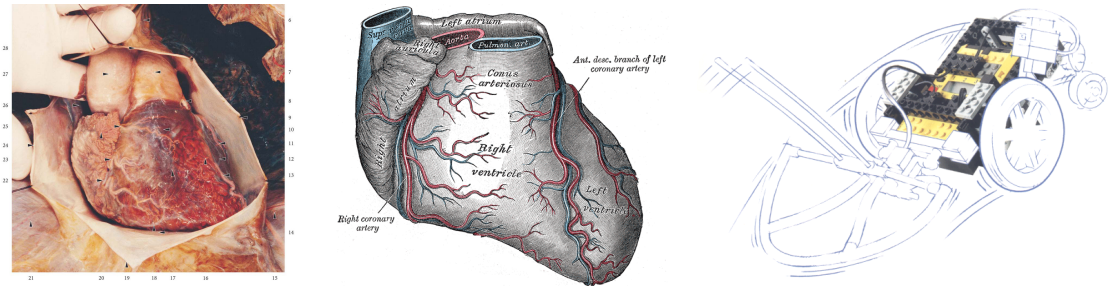
NPR offers a wide range of abstraction techniques. If high abstraction is required, illustrations often use line renderings to convey shape information only (see (a) in figure 2.19). Line renderings are very effective as they use only very sparse graphics [111] [53]. However, if more detail is required, illustrations commonly apply hatching (b) or stippling techniques [63] to abstract the shading of an object. Stippling, hatching and line renderings usually do not use colors. However, if an abstraction of shade information is desired the illustration may exploit a cartoon shader (c) which reduces the amount of shades applied to simplify the rendering [84]. In addition to these abstraction techniques, Gooch noticed that colorized technical illustrations can apply some kind of tone shading to better communicate depth arrangements [51]. The tone of an illustrated object is

(a) (b) (c)

Figure 2.19: Non-Photorealistic Rendering (NPR) techniques provide the tools to abstract information in illustrative visualizations. (a) Line drawings abstract shape presentation using a minimal set of graphical items. (Image courtesy of C. DeCarlo [34].) (b) Hatchings and cross hatching abstract shading by varying its density. Dark areas use dense pattern or cross hatchings while light regions apply sparse hatchings. (Image courtesy of E. Praun [99].) (c) Cartoon shading reduces the number of different shades. (Image courtesy of A. Lake [84].)

often altered depending on the intensity of its shading. Since cool colors are perceived of as being further away, technical illustrations often vary tone from cool to warm colors instead of shifting from light to dark.

In contrast to non-photorealistic renderings, real world objects present texture and shape in full detail as well as in correct perspective and scale. In figure 2.20 the images (a) and (b) offer a comparison of photo-realistic and illustrative presentation techniques. The images are both taken from anatomy texts so it can be assumed that both have the same intention (they try to communicate anatomical structures of the human heart). However, even though the illustrative rendering seems to be more effective in communicating the information, medical students usually have to dissect organs and muscles as part of their training. The amount of detail and the realistic nature of a photography makes the presentation more believable.



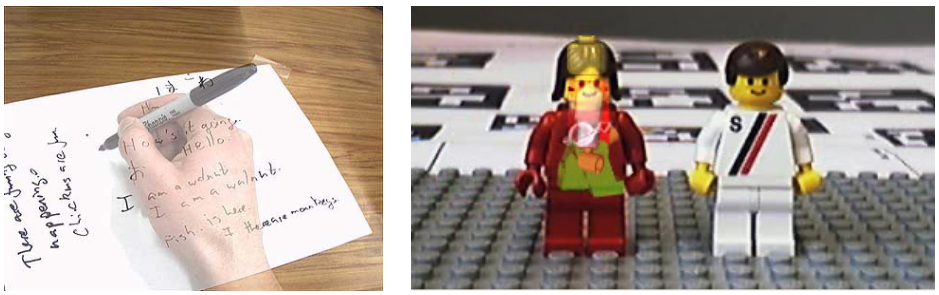
(a) (b) (c)

Figure 2.20: Photorealistic versus illustrative presentations of the anatomy of the human heart.(Image (a) courtesy of Gray's Anatomy of the Human Body [55], image (b) courtesy of Walter Thiel Photographischer Atlas der Anatomie [119]) The illustration in (c) uses a photorealistic presentation of the focus element and a non-photorealistic rendering of contextual information. (Image taken from construction manual of the Lego Mindstorm edition [30])

This thesis combines both presentation techniques by rendering illustrative visualizations within AR environments, taking into account real world information while applying emphasis techniques from illustrations. In particular this thesis exploits illustrative techniques to present hidden structure along with their occluding contextual information. The following sections of this chapter present an overview of illustrative x-ray visualization techniques and the state of the art of their adaptation to computer graphics.

2.4.1 Ghosting

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 2.21 (b) demonstrates the presence of clutter after blending the occluding structure uniformly with hidden objects. Even though very uniformly colored occluders covering high contrastive structures (as shown in (a) of figure 2.21) may allow one to perceive both types of data, the experiments of V. Buchmann et al. [17] showed that spatial relationships are lost if transparency is uniformly altered. The user perceives the hidden text as an overlay on top of the hand if transparency is too high.



(a) (b)

Figure 2.21: Uniform Transparency Modulation (a) Uniformly colored occluders allow one to see through them. However Volker Buchmann and his colleges noticed that the hidden text is perceived of as being on top of the occluder. (Image courtesy V. Buchmann [17].) (b) More complex color distribution result in an ambiguous presentation if a uniform transparency modulation is applied.

Since uniform modulations of transparency create a number of perceptual problems, illustrative x-ray visualizations vary transparency values non-uniformly over the object (figure 2.22). This results in a so called ghost presentation of the occluder. 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 A. Crow proposes a function of the cosine [33] of the angle between a normal surface point and the current viewing vector. To simulate different transparent media, Crow additionally uses the cosine to a modifiable power which is can vary the drop-off function of his transparency modulation. 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 the normal vector of

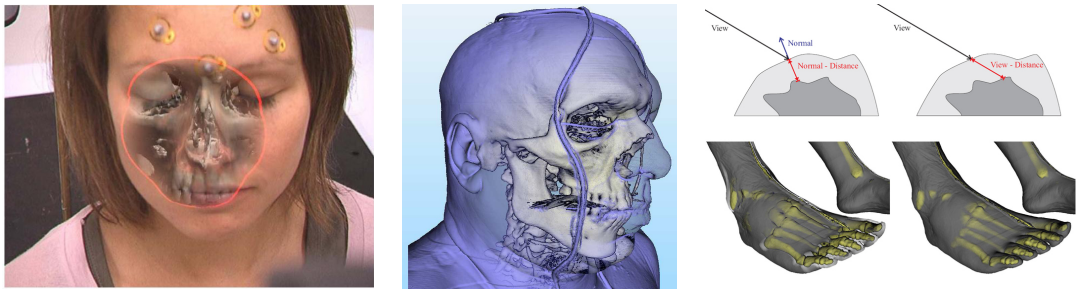
(a) (b)

Figure 2.22: Two Types of Ghosting. (a) Ghostings of hidden objects shine through occluding structures (Image courtesy of Albert Maier, www.illustrationz.co.nz)(b) Occluding structures use a ghost presentation to uncover the inner parts of the sports car (Image courtesy of Kevin Hulsey, www.khulsey.com)

this point [52]. Even though he didn't explicitly points it out, Crow's approach allows for transparency decrease close to the silhouette of the occluding structure.

A more direct approach to compute a ghost representation which depends on the silhouette of the object is presented by Diepstraten et al. [36]. First the silhouette is computed by comparing the orientation of adjacent faces (which is similar to Apple's algorithm [3]) followed by a computation of the distance in 2D between vertices and the detected silhouette. To encode different material, the transparency is encoded by weighting the distances.

Non-uniformly modulated transparency presentations preserve certain parts of the object while other parts are completely removed. In Crow's approach the ghost presentation preserves the silhouette of the occluding structures. However, other line renderings are also able to convey shape information very effectively. For example, crease lines or suggestive contours [35] have been proven to be good shape communicators [121]. It seems natural that illustrators and computer graphics researchers exploited these types of lines to generate ghost representations. For example, V. Interrante demonstrated the effectiveness of preserving crease lines in transparent visualizations [66]. In addition to a uniform transparency modulation of de-saturated occluding structure, she emphasizes valley and ridge lines by rendering them fully opaque using an increased saturation. C. Tietjen uses silhouette lines to present context in an otherwise volume rendered image [120]. Krueger and his colleagues use a curvature estimation in 2D to vary transparency values [83]. Moreover, instead of a discrete emphasis of crease lines, Krueger's technique enables one



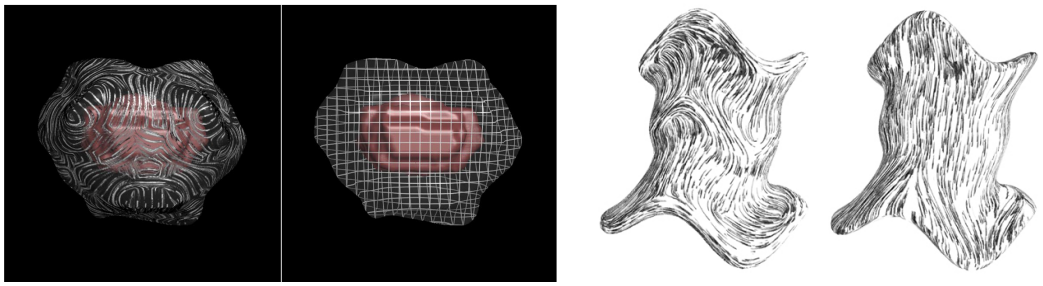
(a) (b) (c)

Figure 2.23: Different Strategies for Modulating Transparency (a) Curvature dependent transparency modulation using a user controlled 2D mask in AR. (Image courtesy of C. Bichlmeier [11].) (b) Curvature driven values opacity modulation in volume rendering. (c) Distance dependent transparency modulations. On the left distance is measured along the normal vector of the occluding surface point and on the right distance is defined along the viewing path. (Image courtesy of A. Krueger [83].)

to smoothly vary transparency values. In addition he used the relationship between occluding and hidden structure to control transparency modulations. For example, figure 2.23(c) shows a transparency modulation using the distance between the occluding and hidden structure. Krueger also demonstrated the effect of interactively weighting the transparency values (computed from curvature or structure relations) using a 2D Magic Lens. Transparency values at the center of the Magic Lens are increased while those close to its boarder either remain unchanged or are increased (figure 2.23(b)).

While Krueger combined creases (due to their curvature approximation) and a user controllable mask, Stefan Bruckner and his colleges combine shape and view dependent information. They evaluate shade information, which is previously computed using the Blinn-Phong mode, to subsequently modulate transparency. Since lighting can be easily altered, a shade based transparency modulation allows one to preserve silhouette and crease lines in an interactive manner. To also mimic the behavior of a clipping plane, the distance between the viewer and the model is encoded in Brucker's system. Interaction is accomplished by changing the distance between the camera and the object itself.

C. Bichlmeir and his colleges applied a similar system to the one presented by Krueger to an AR environment [11]. It modulates the transparency of previously classified video information with respect to a mixture of curvature and a user controllable 2D Magic Lens. Their classification of video data is done using a three dimensional registration of the virtual objects relative to its real world counterpart (figure 2.23(a)).



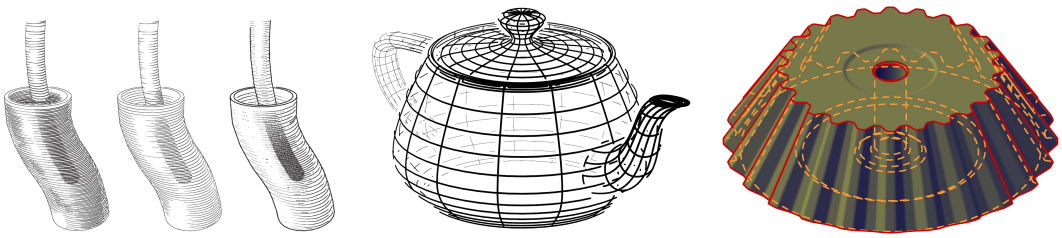
(a) (b)

Figure 2.24: . Ghost Presentations Using Texture to Communicate Shape (a) Left: stroke direction follows a principal curvature. Right: "solid grid" generated by cross-section (b) Left: texture orientation follows the direction of a principal curvature. Right: texture orientation follows a constant up direction. (Images courtesy of V. Interrante [67] and S. Kim [77])

Even though a modulation of transparency values based on certain features is able to preserve the shape information of some objects, V. Interrante and E. Hodges noticed that not all shapes present characteristic shape features which are worth being preserved [68] [63]. Instead smooth shapes, for example a sphere or a cylinder, may not be correctly preserved if high transparency values cause very sparse ghost presentations. Therefore, instead of modulating transparency depending on a certain set of features, V. Interrante and S. Kim experimented with textures which introduce feature which are able to convey shape information [77]. Inspired by hatchings which abstract shading (see (b) in figure 2.19), they evaluated the perceptual applicability of directional stroke textures to convey both shape and depth information simultaneously (figure 2.24). They could prove that strokes aligning with a principal direction are able to convey shape information better than strokes aligned in a constant direction (b).

Nonetheless, Interrante could not measure a statistically significant difference between a constant pattern and curvature directed strokes for the communication of depth values in x-ray visualizations. However, since the difference seems to be very obvious, she attributes the rejection of her hypothesis to the design of the experiment.

While V. Interrante used hatching to present the occluder covering a smoothly shaded hidden object, J. Hamel and S. Schlechtweg [57] incorporated transparent surfaces using hatching on hidden and occluding structure. Inspired by an illustration made by G. P. Hodges, they darken the hidden object where it enters the transparent occluder while they



(a) (b) (c)

Figure 2.25: Communicating Depths via Line Stylizations (a) To allow hidden structure to shine through, hatching of occluding structure is modified by altering the thickness of hatch lines, the amount of hatch lines and the style. (Image courtesy J. Hamel [57].) (b) Line halos communicate depth arrangement in line drawings. In addition, thickness and brightness of hidden lines decreases with increasing distance. (Image courtesy G. Elber [42].) (c) By convention, hidden lines are often dashed. (Image courtesy T. Isenberg [70].)

lighten the hidden structure where the edges at the entry point of the occluder overlap. In addition, they alter the hatching of the occluder where hidden structure "shines through" by darking the texture with increasing distance to the previously lightened edges. The occluder's hatching is darkened by either changing the thickness of its lines, by modifying their density or by a combination with a set of stipplings (see (a) in figure 2.25, left to right).

A similar technique is suggested by E. Hodges if applications require one to present occluding structure as mostly visible. She proposes the generation of ghost presentations of hidden structure by lightening their features so that hidden structures "shine through". The appearance changes only where hidden and occluding structure overlap in 2D. Figure 2.22 (a) shows an example application of this technique in a hand made illustration by A. Maier (www.illustrationz.co.nz). Notice, Maier also lightens occluding features to emphasize their shape information.

Both hidden and occluding lines have also been shown to be useful for architectural visualizations [97] using a blueprint rendering technique. The blueprint is rendered by blending all possible edges into a single rendering. Nienhaus detects the edges from a set of 2D layers which are generated by using the depth peeling [44] technique.

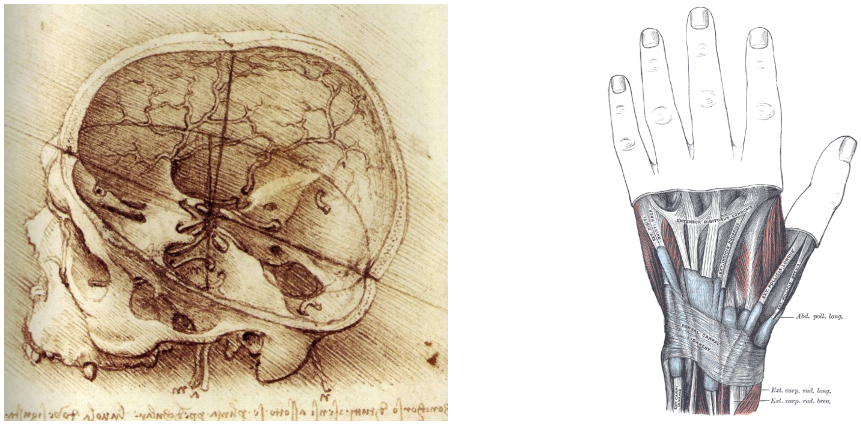
However, when line renderings are used on visible and otherwise invisible information, cluttering may result. To counteract cluttering in line renderings, a stylization of the lines often helps to communicate depth arrangements. For example, (b) in figure 2.25 shows a line rendering by E. Elber [42] using thick and dark lines close to the camera while thin

lines represent structures further away. This rendering also demonstrates the effect of using line halos around line in front [3]. By introducing discontinuations of otherwise hidden lines at those places where they intersect, the impression of occluding lines is caused. A similar technique for presenting both hidden and occluding line rendering is demonstrated in (c) of figure 2.25. However, while occluding lines are still rendered continuously, hidden lines have been dashed to uniformly introduce discontinuations.

M. Livingston evaluated additional visualization parameters to the style of line renderings in visualizations of multiple occluding structures in AR environments [89]. He considered the object's relative intensity, the structure's opacity and whether they are drawn using the outline alone or in combination with a single colored virtual phantom object. Similar to the experiments of V. Interrante, they asked users to determine the location of a hidden structure. M. Livingston compared the error measurements in different combinations of the visual parameters with the error measured using a ground plane which introduces perspective distortions of a known pattern (this is known to be a good indicator for depth [50]). His study showed that a configuration of an outline together with a single colored phantom rendering in combination with decreasing opacity and intensity at increasing distances was almost as powerful as the ground plane visualization in indicating the depth of hidden objects.

2.4.2 Cutaways

Instead of trying to preserve depth cues by preserving a certain amount of characteristic information, cutaway illustrations completely remove occluding structures. However, while the careless removal of occluding information will most likely result in ambiguous presentations, cutaway illustrations design the cut out in a way that depth cues are introduced and missing shape information can be mentally approximated. To be able to mentally close the introduced openings, they make use of the Gestalt Laws outlined in section 2.2 For example, figure 2.26 shows two cutaway illustrations which both make use of 'closure' (see also (b) in figure 2.9). Our cognitive system closes a familiar pattern which allows the presentation of hidden structure in full detail without any occlusion. Even though no information is present in front of the object of interest, depth cues exist implicitly via hinting at the missing information. The image on the left in figure 2.28 shows a drawing by Leonardo da Vinci from the 16th century in which he presents structures of the skull as context information. Our common knowledge of the look of a skull in combination with his contextual indications makes us mentally close the skull which in turn helps in



(a) (b)

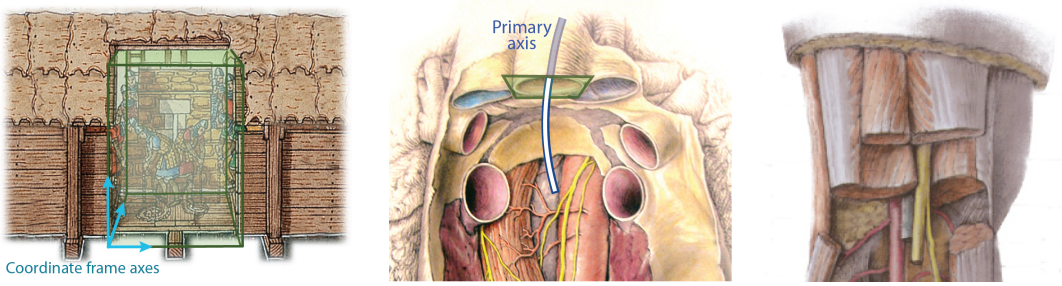
Figure 2.26: Illustrations using a Cutaway Presentation Technique (a) Inset cuts. (Image courtesy of Leonardo da Vinci "View of a Skull, c. 1489".) (b) Inset cuts. (Image courtesy of H. Gray [55].)

understanding the spatial relationships of all displayed structures. The presentation in (b) of figure 2.26 exploits the same cognitive behavior; the common pattern of the occluders is mentally continued even though they are not directly visible.

However, to support the identification of contextual structure as a part of some known object, other Gestalt Laws like symmetry or similarity (see section 2.2) should be taken into account. In addition, cutaways are usually designed to include depth cues to communicate the distance between the removed occluder and uncovered hidden structure (notice the transversal cut at the edges of the cut out to indicate thickness of the removed skin in (b) of figure 2.26).

As W. Li and his colleagues noticed [88], illustrators often use only a few object aligned cut-out shapes to meet these requirements. He noticed that mostly box shaped structures use a box shaped cutaway to remove occluding information. The cutaway object is aligned to the local coordinate system of the occluding object as in (a) of figure 2.28. However, a number of shapes cannot be well approximated by using a box. By examining hundreds of manually drawn illustrations, W. Li and J. Diepstraten [37] noticed that cylindrical objects are often exposed by using a wedge like shape aligned with the objects main axis (which is the axis with the largest section). The two planes of the wedge most often intersect at an angle between 90 and 140 degrees.

However, if multiple objects occlude an object of interest, the second level occluding



(a) (b) (c)

Figure 2.27: Different Cut-Out Shapes (a) Box Cut (b) Transverse Tube Cut (c) Inset Cut (Images taken from [88])

structures (which are defined as the structure in between the main occluder and the hidden object of interest) are not just opened but completely cutaway including their front and back faces. Second level occluding tubular structures are commonly cut away using a cutting plane which is oriented perpendicularly to their primary axis (see (b) in figure 2.27). Such cutaways allow one to indicate the overall path of the tube. Second level occluding structures have also been presented by illustrators by applying a relative inset while the global cutaway still follows the side walls of the main cutaway shape as in figure 2.27 (c). Inset cuttings allow one to present more detail of the structures which fall between the enclosing object and the object of interest.

For complex and thin enclosing structures, W. Li noticed illustrators often use window cuts, which are 2D shapes aligned with the object's surface. Window cuts indicate the shape of the occluder by implicitly bending the outlines of the windows which follow the occluder's shape. Therefore, window cuts often emphasize the contour of their cutaway. An implementation of window cuts on real world objects was demonstrated by C. Bichlmeier [9]. As M. Bajuraand and H. Fuchs outlined [5] and other researchers further evaluated [48] [114], object aligned cutaways implicitly add depth information by introducing a number of static and dynamic depth cues at the walls of the cut-out shape.

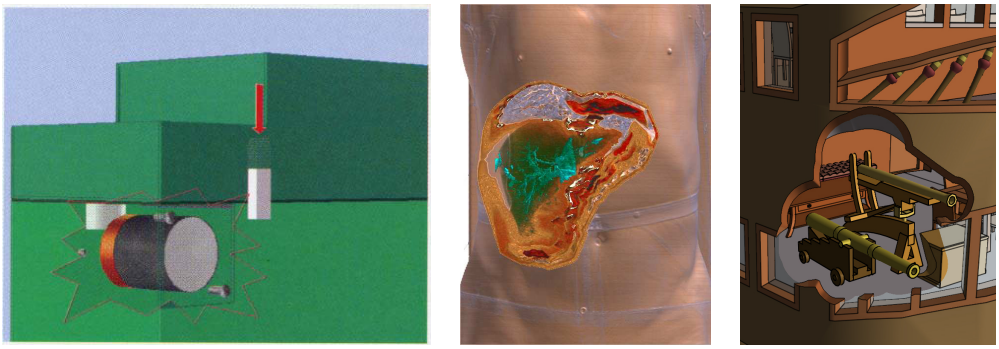
Interactions in object aligned cutaways have to be specific to the cut-out shape (otherwise the relation between the object's overall shape and the applied cutaway shape may be negatively affected). Therefore, W. Li allows interaction of a cutaway specific parameter space. For example, while box cuts are allowed to be translated along the object's main axis, interactions on transversal tube cuts are restricted to a simple adjustment to the position of the cutting plane along the tube.

While W. Li uses widgets to change the values, Knoedel an his colleagues implemented gestures to control the parameters of cut-out shapes [80]. For example, simply a scratch in x-direction followed by a scratch in y-direction defines the opening of a wedge cut. Besides gestures, Chen et al. demonstrated pressure based interactions (implemented using a stylus pen) to control the depth of a cutaway in a volume rendering [25]. He showed three different techniques which are designed to mimic natural techniques like drilling or lasering a cutaway.

Cutaway shapes as identified by W. Li are object aligned, and thus view independent. However, other automatic cutaway systems have been proposed considering the current point of view to ensure the complete uncovering of hidden structure. The first automatic cutaway rendering was presented by S.Feiner and D. Seligman as part of their Intent-Based Illustration System (IBIS) [113]. Their system first classifies obscuring, non-obscuring and hidden objects [45]. After rendering non-obscuring and hidden structure they render a 2D cut-out-mask to the z-buffer only. The cutaway mask is placed and scaled to cover the structure of interest while its depth values are set to be in front of all objects in the scene. By setting the depth values of the mask to a value smaller than all others, the subsequent rendering of obscuring objects will reject all fragments which otherwise hide the object of interest. To visualize the cutaway mask, they additionally render its outline to the frame buffer (see (a) in figure 2.28) (notice that this is only a 2D outline which does not reflect the object's form).

Since Feiner's algorithm makes use of hardware capabilities, it is able to render in real time. However, the resulting presentations only uncover the hidden object without any indication of the thickness of the structure which is cut away. To indicate the thickness of occluding objects, view aligned conical cuts have been demonstrated by I. Viola et al. on volumetric data as shown in (b) of figure 2.28 [124] and later by M. Bruns et al. on 3D geometrical representations of the scene [18]. They both use a 2-1/2D cutaway mask which is generated from the depth values of the back faces of the object of interest. The depth mask controls the rejection of fragments inside the cutting volume. Every fragment which has a depth value smaller than the corresponding value in the mask is rejected. To generate the conical shape the depth buffer is furthermore enlarged by calculating a distance transformation from it. The resulting 2D distance values are combined with the maximal depth in the depth buffer. The conical side wall results from gradually decreasing the depth value with increasing distance in 2D.

While volumetric data defines the structure at each point of the conical side walls



(a) (b) (c)

Figure 2.28: Adaptive Cutaways. Images (a) and (b) show view aligned conical cutaways using a 2-1/2D cutaway mask. (a) A 2D cutaway mask. (Image courtesy of D. Seligman and S. Feiner [45] [113].) (b) Cutaway rendering is applied on volumetric data (Image courtesy of I. Viola [124].) (c) The cutaway mask interactively rejects fragments of 3D geometrical objects (Image courtesy of M. Bruns [18].)

implicitly, polygonal data often only models the outer surfaces. If the system cuts a single surface material this modeling technique becomes visible. Therefore, cutaway objects have to be modeled with inner and outer surfaces separately. If this is not the case, C. Coffin and T. Hoellerer showed an approach which displaces the front face of the model to generate a second one online [28]. However, even if the model defines a surface for both sides of the material, the area between both faces is usually empty (if no inner parts are modeled). Thus, Choffin and Hoellerer automatically generate geometry to render the border between inner and outer surfaces. In contrast, M. Burns showed how additional additional faces can be avoided by shading the fragments of inner surfaces using a normal shape derived from the cutaway shape (see (c) in figure 2.28).

In contrast to object aligned cutaways, view aligned excavations always reveal selected hidden objects. Also, interactions have been proposed to define a mixture between both approaches. Interactive cutaway renderings have been demonstrated using the Magic Lens metaphor (However, notice those renderings which do not necessarily follow illustrators techniques as presented by W. Li anymore). For example, T. Ropinski demonstrated the use of arbitrarily shaped geometrical and volumetrically 3D Magic Lenses [108]. R. Bane et al. implemented the reverse a conical shape, the x-ray tunnel, which is a frustum inside the viewing frustum positioned along the viewing direction [7]. The tunnel is divided into different regions which have been assigned to rendering styles. The user controls

a single region by adjusting its specific front and back plane. A cutaway is realized by setting the rendering style of the first region to completely transparent. This system furthermore allows the combining of ghost presentations with cutaways by switching to wireframe mode in the region behind the cutout area. Combining ghostings and cutaways is a very common presentation technique, demonstrated e.g. in the previously mentioned systems of Burns, Seligmann, Knoedel and Viola. However, while the former three use a pre-defined very sparse ghosting (implemented as line drawing), I. Viola exploits the importance of the stylized object to choose the level of sparseness online with respect to other covered objects. This allows one to change the structure which is ghosted inside a cutaway volume at runtime by changing the importance distribution.

2.4.3 Explosion Diagrams

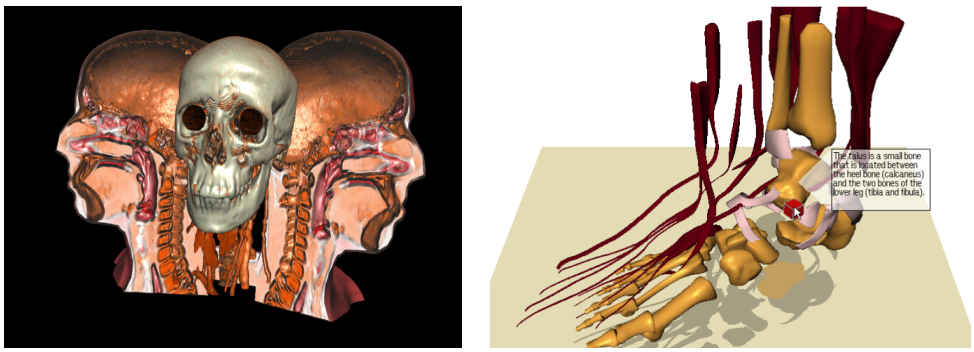
Both ghost presentations and cutaways remove information from the illustration. Even though our cognitive system is able to guess the missing shape information, detail and texture is lost. However, displacing and/or distorting the occluding structure allows one to keep them in the F+C presentation (section 2.3). Illustrators exploit this F+C technique by generating an exploded view. The explosions are designed to make use of the cognitive ability wherein one can mentally reverse the introduced modifications (which were necessary to uncover the object of interest). To support the mental inversion, explosion views also use the power of the Gestalt Laws. For example, the laws of closure, symmetry and simplicity & good continuation (see figure 2.9). The layout of displaced objects indicates

Figure 2.29: Exploded View. The objects have been displaced to resolve occlusions. The presentation uncovers the internals of a car by moving occluding structure out of the line of sight. (Image courtesy of K. Hulsey, www.khulsey.com)

occlusions even without actually covering the object of interest. For example, figure 2.29 shows a manually drawn illustration of otherwise hidden mechanics of a car. The common knowledge about the construction of the car, in combination with a single direction of exploded elements supports the Gestalt Laws of 'good continuation' and 'closure'.

Computer graphics researchers have tried to automate this technique. H. Sonnet and his former colleagues [115] presented an interactive system which moves the parts of an object out of the 3D volume of his explosion probe. This interaction tool defines a 3D frustum starting from the near plane of the user's viewing frustums and it ends at a user controllable 3D cube. Every part which falls within his 3D explosion volume will be moved outside to ensure visibility at the cube's location. According to this, the cube at the tip of the explosion volume has to be moved close to the object of interest to uncover it as in figure 2.30 (b). While the system of H. Sonnet focus on the revealing of a specific hidden structure, he does not explicitly consider the explosion layout. However, in a psychological study design parameters (similar to the Gestalt Laws) were carried out which effect the understanding of the layout of an explosion diagram [61].

Instead of using a cone-like explosion volume which pushes the parts out of the line of sight, S. Bruckner and his colleagues [16] explicitly define a set of different forces which push the objects relative to each other rather than relative to the volume defined by the explosion probe. As the parts of an object move relative to each other, this approach allows the keeping of the spatial relations between exploded parts visible in the presentation. To apply forces between parts of the object, he first splits the object into a set of box (or slice)



(a) (b)

Figure 2.30: X-Ray Visualization Using Explosion Views (a) Exploded view of volumetric data. The explosion is constrained using a hinge joint. (Image courtesy of S.Bruckner [16]) (b) Caption. (Image courtesy of H. Sonnet [115])

shaped parts. His system applies four different types of forces. Visibility is controlled by a viewing force (similar to H. Sonnet's system) but in addition, forces to repel, attract and control the spacing between parts can be applied. The strength of the forces is interactively adjusted which implicitly alters the appearance of the explosion. To even further control the layout of the explosion, S. Bruckner allows one to constrain the forces by applying different types of joints. For example, figure 2.30(a) was generated by constraining the forces using a hinge joint. Nevertheless, other types like slider joints were shown to be useful to explode similar parts of the object in a similar way.

Even though exploded views present a powerful technique for uncovering hidden structure, x-ray visualizations are not their traditional area of application. Instead, they have been utilized most often in assembly instruction and overview presentations. While illustrators have been manually creating explosion diagrams for centuries, a system to automatically generate a comprehensible assembly plan was presented by M. Agrawala in 2003 [1]. His implementation follows the findings of a psychological study [61] which included the usability parameters of a traditional assembly plan. For example, one finding was that assembly plans should present step-by-step instructions rather than a complete explosion of the object. Each step adds a set of important objects to a completely collapsed presentation of the then assembled object. Agrawala's system generates assembly instructions by rendering explosion diagrams with the constraint that in each presentation, only those objects are added to a previous presentation which do not intersect with other not yet assembled objects and which are clearly visible from a common point of view (figure 2.31)

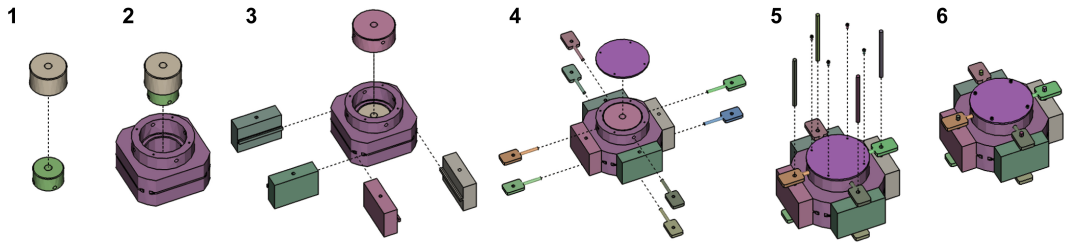


Figure 2.31: Automatically Generated Assembly Instructions(Image courtesy of M. Agrawala [1].)

Since assembly plans only consider a few objects in each presentation, they are usually not subject to clutter as much as explorative explosion diagrams are (which try to present as much of an object as possible at once). Thus, explorative explosion diagram are often difficult to understand. As a remedy, interactive explosion systems have been proposed which allow one to change the presentation at runtime to control the amount

of collapsed and displaced parts of an object. This way, interactive explosion diagrams combine the ability to communicate relationships with interaction pattern which support the understanding of the explosion.

For example, F. Ritter et al. [105] present a system to support learning spatial relationships by changing the explosion diagram at runtime. To furthermore reduce the possibly for clutter they modify the explosion 'in-place' by scaling down the parts of an object, instead of displacing them (a similar technique was presented by A. Raab [100] in his focus and context explorations). Other systems have been designed to support interactions of displaced explosion diagrams. For example, W. Li and colleagues [86] added a number of interaction techniques to Agrawala's system which allowed them to modify the positions of exploded parts relative to each other. They demonstrate direct manipulation of parts along their explosion path as well as an exposure of parts. This is implemented by increasing the explosion distances of adjacent parts to the one which is selected by a 'mouse over' technique. This presentation with a quick selection method enables one to look quickly through parts along an explosion path.

W. Li et al. also presented an interactive system to support the understanding of existing 2D image based explosion diagrams [87]. After segmenting the individual parts he manually defines occlusions between them and an explosion direction to allow the interactive collapsing and expanding of the segmented image regions (figure 2.32).

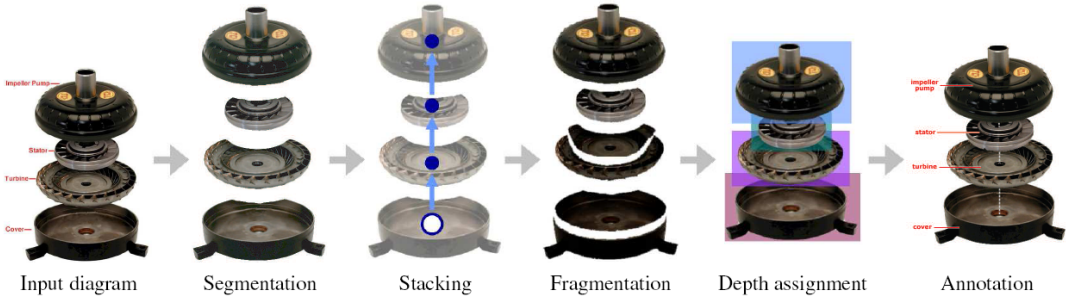
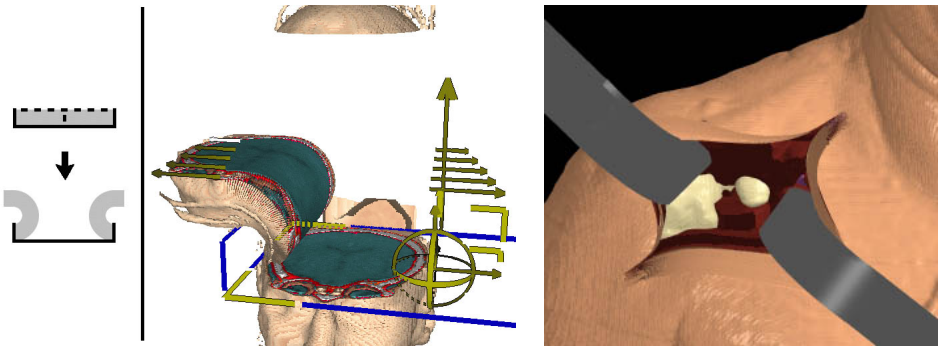


Figure 2.32: Interactively Reassembling 2D Explosion Views (Image courtesy W.Li [86].)

In contrast to W. Li's rather high amount of manual authoring in his image based system, other systems further automate the design of the explosion layout. However, since a high number of displaced objects easily produce cluttered presentations, explorative explosion diagrams have to carefully select the direction and ordering of part separation. To compute a valid order for part removal, Agrawala computes a sequence based on the work of geometric analyses by Romney [107]. They both remove one part after the other,

constrained by blocking information, so that only parts are removed which do not intersect with other parts along their explosion path. If a part can be removed from the object in multiple directions Agrawala proposed using the one which escapes the bounding box of the object (computed before the part is removed) most rapidly. In addition, parts in his system move relative to other parts by stringing their displacements together. Agrawala chooses to let a part follow the movements of the biggest part it has contact with.

However, Agrawala's system needs additional semantics about the object such as which faces belong to a part or which parts form an unexplodable grouping. Niederauer and his colleagues presented a system which generates an explosion diagram from a set of triangles, but only if they represent a building with multiple floors [96]. By assuming that two floors are being separated by a roof, Niederauer searches for faces which belong to a roof. Given an 'up direction' of the building, he analyzes the orientations of the faces and he groups those which face opposite to the up direction at a similar height. To avoid selecting faces of the inventory (for example) he identifies a roof only if a certain number of faces at a similar height was selected.



(a) (b)

Figure 2.33: Distorted X-Ray Visualizations. (a) Physically correct distortions help to understand the modifications (Image courtesy M. McGuffin et al. [92].)(b) Additional presentation of the tools used to create the distortion helps to understand the opening which subsequently supports the mental inversion of the distortion. (Image courtesy S. Islam et al. [71].)

Besides explosion diagrams of ridged objects, physics based deformations present a special kind of explosion. To uncover hidden objects they bend or squeeze occluding information in a physically correct way [32]. The realistic behavior of the distortion allows the presentation to make use of Gestalt Laws which in turn allow one to mentally undo the

distortion. Figure 2.33 shows two examples of such x-ray visualizations from M. McGuffin [92] and S. Islam et al. [71]. The image on the left shows how a realistically presented folding of a layer supports the understanding of the behavior. The same cognitive behavior is utilized in the image on the right side. However, the introduced distortion is furthermore emphasized by displaying the tools which are commonly used in real surgeries to generate the opening. Since the additional items help to understand the process which had to be made to create the image, they also support the mental inversion.

Chapter 3

Rendering X-Ray Visualization in Augmented Reality

Content

If we consider Augmented Reality (AR) as a visualization technique, the relationship of real and virtual objects is one of focus and context: Either we want to provide additional virtual context to an important object of interest in the real world or we want the user to focus on a virtual object embedded in a real world context. In case of x-ray visualizations, we consider hidden structure as information in the focus of attention, while occluding objects represent their context. Following this line of thought, a rendering system for

x-ray visualization in AR environments can be considered as a Focus and Context (F+C) visualization framework. Since F+C (and thus x-ray) visualizations demand that we visually discriminate objects depending on whether they are marked as focus or context, the rendering system has to provide tools to first classify the content of the 3d scene, followed by the rendering of both using different styles.

The information commonly available to the AR system is a set of virtual three-dimensional objects in combination with two-dimensional images presenting real world information which are received from the AR system's video feed. Since three-dimensional virtual objects offer much more information about its shape than a two-dimensional video image is able to provide, an F+C classification and its subsequent stylization will most likely produce visually more appealing results if generated from registered virtual 3d objects. However, due to the complexity of real world scenarios, a complete virtual representation of all real world objects is difficult to prepare. Even if the scene is prepared to provide virtual counterparts of real world objects, a number of other errors may be introduced by the virtual data (see section 1.1.3), making its usage at least questionable.

Consequently, the framework has to be able to handle both cases, image and object space classification and stylization. While differently prepared situation or different conditions, like the number of occluding layer, require different visualizations, a system to support x-ray visualizations in real environments has to provide a modular software architecture allowing to easily exchange the visualization modules depending on the current situation. Also a rendering framework for x-ray visualizations in AR has to be able to handle all issues involved in placing virtual augmentations in real world environments itself. For example, since a number of different data sources are required to place an augmentation, synchronization deficiencies may result in a misplacement of the virtual objects. Since an object's stylization has to be placed very close to its real world location, unsynchronized overlays will disturb the impression of an x-ray visualization.

The following list summarizes the requirements of a rendering framework to support stylized x-ray visualization in Augmented Reality environments:

(1) Tools to classify real and virtual information into Focus and Context groups.

(2) Tools to allow the stylization in both, image and object space.

(3) A system to integrate stylized x-ray visualizations in real world environments. This includes methods to synchronize all data involved in the generation and placement of virtual content.

(4) A modular software architecture which enables an easy exchange of the x-ray visualization technique in order to adapt to the current environment and the available data.

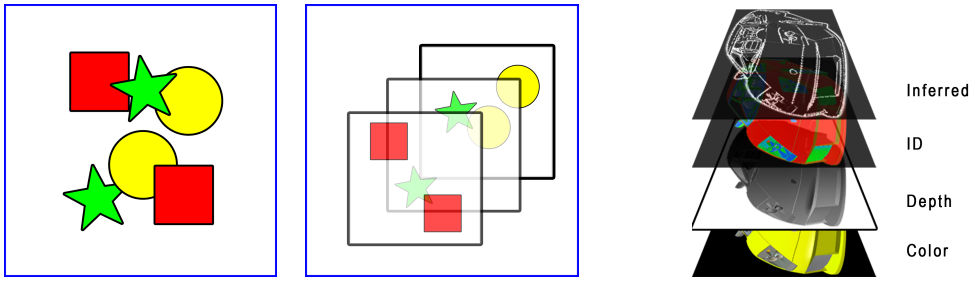
While this chapter describes a system to render x-ray visualizations in real world environments, the subsequent chapters of this thesis demonstrate its usage. The following sections first present the basic rendering algorithm, followed by an overview of the AR framework. This chapter closes with the approach of Multi-Level Focus and Context Templates, which enable an effective visual communication of x-ray visualizations in a real world environment.

3.1 Rendering Technique

Recent work on volume visualization, such as that of Krüger et al. [83] and Hauser et al. [60], 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 approach, which we found suitable for our aims. Consequently, the core of the presented rendering approach for x-ray visualizations in AR is a general GPU-based rendering framework similar to [83]. However, instead of simply extracting layers of the three-dimensional scene, we are aiming at arbitrary groups of scene content. Therefore, we had to add a third step which composes the final result per pixel while executing a set of rules which defines the composition of a particular pixel.

Buffer Rendering. Rather than just making a binary decision between focus and context, we classify the objects in the scene into multiple families, using a scene graph markup mechanism [104]. For every context family, a separate g-buffer (which is a set of buffers [110], Figure 3.1(c)) is allocated, which is used to render all object families into distinct buffers. This enables us to apply different treatments, such as image manipulation, in the next step. However, since three-dimensional geometry is rendered in this step, we are also able to apply any object based stylization during Buffer Rendering.

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



(a) (b) (c)

Figure 3.1: Rearranging Scene Elements into G-Buffer (a) An illustration of a scene. (b) One possible G-Buffer volume. Notice that a G-Buffer does not represent a depth layer (c) Conceptual representation of the image buffers contained by a single G-Buffer.

Buffer Processing. Once the buffers have been rendered, we process each of them to visually discriminate focus from context. During this step the visual styling is achieved using image operators, which may include edge enhancement and color or transparency manipulation. The actual modification is highly application dependent which demands a flexible authoring technique. Within the scope of this thesis different approaches to define and control the stylization of the content of a set of G-Buffer have been implemented. They range from a mechanism similar to shade trees [31] to a set of explicit calls of GPU shader (see section 3.3.2.1 for more details about the provided options to configure the processing of G-Buffer content). However, the general idea of this step can be outlined as following.

```
For every G-Buffer(i) {
  Modify G-Buffer(i) according to a set of rules;
}
```

Scene Compositing. This step combines the information contained in the processed buffers. They are composed in front to back order, which are evaluated by sorting depth values associated with every fragment. To allow for an interaction with the buffer's content, the scene composition is not fixed but may also be scripted, which can vary on a per-pixel basis, enabling different compositing strategies for different areas of the image.

```
For every pixel(x,y) {
 r=execute-compositing-strategy-at(x,y);
 return r as final pixel;
}
```

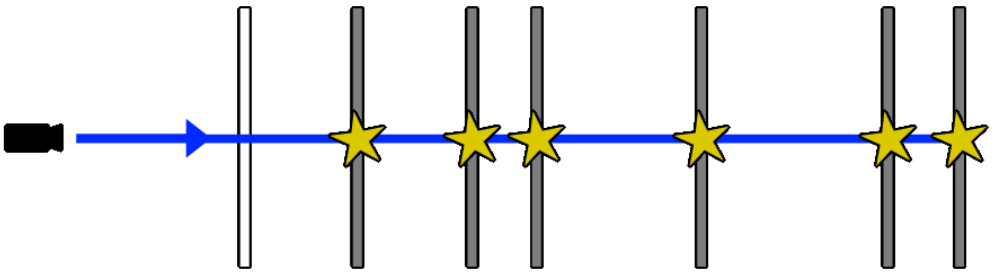


Figure 3.2: Scene Compositing. Non-Uniform raycasting through the G-Buffer volume

3.1.1 Buffer Rendering

From the algorithm, it is clear that a single frame buffer does not fulfill our requirements. We need not only color+alpha but also at least depth information for the scene compositing and, potentially, several additional buffers to control the Buffer Processing. To store this information, Geometric Buffers (G-Buffers) [110] 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 view-dependent information about objects, including the current depth and can therefore be seen as a 2.5D scene approximation. As many G-Buffer implementations do, we extend the information held in the G-Buffer to include inferred information from the processing step. This inferred information can later be used to change the visual appearance of objects belonging to a particular family. Figure 3.1(c) provides an illustration of a G-Buffer containing four different bitmaps: color, depth, ID, and inferred edge information. A single G-Buffer contains an approximation of those objects in the scene belonging to a particular context family. We can thereby isolate the styling applied to different families, while the collection of all G-Buffers approximates the whole scene. For example, the objects in Figure 3.1(a)/(b) have been bound to different buffers based on their family membership; in this case, the family membership is exemplified by shape.

Assuming a scene graph, as it is commonly used in VR and AR applications, a simple approach places focus objects and context objects in separate branches of the scene graph. This, however, would limit the possible data sources to those with a specific separation of these two elements. Moreover, it would make interactive changes of F+C separation only possible if the application is tightly coupled with the scene graph data structure. Instead of depending on a hierarchy that fulfills our requirements, objects are marked with contextual information and scattered throughout the scene graph in any naturally occurring order

without any enforced grouping. Sorting objects by context family happens implicitly during the scene graph traversal, using a parameterized scene graph which enables a context-sensitive scene graph traversal. Using the of context sensitive scene graph traversal we are able to use the regular rendering pipeline to extract all the necessary information that will be used in the next steps of our rendering algorithm. The scene is traversed in a single pass with multiple render targets and every object is rendered to exactly one G-Buffer, which is determined by its family membership given as semantic markup (see Figure 3.2).

The idea of a context-sensitive traversal of scene graphs was introduced in a publication by Gerhard Reitmeyr [104]. The grouping of scene graph elements into G-Buffer build upon this work, therefore its important points are summarized here for convenience. A set of context parameters is maintained throughout the traversal of a scene graph. This allows parameterizing and repurposing sub graphs in various ways. The context parameters are maintained as part of the state of the scene graph traversal, which makes them independent of the scene graph structure. The context parameters are modeled as an associative array of key-value pairs, where the values are either strings or pointers to sub graphs. By using pointers as parameters, such template sub graphs can be inserted multiple times during the traversal. In contrast to a conventional directed acyclic graph structure, the binding of child nodes to their parents happens very late, during the traversal itself, so the nodes can be changed for each traversal and provide a very flexible way of assembling complex scene graphs. Transparent caching of the traversal outcome in display lists ensures that rendering performance is not adversely affected.

The context sensitive scene graph traversal shifts the complexity of managing multiple representations from the application code to the scene graph itself. Rather than writing application code to modify the scene graph or change rendering parameters based on user interaction, the application only needs to change the context parameters for high level control of the visual effects. The scene graph will adapt the visual appearance of its contained objects to dynamically changing requirements, and, even compose sub graphs on the fly. A particular visual representation is simply an instance of a template sub graph combined with a specific choice of context parameters. Combinations of content and visual interpretation are created during traversal only, which is the actual moment in time they are required. Figure 3.3 shows such a conceptual grouping of objects in the scene graph (highlighted in red and blue), regardless of their occurrence in the graph. The implementation of this marked-up scene graph is based on Coin3D (www.coin3d.org), but

the mechanism itself is independent of any specific platform and is easy to duplicate on other systems.

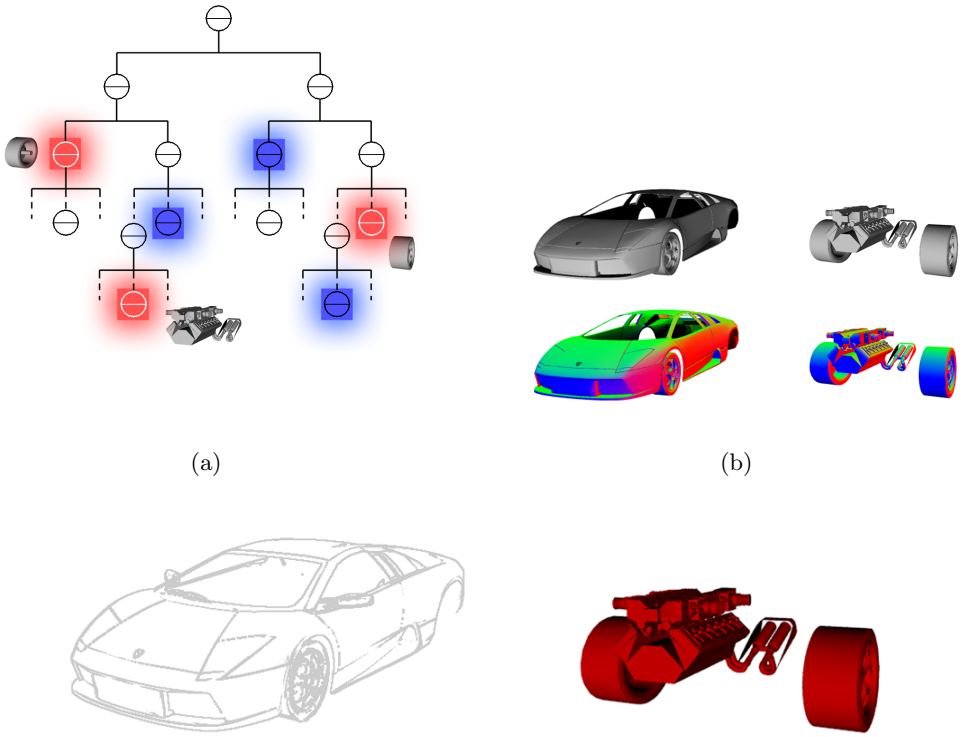


Figure 3.3: Data processing before compositing. (a) Conceptual illustration of a context sensitive scene graph. Context families of objects with the same contextual information (in blue and red) can be jointly referenced regardless of their position in the scene graph (b) Result of the first step of our algorithm (G-Buffer rendering). (c) A computed sparse representation and the shaded focus objects. This is the result of the second stage of our rendering pipeline. The final composition is shown in Figure 4.9

3.1.2 Buffer Processing

To stylize the content in 3d space, image processing techniques can be applied on every buffer. This allows not only to compute their stylization but also to infer additional information which may be used to either control the subsequent scene composition or to stylize other G-Buffer's content. For example, the information in all G-Buffer may be used to detect shape cues like edges or regions with high curvature, by may also be used

to mark regions with a particular color or depth values. While some techniques need to consider more than the fragment's value (e.g. other types of values in its neighborhood), fragment values of other buffers in the same G-Buffer, or fragment values from different G-Buffers can be taken into account during Buffer Processing.

A sensitive stylization of x-ray visualization demands a complex classification. However, rather than relying on the user to entirely provide such an object classification, the implemented system allows to author x-ray visualizations through heuristic rules during Buffer Processing. Therefore, complex F+C scenarios are supported by allowing to cascade simple F+C separators, using a directed acyclic graph structure on top of the set of G-Buffers. Such a graph is easy to create and to understand, and it allows the explicit modeling of the control flow and data flow of the rendering process. The graph separates focus and context discrimination in interior nodes, while the leaf nodes execute rendering operations (see section 3.3.2.1 for more details about the implemented options to configure the processing of G-Buffer content).

3.1.3 Scene Composition

During scene compositing, the information from the set of G-Buffers is merged into a final image using a GPU raycasting algorithm with a nonuniform step length. Notice that simple blending of the G-Buffers is not enough since per-pixel occlusions are essential for the desired effects. The ray has to traverse the scene approximation of the G-Buffers in front to back order (as illustrated in Figure 3.1 and Figure 3.2). 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 been done, we proceed to compose all the fragments into one single output. The composition, however, is based on compositing rules that can arbitrarily alter the contribution of a particular pixel from one of the G-Buffers. For example, the color or transparency of a particular pixel may be modified based on the importance of the pixel that was visited along the ray before the current one. The presented system even allows to apply different compositing rules on different regions in the image (Figure 3.1.3). This is a particularly useful operation for complex AR scenes to interactively suppress pixels that are unimportant or confusing. The following two examples show different strategies of composing the fragments of all involved G-Buffers per pixel. While the first example uses all available information, the second example demonstrates using information about the depth order of the fragments. In chapter 4 the usage of a compositing strategy is shown which enables to filter contributing

fragments in order to control the depth complexity. By composing the three-dimensional scene using the result of previous F+C classifications, the system becomes able to only select the first contextual (in depth order) and the focus' fragment during the composition of the pixel's displayed color. However, there is no restriction to depth information as compositing parameter, other G-Buffer content may be used to control the composition of the x-ray visualization as well.

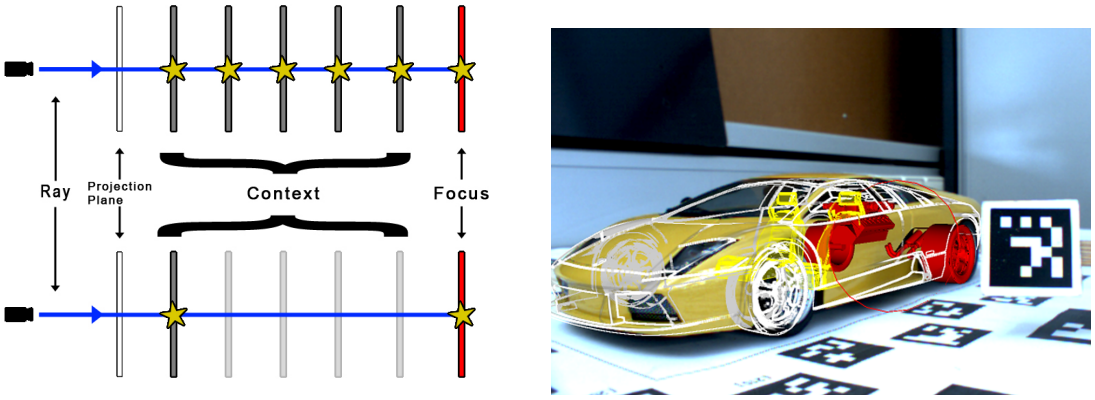
```
composing-all-fragments{
  Sort all G-Buffers by depth d;
  For all G-Buffers(j) in order given by d
    r = compose(fragment(G-Buffers(j),x,y),r);
  return r;
}

composing-first-and-last-fragments-only{
  Sort all G-Buffers by depth d
  r = compose(fragment(G-Buffers(0),x,y),r);
  r = compose(fragment(G-Buffers(numOfGBuffer()-1),x,y),r);
  return r;
}
```

3.1.4 Implementation

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

To address a certain buffer of a specific G-Buffer a simple texture tiling technique is used, where each tile represents a G-Buffer within the texture representing the spe-



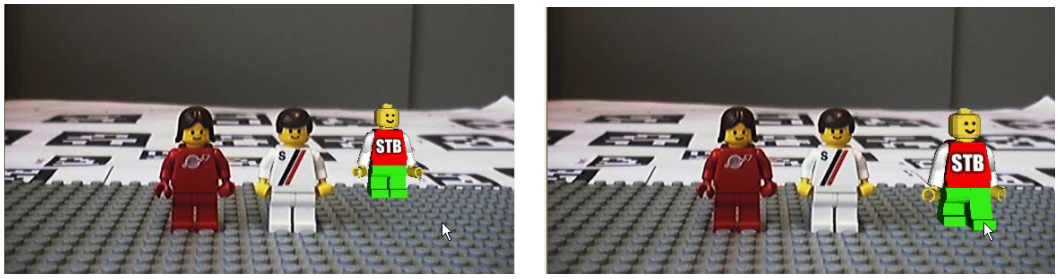
(a) (b)

Figure 3.4: Filtering during raycasting using two strategies of compositing. (a) Illustration of the applied compositing strategies. 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 color value (b) Interactively applying the different compositing strategies to reduce the number of context information inside a flat Magic Lens

cific buffer. Therefore, switching G-Buffers from a different context family merely means looking up corresponding viewport parameters of a single large target texture.

3.2 An Augmented Reality Framework

To believably fuse real world imagery with renderings of virtual 3d objects, the virtual imagery has to adapt to the real world environment. Since depth cues are the most important visual communicators to identify an object's position and orientation in 3d, the rendering of virtual structures has to adapt to match the real world depth cues. However, if the goal of the visualization allows to present the virtual content in a way so that it is easily identified as being a virtual add-on to the real world environment, a subset of all existing depth cues may be sufficient. For example, Figure 3.5 shows the same virtual figure in exactly the same real environment but rendered in different sizes using different perspective distortions and placed in different heights in 2d image space. Even though the virtual rendering ignores depth cues which result from the current real world light conditions (e.g. shadows or reflection), the differences between both images let us believe that the figure in the left image is behind, while the figure in the right image seems to



(a) (b)

Figure 3.5: Impression of 3d location and orientation is guided by depth cues. Differences in perspective distortion, relative size and relative height in 2d image space generate the impression that: a) The virtual figure seems to stand behind the real figures b) The virtual figure seems to stand next to the real figures.

stand next to the real one (see section 2.2 for more detail on depth perception)

This example demonstrates the importance of those depth cues which result from the placement and orientation of the virtual content. It also shows the importance of those cues which result from the parameters of the camera e.g. to control perspective distortion. Following this line of thought, important requirements of an AR system are the ability to register virtual 3d objects in real world 3d space and the capability to render the virtual content with approximately the same camera parameters as being used to capture the real environment. Once the application also requires to analyze real world information, like colors or shades, the AR system has to supply also a 2d video stream of the real world environment. Nevertheless, video data is often not only required to analyze the visual situation, but often used to present to real world information in the AR rendering, if e.g. the display device of the AR system does not allow seeing through it.

Figure 3.6 shows an example of a common AR system and the data which is acquired, computed and presented. The application in the example uses video data as input and computes the tracking data from the video. Subsequently, the virtual object is rendered using the previously derived tracking data, which defines the position and orientation of the virtual object relative to the real world camera. The virtual camera uses pre-calculated parameters reflecting the characteristics of the real camera to generate a 3d rendering of the virtual object which automatically provides depth cues matching to those present in the real video stream. This algorithm allows a simple overlay of the virtual information to generate the impression of a virtual figure standing on a real world card. This example demonstrates the usage of the most important data in a visual AR application.

Even though rendering and tracking data are the primary ingredients of any AR application, the hardware devices used to acquire or display the information may vary for each setup. Different input or output devices usually require to use different drivers, different display modalities or even different data management. Consequently, this wide variety of possible input and output devices requires a software infrastructure supporting hardware as well as its configuration independent application development.

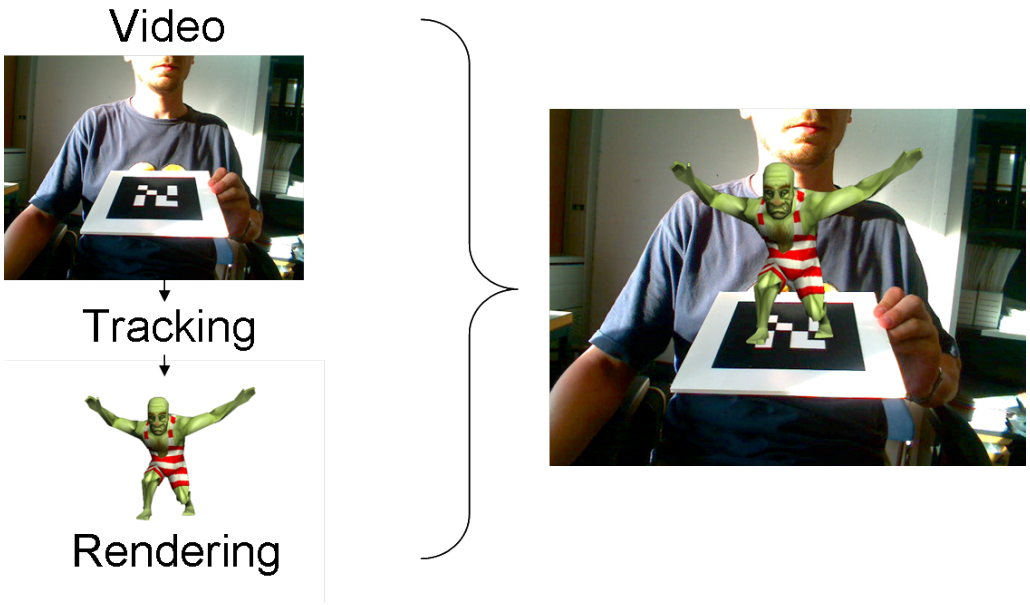


Figure 3.6: Data flow in a common AR application using vision tracking. Real world imagery is delivered by the system's video feed and processed by vision based tracking algorithms. To align virtual and real data, the tracking data has to be applied to transform virtual content followed by an overlay of the rendering result on top of the video feed.

Hardware devices which even need their own software to compute e.g., tracking data out of the information seen by a set of camera images (like ART tracking technology) usually require an independent threads of control. Those data which are acquired in asynchronously running components of the AR system usually require special care when applied to further steps in the AR rendering pipeline. For example, Figure 3.6 shows an application which uses tracking data directly computed from the AR system's video feed. The same video information is subsequently used to overlay the virtual rendering. While the setup in Figure 3.6 uses tracking data directly computed from the same video information, which is used in the final AR rendering, synchronization of the video and rendering can be easily achieved. In contrast, the setup in Figure 3.7(a) retrieves tracking data from an external application running in an independent thread of control. The

external application uses a set of cameras (usually fixed in the environment) which are all different to the one which is used to deliver the video in the background of the AR rendering. While tracking data comes from a different source than video on which the computer graphics is overlaid, the system has to synchronize video information used in the background and those used to compute tracking data from (see Figure 3.7(a) for an unsynchronized augmentation)

This section describes the system's main components. The kernel itself and how it manages integration and synchronization of the components is discussed in the following section.

3.2.1 Data handling

Tracking data: Real-time acquisition and streaming of tracking or other multi-modal device data is done with a tool called OpenTracker, described in detail elsewhere [103]. OpenTracker follows a data flow graph architecture which can be distributed across multiple networked computers if desired. Source nodes for a large variety of tracking devices exist to acquire the device data. Filter nodes can further process the data, and sink nodes deliver the data to an application. Multiple components can register to receive data from a single source in OpenTracker, which allows to achieve synchronized behavior of multiple objects in the application.

Video data: Video management is handled by a tool called OpenVideo, which implements a general dataflow system for video data based on a pipes-and-filters architecture [19] in a similar way OpenTracker is handling tracking data. However, in contrast to trackers and conventional sensors, video sources consume very high bandwidth and consequently require specific techniques to avoid undesirable behavior such as frame drops or stalling. Existing video libraries such as Microsoft DirectShow achieve such efficiency by platform specific optimizations. Consequently, OpenVideo was designed as a wrapper library encapsulating efficient single-purpose video libraries. The implemented tool provides a consistent scriptable interface as well as missing features such as network streaming, simultaneous delivery of video to multiple application objects, and a notification service for components interested in video updates.

OpenVideo encapsulates raw video data in video objects, which reside in shared memory and are passed around by reference. Video objects have a number of advantages over a fixed frame structure. They can be annotated with arbitrary meta data, allowing simple and experimental multi-modal data structures. For example, adding resolution and format

description allows a receiver to deal with a variety of sources. A tracked camera or image probe can deliver a video stream, where each frame is annotated with the corresponding pose information from the attached tracker, making it possible to explore an object from multiple points of view. To support data synchronization, the video object stores video data in multiple buffers.

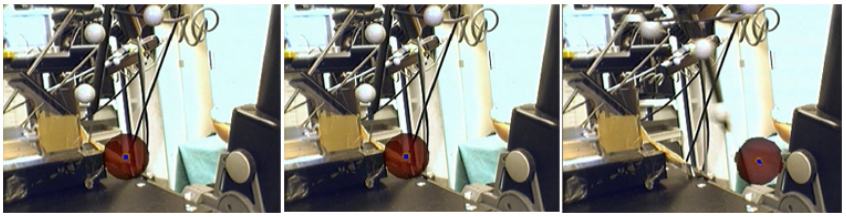
Rendering: The rendering component of the implemented system is based on Coin3d (www.coin3d.org), a scene graph library which implements the Open Inventor [128] standard. Through extensions of Coin's class hierarchy of scene graph nodes, the special requirements of AR renderings can be addressed. An AR rendering system needs not only to answer What and How to display but also Where to display it. A special component of the implemented framework was designed to address this question. It is able to create one or multiple views of a scene, accommodating a wide variety of display options, including frame buffer configuration and camera manipulation (including real-time image warping and off-axis rendering). Through configuration options it also allows to accommodate all viewing conditions including VR, AR and conventional displays. For example, the same application is able to display its content on a classic 4-panel view on a single monitor screen as well as on a stereoscopic head-tracked video-see through display. By completely decoupling viewing conditions from viewed content and application, the same features and rendering quality are simultaneously available to all applications in all setups.

This flexibility is achieved by encapsulating all relevant viewing commands affecting the viewing condition as nodes in the scene graph. The scene graph can then be scripted to deliver any viewing for any content, by mixing and matching viewing commands with content. For example, a stereoscopic display can be scripted as a two-pass rendering method while the same scene on a monoscopic display simply omits the second pass.

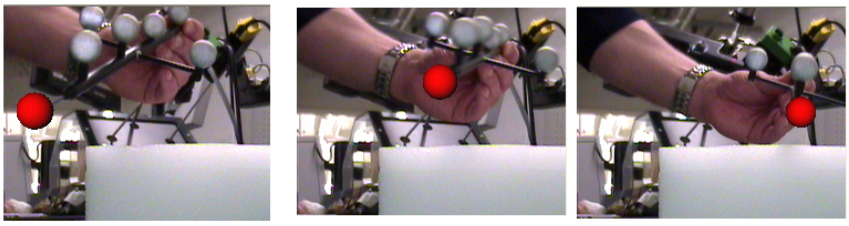
3.2.2 Scheduling

The presented architecture builds on a micro-kernel pattern [19], which implements the basic functionality to load and unload the system's components and it provides scheduling strategies supporting different setups. If none of the system's components need to be executed in a different thread of control, the data used to generate the AR rendering can be requested one step after the other. If furthermore tracking data is computed using the same video information as used to overlay the computer graphics, the AR fusion can be assigned to the exact point in time when the video image was captured. However, some systems require to use different sources for tracking and video information. For example,

an ART tracking installation does not deliver video and thus requires an external video source. To synchronize tracking, video and rendering using different hardware sources, we acquire video and tracking data after the other, and record the offset between the data. For example, if we first compute the tracking data, followed by an acquisition of the video information, the video data is 'behind' the tracking information approximately by the time consumed to acquire and process the tracking data (not counting the latency of the hardware itself). To reduce this offset, the implemented system allows to process the data in different threads of control, while acquisition from hardware is still going on. This strategy reduces overall latency.



(a)



(b)

Figure 3.7: Synchronized vs. unsynchronized overlay. A virtual sphere is attached to the tip of a tracked tool in both figures. a) the virtual sphere is misplaced due to an missing synchronization of video and tracking data b) Tracking and video data is synchronized in each frame, right before the rendering of the virtual content is started.

However, the overall display rate of the scheduling still depends on the slowest part in the rendering pipeline. If the rendering is very slow, the overall display rate will drop. This includes the frame rate of the video information in the background of the AR rendering, so unusable applications may result. Since OpenTracker and OpenVideo nodes can be executed within an independent thread of control (e.g. to service an independent hardware device), a strategy of asynchronous data acquisition can be used to accelerate the overall

rendering. However, the asynchronous processing of data of individual system components needs to be handled with care, since two separate sources can deliver data at different rates, resulting in an error visible in the AR presentation. The application programmer has to decide which strategy to use: if the introduced synchronization error is affecting the application goals less, asynchronous processing might present the best choice. Error visualization or multi-framerate rendering in combination with image based rendering are able to reduce the influence of the introduced error.

3.3 Focus and Context based Integration of Virtual Data

The strength of AR originates from a fusion of real and virtual imagery. Virtual data either presents additional contextual information to real world objects, or it shows the object of interest itself. Consequently, AR visualizations can be considered as Focus and Context visualizations by their origin. On one hand, photorealistic AR renderings utilize the real world imagery as a presentation platform only. They try to adapt the appearance of virtual objects as good as possible to the visual circumstances of the real world environment. On the other hand, non-photorealistic AR visualizations try to convey a meaning (like the spatial relationship between objects, or functional information, like which button to press next to turn on a machine). While ordinary AR presentations simply render virtual content over real world imagery, hoping that it fits into the environment, comprehensible visualizations trade realism for communicative emphasis. This thesis focuses on comprehensible x-ray visualizations. This section presents the authoring and rendering of Focus and Context visualizations in AR environments.

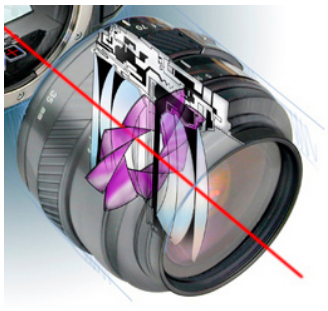


Figure 3.8: Illustrative x-ray visualization. Notice, how the hidden object is shaded in a way which communicates its importance, while occluding structures are presented as context without distracting the user's attention (Image courtesy of Beau and Alan Daniels, www.cutaway-illustration.com).

In case of an x-ray visualization, a comprehensible presentation of hidden and occluding structure has to be generated. Therefore, both have to be identified easily. However, the presentation has to simultaneously encode the fact that hidden objects are the most interesting elements in an x-ray visualization. For example, Figure 3.8 shows an illustration of the otherwise hidden mechanics of a camera, which are presented in combination with the occluding casing. While the occluder is visualized using very de-saturated colours, the hidden aperture appears in highly saturated purple. By using different level of saturation, the visualization is able to draw the user's attention to the object of interest (see section 2.3 for other types of visual encodings)

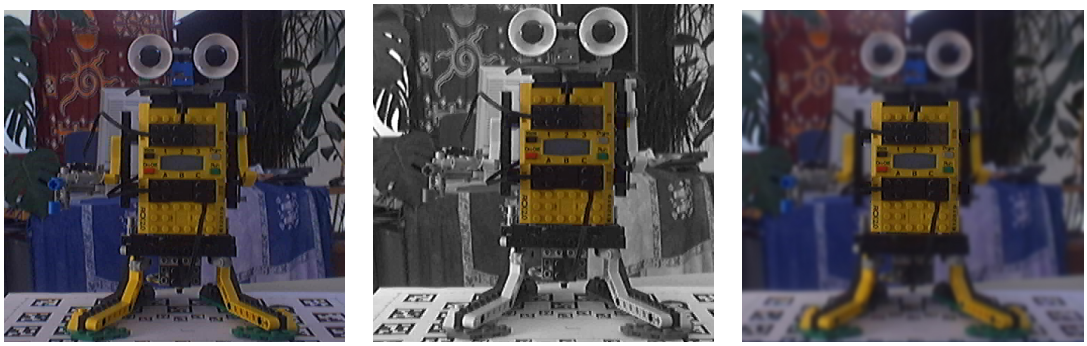
3.3.1 Single-Level Focus and Context Visualization

Conventionally, F+C visualizations consider a single level of discrimination into focus and context. However, if the presentation's environment is very complex, single level F+C classifications may result in a presentation of visually isolated elements. Figure 3.10 and Figure 3.9 demonstrate the different effects of F+C visualizations in purely virtual compared to complex AR environments. All visualizations use a single level of F+C discrimination (meaning that visual elements either belong to the focus or the context group). The goal of all visualizations is to direct the user's attention to the central part of the robot. It can be seen that the visualizations in Figure 3.9 effectively communicate their message, while the renderings in the real world environment are more difficult to

(a) (b) (c)

Figure 3.9: F+C Visualization in a Virtual Environments. (a) Rendering of a Lego robot in a VR environment (b) F+C visualization in VR using de-saturation to suppress visual attention of context structure (c) F+C visualization in VR using blur to communicate the F+C roles

read. For example, even though the focus clearly stands out (as in Figure 3.10(d)), its remainder cannot be discriminated from the information in the background. In contrast, the same visual discrimination is effective in purely virtual environment (Figure 3.9(c))



(a) (b) (c)

(d) (e) (f)

Figure 3.10: F+C Visualization in complex Real World Environments. (a) The Lego robot presented in a real world environment without any stylization (b) The F+C visualization is using de-saturation to suppress visual attention of context elements. (c) F+C visualization by applygin a blur on contextual fragments (d) F+C visualization using de-saturation. In contrast to the visualization shown in image b, the F+C roles are identified in object space resulting in a de-saturation of only those structures which are in direct contact with the focus object (e) F+C visualization using the same classification as image d, but applying a blur to visually communicate the roles.(f) A combination of F+C communicators. Context information is visually suppressed using blur and de-saturation, while saturation is increased at focus elements.

As outlined in section 2.3, some applications add weights [40] to object classifications, so that membership to focus or context is no longer binary but continuous. However, 3D AR deals with objects presenting real world (often complex) information which demand more complex classifications to keep important information visible in contextual areas. For example, in classical F+C illustrations, such as for textbook figures, no disturbing background is present allowing classical F+C visualizations to be very effective (similar to Figure 3.9). However, in a real-life AR environment, we cannot limit the range of visible objects exclusively to those which are mostly needed in the presentation. Consequently, F+C visualizations have to consider a higher amount of information which consists of useful but also of possibly distracting elements. This often causes traditional F+C visualizations to fail in AR environments(Figure 3.10).

3.3.2 Multi-Level Focus and Context Visualization

As Figure 3.10 demonstrates, a single level of Focus and Context discrimination is often not sufficient to generate comprehensible presentations in complex real world environments. Even though continuous degree of interest functions are able to generate a wider range of classifications, F+C visualizations using a single level of classification do not ensure a suitable discrimination in AR. Even classifications, which consist of more than two groups of objects may result in incomprehensible renderings, if multiple F+C visualization have been applied heedlessly (Figure 3.10(f)).

Thus, this thesis proposes to render a combination of F+C visualizations, which have been carefully combined in a way to fit to each other. Their visual harmonization is controlled by applying a manually designed template which consists of combinations which are known to harmonize with each other. The presented approach aims to classify the data through cascading F+C classifications which results in an implicit multilevel F+C separation, where F+C separators are applied consecutively on the results of previous separation steps. Since the emerging template only assumes a certain object classification it can be applied to similar object arrangements generating similar visualizations. For example, Figure 3.13(b) is using the same visualization template than the image in 3.11. In both examples, it can be seen that surrounding real world information is preserved while at the same time the users attention is similarly well directed to the focus element as within a single level F+C visualization in a virtual environment.

To avoid incomprehensible visualizations, the F+C visualization template which was used to render Figure 3.11 uses a set of different in-place Focus and Context presentation

Figure 3.11: Multi-Level F+C Visualization

techniques (see section 2.3.2 for a list of in-place encoding techniques) which are carefully combined in a way to fit to each other. Their visual harmonization is controlled by applying a manually designed combinations of visualization techniques which are known to harmonize with each other.

To enable an automatic harmonization in a future system, the following paragraph will outline the rules which have been applied manually to ensure a visually pleasing complementation of techniques. The rules will be described by explaining the local and global goals of the applied visualization techniques.

The visualization template first directs the attention of the user to both, the actual focus element (the center of the robot) and the objects in its semantic proximity (the extremities of the robot). This is achieved by reducing the visual importance of the background information (the first level context) using a de-saturation (computed by reducing the saturation component in HSV color space) and a blur encoding (computed by iteratively apply a Gaussian function in a 7x7 window). In addition, both the focus and its direct context (the extremities of the robot) are framed by applying a halo rendering (the halo is computed using the algorithm proposed by [15]). Besides discrimination between background and foreground objects, this step of the visualization puts emphasis on the close relation between the focus object (the central part of the robot) and the extremities of the robot. To be able to further guide the user's attention to the actual object of interest, a second (knowledge based) separation splits the focus from its proximity. To ensure harmonization, the F+C presentation is configured to again apply a de-saturation to the current contextual information while the focus is framed. However, to not to break the

previously introduced visual connection between the focus and the contextual data, a less strong F+C discrimination is applied. Thus, instead of directly rendering the frame, the halo values are only used to control the saturation values in the contextual area. Since, a de-emphasis often comes to the prize of a reduced clarity, another F+C separator is applied to increase the clarity of characteristic shape information (which are computed by appling and edge detector). To prevent the visualization technique from voiding out previously applied F+C techniques, emphasis is put on focus elements using an encoding which was not used before. (in this case, the brightness of the focus elements is set apart from contextual information (the edges have been darkened).

To ensure harmonizing combinations of F+C visualizations the example makes use of two rules. The visualization uses the same F+C visualization techniques to distinguish the central part of the robot from the elements in its direct context as it uses to distinguish foreground and background objects. The visualization applies a higher de-saturation and a second visual discriminator (the blur) on lower level context information (where the first F+C discrimination defines the lowest level). In general these following two rules have been applied to choose the techniques and to define there parameter:

- Suggestion 1: Higher level context uses a subset of lower level encodings, or if only one encoding was used, higher level context applied the same encoding technique. In both cases, parameters are set to encode the F+C roles less strong.
- Suggestion 2: Higher level focus uses different techniques than lower level F+C presentations

3.3.2.1 Implementation

The implementation uses the framework's scene graph, besides for rendering the scene, also as the software infrastructure to combine F+C shader. Exploiting a graph for configuration as well as rendering has a number of advantages. It is easy to create and to understand, and it allows the explicit modeling of the control flow and data flow of the rendering process. It directly makes use of important scene graph capabilities like in-order traversal to visit the nodes in a certain order, while field connections enable out-of-order data flow dependencies.

```
SoGBShader{
        SoSFString   gBufferName;
        SoSFString   fragmentShader;
        SoSFString   vertexShader;
        SoSFString   geometryShader;
        SoMFString   parameterNames;
        SoMFEnum     parameterTypes; #UNIFORM1F,#UNIFORM2F,...
        SoMFString   parameterValues;
        SoMFEnum     readFrom; #COLOR,NORMAL_DEPTH,AUX0,AUX1,...
        SoMFEnum     writeTo;  #COLOR,NORMAL_DEPTH,AUX0,AUX1,...
}
```

Listing 3.1: Interface of the basic GB-Buffer processing node

Thus, all elements of the visualization framework are implemented as node extensions in Coin3D. This makes the system easily extensible, while the file format of Coin3D allows convenient user scripting of all content, facilitating the development of authoring tools for end users. To process the content of a specific G-Buffer, we have implemented a scene graph node (*SoGBShader*) which executes a single GLSL-shader at a time. This node has to be configured to define the buffers (out of all available G-Buffers) a GLSL-shader reads

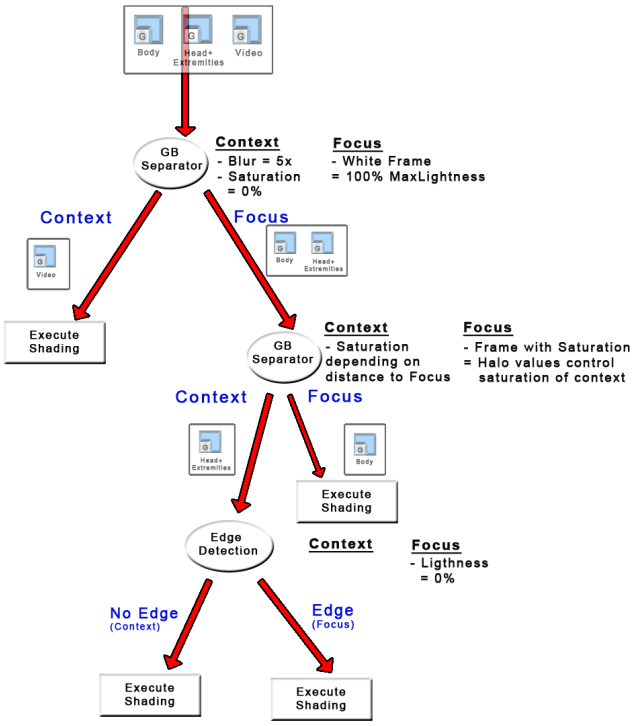


Figure 3.12: Illustration of the combination of F+C visualization techniques.To ensure visual harmonization higher level context elements use lower level visualization techniques with decreased values.

.

from, as well as to define to which buffer of the currently processed G-Buffer it writes to. The *SoGBShader* inherits from *SoSeparator* and executes the defined GLSL-shader on the nodes below its path. To execute the shader script for each pixel in a G-Buffer, the *SoGBShader*-node usually contains a (*SoOrthoQuad*-node which renders a screen aligned quad.

The *SoGBShader*-node allows a very flexible configuration of the G-Buffer processing. However, as soon as scene stylization becomes complex, an explicit authoring of each single step in the G-Buffer processing pipeline becomes challenging. We therefore encapsulate the atomic *SoGBShader*-nodes into a higher level abstractions (the *SoFCShader*), which allows to focus on the combination of visualization techniques while designing the stylization. For example, to compute a blur or a halo effect, multiple iterations over a predefined set of *SoGBShader*-nodes have to be executed. Depending on its configuration (see listing 3.2 for an example) the *SoFCShader* generates the necessary *SoGBShader*-nodes.

```
SoFCShader{
        shaderPath      = USE SHADER_PATH.oa
        focusGBuffer ["gbExtremities","gbBattery"]
        focusShader [FRAMING]
        focusShaderValues ["5","1 1 1","1"] #[Size,RGB,A]
        contextGBuffer ["gbVideo"]
        contextShader [SATURATION,BLUR]
        contextShaderValues ["0.0","4"]#[SaturationValue,NumberOfGaussIterations]
        contextBranch SoFCExecute{}
        focusBranch SoFCShader {
                            shaderPath = USE SHADER_PATH.oa
                            focusGBuffer ["gbBattery"]
                            contextGBuffer ["gbExtremities"]
                            contextShader [SATURATION_BY_FOCUS_FRAME]
                            contextShaderValues ["3","1.0"]#[Size,MaxSaturation]
                            focusBranch     SoFCExecute{}
                            contextBranch SoDetectEdges{
                                detector "fs_gradient.glsl"
                                parameterName ["threshold"]
                                parameterType [UNIFORM1F]
                                parameterValues ["0.7"]#[ThresholdValue]
                                inputBuffer [COLOR]
                                focusShader [LIGHTNESS]
                                focusShaderValues   ["0.0"]
                                contextBranch SoFCExecute{}
                                focusBranch SoFCExecute{}
                              }
                    }
}
```

Listing 3.2: Example configuration of combinations of FC visualization techniques

3.3.2.2 Multi Level Focus and Context X-Ray Visualization

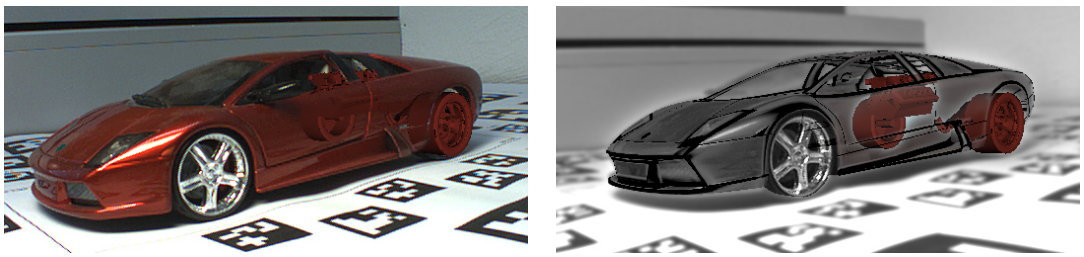
The F+C template is a powerful tool, but it is independent of the current viewing direction and therefore does not take occlusions into account. However, the goal of this thesis is

displaying x-ray visualization to reveal hidden structures. To allow transparency within Multi-Level F+C (MFC) classifications, we have to define a behavior for all different combinations of overlapping MFC classified fragments.

To render MFC x-ray visualizations the system has to analyze the F+C affiliation during image compositing for each fragment. A fragments highest F+C affiliation is used in the current implementation, resulting in either focus or context information along a viewing ray. By applying the following rules, the most important elements will be selected if fragments overlap.

- Situation 1 (S1)- Focus covers other Focus: In this situation the system follows the current depth arrangements. The one closest to the camera will be used, irrespectively of the position in the FC graph.
- Situation 2 (S2)- Context covers other Context: Similar to S1, the system follows the current depth arrangements
- Situation 3 (S3)- Focus covers Context or Context over Focus: The system always favors focus over context fragments, no matter which level they are in the hierarchy

Notice the clarity of the information in Figure 3.13(b) compared to the image in Figure 3.13(a). The improvement appears due to the control of the MFC stylization template. The combination of visualizations was designed to ensure a homogeneous stylization with the goal to direct the user's attention. However, Figure 3.13(b) also demonstrates that rather heavy modifications may result by applying a stylization template. In order to reduce to amount of real world modifications, the stylization may be restricted using an

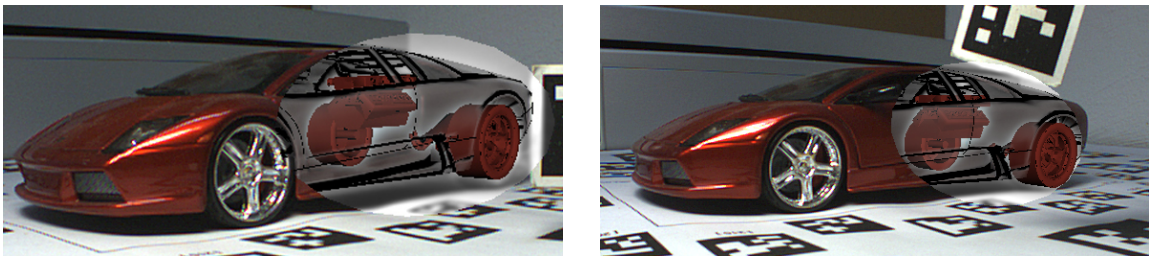


(a) (b)

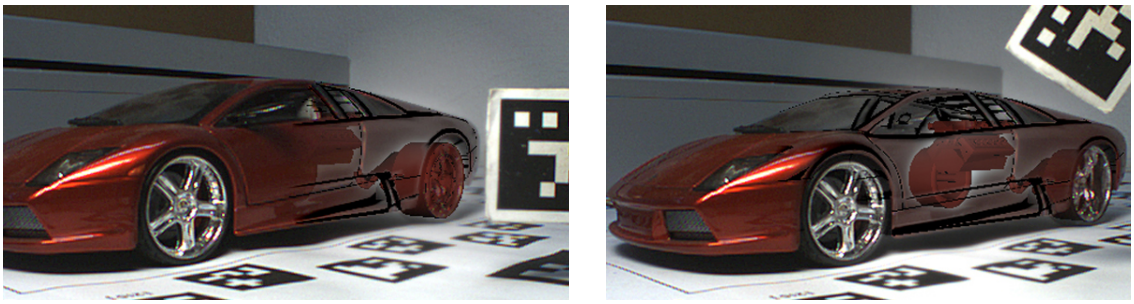
Figure 3.13: Multi-Level F+C (MFC) X-Ray Visualization. (a) Problematic visual interaction between virtual and real world elements (b) The same stylization as demonstrated in Figure 3.11 is applied

interactive 2d mask (see Figure 3.14(a)). To further avoid abrupt changes Figure 3.14(b) demonstrates a soft blend between the MFC visualization and the current video data. The blending is controlled by rendering a gray scale texture on a flat circle, which is used to generate the lens mask. The texture values are used scale the opacity values of video information up, from the center towards the border of the lens, while opacity values of the MFC visualization decrease inverse, so that both sum up to one (see equation 3.1).

$$rgb = (1 - lensTextureIntensity) * mfcVis + (lensTextureIntensity) * video \quad (3.1)$$



(a)



(b)

Figure 3.14: Masked MFC Visualization to reduce the modification of real world imagery. (a) A 2D Magic Lens controls where the stylization is used (b) The stylization is blend with video data using a blend values which are introduced by an intensity texture of the Magic Lens.

Chapter 4

Ghosting

Content

The perception of spatial relationships between objects in 3d space is guided by a number of depth cues (section 2.2). If hidden information is in the focus of a presentation, non-uniform transparency modulations have been successfully applied to preserve depth cues. As outlined in section 2.4.1, a number of different methods have been proposed to automatically generate a comprehensible ghost representation (non-uniform transparency

modulation) of occluding structure in illustrative renderings. This chapter presents methods to automatically modulate the transparency of real world structure to reveal virtual objects in an AR environment. This chapter first describes different algorithms to attribute the pixel of a video stream with a certain transparency value (section 4.1). Subsequent to the classification, algorithms to comprehensibly present the computed ghost in an AR environment are demonstrated in section 4.2, while a discussion on possible sources of errors and attempts to overcome them will close this chapter.

4.1 Classification

To be able to modulate transparency values of fragments representing real world occluding structure, the system has to first identify those pixel in the video feed. A common method to handle occlusions in AR is *Phantom Rendering* [13]. Phantom objects (which represent 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, by rendering a virtual counterpart of each real world object, the application will reject hidden virtual fragments using using ordinary Open-GL depth testing. This reveals occlusion by allowing only visible fragments of virtual structure to pass (Figure 4.1). The phantom algorithm can be outlined by the following steps:

Algorithm 1 Phantom Rendering

1. Draw Video
2. Disable writing to color buffer (glColorMask or glBlendFunc(0,1))
3. Render virtual representations of real scene (Phantoms)
4. Enable writing to color buffer
5. Draw virtual objects

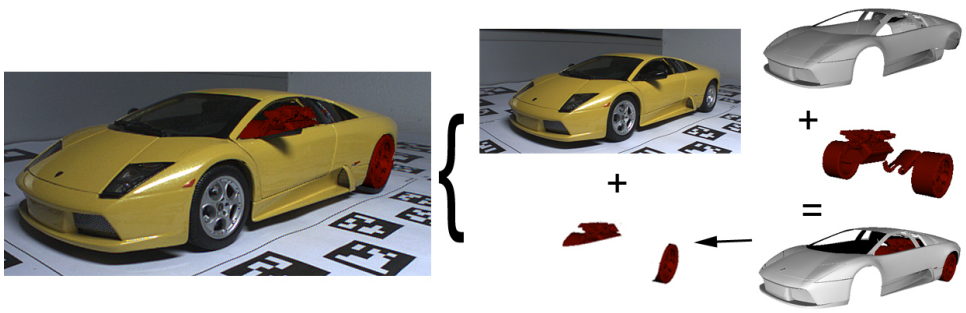


Figure 4.1: Phantom Rendering.

4.1.1 Object based Ghosting

From looking at the results of the algorithm it is clear that phantom rendering only prevents virtual fragments from being rendered on top of real world information. The algorithm does not specifically identify real world pixel, which can be considered to turn 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 (Figure 4.2). 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 (Figure 4.2 left side), a texture lookup at the fragment's position after its projection to image space provides the output values (Figure 4.2 right side).

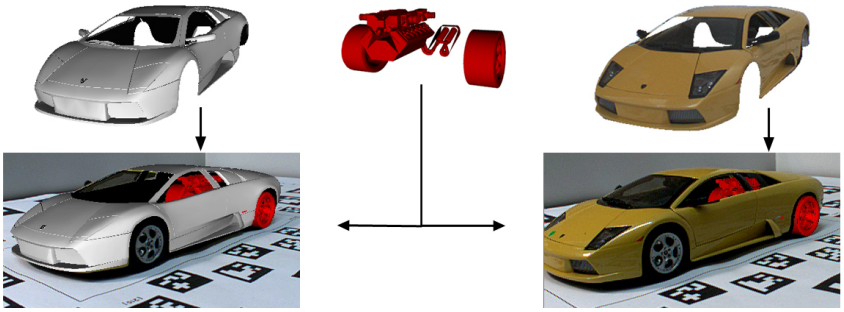


Figure 4.2: Video Phantoms. Since ordinary phantom objects resolve occlusions by rejecting occluding virtual fragments occluding structure are not directly identified. Video phantoms (image on the right hand side) make use of current real world video information instead of virtual shades.

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 4.3 shows an example of a ghost rendering, which was generated from a 3d video phantom. The stylization uses principal curvature information [81] of the registered 3d mesh. The curvature values are defined per vertex and computed in an offline process before the application is started (the implemented system uses Stanford's trimesh* library to compute principal curvature values). During rasterisation the curvature values are interpolated between vertices, so that for each fragment (and in case of a video phantom each video

*http://www.cs.princeton.edu/gfx/proj/trimesh2/

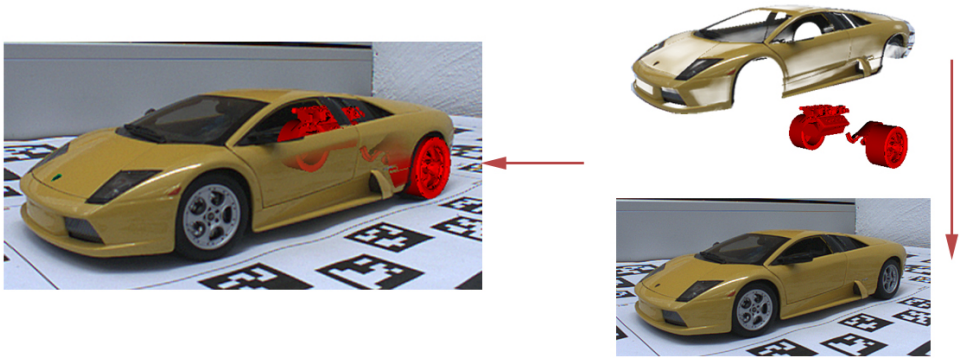


Figure 4.3: Non-Uniform Transparency Modulation using a Curvature Analyses of a Video Phantom.

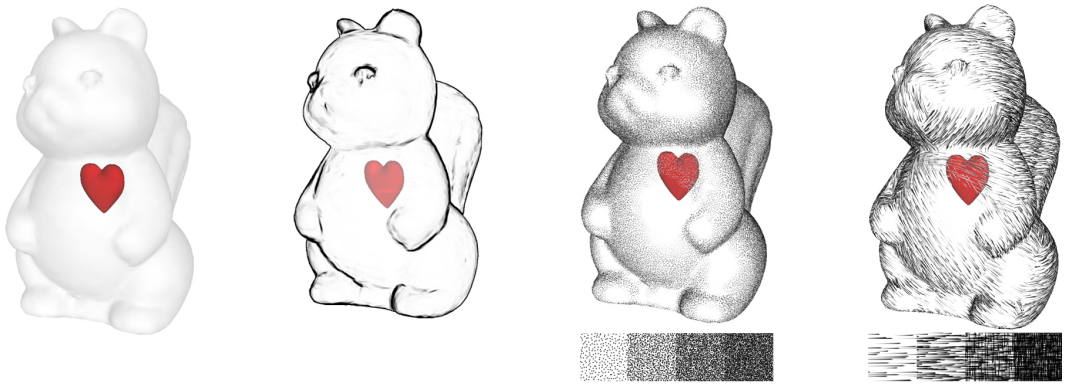
pixel) its maximal curvature value is carried out. Finally, curvature values are linearly mapped to transparency values using equation 4.1. To control the object's transparency the parameters k and *tShift* have to be modified. While *tShift* linearly changes the mapping from curvature to opacity values, 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.

$$O_i = \frac{C_i}{k} + tShift;\ k \neq 0.0 \tag{4.1}$$

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

An object based feature preservation works well for objects which provide enough characteristic features. However, some shapes do not consist of enough features to be well preserved (see Figure 4.4(a)). To introduce features, stroke textures which follow principal curvature directions were proposed [68](see section 2.4.1). This technique is proven to increase the understanding of shapes [77]. Moreover, automatic stippling techniques (which abstract shading only with a set of uniformly but randomly distributed dots) have been successfully applied in NPR to abstract shading while preserving shape information. Consequently, the system presented here is able to apply stroke and stipple textures to registered 3d video phantom objects (Figure 4.4).

As demonstrated in Figure 4.5, hatching and stippling techniques provide a powerful tool to comprehensibly control the amount of occlusions for an x-ray visualization also in AR. Since the density of strokes or stipple elements varies propotional to object's intensity,



(a) (b) (c) (d)

Figure 4.4: Non-Photorealistic Rendering to Preserve Depth Cues. (a) A uniform transparency modulation is unable to communicate the relation between hidden and occluding structure. (b) As noticed by Interrante [68] and Hodge [63], some shapes do not consist of enough characteristic features to communicate shape and thus depth cues. (c) Stippling to visualize occluding structure. The transparency value depends on the brightness of the stipples which allows to preserve a few stipples in front of hidden structure (d) Hatching is used to present the occluding shape. Transparency depends on a fragments brightness value.

we are able to interactively change the density and thus the opacity of the occluding structure by changing the illumination of the object (e.g. by changing the position or intensity of the light source, similar to the approach in [14]).

The images in Figure 4.4 demonstrate the effect in a virtual environment and Figure 4.5 and Figure 4.6 show their integration in a real world environment. It can be seen, that preserving a few stipples or a few strokes in front of the hidden object enables the visualization to indicate the transparent surface in a VR (Figure 4.4) and in low frequent AR scenes (Figure 4.5). However, if a hatching texture is applied on a real world texture with higher frequencies, the texture patterns may interfere, resulting in an ambiguous presentation (Figure 4.6(b)). In contrast, stipplings introduce less prominent artificial features. Consequently, they interfere less with real world texture information (compared to stroke patterns), why we consider them as a safer technique to introduce artificial occlusions in unknown scenes with potentially high frequency textures(Figure 4.6(b)).

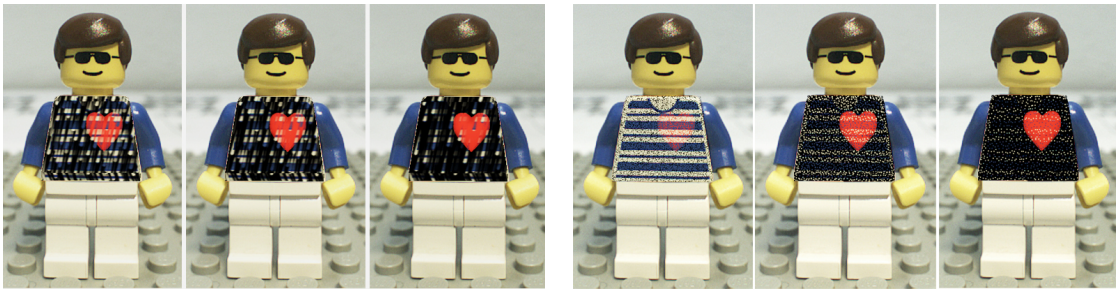
The implementation of a ghosting with artificial hatch and stipple pattern uses the real time approach of Praun et al. [99]. It applies Tonal Art Maps (TAM) using a set of precalculated texture coordinates [98], which control stroke directions and stroke

(a) (b)

Figure 4.5: Ghosting using Hatching and Stippling on Low Frequent Textures in AR (a) Hatchings are able to communicate the occluding shape information with few strokes (b) Stippling need higher opacity setting to preserve the shape information

scale depending on the shape of the occluder. As suggested by Interrante, our texture coordinates follow the principal curvature. However, since the vector field is computed from curvature information of an ordinary 3d mesh, the derived directions are prone to irregularities (see [132] for details). Thus, our implementation smoothes the initial direction filed depending on a user controllable value which is similar to the system of Schlechtweg and colleagues [132] (see [38] for more details on our implementation of the real time hatching approach of Praun et al. and the quality improvements of Schlechtweg and his colleagues).

Since our system applies an automatically computed Tonal Art Map on a set of automatically generated texture coordinates (which are consistently distributed over an arbitrary 3d mesh), the system is able to easily change the TAM, in order to control the way opacity values are being altered. Figure 4.5 demonstrates the results of a transparency modulation using two different TAMs (a stippling and a stroke texture). This flexibility allows us to choose a transparency modulation depending on the parameters of the current scenario. For example, if applied to a low frequency real world object, a hatching seems



(a) (b)

Figure 4.6: Ghosting using Hatching and Stippling in AR on High Frequent Objects (a) Hatching is not aligned with real world texture which results in a disturbed appearance of both (b) Stippling is less interfering with real world texture information if used to control preservation of occluding fragments

to be more effective in communicating shape information than stipplings (Figure 4.5). This is because the bending of the strokes implicitly encodes shape information, while stipplings have to preserve a higher amount of the occluder to communicate a comparable amount of shape information. However, if a hatching is applied on high frequency real world texture, the strokes may interfere with the real world texture, resulting in an ambiguous presentation (Figure 4.6(a)).

However, an automatic verification of the current situation followed by a selection of the TAM which fits best to the appearance of the occluder is left for future work. Moreover, a future system will be able to alter texture coordinates online, to align with the directions of real world texture. This will allow to apply a consistent preservation of real world texture information in combination with object based features.

4.1.2 Image based Ghosting

Using a registered video phantom allows to create ghost presentations in AR by analyzing the parameter of a registered 3d mesh. However, an important source for AR scenes is the video stream used for video see-through augmentation. Since a ghosting from a video phantom only depends on the parameter of a 3d mesh, the result does not take into account important real world information such as landmarks or texture. In addition, features like the silhouette of an object are view dependent and thus have to be computed online for the current point of view.

Following this line of thought, the system has to identify an occluder's ghost presen-

tation not only in object space, but also in image space. Since the rendering algorithm described in chapter 3 is only dependent on the fact that all input data is available in a G-Buffer, the sources of data can vary. Thus, a video stream delivered into a G-Buffer can be used as a source for stylization even without a registered phantom. Feature classification to subsequently compute a ghost presentation is implemented as a set of image based operations (see chapter 3.1 for details on the rendering algorithm). Notice, that if no phantom object exists, depth order has to be defined manually on a G-Buffer level. In addition, if no phantom object exists, depth order can not be computed, and thus care has to be taken to classify scene content into non-intersecting G-Buffers for any point of view.

Image space classification is implemented in addition to object space classification. This allows either to combine both techniques or to flexibly switch, depending on the demands of the specific algorithm. For example, since object space approaches allow greater freedom for line parameterization and thus for further stylization, consistent line stylizations are preferably computed in object space [69]. G-Buffers allow to detect crease and silhouette lines easily and fast in image space [110]. In addition, image space operations follow the rendering of 3d scene content into 2d image presentations. That implies the possibility to further classify phantom objects in 2d image space allowing to consistently apply a set of operations on both 2d and 3d data sources.

To generate ghost presentations from a set of G-Buffer, image operators have to identify characteristic features of the occluding objects. Research on line rendering has achieved a

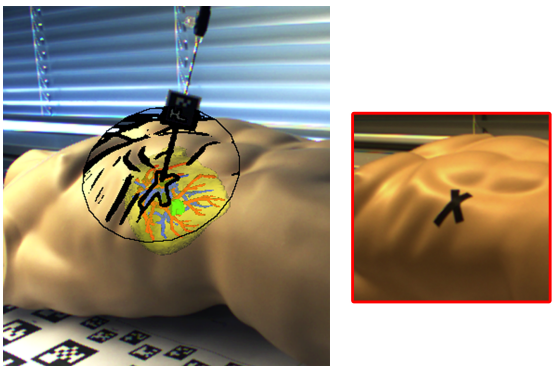
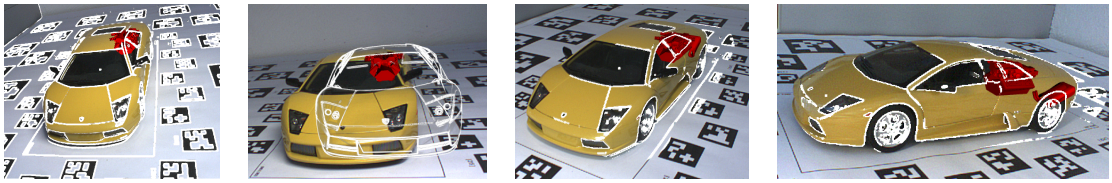


Figure 4.7: Image based Ghosting using Edge Preservation (a) Preserving edges on the entire image clutters the presentation. An interactive Flat Magic Lens allows to control cluttering video ghosting. Edges have been preserved by operators in 2d image space only. Each video frame is processed and discrete edges are detected using and ordinary edge detector (like the canny operator [21]



(a) (b) (c) (d)

Figure 4.8: Hybrid Ghosting with Phantom Objects. (a) Image based ghosting results in cluttered images (b) Misregistered Phantom Object (c) Edges have been masked with the misregistered phantom object(d) Hybrid ghosting has to be applied with care because the registration error may be close to invisible from certain perspectives if edges align with real world structure (Compare with Figure 4.16)

number of algorithms to compute a silhouette and crease lines in 2d image space as well as on 2-1/2d G-Buffer data structures. For example, if 2d video data is the only source for augmentation, ordinary edge detectors (like the well known canny operator [21]) may be applied to each of the video frames. Figure 4.7 shows an example of edge preservation by applying an edge detector to the current video feed. Notice, that by using only the video data as a source for preservation, other features than those which belong to the occluding object have been detected. For example, Figure 4.7 shows a simulated radio frequency ablation of a liver tumor. Besides the edges on the puppet also the edges of the ARToolkit marker and the needle (which is inserted into the tumor) have been detected and emphasized in black.

Even more disturbing than edges from unwanted objects, is that edges extracted from video may clutter the whole view (Figure 4.9(a)). As a remedy, a hybrid approach using features from video and tracked objects as stencil masks is able to reduce the image clutter. Figure 4.7 has been enhanced with edges from the video stream which were stenciled by the region covered by a Flat Magic Lens (see section 2.3.1). 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.

Thus, other 3d objects may be introduced and tracked relatively to the hidden object to generate the mask. For example, if a phantom object is available but cannot be used because of its quality, it may still be sufficient to generate the mask. The images in Figure 4.9 demonstrate the enhancement of edges from the video stream, which were cut by the 2d footprint of the rendering of the miss-registered phantom object (which is shown in

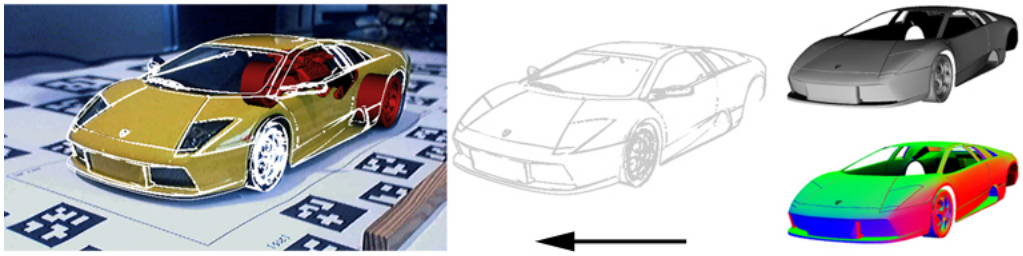


Figure 4.9: Ghosting by Edge Preservation. Edges have been computed on 2-1/2d G-Buffer information. In this case, discontinuities in the normal and depth buffer have been used.

Figure 4.9(b)). Even though the tracked object is badly registered, it still can be used to limit the area of edge preservation. This technique enables us to use the registration quality of edges from video without cluttering over the whole screen. Notice, that under a certain perspective, the registration error is almost invisible because its 2d footprint closely aligns with its real world counterpart (Figure 4.9(c)). Thus, hybrid ghosting has to be carefully applied. If the ghosting uses a mask which is generated from a miss-registered phantom object, the resulting visualization hides the registration error. Consequently, hybrid ghosting in AR can only be a matter of choice if the registration error of hidden structure is irrelevant to the application. Otherwise, the error has to be communicated (see the section 4.2.1.4).

Edge preservation computed directly on a video stream provides a powerful tool to overcome the necessity for 3d virtual counterparts of real world objects in AR environments. However, feature preservation on highly textured structures will likely result in a number of features which are irrelevant to communicate shape information. Moreover, a number of curvatures of occluding structure will probably not be detectable in the video frame under e.g. poor illumination. Thus, to be able to avoid under or over detection, a certain quality of the data source has to be ensured. For example, the lighting in Figure 4.7 had to be manually controlled by using an extra lamp which is positioned above and in an angle of about 45 degree to the puppet.

If in addition to video data, other data sources are available, a combination of feature detection may produce better results to communicate shape information. For example, to generate line renderings in VR, Hertzmann et al. [62] proposed to search for discontinuities on both, normal and depth buffer data. Since our rendering framework allows to generate the proposed data sources we are able to apply Hertzmann's technique to the rendering of a phantom object (see Figure 4.9).

In case the system can choose from multiple sources, the choice of which data source to use, depends on one hand on the demands of the algorithm, which generates the ghost presentation (e.g. a 3D mesh analyzer requires a 3d mesh to input) and on the other hand on the quality of the sources. For example, ghost presentations may be easier obtained from renderings of a virtual phantom objects (or if possible, directly on a 3d mesh analysis) than from real world video data. This is because rendered objects are not affected by image noise or poor illumination and are therefore processed more easily. However, generating a ghost presentation from a registered 3d phantom object is prone to registration errors. This can be a consequence of a mismatch between real and virtual objects, poor camera calibration or tracking errors. Combinations, which use either one of them, have to analyze the current data sources to choose the best stylization in certain situations. Adaptive ghost generations are left for future work.

No matter where the features (e.g. the edges) come from, the resulting ghosting only consists of either fully transparent or completely opaque structures if a discrete thresholding is used. Therefore, to compute a transparency fall-off in 2D image space, one can either apply an operator which extracts smoothly changing image features (which directly map to transparency values) or one computes transparency values in a subsequent operation using previously detected features as input. For example, Figure 4.10 and Figure 4.12 applies transparency values depending on the result of a 2D distance transform which is computed from a set of previously detected features. The features were computed by calculating the sum of the squared differences of the intensity values in a 3x3 window (see equation 4.2). The distribution of the feature values were computed using the halo operator proposed in [15].

Figure 4.10: Feature Based Transparency Distribution. Edges have been detected in image space and distributed using a 2d distance transform

$$F_i = \frac{\sum_{j=1}^{9} (x_j - X)^2}{9}; \; X = \sum_{j=1}^{9} x_j \tag{4.2}$$

4.2 Presentation

The previous section presents techniques to non-uniformly modulate transparency of occluding structures. The transparency values are computed from either 3D registered 3D phantom objects or from the system's video feed itself. However, depending on the complexity and the quality of the data, different presentation techniques have to be considered in AR environments. This section will first discuss variations of ghost presentations of single object occlusions in AR, followed by a set of filter techniques to allow the combination of ghost presentations of even multiple occluding objects.

4.2.1 Single Object Occlusions

To limit the range of influencing factors, this section only considers ghost presentations of a single occluding object. The subsequent section focuses on presentation techniques allowing to also handle multiple object occlusions.

4.2.1.1 Video Ghosting

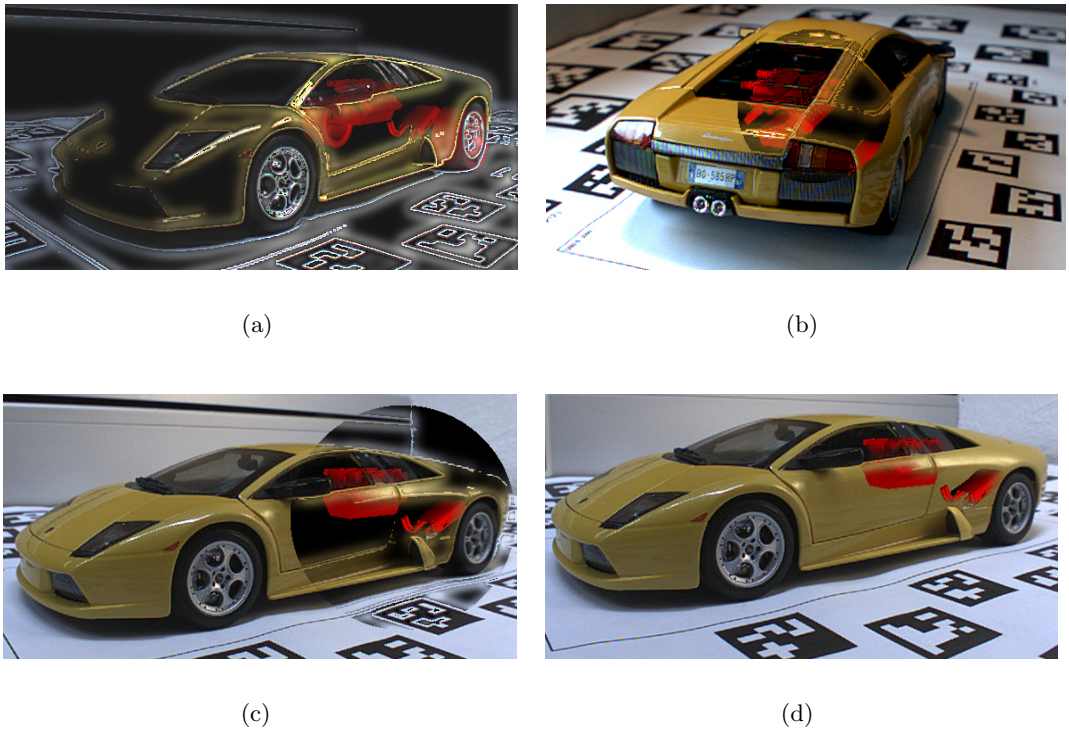
If a perfectly registered virtual model of otherwise hidden structure exist, the ghost presentation may consist of a nonlinear transparency modulation of real world occluding structure, which is presented over the rendering of the virtual objects (Figure 4.11) shows

(a) (b)

Figure 4.11: Ghost Presentations using Video Phantom Objects (a) Dense ghost presentation (b)Sparse ghost presentation

an example of two different opacity settings in such a configuration. Such ghost presentation of video information allows to see the inner parts of the car while believably presenting the occluding information

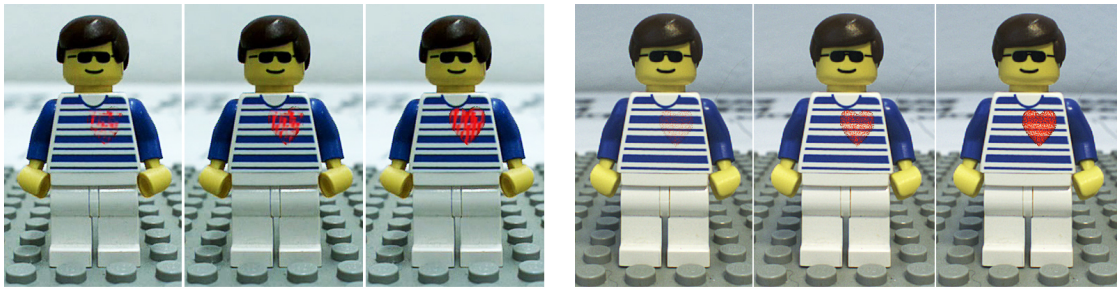
However, Augmented Reality environments often suffer from incomplete virtual data. Often only the video in combination with the object of interest is available to the AR system. In cases where everything else is missing, the resulting ghost presentation may be difficult to understand (Figure 4.12(a) demonstrates the problem). Even if we restrict the area of ghosting with e.g. a Magic Lens tool Figure 4.12(b) or the enlarged footprint of the object of interest, the missing information becomes exposed as soon as the area of augmentation becomes large enough (Figure 4.12(c)). Thus, if little information is present behind the visible structure, we propose to instead present the video information in the background (Figure 4.12(d)). Since the current implementation only allows to



(a) (b)

(c) (d)

Figure 4.12: Image Based Video Ghost in Incomplete Scenarios (a) (b) The Magic Lens tool which is able to restrict the ghost presentation to directly occluding information. This limitation of augmentation is able to limit the missing virtual objects (c) However, if the Magic Lens tool has to cover a large area in 2d image space the empty environment becomes visible again (d) Video background has been added to hide the lack of information



(a) (b)

Figure 4.13: Video Ghosting with an Artificial Pattern (a) Video ghosting using a hatch pattern to preserve occluding structure. The pattern is not communicated. Both, hidden objects and occluding texture are difficult to understand (b) Stipple pattern cause less artifacts. However, the hidden object seems to fly in front of the occluder

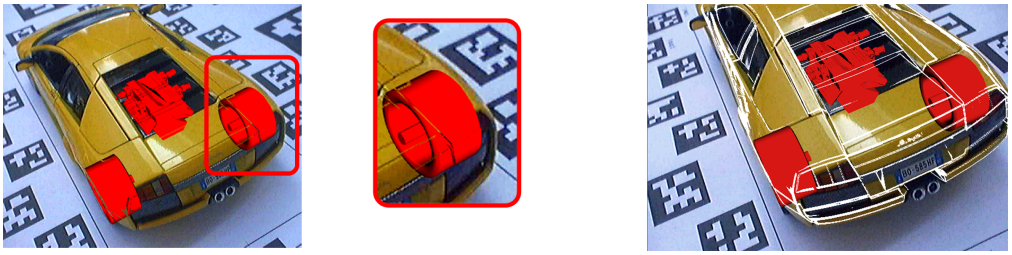
manually configure which information to use in ghost presentation, a future system should be able to detect the current amount of hidden information to be able to adaptively alter the presentation technique.

Notice furthermore, if an artificial pattern preserves occluding information, an additional video background may hide the applied pattern. To be able to communicate the artificial pattern, it has to appear as all over the object (see Gestalt Laws in section 2.2 and compare Figure 4.13 with Figure 4.6).

4.2.1.2 Virtualized Ghosting

The limits of human perception impose a trade-off between the amount of preserved features and the clarity of revealed hidden structures in AR environments. Therefore, if the visualization demands a high visibility of hidden structures, a very sparse representation has to be used for occluding objects. However, such very sparse representations necessarily provide only a small amount of features, which may render it difficult to mentally reconstruct the object's shape, especially if its appearance is similar to its surroundings as it appears in video ghostings. Figure 4.14(a) shows such a very sparse preservation of real world imagery. Notice the difficulty to perceive the preservations of the real world imagery.

The visualization should aim to contribute features of optimal clarity. In the case of x-ray visualization occluding shape has to be communicated. Thus, if very sparse representations have to be used, the clarity of the presentation increases if features appear



(a) (b)

Figure 4.14: Sparse Ghost Presentation. (a) Sparse real-world preservations may be hard to perceive.(b) A uniformly colored accentuation of a sparse ghosting is able to provide a good impression of the occluding object.

similar to each other (see the Gestalt Laws in section 2.2). Characteristic features of sparse representations are more comprehensive, if they have a consistent tone or color, which stands in strong contrast to their surroundings. Unfortunately, real world imagery does not guarantee that features appear following these criteria. We therefore use virtual colors for the ghost presentation (as demonstrated in Figure 4.14(b)). We call this a virtualized ghost. As outlined by Hodges [63], hidden structures do not shine through transparent surfaces where high light reflections appear. Therefore, the virtual ghost presentation tries to mimic light reflection by applying unsaturated and very light colors, typically shades of white. A virtualized ghost is computed using equation 4.3.

$$\begin{aligned} RGB_i &= NdotL_i * intensity * ghostColor + eShift \\ O_i &= \frac{C_i}{k} + tShift;\ k \neq 0.0 \end{aligned} \tag{4.3}$$

i:Current fragment; RGB: Resulting color value; NdotL: Dot product of normalized normal and light vectors; O: Opacity; C: Maximal curvature value; k: Curvature weight; tShift: Transparency shift; eShift: Emphasis shift

4.2.1.3 Emphasized Ghosting

Since virtualized ghostings add a virtual representation of the occluding structure, independent of its level of sparseness [†], dense virtualized ghostings will cover a high amount of real world information (see Figure 4.15(A)). To preserve real world imagery in dense configurations and to communicate the occluding structure effectively in sparse configurations, we apply an emphasis of the ghost presentation which is dependent on the occluder's current level of sparseness. By applying equation 4.4, only sparse representations will appear in synthetic shades.

$$\begin{aligned} O_i &= \frac{C_i}{k_1} + tShift \\ RGB_i &= (1 - (\frac{C_i}{k_2} + eShift)) * videoRGB_i + ((\frac{C_i}{k_2} + eShift) * NdotL_i * intensity) \\ k_2, k_1 &\neq 0.0 \end{aligned} \tag{4.4}$$

i:Current fragment, RGB: Resulting color value, videoRGB: Real world rgb values, NdotL: Dot product of normalized normal and light vectors, C: Maximal curvature value, O: Resulting opacity, k_1, k_2*: Curvature weight, tShift: Transparency shift, eShift: Emphasize shift*

Figure 4.15 demonstrates the difference between virtualized and emphasized ghost presentations. Similar to the mapping between curvature to transparency values (see 4.1), the emphasis can be adjusted by modifying the parameter *eShift* in equation 4.4. If the parameter k_1 and k_2 are equal, the emphasis depends on the current level of sparseness. However, if both are independent, a sparse virtual emphasis can be generated even on a dense (otherwise real world) ghost presentation. This may especially be useful to communicate the current registration error (see next section).

The presented techniques are mainly useful if the virtual data is incomplete and a video background has to be used in addition to a ghost presentation. However, the examples in Figure 4.13, Figure 4.14 and Figure 4.15 use a registered phantom object only to detect good features. The presentation technique is independent from the applied classification

[†]We call sparse presentations those with a high but non-uniform transparency modulation, which results in a preservation of only a few features of the object. Respectively, dense ghostings are those with a high amount of visible object elements.

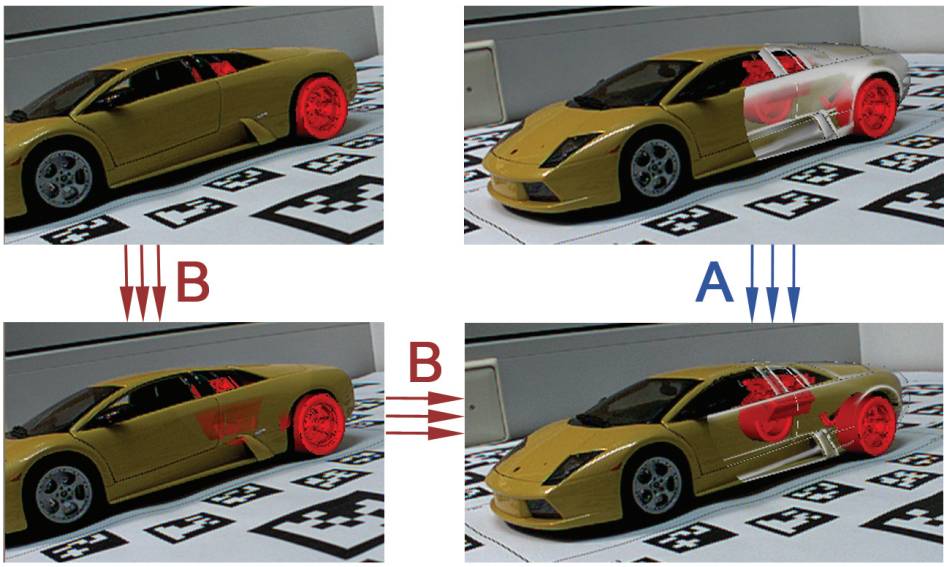


Figure 4.15: Emphasized versus Virtualized Ghosting. (A) Virtualized Ghostings are able to communicated sparse ghost representations but hide the real world object in case of dense presentations (B) By rendering a virtual ghost only in case of a sparse presentation, the rendering keeps real world information in dense presentations, while it is still able to communicate the shape via virtual structure in sparse presentations of the occluding structures.

technique.

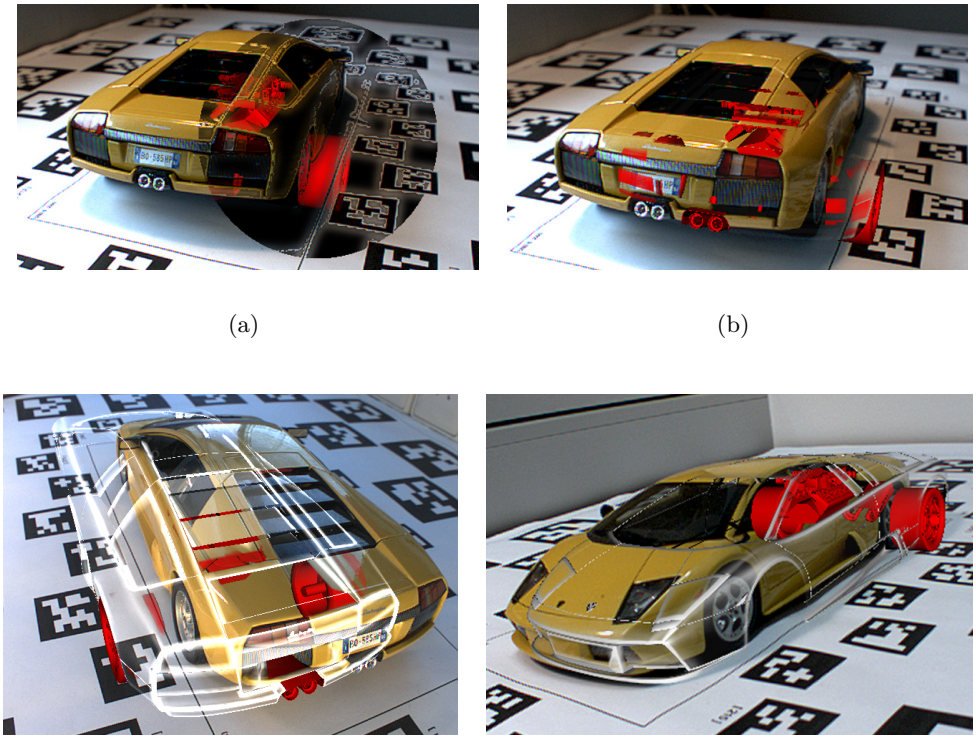
4.2.1.4 Error Friendly Presentation

Visual augmentations often suffer from erroneous data causing falsly registered 3d virtual objects [64]. Application which demand very accurate data (like in medial AR) restrict their working area to the lowest possible limit. Such limitations allow to better capture the environment by using more accurate and stable tracking technology, by acquired very precise phantom objects or by e.g. pre-computing light distributions.

However, such limitations are not always possible, e.g in outdoor AR applications. There are many different sources of defective data, so that a perfect augmentation may be very difficult to achieve in a dynamic environment (see [64] for an overview of possible sources of error). Consequently, if ghost presentations are subject of erroneous data, the augmented shape information does not align with the real world object any longer, which

may renders the visualization incomprehensible (Figure 4.16(b)).

Even though the system is not perfect, we are able to compute a probability of the current error [27]. Similar to [91], this probability value can be used to alter the way we present the scene content. We use the probability of the current error to increase the brightness of the fragments of the current ghost presentation. If the ghosting was generated using a phantom object, such emphasis allows us to communicate the phantom object, and regarding to Robertson [106], this enables a mental amelioration of the error (Figure 4.16(d)). However, a dense and emphasized ghost presentation may hide the relationship between the offset ghost and the real world object (similar to Figure 4.15).



(a) (b)

(c) (d)

Figure 4.16: Error Communication. (a) Video based ghosting using erroneous registration data (b) Video Ghosting is not able to communicate the error. (c) Emphasized Ghosting communicates the error but is not able to believably present hidden structures in case of erroneous data. Notice, due to independent parameter k_1, k_2 in equation 4.4, the emphasis is independent from the current level of sparseness (d) Increasing transparency in combination with emphasized or virtualized ghosting is able to communicate the error as well as hidden structure. Notice, the difference between (c) and (d) is there level of sparseness

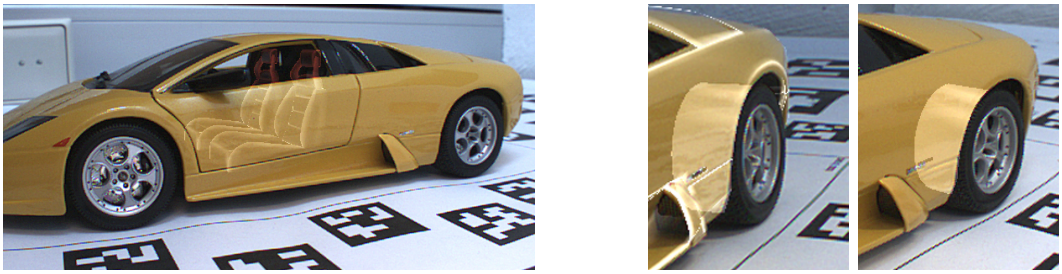
Therefore, error-based emphasis is only applied to a very sparse ghost classification. To avoid preservation of occluding structure, which does not occlude the virtual object in its correct position (as demonstrated in Figure 4.16(c)), the current transparency value is altered with respect to the error estimation. Thus, the error-based emphasis presentation technique uses a very sparse ghost representation, which is virtually emphasized.

Nevertheless, this error based emphasis is only effective if a phantom object is present. Otherwise, emphasis of the features which have been detected directly on the video frame do not help in communicating the current error. In fact, they may even hide the current error (Figure 4.16(a), Figure 4.9(c)). Thus, if no phantom object can be used, introducing additional primitives to the scenario, as proposed by Robertson and McIntyre [106], can help demonstrating the error. However, since Robertson evaluated the effectiveness of external context information only on opaque presentation, the cognitive impact of such error communicator in x-ray visualizations is unclear. Thus, a future project will consider an evaluation of external error communicators for x-ray visualizations.

4.2.1.5 Shine Through of Hidden Objects

Some situations might have more than one focus object. In those cases, the relative importance between occluding and occluded objects cannot be clearly resolved. If such objects overlap in screen space, a strategy to select the contributing fragments has to be defined. A naive approach is to simply blend between such objects to keep both foreground and background object visible in the same amount. However, as outlined before, simple blending often fails to convey the spatial arrangement and is therefore mostly inappropriate for our purposes.

As Diepstraten pointed out [36], a better solution is to lighten those fragments which occlude characteristic features of hidden information (see section 2.4, Figure 2.24a). The presented system generates such combinations of sparse hidden and densely occluding elements by blending the non-modified occluding structure with a desaturated and lightened ghost presentation of the hidden elements(Figure 4.17(a)). Similar to Viola's approach [124], the density of the ghost presentation depends on the relation between importance values of occluding and hidden structure. Since a more important occluding element generates a rather sparse hidden ghosting, similar importance values result in rather dense hidden representations. Notice, even though an emphasized ghost presentation of the occluding structure helps to accentuate its shape (Figure 4.17(b) left), such visual modification is not essential to communicate spatial relationships between hidden and already



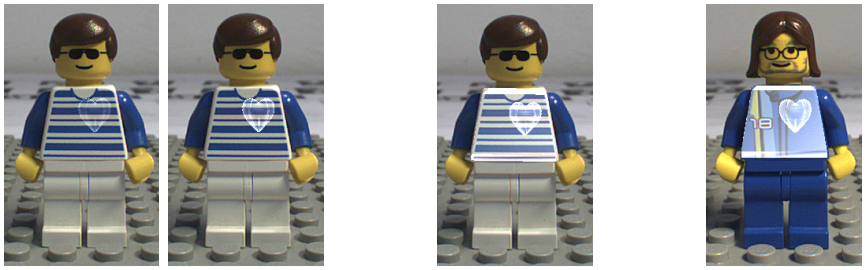
(a) (b)

Figure 4.17: Ghost Presentation of Hidden Structure. (a) Occluding structure have been lightened, where a ghost presentation of hidden elements 'shine through' (b) Even though an additional emphasized ghost presentation increases the perception of the shape of occluding elements (left), it is not essential to communicate spatial relationships in this type of presentation (right).

visible elements (Figure 4.17(b) right) if the hidden object *shines through*. This is because the visualization only modifies the lightness values of the occluding elements.

However, if the occluding structure introduces a texture, which interferes with the hidden ghost, the presentation becomes difficult to understand with an increasing density of the hidden object. An exception are already bright structures that are not affected by an increasing lightness (Figure 4.18(b). Thus, bright elements of an occluding texture are not disturbed, if fragments are lightened, where hidden elements shine through. Consequently, if bright elements exist in an occluding texture, we can use them to keep undisturbed texture elements in the presentation by increasing the lightness values of the occluder texture relative to the density of the hidden ghost. Even though this technique increases the understanding of ghost presentations of hidden structures, if no bright elements exist, it decreases the quality of the presentation without supporting the communication of spatial relationships 4.18(c).

Since the current implementation needs a manual selection of the applied stylization, the system is not yet able to alter the applied visualization technique depending on the visual appearance of real world structure. A future system will evaluate material parameter of real world structure to enable choosing an appropriate presentation of sparse hidden elements at runtime.



(a) (b) (c)

Figure 4.18: Lightening Textured Objects. (a) Occluding texture interferes with an increasing density of a hidden ghost presentation (b) Lighten bright elements makes them noneffective to the presentation technique which keeps these elements undisturbed in the presentation (c) Increasing the lightness decreases the quality of the result if less bright elements exist

4.2.2 Multiple Object Occlusions

Real world scenarios usually do not consist of a single occluding object. Instead, complex scenarios including complex depth arrangements exist. Thus, the aforementioned ghost presentations can easily lead to excessive contextual information and cluttered image content, if directly applied to all structures in-between the visible and the object of interest (see section 1.1.3).

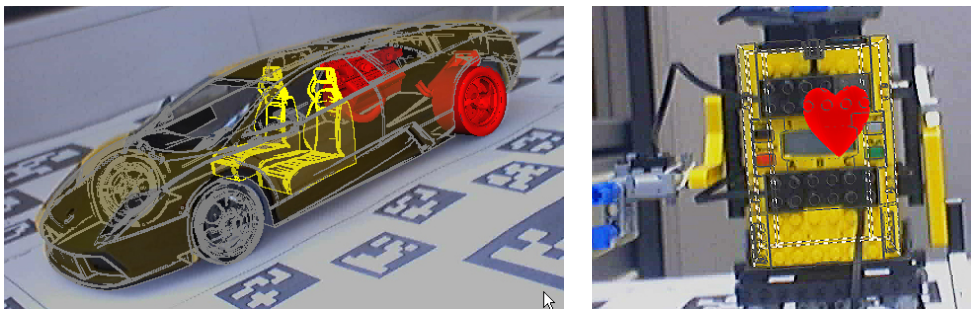
To allow generating ghost presentations of complex scenarios, the presented system is able to not only extract key features from occluding objects, but also to control the amount of information visible in the final image. Since filter operations have to be applied depending on the structure of the scene and the intent of the visualization, the presented system furthermore allows flexibly changing the way information is filtered. This is implemented by supporting a set of filter operations on the data which occurs in the different stages of the rendering pipeline (see section 3). 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. The following subsections discuss the advantages and drawbacks of these two different approaches.

4.2.2.1 Filtering by Stylization

A proper visualization is able to reduce the visual complexity of the augmentation. Thus, the presented system allows stylizing the data to visually encode properties like the object's

group affiliation or spatial relationships. By altering the appearance of scene elements, the system enables filtering via processing of the information by our cognitive system. Figure 4.19(a) shows an example of a coloring which encodes both group affiliations and importance values.

To control which parts of which G-Buffers get affected by a certain set of stylizations, the presented system uses F+C graph structure during G-Buffer Processing (see section 3). Recall that F+C visualization is approached not only by defining a set of visualization techniques, but also through a hierarchy of filter operations. Nodes in the graph represent filter or shade operations on G-Buffer content, while the data flow defines on which part a subsequent filter or shader is applied to.



(a) (b)

Figure 4.19: Filtering by Stylization. (a) To encode group affiliations, coloring has been changed. In addition, toon-shading has been used on the seats to reduce visual complexity relative to the main objects of interest (the engine and the wheels) (b) Hidden lines have been encoded using a dashed line pattern. Since dashed lines introduce an artificial pattern, they are more difficult to understand in combination with complex object presentation in AR environments.

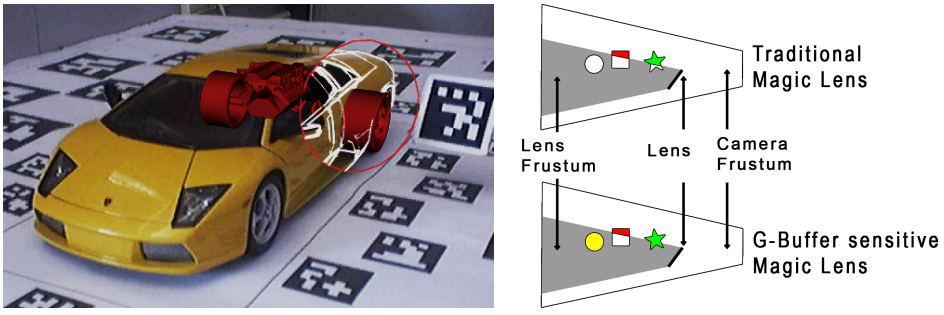
As outlined in section 2, stylizations are not only capable of encoding group affiliations or importance distributions, but also to communicate spatial relationships. For example, line renderings may use edge halos on front facing structures [3] or a dashed presentation of hidden ones to present their relative depth arrangements. An example of the latter is shown in Figure Figure 4.19(b). However, these stylizations introduce discontinuities and artificial pattern which are independent of real world texture information. Similar to the problems of applying a stroke pattern to textured objects (see section 4.1), a dashed pattern may interfere with real world information present in its close proximity of its presentation space. Thus their usage has to be carefully minded.

However, the current system does not support any automatic verification of either the comprehensibility of the result nor the applicability of a specific stylization in the current scenarios. In a future project we will first investigate methods to allow an online verification of several stylization techniques in AR, followed by an implementation of adaptive stylization techniques to consistently generate the most valuable presentation for the current AR environment.

4.2.2.2 Hybrid Filtering by G-Buffer Masking

Stylizations controlled by the F+C graph do not directly incorporate any spatial properties. To spatially restrict the stylization, the presented system integrates a Flat Magic Lens tool (see section 2.3.1) into the F+C graph hierarchy. This is implemented by simply adding to scenario a 2d geometry which is tracked in 3d space. The information in its corresponding G-Buffer is used to control subsequent stylizations in the F+C Graph. Thereby, the system is able to use the Magic Lens tool to interactively control the spatial appearance of stylizations of scene elements relative to the hierarchy describing the combinations of F+C visualization techniques.

Since the Magic Lens tool is integrated in the hierarchy of the F+C rendering, the system becomes able to control the spatial appearance of only distinctive G-Buffers. The filter affects only thus fragments which are processed below its corresponding node in the F+C hierarchy. For example, Figure 4.20(a) shows how only the chassis of the car get affected by the G-Buffer-sensitive Flat Magic Lens, while the other objects remain



(a) (b)

Figure 4.20: Interactive Filtering using a G-Buffer sensitive Flat Magic Lens (a) The Magic Lens tool controls the occurrence of the ghosting of a single object (b) Illustration of the difference between ordinary Flat Magic Lens (top) and the G-Buffer sensitive Flat Magic Lens

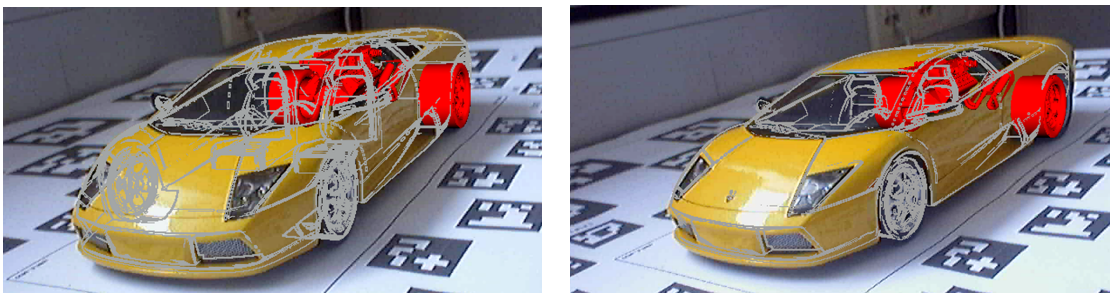
unchanged. This filter interactively limits the spatial occurrences of the ghost presentation. However, it only takes the fragments of the chassis of the car into account (notice the wheels and the engine are not affected by the lens).

The difference between conventional Magic Lenses and our implementation is illustrated in Figure 4.20(b). On top it shows a traditional Magic Lens, affecting everything in its frustum, regardless of whether the effect is desired for a particular object. The bottom shows our implementation and how it affects only a certain set of G-Buffers (only the red square is stylized by the lens).

4.2.2.3 Per-Pixel Filter by G-Buffer Grouping

Depth complexity increases if multiple objects in front of an object of interest overlap in screen space. The presented system allows to reduce the number of contributing fragments on a per pixel level by either adjusting the grouping of scene elements during G-Buffer rendering or by a special filter strategy during scene compositing.

If more than one fragment falls onto the same pixel, a test such as the depth buffer test is required to choose one remaining fragment. The way we group objects into families defines which fragments are tested against one another. Thus, the grouping of scene elements in combination with the choice of the fragment tests defines the content of the G-Buffers and accordingly the result ghost presentation. For example, Figure 4.21(a) shows a setup of four different G-Buffers, while Figure 4.21(b) uses a simplified grouping into only two G-Buffers. It can bee seen that a grouping of the scene elements into fewer G-Buffer already rejects a number of fragments and thus reduces the amount contributing



(a) (b)

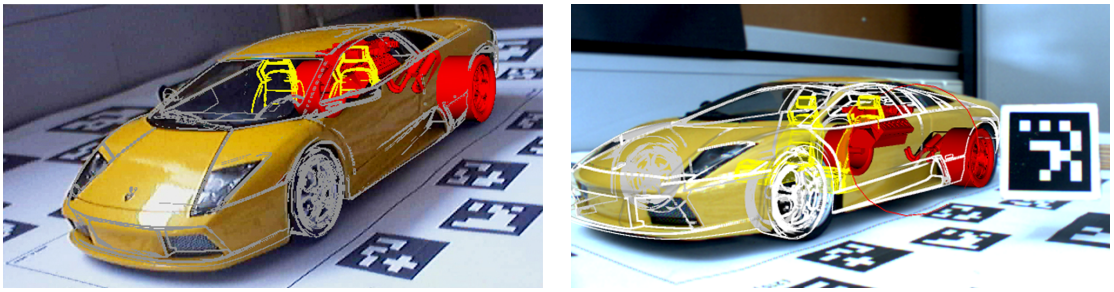
Figure 4.21: Filtering by Grouping Scene Content. (a) Four G-Buffers hold the content of the scene (b) The same scene is rendered into only two G-Buffer, resulting in fewer fragments contributing to the augmentation

structures.

Such filtering by first selecting a set of objects followed by regular OpenGL fragment testing is a versatile tool to control the amount of visible information. However, the total of augmented information in the final image depends on the number of G-Buffers. Even though fewer G-Buffers create less clutter and are usually faster to process, in order to define appropriate filter strategies we need detailed knowledge of the scene and its object structure to define a suitable object grouping. Nevertheless, this may not always be available, in particular in dynamically changing scenes.

4.2.2.4 Per-Pixel Filter by Compositing Strategies

Statically reducing the amount of contributing pixels may result in the desired amount of information in one place, while at the same time in a lack of information in different place. Thus, we have implemented another technique, filtering during scene compositing, which reduces the amount of visible information per pixel regardless of the number of G-Buffers (see section 3.1.3). Any filter strategy that can be formulated using pixel values and sorted depth values can be applied to filtering during scene compositing. The system offers some simple heuristics that can successfully limit clutter in typical situations.



(a) (b)

Figure 4.22: Filtering Through G-Buffer Compositing. (a) The First-Hit-And-Focus strategy was applied on the entire framebuffer (b) The First-Hit-And-Focus strategy is applied only inside a Flat-Magic Lens tool while outside, all available fragments contribute to the image composition

Figure 4.22 shows one such strategy, where the first fragments (after sorting the depth values in front to back order) and those belonging to the focus objects have been used (called first hit and focus strategy). The image on the left shows the strategy applied on the entire frame buffer while the image on the right demonstrates its results if it is

only applied inside a Flat Magic Lens. While four different G-Buffers are used to visually discriminate the scene elements, such heuristic successfully reduces the amount of visible fragments inside the lens.

Filtering during scene compositing allows using arbitrary compositing rules in any 2d spot. Nevertheless, compared to the other filter techniques it is generally more expensive since fragments discarded in this stage have already gone through the entire pipeline before they are finally eliminated.

Chapter 5

Cutaways

Content

To present comprehensible x-ray visualizations, hidden and occluding structures have to be presented. The previously discussed ghost presentations render a sparse variant of the occluding object which allows seeing behind. However, this technique often generates cluttered and ambiguous visualizations if multiple objects hide the object of interest. In addition, if occluding elements have been preserved the hidden object can not appear in full detail. Instead, they override some portions of the image to introduce occlusions as a direct depth cue (see section 2.2). As an alternative, cutaway illustrations are able to completely uncover the hidden object (Figure 5.1(a)) even if it is occluded by a number of objects.

Even though cutaway renderings have been subject of research in AR environments before [48] [10], this chapter presents cutaway renderings in AR within the scope of this thesis. This chapter discusses cutaway generation and their presentation both, in complete and perfectly registered virtual environments as well as from incomplete and erroneous data. Thus, this chapter is not only provided for completeness' sake, but it allows to make a fair comparison between all major illustrative x-ray visualization techniques under

(a) (b)

Figure 5.1: Cutaway Presentations (a) Wedge Cut. Notice how the bending of the contour of the cut-out area indicate the shape of the object (b) A Flat Magic Lens controls the cutaway of occluding information. Simple Magic Lens based cutaway classifications are not able to indicate thickness or other occluding structures in-between the occluder and the object of interest. Since no occluding structure is preserved, the presentation fails to indicate the shape of the occluder.

similar environmental conditions.

As discussed in section 2.4.2, a cutaway presentation removes occluding information entirely. A similar approach was already applied in the previous chapter in the context of filter techniques to control multiple occluding ghost presentation. Nevertheless, the previously introduced filter techniques focus only on strategies to reduce the amount of contributing fragments. Even though the results already enhance the comprehension of multi-object ghost presentations, they do not consider features outside the filtered area or take shape information into account. To enhance the mental reconstruction of the missing information, the rendering has to encode the shape information at the border of the cutaway. In fact, if simple 2-1/2D Magic Lens filters are applied, the result may not be able to indicate missing information. Such Magic Lens cutaway renderings often cause the impression of a floating virtual object in front of real world video data. Figure 5.1(b) demonstrates this phenomenon. Since the rendering does not take any information about the occluding structure into account, it is not able to indicate the shape of occluding structure.

In contrast, illustrative cutaway presentations have been specifically designed to support mentally closing the introduced excavation. Once the removed information is mentally reconstructed, the perception of spatial relationships are directly inferred by our cognitive system. The following section discusses methods to automatically compute illustrative

cutaway renderings in AR (section 5.1), followed by an outline of presentation techniques to overcome the difficulties, which the AR environment adds to the application (section 5.2).

5.1 Classification

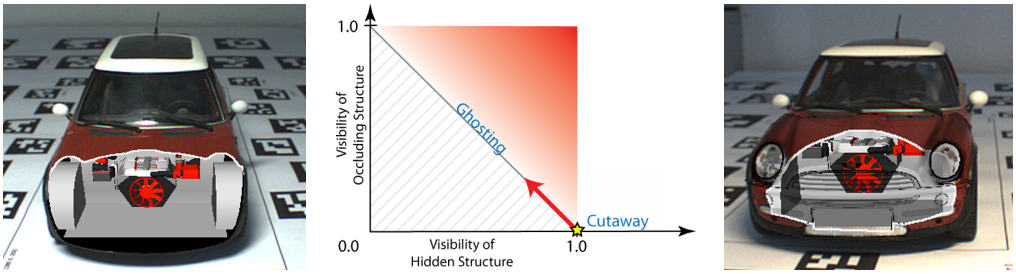
The key element of comprehensible cutaway presentations is their ability to indicate missing information, so that our cognitive system is able to mentally close it. This section discusses techniques to identify the information, which has to be removed to uncover the hidden element, and the information which has to be preserved to indicate the missing structure.

5.1.1 Object Based

Recent work has identified a set of object aligned cut-out shapes, which are frequently used by artists [88]. These techniques indicate missing information by the way they cut occluding structure (for more detail see section 2.4.2). However, to effectively communicate spatial relationships, they require a certain relation between the point of view and the object of interest. In contrast,Viola and his colleagues [125] as well as Burns et al. [18] demonstrated the power of view aligned cutaway renderings, which ensure visibility from any point of view but they do not directly take the shape of occluding objects into account. Since AR environments are interactive by their definition, a dynamic point of view has to be considered which may conflict with an object based approach.

On the other hand, object aligned cutaways have been specifically designed to support mentally inverting the cut operation, which generates better results in situation where a suitable point of view is reachable. Following this line of though, an AR system supporting cutaway renderings has to offer both, view and object aligned cutaway classification techniques. If the application is able to assume a rather small range of different points of view, relative to the object, which is about to be cut, and if in addition the current location of the user falls within an area which defines a good point of view for an object based technique, the system should apply an object based cutaway rendering, otherwise it should fall back to a view dependent cutaway variant.

The current system supports view aligned wedge cuts. To be able to use the system in an arbitrary environment, a future system could offer object aligned cutaways. Furthermore, it will be able to evaluate the current viewing condition to allow an interpolation



(a) (b) (c)

Figure 5.2: Trading Visibility for Shape Perception (a) Large cutaways may not allow to mentally reconstruct the cutout shape (b) Reducing the visibility of hidden information allows to preserve shape information of occluding structures. (c) Reducing the visibility of hidden structure by adding a ghost presentation generates the most effective shape communication using the least amount of additional occlusions

between view- and object aligned cutaway pattern to automatically choose the most appropriate style. The future system will also be able to adaptively combine ghost presentations with cutaways to counteract problems of large cut outs which are demonstrated in Figure 5.2a. If very little information is left to mentally reconstruct the cut out information, a ghost presentation will used to preserve the most important shape features (Figure 5.2b). By adding a very sparse ghost presentation we trade visibility by shape perception (Figure 5.2c)

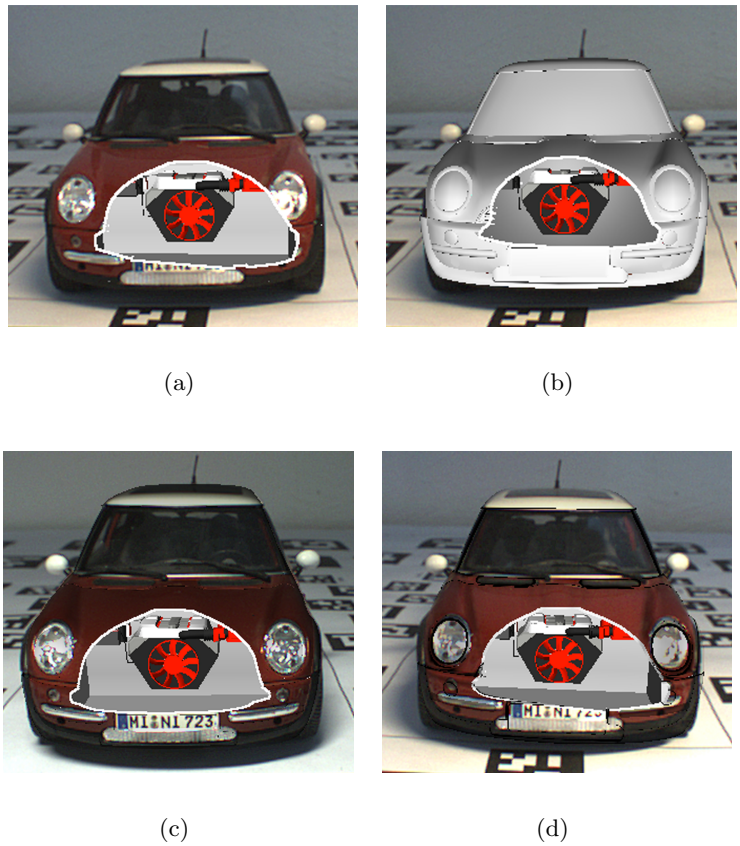
5.1.2 Image Based

If the AR system does not offer a registered 3D phantom object, the previous classification techniques can not be applied. In those cases a Flat Magic Lens may be used to identify removable video information. However, as already mentioned the results may be difficult to understand (see Figure 5.1(b)).

5.2 Presentation

The examples presented in the previous section already demonstrate the ability to indicate the removal of occluding information by using a cutaway presentation (e.g. Figure 5.1(a)). A very important communicator to infer missing shape information of occluding structures is the contour of the cut-out shape. Its bending encodes the shape of the occluding structure. Since this is one of the key elements of a cutaway illustration, the success of

the presentation highly depends on a believable fusion of the virtual contour on top the real world shape.



(a) (b)

(c) (d)

Figure 5.3: Re-shading Cutaways. (a) No re-shading is applied. Even though the virtual contour of the cutaway follows the almost perfectly registered phantom object, the contour does not completely integrate with the real world information about the shape of the car. (b) Cutaway in combination with the diffuse shaded phantom object (c) Re-shading the real information using a diffuse virtual rendering of the phantom object helps to combine virtual and real world information (d) Using an curvature information allows to specifically emphasize characteristic shape information which reduces the overall modifications

Therefore, it is not only the bending of the contour which has to be considered, but also shading information of the remaining shape of the occluding object (see Gestalt Laws in sections 2.2). Consequently, both circumstances, small differences between the shape of the phantom and the real world object, as well as adverse real light conditions will result in badly integrated cutaway renderings in AR (Figure 5.3a). To overcome these deficiencies, we compute a diffuse shading of the virtual object which we use to re-shade the real world object using equation 5.1 (Figure 5.3c). Since the virtual shade information perfectly

aligns with the virtual contour (see Figure 5.3b), the re-shaded real-world object enables a better integration of the virtual contour.

$$RGB_i = videoRGB_i * (NdotL_i * intensity + rShift) \tag{5.1}$$

i:Current fragment, RGB: Resulting color value; videoRGB: Real world rgb values; NdotL: Dot product of normalized normal and light vectors

Since virtual re-shading is applied all over the real object, it may strongly changes its appearance (depending on the virtual lighting). However, as outlined in the previous chapter, a sparse ghost presentation is able to affectively communicate shape information. Moreover, a ghosting reduces the amount of modifications to the real object, while it is still able to communicate the required shape information. To even more increase the ability of shape communication we use shade information for the ghost's virtual color using equation 4.4 with O_i set to 1.(Figure 5.3d).

5.2.1 Registration Error

Cutaway presentations are not directly able to communicate erroneous data (Figure 5.4a). However, as Figure 5.3(c) also demonstrates, by re-shading the real world object using the 3D-registered virtual phantom, the presentation is able to hide small differences between the phantom object and its real world counterpart. Adding an emphasized ghosting (similar to 5.3(c)) even allows to communicate larger errors. An augmentation of a ghosting places characteristic virtual features next to their real counterparts. This results in a direct presentation of the offset which allows to infer the current error.

If many of the characteristic features are located inside the cutout area, a ghosting outside the cutaway may not be able to present enough features. In contrast, an additional ghosting of only those structures, which fall inside the cutaway, may also fail to link real landmarks, especially if the cutaway occludes the real world features (Figure 5.4(c)). Consequently, a combination of an emphasized ghosting outside and inside the cutaway area will render the safest result. However, notice again, applying a ghost presentation to communicate the current error inside the cutaway will re-add occlusions to the presentation (see the diagram in Figure 5.2b). If the goal of the visualization is to ensure a certain visibility, any rendering of occluding structure may be counterproductive, even if only a very sparse one is applied.

(a) (b)

(c) (d)

Figure 5.4: Ghosted Cutaways allow to Communicate the Current Error (a) The error is not communicated (b) Cutaways in combination with an emphasized ghost presentations are able to communicate the error (c) However, the ghosting is more effective if it appears outside the cutaway volume. (d) Adding both only improves depth perception but does not influence the communication of the error.

5.2.2 Incomplete Data

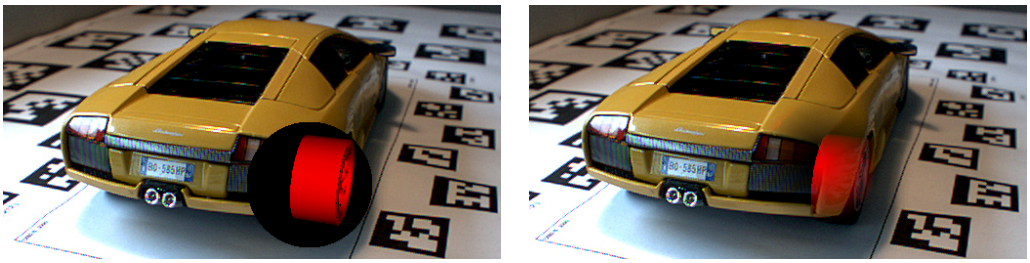
If 3D structure between the visible and an uncovered object exist, a good estimation of depth values between hidden and occluding elements is possible in a cutaway presentation (see all previous Figures in this chapter). However, if the virtual model is incomplete, no structure exists to help to indicate depth values or occluding shape (Figure 5.5a, Figure 5.6a).

While partial occlusions help to communicate the depth order (Figure 5.5(b)), if the cutaway becomes too big to overlap in screen space (Figure 5.5(a)), spatial arrangements between real and virtual objects may be lost. Therefore, to communicate depth and thus spatial arrangements, we render the cutaway volume (Figure 5.5(c)) visible in cases where the virtual model is incomplete.

(a) (b) (c)

Figure 5.5: Incomplete Hidden Data. (a) Cutaways with incomplete data are only able to present spatial relationships between occluding and hidden structure, if partial occlusions exist. (b) If no occlusion exist depth arrangements may become ambiguous (c) Rendering the cutout volume re-adds depths cues indicating spatial arrangements.

Nevertheless, if no phantom object is available, only depth differences between the position of a Magic Lens tool (which controls the cutaway) and the virtual object of interest is possible. Since we assume that the order of elements is more important than such depth differences, we add an image based ghost presentation if no other data than the video and the otherwise hidden object of interest is available (Figure 5.6(b)). However, notice once again, by transforming the cutaway into a ghosting presentation, the system trades visibility for comprehensibility. In addition, to hide the flat character of the Magic Lens, our rendering abandons to use the contour of Magic Lens Tool.



(a) (b)

Figure 5.6: Incomplete Hidden Data Without Phantom Model. (a) Spatial relations are completely lost (b) Adding an image based ghost presentation trades comprehensibility by visibility

Chapter 6

Explosion Diagrams

Content

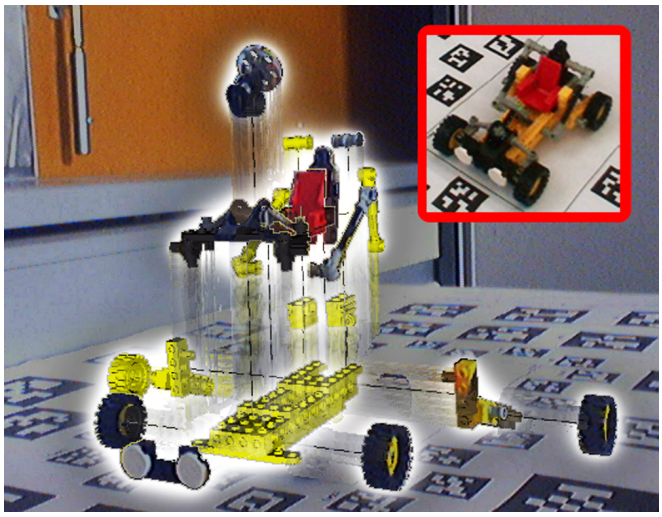


Figure 6.1: Explosion Diagram in Augmented Reality. Real world information is displaced using *synchronized dual phantom rendering*. Exploded parts are visually unified using either pure virtual or real world combined with restored real information. A halo outline discriminates the parts from background information and their connection lines are embedded in a motion blur effect resulting from their explosion.

The previously introduced techniques generate x-ray visualizations by removing parts of the occluding structure. The remaining information is presented in a way that supports mentally reverting the removal. However, also possibly important information may be removed. As an alternative to remove information, this chapter investigates explosion diagrams in AR environments, which are able to present hidden and occluding objects in full resolution (Figure 6.1). Explosion diagrams are traditionally used in technical illustrations to present the assembly of an object (see section 2.4). Unlike cutaway- or ghost-visualizations, they do not try to find a tradeoff between visible occluding- and uncovered hidden structures. Instead, they present either information next to each other, using an arrangement of parts that makes it possible to mentally reassemble the object. The example in Figure 6.1 demonstrates that explosion diagrams hide only those structures which are considered as less important background information. Assuming that the occluding information of an otherwise hidden object of interest is more important than the information in the background, explosion diagrams increase the amount of relevant information, while communicating the spatial arrangements in x-ray visualizations.

The key element of a comprehensible explosions diagram is the layout of the exploded parts of an object. This spatial arrangement encodes contextual relationships among the

parts, which in turn is able to support the process of mentally reassembling the exploded object. However, due to the amount of information in a real world environment, an explosion diagram in AR demands special visualization and rendering techniques, in addition to an expressive layout. They require rendering techniques to relocate real world information as well as restoration techniques to fill out arising voids after relocating a real world object.

Furthermore, traditional illustrations usually present the explosion diagram in front of a uniformly colored (most often white) background (Figure 6.2). In contrast, AR displays have to deal with a high amount of real world information in the background. Since the surroundings of an object contain its contextual information, an explosion diagram in AR moves its parts out of its original, into a new collection of contextual information. Therefore, an explosion diagram in a real world environment has to use special visualizations to avoid the influence of misleading contextual information. The same has to be considered to visually link between the parts of an explosion diagram in AR. For example, notice in Figure 6.5(e) how difficult it is to see a single colored dashed line in a non-tuned visualization in AR.

This chapter addresses the task of integrating explosion diagrams in an AR environment. The layout of the diagram is automatically computed by applying a set of search strategies on a data structure which hold all possible explosion diagrams. In the second part of this chapter, algorithms to comprehensibly present the explosion diagram in an AR environment are discussed. It shows how to relocate video information based on the explosion diagram and it presents algorithms to comprehensibly integrate relocated information back into the real world environment.

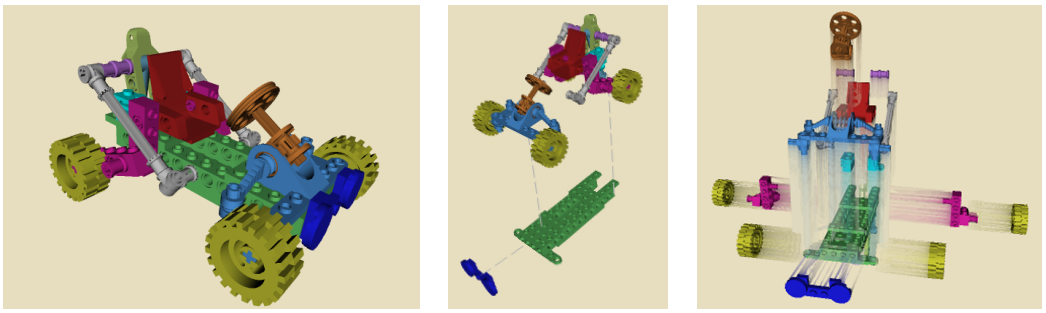
6.1 Classification

To support the mental ability to reassemble an exploded object, Gestalt Laws (section 2.2) have to be minded. In addition, since different tasks demand different layouts (see section 2.4), the layout of the explosion diagram has to be chosen appropriate to the current task. For example, an explorative explosion diagram aims to present a high amount of different parts at once (Figure 6.2(c)). When exploring an object, its overall structure is of same interest as each single part of object. Thus, the object has to be be split into single parts (to present each single one) positioned relative to each other (to communicate the overall shape).

Conversely, a single step of an assembly plan only presents the parts which are neces-

sary for the current step. Heise et. al. [61] noticed that in a presentation of an assembly plan, symmetric parts should be presented in the same so called 'action diagram' which describes a single step out of the whole set of instructions of the assembly plan. For example, the assembly of all wheels should be presented in the same picture. We assign these findings to our animation styles, which explode symmetric parts, either at the same point in time or in a row without interference of other non-symmetric parts.

In case of an x-ray visualization, we want to uncover the hidden object rather than exploring every single part. Thus, an explosion diagram, which is specifically designed to be used in an x-ray visualization, should aim for a layout which keeps the maximal amount of objects unexploded, while the object of interest in uncovered. Those layouts are computed by removing parts from the object until the focus element is isolated (Figure 6.2(b)). Notice, the implemented system is able to generate both grouped explosion diagrams as well as single part explorative layouts.



(a) (b) (c)

Figure 6.2: Different Layouts. (a) Unexploded presentation of the object (b) A grouped layout is used to communicate the roles in an x-ray visualization. The least amount of groups have been found to isolate the object of interest (c) The layout presents a high number of detached parts. Even though the overall shape of the object is preserved the presentation is more difficult to understand than the grouped layout.

No matter which task the layout was intended for, Gestalt Laws have to be considered for all of them (see section 2.2). For example, in a psychological study on the perception of assembly plans, it was found that symmetry is an important criterion for creating well structured explosion layouts [61].

6.1.1 Object based Layout Computation

Several parameters influence the arrangement of parts. Firstly, an explosion sequence has to be found, containing information about which parts have to be removed, before other parts can be detached, without colliding with any of the still assembled objects. Secondly, the exploded parts have to be associated with those (parent) parts which they move relative to. Thirdly, separation directions resembling valid assembly directions as well as separation distances have to be determined. Fourthly, to be able to explode a whole group of symmetrical parts or to quickly reveal a focus element, layers (groups of partitions) are introduced.

Each of these parameters can be influenced independently by using a certain search strategy on a data structure which holds all possible explosion layouts. In the following, an overview of the parameters which are used to control the explosion layout will be described. A more detailed discussion of all different combinations can be found in [118].

6.1.1.1 Partitioning

To automatically compute a layout, we have to find a valid partitioning of the object. A partitioning is a sequence of part removals, in which each removed part does not collide with any other parts. To correctly disassemble an object, a number of different valid combinations exist and result in different layouts. Since most of them do not comply to the aforementioned rules, the implemented system offers a set of search strategies to control the selection of the displayed partitioning.

The range of all possible explosion sequences is determined by using a technique from the assembly planning domain, which follows the approach of assembly-by-disassembly. Parts or groups of parts, which are separable along their assembly direction, are removed until the whole product is disassembled [65]. A part or a group of parts is considered as being separable if it does not collide with other parts along their assembly path. All potential sequences are collected in a single *AND/OR* graph which represents the assembled model in its root node, while all pairs of disjoint groups of parts (further referred to as *partitions*) are being represented as children. By recursively splitting each partition into its set of pairs of valid partitions, a graph is built, which contains all possible partitionings.

Following the graph from its root node to its leaves disassembles the model. However, it is very likely that an assembly can be split into several different combinations of two partitions. Therefore our system implements different strategies to choose a certain path through the AND/OR graph. For instance, a strategy used to create a symmetric layout

removes the symmetric parts one after another before considering other parts. Since the goal of the presented system is to automatically compute a layout, the symmetrical parts are selected by their size and their position in the AND/OR graph. Therefore, the size of the current removed part is compared to all possibilities which can follow this part. Each subsequent part having a similar size to the current one is identified as being symmetric. The algorithm terminates the search for symmetric parts, after it finds a part which is not of similar size to the current one.

A symmetric partitioning introduces a hierarchy of parts by removing similar ones from the assembly in a row. However, other search algorithms on an AND/OR graph can be implemented, leading to other layouts. For example, a partitioning strategy to reveal a focus object may always detect pairs of partitions, where one of them contains the most parts excluding the focus element. By recursively applying this strategy to the partition containing the focus part, a layout is achieved which reveals the focus while all other parts are being presented in the smallest possible number of groups (Figure 6.3(a)). Notice, that symmetric considerations are not directly taken into account when revealing a focus part.

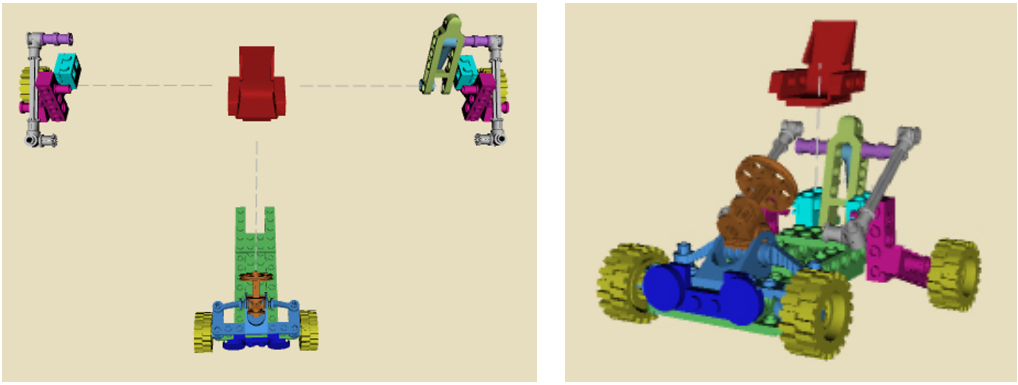


Figure 6.3: Focused Explosion Layout. The seat of the car has been selected as the object of interest.

6.1.1.2 Part Relations

After determining the partitioning of the assembly, a relation strategy decides on which parts move relative to others by assigning parent and child relations. The decision is made on two levels. On partition-level, one partition of each partitioning is chosen to be the static parent, while the second one is the child moving relative to its parent. The

decision is further refined on part-level. Out of the set of parts being in contact between both partitions, corresponding parent and child parts are selected, thus establishing a connection between the partitions. To prevent children from being exploded relative to several different parents, a simple restriction is introduced: each partition must not contain more than one child.

This ensures that partitions, which already contain a child, are always selected as parent partitions. Both levels can be influenced by different criteria independent from each other, thereby creating different layouts. A useful criterion can be the size of partitions and parts. For instance, good results could be achieved, when first deciding to move the partition smaller in size relative to the bigger one. Additionally, on part-level, the biggest parts (using the size of the bounding box) were defined to be parent and child parts. In combination with the previously described symmetrical partitioning style, symmetric layouts were generated.

For focused layouts, symmetrical considerations are of less importance. It is sufficient to ensure, that the focus element stays static, while the partitions are moved away from the focus element. A different approach is to move the focus element only once, if it can be separated from the rest as single part. To be able to clearly distinguish the focus from the rest, it can be exploded a greater distance than the rest.

6.1.1.3 Directions and Distances

To avoid display clutter by abundance of explosion directions, we restrict the overall number to only six, which follow the object's main axis. Having computed information about symmetric parts or relations between focus and context parts, we can further proceed to set up distances and directions according to group memberships. For example, all objects of the same group can be set up along the same main axis and within the same distance to their parents. Such a layout is able to visually underline the relationships among the parts.

Besides group information, the size of a part (e.g. the extent of its bounding box) may be used to influence its distance of explosion, leading to a layout where smaller parts, like screws, are offset a small distance from the parts they are attached to.

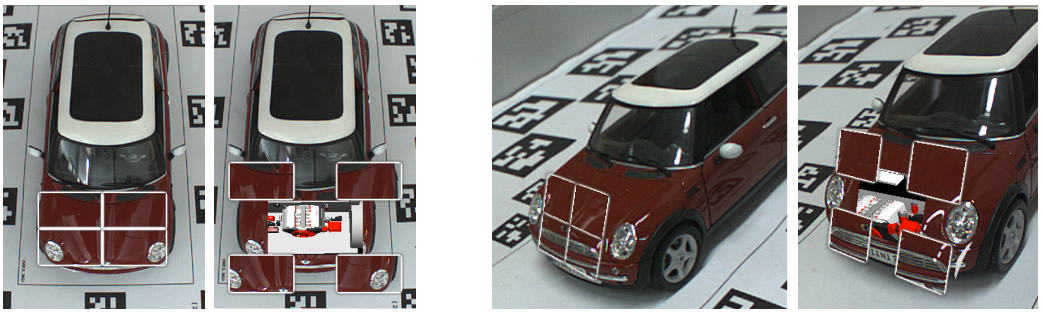
6.1.2 Pixel Transfer using Exploded Geometrical Primitives

Object based explosion diagrams do not only require a 3d model to classify video imagery, but also information about spatial relationships of parts (such as information about parts

which block other parts) to be able to compute a layout. If no phantom object exist, neither an image segmentation can be carried out, nor a comprehensible explosion layout can be computed and applied in AR.

However, an exclusively image based approach can only explode parts of the image depending on its position in screen space. As soon as the camera starts moving, this approach would explode different real world content in consecutive frames, and frame inconsistencies will result in disturbing presentations. Therefore, to limit the difference between the exploded content in subsequent frames, the system explodes geometrical primitives which are textured with a projection of the current video data. Notice, that the error depends on the angle, the distance and the difference in shape between the exploded 3d object and the primitive.

Figure 6.4 demonstrates a plane based explosion in AR. The images in (a) demonstrate an explosion where the explosion plane closely approximates the shape of the exploded object. Such setup almost completely hides away the lack of a 3d phantom geometry. However, the images in (b) demonstrate a configuration where the explosion plane is offset by an angle of about 45 degrees, which renders the explosion more difficult to understand.



(a) (b)

Figure 6.4: Primitive Based Explosion. (a) The explosion plane almost perfectly approximates the occluding shape (b) The explosion plane does not line up with the object.

Even though this allows to see behind the primitive, it is more difficult to imagine the exploded content as being part of its real world object. Visually linking the content (see section 6.2.2.3) helps to improve the understanding between exploded and unexploded primitives. However, the quality of the visualization still depends on the approximation of spatial and shape parameter between the primitive and the real world object.

The comprehensibility of a primitive based explosion highly depends on the quality of

its approximation of the current real wold object. Consequently, in a future project we will automatically fit primitives into a real world environment.

6.2 Presentation

6.2.1 Rendering Explosion Diagrams

Explosion diagrams in AR 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. In this section, we present algorithms to relocate real world information, which fulfill these three requirements. We also discuss the advantages and disadvantages of these approaches.

6.2.1.1 Video-Textured Phantoms

Convincing AR renderings must resolve occlusions between virtual and real world objects. To find out which fragments are visible, a common approach is *phantom rendering* [13] which uses the depth values of real world objects which are computed by adding the virtual counterpart, the phantom object, of real objects to the scene description. Phantom objects are rendered invisibly, only to the z-buffer, which enables the application to subsequently resolve occlusion among real and virtual objects using the z-buffer, assuming that the phantom objects are properly registered with their real world counterparts. While phantoms themselves are rendered fully transparent, the real world information from the video background is kept where a phantom occludes all fragments from virtual objects.

Simple phantom rendering does not take into account the relocation of objects, so it is not suitable for our goal of combining real, virtual and relocated real objects. Specifically, information from the video background in the color buffer must be transferred to its new location, when the phantom of a relocated object is rendered. Thus, we extend the original idea of rendering transparent counterparts of real world objects to video-textured phantoms (Figure 6.5(a), Figure 6.5(b)).

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 simply multiplying each vertex of a phantom with the combined model-view-projection (MVP) matrix representing the model (the phantom) before explosion transformations are applied.

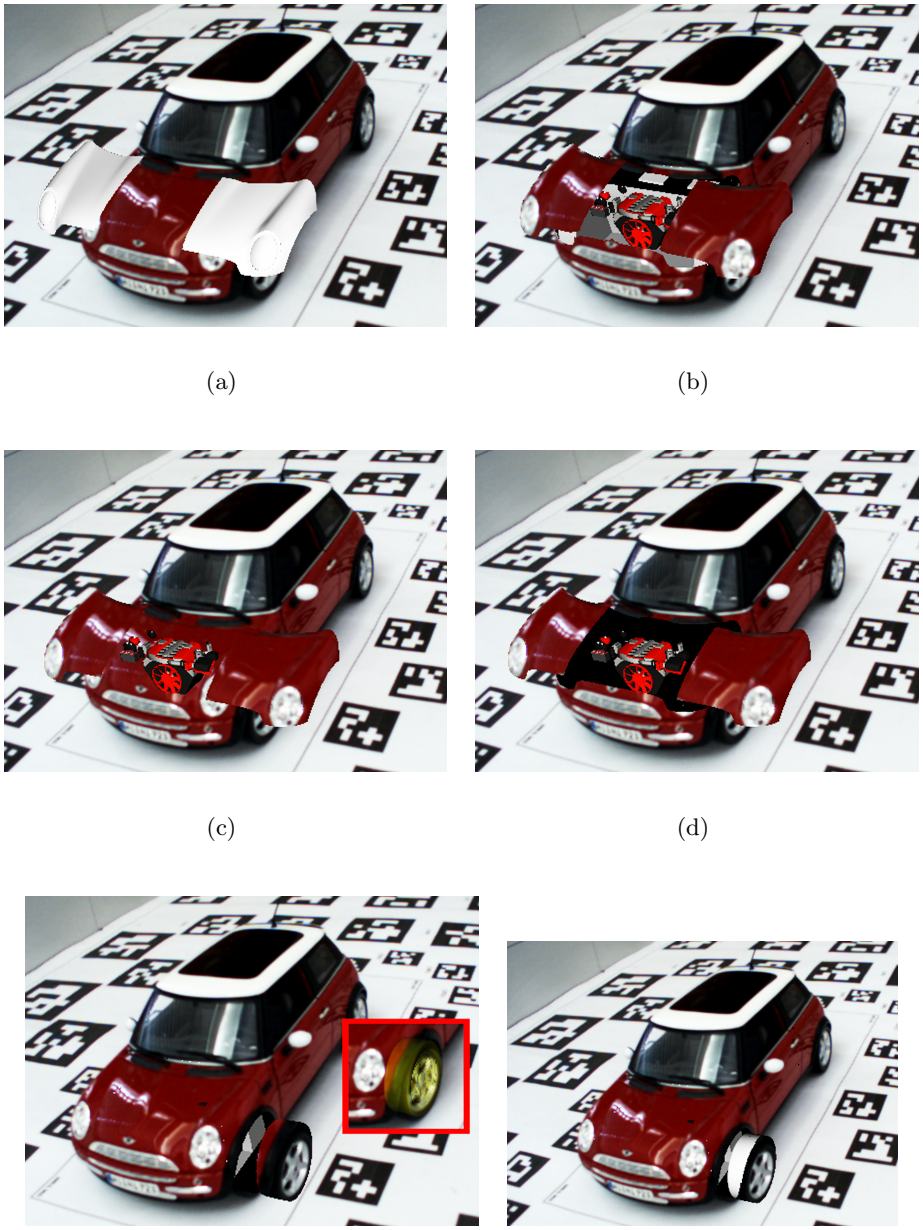
Using a vertex shader, this can be achieved in two ways. We can either render the phantoms in their real world location and pass the transformation to the new location to the shader, or we compute the MVP matrix for each phantom beforehand and use it in the shader while the phantom is rendered in its 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. Consequently, 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.

6.2.1.2 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 (Figure 6.5(c)). 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 (Figure 6.5(d)).

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 either remove redundant real world information, resulting in, e.g., a black background where no information is available, or the mask can be used to supplement incomplete hidden structures (section 6.2.1.4). The algorithm, called *dual*



(a) (b)

(c) (d)

(e) (f)

Figure 6.5: 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

phantom rendering, 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 pixel)
2. *Render all video-textured phantoms (as described in section 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 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.

6.2.1.3 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 (Figure 6.5(e)). 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 (Figure 6.5(f)).

Therefore, instead of directly texturing a relocated phantom, we will first render the phantom at its original location, as in the previous approach. 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 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.

6.2.1.4 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 (Figure 6.6).

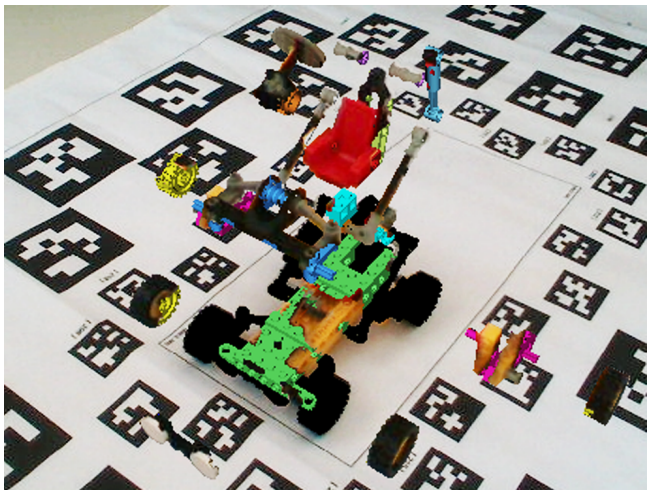


Figure 6.6: Bad example of an explosion diagram in AR. No further shading of the transferred real world information is used. Notice the clutter of real and virtual information.

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 of an object. The empty area is filled using the mean value of the all identified real world background information (Figure 6.2.2.1).

This technique was chosen because it is simple and fast, and leads to acceptable results for the applications we are currently considering. However, more advances techniques such as inpainting [8] exist, but are left as future work.

6.2.2 Visualization

While conventional illustrations can arrange all visual elements on a uniform background, AR visualizations must deal with cluttered real world images. A relocated object will therefore not only be moved out of its natural context, but will be transported in a new, potentially even worse context. The new context is usually not related to the exploded part at all and can potentially lead to difficulties in understanding the explosion diagram.

We therefore must emphasize the exploded parts, so that they successfully stand out from their background. We will first demonstrate visualizations of single parts, before we discuss visual linking between associated parts in AR.

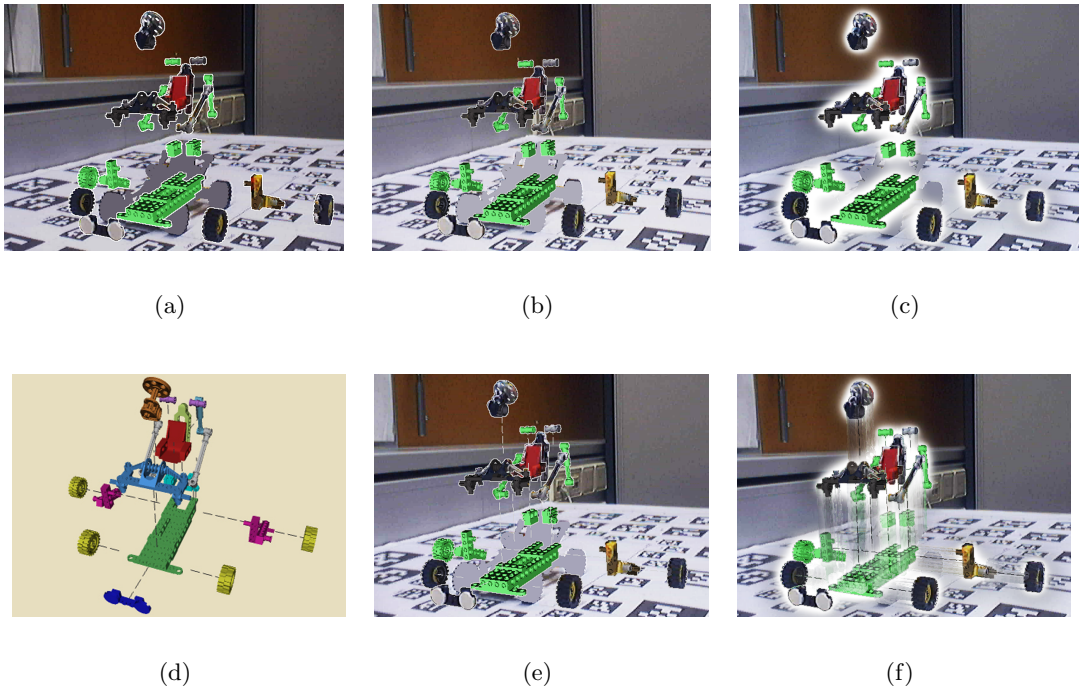
Figure 6.6 shows an explosion diagram in a real world environment. The image was rendered using the techniques discussed in the previous section. It clearly shows two problems of a visualization of an explosion diagram in AR. Firstly, the exploded parts of an object are hardly distinguishable from its surroundings in its new location. Secondly, a combination of real and virtual imagery is rather confusing if presented at the same object.

6.2.2.1 Visual Part Discrimination

To support the visual distinction between the parts of an explosion diagram and its current video background, we highlight the border of the part's 2D footprint (Figure 6.7(a)). The border is identified in 2D by applying an edge detector on an ID buffer which is created during rendering the parts of the explosion diagram.

However, while a uniformly-colored boundary adds a very strong discriminator in situations where a high contrast between background and foreground information is given, the same techniques fails if the boundary is of similar color as the information in the background. Figure 6.7(a) and Figure 6.7(b) show the same borderline in two different level of brightness. While one is easily noticeable, the other can be hardly distinguished from the video background.

Consequently the discriminator has to be adapted to the current situation. In traditional illustrations the background color is usually known. In contrast, AR must deal with dynamically changing information in the background of an explosion diagram. Dynamically computing a suitable high contrast color for each object's boundary per frame creates disturbing temporal variations. Similar to the idea of an *Anti-Interference Outline* [58], we render the borderline using multiple shades (Figure 6.7(c)). In order to compute multiple shades we compute a halo, which softly changes the brightness of an object's borderline depending on the distance to it. To not cover the explosion diagram itself with the halo, we use it only on top of the video background, while the borderline is used as visual discriminator over the parts of an explosion.



(a) (b) (c)

(d) (e) (f)

Figure 6.7: Visualization techniques. (a) White object boundaries used to discriminate objects from background (b) Grey outlines are almost invisible (c) Halos and edge combination are better able to visually discriminate an object from its background. Halos are only used over background information while edges emphasis the parts of an object where they overlap other parts (d) Explosion diagram using dashed lines to visually link its parts in VR (e) Explosion diagram using dashed lines in AR (f) Using motion blur in addition to visually link parts in AR.

6.2.2.2 Visual Part Unification

The second problem of simple transformations of video texture appears when a mixture of real and virtual data appears on a single part of an explosion (see Figure 6.6). While an explosion diagram reveals formerly hidden parts, not enough video information is available to cover the entire visualization. Therefore the video information must be supplemented with virtual information, similar to the requirements of the restoration of hidden information.

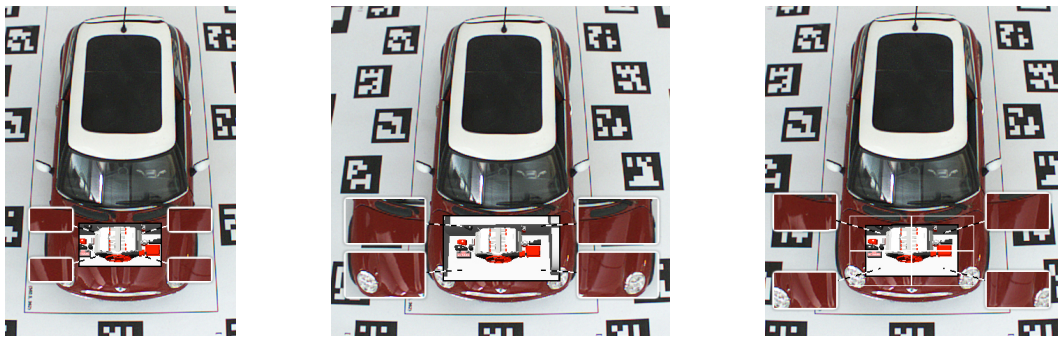
The chosen strategy to visually unify the occurring material depends on the ratio of visible virtual to real world information. The visible pixels are counted with an occlusion query before pixel shading is applied. If the amount of available video pixel is too small (empirically set to less than 50%), we will only use the virtual color of an object (Figure 6.1, Figure 6.2.2.1). 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 devided by the amount of pixel at the border. Note the change in color on the front left wheel in Figure 6.6 and Figure 6.1.

6.2.2.3 Part Linking

To support the process of mentally reassembling an explosion diagram, traditional illustrations commonly use connection lines to visually link the corresponding parts of the object. The drawings often use rather thin and dashed (or dotted) line styles, which must be drawn with high contrast to be perceived well (Figure 6.7(d)). This is usually achieved through a homogeneously inked background and a conspicuous colorization of the connection lines, relative to the background color.

In contrast, real world environments are by no means uniform and thin lines will be easily overlooked (Figure 6.7(e)). Therefore, we add visual discriminators between the objects and the background imagery to emphasize the foreground objects. An example is the haloed outlines presented in the last section. To furthermore avoid discontinuities, we



(a) (b) (c)

Figure 6.8: Linking Content in Image Based Explosions (a) Even though dashed lines are used to link the exploded parts to its original location it is difficult to reassemble the occluding structure (b) By duplicating the information we are able to use content itself to visually link the parts of the explosion (c) Visualizing the original location helps to clarify the content of the exploded parts

embed the virtual information within a uniformly-colored background. For this purpose we do not only use dashed connection lines, but rely on motion blur effects of the exploded parts as they form a relatively uniform visual discriminator (Figure 6.7(f)).

In a three-dimensional explosion diagram, the content of each exploded part may already indicate its original location. However, in case of an explosion presentation, which uses a geometrical primitive (section 6.1.2), its parts may not present any content which allows to infer its original location. Even if visual links are used, the exploded content may difficult to relate to the unexploded remainder (Figure 6.8(a)). To enhance the understanding of the relation between the exploded and unexploded parts the presented system allows to increase the size of each exploded part until it includes real world landmarks. However, to use real world landmarks as visual links, we have to keep them in both parts of the presentation (in the exploded and the unexploded parts, Figure 6.8(b)). Since a duplication in an explosion diagram may confuse its user, we furthermore link the exploded part to its original location by presenting its outline in its unexploded location (Figure 6.8(c)).

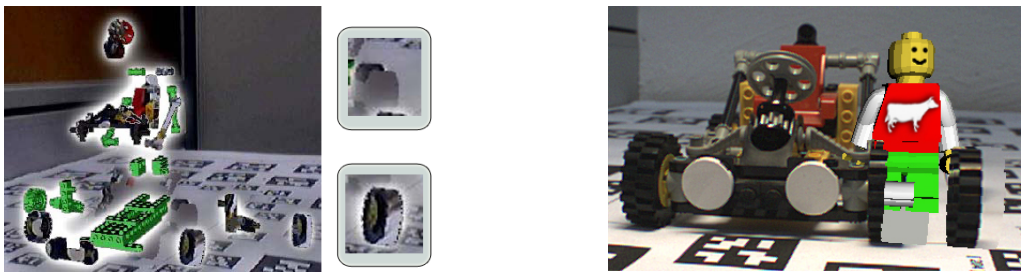
6.2.2.4 Error Friendly Visualization

If the explosion diagram is generated by using registered 3d phantom objects, imperfect tracking data and other sources of registration errors are causing false video masks of the exploded parts. The arising segmentation error causes a mixture of incomplete parts of different objects inside the mask. Such a combination of video data of different incomplete parts will most likely generate an incomprehensible explosion diagram (Figure 6.9(a)/(b)).

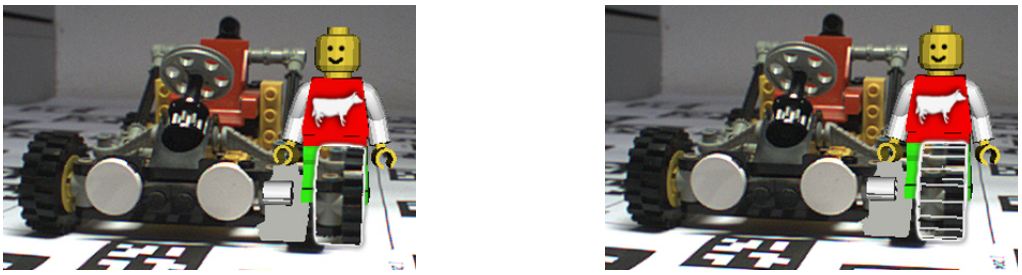
To ameliorate the error, Figure 6.9(c) demonstrates the use of contour lines as proposed in the previous section. Even though contour lines are able to visually group the repositioned parts of the different objects they fail to clarify the presentation. A more unambiguous presentation is possible by using an emphasized ghosting next to contour lines. The additional ghost presentation is able to encode the shape of the exploded part (Figure 6.9(c))

Even though the presentation in Figure 6.9(c) is able to indicate the shape of the exploded part, no visual indication of the current error is available. An emphasized ghost presentation on an explosion view is only able to communicate very small errors. However, with an increasing error of the current registration data, the real world information makes the understanding of the presentation more difficult than helpful. Consequently, a virtual presentation will generate more comprehensible results (Figure 6.10(a)). Nevertheless, if

the communication of the error is important to the application, a purely virtual presentation of exploded parts is inappropriate, because the error can only be assumed from the 2d contour of the video cut out of the unexploded object. To be able to communicate the error, we increase the size of the exploded parts, which allows to make use of an emphasized ghosting to present the current error (as presented in section 4.2.1.4). Notice, that the resized part has to be big enough to contain both the exploded real world part and its registered virtual counterpart from which the ghost presentation is computed (Figure 6.10b).

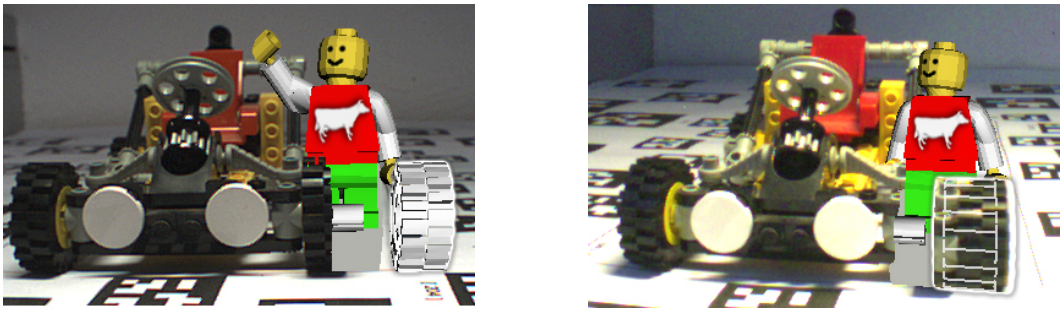


(a)



(b) (c)

Figure 6.9: The Effect of Erroneous Registration Data to an Exploded View. (a) If real world information is used on an exploded part, the registration error affects the comprehensibility of the parts itself. (b) Using the outline of the exploded part helps to differentiate the content from the background. (c) Adding an emphasized ghost presentation increased the comprehension of the parts. The real world information is irrelevant for the visual communication of the parts.



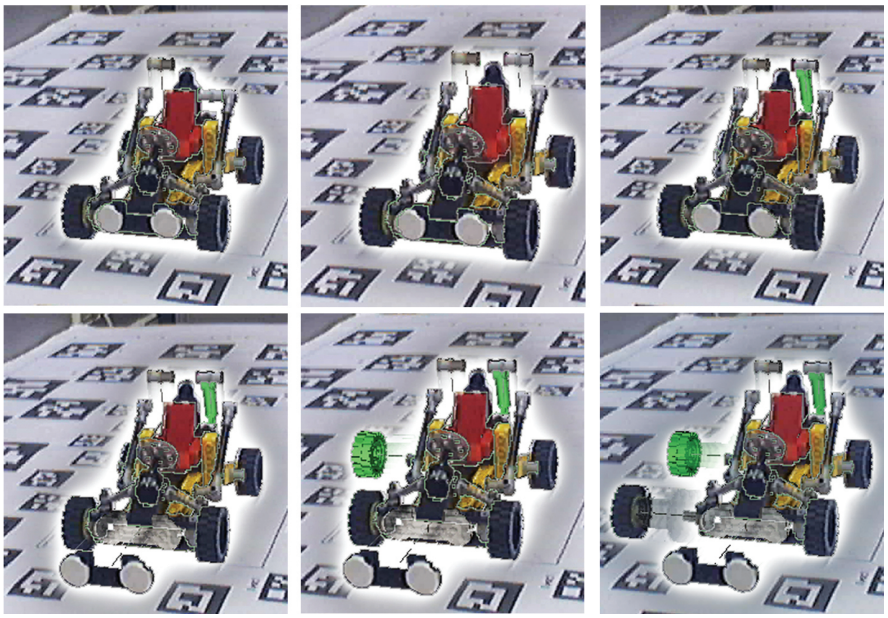
(a) (b)

Figure 6.10: Error Friendly Visualizations (a) A virtual representation of the exploded part is able to hide the registration error, resulting in a more comprehensible visualization. (b) If a communication of the error is relevant for the application, an enlarged exploded part in combination with an emphasized ghosting is able to present the error even in an explosion diagram.

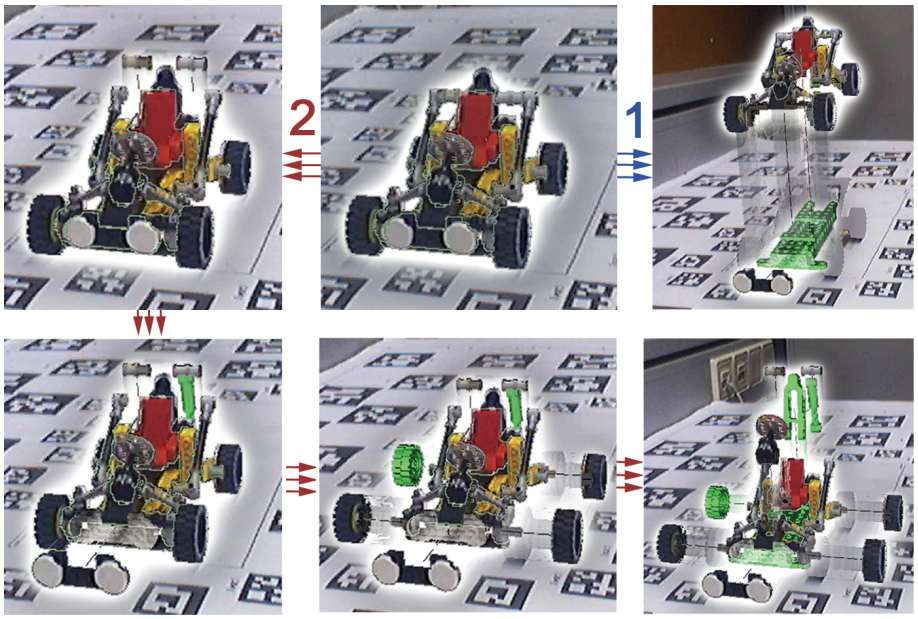
6.2.3 Animation Styles

Since we are aiming for interactive presentations of explosion diagrams in AR, we are able to utilize animation styles to further enhance the comprehension of our presentations. We have implemented three different animation styles. An explosion of all parts at once ((Figure 6.11(b)-1) and two styles supporting grouped explosions (Figure 6.11(a), Figure 6.11(b)-2).

To follow the assembly, the system implements animations which remove all parts one after the other (Figure 6.11(a)). To outline semantic groups of parts, the system also supports exploding all parts which belong to the same group at once (Figure 6.11(b)-2). These two strategies allow to explore an object's assembly. However, notice once again, x-ray visualizations do not require an explosion diagram which presents the whole disassembly of an object.



(a)



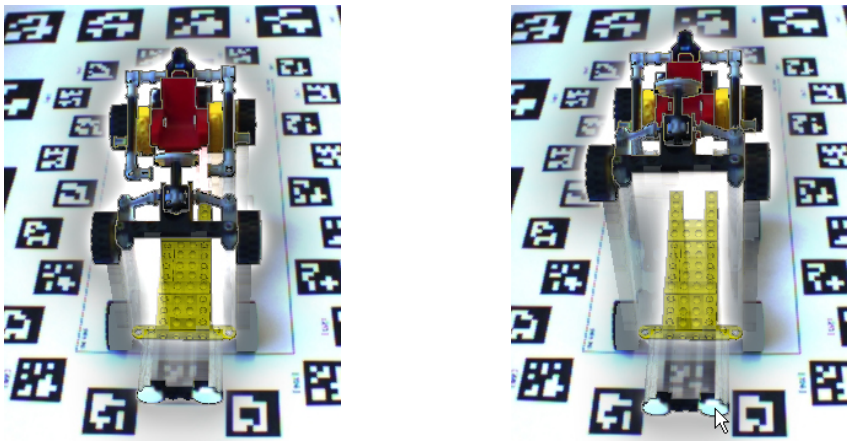
(b)

Figure 6.11: Animation Styles. (a) Single part explosion. Parts are exploded one after the other (b) Grouped explosions. 1- All in one animation applied on Focus and Context groups 2- semantically grouped items exploded at the same time

6.2.4 Viewpoint Integration

The layout of the explosion diagram enables a comprehensible exploration of otherwise invisible parts. However, the layout is independent of the current viewing direction and thus the visibility of hidden structure cannot be ensured. Even if the visual complexity of the scene is reduced by using a layout, which generates a minimal set of groups to separate the structure of interest from other parts of the object, these parts may obscure the focus element at a certain angle. Figure 6.12(a) demonstrates this problem. Even though the layout enforces a minimal set of exploded groups the object of interest is partially occluded from the chosen point of view.

To ensure the visibility of the object of interest in an explosion diagram, the system is able to alter the distances of exploded parts, similar to [115]. In each frame, a 3d cone is rendered with its bottom at the camera's viewing plane and its tip at the back-face of the bounding box of the object of interest. All parts of the explosion diagram are tested against the volume of the 3d cone. If one falls inside, its explosion distance is adjusted, so that this is no longer the case.



(a) (b)

Figure 6.12: Viewpoint integration. (a) The explosion diagram is presented without considering the current point of view. Consequently, the visibility of the focus element is ignored. (b) A invisible 3d cone with its tip at the back face of the bounding box of the object of interest ensure the visibility of the focus object.

While the grouped explosion layout does not include too many different explosion directions, it allows to clearly see the object of interest from quite a number of different points of view. However, from some points of view it occludes the object of interest. For

example, if the camera is positioned in front of the car, only moving the car's light behind the camera will allow complete uncover the object of interest. This strategy removes the part from the scene and thus is not an ideal solution. It might be better to move the part up with the other parts, even though they are not directly connected. Following this line of though, explosion layouts which consider the current point of view may even use indirect part relation. A future study should investigate automatic layout computations, which consider indirect part relations to ensure constraints. Since the viewpoint usually changes in an interactive application, the future system should also evaluate the effectiveness of a layouts if visibility constraints have to be considered. Furthermore, if layouts have to change between different points of view, the future project should incorporate properties of explosion layouts to be able to interpolate between them.

Chapter 7

Conclusions

Augmented reality displays extend viewer perception via the addition of computer-generated information. One of the main advantages of AR applications is their ability to reveal hidden objects by presenting their virtual counterpart at exactly the same location as the original object appears. However, the success of an AR application is dependent on to the comprehensibility of its visualization. Thus, this thesis focuses on the application of illustrative x-ray visualization techniques with the goal to enhance AR presentations of otherwise hidden structures.

To visually communicate spatial arrangements between hidden and occluding structures, x-ray presentations have to take special care about the information which is about to be occluded by the overlaid virtual data. This thesis exploits the capabilities of a presentation of contextual information to support the comprehension of x-ray visualizations in AR environments. To identify contextual information, this thesis applies rendering methods which are inspired by illustrative visualization techniques. Ghost presentations (chapter 4) have been applied to render objects non-uniformly transparent, Cutaway presentations (chapter 5) are used to incise a cavity in an occluder and Explosion diagrams (chapter 6) are able to present hidden and occluding structure in full detail. While all of them are widely used in illustrations, this thesis discusses their real-time integration into interactive and visually complex AR environments. It presents the generation of all three methods from either registered phantom models, pure image data or by interactively controlling auxiliary primitives. As real-world data are naturally imperfect, this thesis furthermore discusses the performance of the different techniques using erroneous data sources. Moreover, the complex character of AR environments requires complex visualization techniques to neither isolate certain structure nor to generate ambiguous presentations. This thesis

presents the idea of cascading Focus and Context visualization techniques to handle complex AR environments (section 3.3).

7.1 Closing Remarks

Even though there are many situations in which illustrative x-ray visualizations are able to enhance the understanding of the visualization, none of the techniques is universally applicable to an AR environment. In fact, all presented techniques require certain characteristics of the occluding structures, or else they fail to create expressive visualizations. For example, to effectively use cutaway-visualizations, the occluding object has to be available as a 3d phantom object, or otherwise no contextual information can be computed to indicate the cavity. Ghost presentations in turn may suffer from an inappropriate feature detector, which creates its sparse representations. A ghost presentation computed by using a poor detector or an occluder with too little features may not preserve enough depth cues to successfully communicate object arrangements. Similarly, a poor feature detector in combination with too many features easily causes cluttered displays. Since an automatic feature extraction for ghostings may become difficult, researchers have experimented with textures to introduce an artificial structure, helping to convey the 3D shape of an object [68]. However, using an artificial structure to preserve real world information may interfere with texture information, which is already present on real world objects, making it difficult to apply such textures in AR.

Even more problematic than the amount of preserved features is the fact that both ghosting and cutaway renderings remove information from the occluding objects. As a remedy, explosion diagrams are able to present all objects in uncovered and in full detail. Nevertheless, explosion diagrams require a comprehensible layout and if integrated in AR environments a very precise registration.

Furthermore, any visualization in an Augmented Reality environment consist of a visual interaction between the presentations of virtual and real word structure. This interaction is a key element towards the generation of comprehensible visualizations, and the AR display has to ensure the generation of visually harmonizing elements. To handle visually complex and dynamic environments, AR shading can modulate the appearance in a way that its effectiveness is ensured (see section 3.3). However, such changes are rather pronounced and may disturb the experience of presence in a real-world environment.

7.2 Future Work

While this thesis introduces illustrative x-ray visualizations to AR environments the presented work is just a first step towards interactive and universally applicable illustrative x-ray visualizations. Especially in dynamic real world environments, a single and preconfigured x-ray visualization technique will fail to render comprehensible results. This thesis shows the strengths and the shortcomings of the visualization techniques mostly isolated. An online adaptation of the parameters of a single technique, to the conditions of the real world environment or a substitution of the technique itself or a combination of techniques are all promising directions for future work.

Thus, a future system should be able to evaluate all available data to either adjust rendering parameters (such as the amount or source of features which subsequently generate a ghosting) or to combine different techniques. Such an adaptive system seems to be a promising step towards a universally applicable x-ray visualization, which is not only robust but also able to avoid unnecessary and task irrelevant interactions.

Figure 7.1 shows one possible continuum which defines combinations of the presented x-ray visualization techniques. Depending on the current visibility settings of occluding and hidden elements, a combined technique (such as a ghosted-cutaway or a ghosted-explosion diagram) is defined.

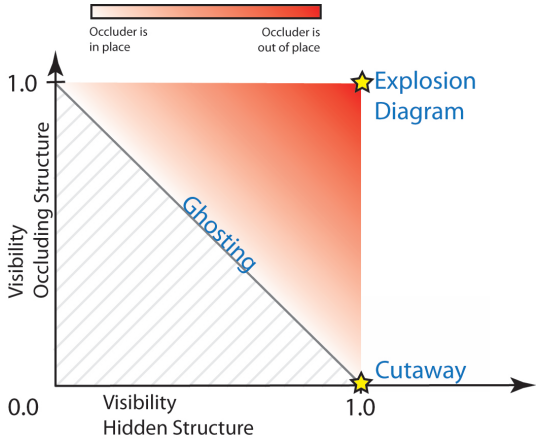


Figure 7.1: Visibility Diagram to combine Cutaway-, Ghost and Explosion Presentations

A future project should also invest further research on the generation of the basic elements of illustrations from and in combination with real world video data. This includes the extraction of characteristic object features from 2d video data as well as from 2-1/2D online depth data (which may be generated by an analysis of the optical flow or given by

the output of a commercially available stereo camera).

Furthermore, basic research on the perception of Focus and Context visualization techniques in AR could support the integration of illustrative rendering techniques in real world environments. User studies have to elaborate the influence of perceptual parameters, such as the environmental clutter or the visual saliency of the involved elements, to control when to apply which illustrative visualization technique in AR best.

Acknowledgments

I would like to thank everybody who supported me during my PhD. For the scientific advisory, I would like to thank my supervisor Dieter Schmalstieg, who always pushed me into the right direction when I was about to lose focus. In addition, many thanks go to all the people at the institute with whom I had technical and non-technical discussions. Special thanks should be expressed to Erick Mendez, Markus Tatzgern, Stefan Hauswiesner, Christopher Dissauer, Alex Bornik and Bernhard Kainz for the discussions, which in some way or the other had a direct input to this thesis.

In addition to the people in my scientific life, I would like to express my deepest thanks to my whole family, especially to Judith for tolerating my often too long working hours.

Bibliography

[1] Agrawala, M., Phan, D., Heiser, J., Haymaker, J., Klingner, J., Hanrahan, P., and Tversky, B. (2003). Designing effective step-by-step assembly instructions. *ACM Transactions on Graphics*, 22(3):828–837.

[2] Ali, K., Hartmann, K., Fuchs, G., and Schumann, H. (2008). Adaptive layout for interactive documents. In *SG '08: Proceedings of the 9th international symposium on Smart Graphics*, pages 247–254, Berlin, Heidelberg. Springer-Verlag.

[3] Appel, A., Rohlf, F. J., and Stein, A. J. (1979). The haloed line effect for hidden line elimination. volume 13, pages 151–157, New York, NY, USA. ACM.

[4] Azuma, R., Baillot, Y., Behringer, R., Feiner, S., Julier, S., and Macintyre, B. (2001). Recent advances in augmented reality. *IEEE Computer Graphics and Applications*, 21(6):34–47.

[5] Bajura, M., Fuchs, H., and Ohbuchi, R. (1992). Merging virtual objects with the real world: seeing ultrasound imagery within the patient. In *Proceedings of ACM SIGGRAPH*, pages 203–210, New York, NY, USA. ACM.

[6] Ball, T. and Eick, S. G. (1996). Software visualization in the large. *Computer*, 29(4):33–43.

[7] Bane, R. and Hollerer, T. (2004). Interactive tools for virtual x-ray vision in mobile augmented reality. In *ISMAR '04: Proceedings of the IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 231–239, Washington, DC, USA. IEEE Computer Society.

[8] Bertalmio, M., Sapiro, G., Caselles, V., and Ballester, C. (2000). Image inpainting. In *Proceedings of ACM SIGGRAPH*, pages 417–424, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

[9] Bichlmeier, C. and Navab, N. (2006). Virtual window for improved depth perception in medical AR. In *AMIARCS - Virtual Window for Improved Depth Perception in Medical AR*, Copenhagen, Denmark. MICCAI Society.

[10] Bichlmeier, C., Sielhorst, T., Heining, S. M., and Navab, N. (2007a). Improving depth perception in medical augmented reality: A virtual vision panel to the inside of the patient. In *Proceedings of BVM 2007*. Springer.

[11] Bichlmeier, C., Wimmer, F., Sandro Michael, H., and Nassir, N. (2007b). Contextual anatomic mimesis: Hybrid in-situ visualization method for improving multi-sensory depth perception in medical augmented reality. In *ISMAR '07: Proceedings of the International Symposium on Mixed and Augmented Reality*, pages 129–138.

[12] Bier, E. A., Stone, M. C., Pier, K., Buxton, W., and DeRose, T. (1993). Toolglass and magic lenses: The see-through interface. In *Proceedings of ACM SIGGRAPH*, pages 73–80, Anaheim, CA.

[13] Breen, D. E., Whitaker, R. T., Rose, E., and Tuceryan, M. (1996). Interactive occlusion and automatic object placement for augmented reality. *Computer Graphics Forum*, 15(3):11–22.

[14] Bruckner, S., Grimm, S., Kanitsar, A., and Gröller, M. E. (2006). Illustrative context-preserving exploration of volume data. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1559–1569.

[15] Bruckner, S. and Gröller, E. (2007). Enhancing depth-perception with flexible volumetric halos. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1344–1351.

[16] Bruckner, S. and Gröller, M. E. (2006). Exploded views for volume data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1077–1084.

[17] Buchmann, V., Nilsen, T., and Billinghurst, M. (2005). Interaction with partially transparent hands and objects. In Billinghurst, M. and Cockburn, A., editors, *Sixth Australian User Interface Conference (AUIC2005)*, volume 40 of *CRPIT*, pages 17–20, Newcastle, Australia. ACS.

[18] Burns, M. and Finkelstein, A. (2008). Adaptive cutaways for comprehensible rendering of polygonal scenes. In *Proceedings of ACM SIGGRAPH Asia*, pages 1–7, New York, NY, USA. ACM.

[19] Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., and Stal, M. (1996). *Pattern-Oriented Software Architecture, Volume 1: A System of Patterns*. Wiley.

[20] Cakmakci, O. and Rolland, J. (2006). Head-worn displays: A review. *Display Technology*, 2(3):199–216.

[21] Canny, F. J. (1986). A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698.

[22] Card, S. K., Mackinlay, J. D., and Shneiderman, B. (1999). *Readings in information visualization: using vision to think*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[23] Carpendale, M. S. T., Tigges, M., Cowperthwaite, D. J., and Fracchia, F. D. (1998). Bringing the advantages of 3d distortion viewing into focus. In *IEEE Visualization*, pages 17–20. IEEE Computer Society and ACM.

[24] Catmull, E. E. (1974). *A Subdivision Algorithm for Computer Display of Curved Surfaces*. PhD thesis, Department of Compuer Science, University of Utah.

[25] Chen, Jason, H.-L., Samavati, Faramarz, F., Sousa, and Costa, M. (2008). GPU-based point radiation for interactive volume sculpting and segmentation. *The Visual Computer*, 24(7-9):689–698.

[26] Close, B., Donoghue, J., Squires, J., Bondi, P. D., Morris, M., Piekarski, W., Thomas, B., Thomas, B., and Au, U. E. (2000). ARQuake: An outdoor/indoor Augmented Reality first person application. In *International Symposium on Wearable Computers*, pages 139–146.

[27] Coelho, E. M., MacIntyre, B., and Julier, S. J. (2004). Osgar: A scene graph with uncertain transformations. In *ISMAR'04: Proceedings of the IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 6–15.

[28] Coffin, C. and Hollerer, T. (2006). Interactive perspective cut-away views for general 3d scenes. In *3DUI '06: Proceedings of the IEEE Symposium on 3D User Interfaces*, pages 25–28.

[29] Cole, F., DeCarlo, D., Finkelstein, A., Kin, K., Morley, K., and Santella, A. (2006). Directing gaze in 3D models with stylized focus. *Eurographics Symposium on Rendering*, pages 377–387.

[30] Company, T. L. (2002). *Robotics Invention System 2.0, Manual Lego Mindstorms*. Lego.

[31] Cook, R. L. (1984). Shade trees. volume 18, pages 223–231, New York, NY, USA. ACM.

[32] Correa, C., Silver, D., and Chen, M. (2007). Illustrative deformation for data exploration. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1320–1327.

[33] Crow, F. C. (1978). Shaded computer graphics in the entertainment industry. *Computer*, 11(3):11–22.

[34] DeCarlo, D., Finkelstein, A., and Rusinkiewicz, S. (2004). Interactive rendering of suggestive contours with temporal coherence. In *NPAR '04: International Symposium on Non-Photorealistic Animation and Rendering*, pages 15–24.

[35] DeCarlo, D., Finkelstein, A., Rusinkiewicz, S., and Santella, A. (2003). Suggestive contours for conveying shape. volume 22, pages 848–855.

[36] Diepstraten, J., Weiskopf, D., and Ertl, T. (2002). Transparency in interactive technical illustrations. *Computer Graphics Forum*, 21:317–325.

[37] Diepstraten, J., Weiskopf, D., and Ertl, T. (2003). Interactive Cutaway Illustrations. *Computer Graphics Forum*, 22:523–532.

[38] Dissauer, C. and Kalkofen, D. (2008). A GPU based framework for real-time NPR. Technical Report TR-08-01, Institute for Computer Graphics and Vision, Graz University of Technology, Austria.

[39] Doleisch, H., Gasser, M., and Hauser, H. (2003). Interactive feature specification for focus+context visualization of complex simulation data. In *VISSYM '03: Proceedings of the Symposium on Data Visualization*, pages 239–248, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

[40] Doleisch, H. and Hauser, H. (2002). Smooth brushing for focus+context visualization of simulation data in 3d. *In the Proceedings of the International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media 2002 (WSCG 2002)*, pages 147–154.

[41] Doussis, E. (1993). *An Ultrasonic Position Detecting System for Motion Tracking in Three Dimensions*. PhD thesis, Tulane University.

[42] Elber, G. (1995). Line illustrations computer graphics. *The Visual Computer*, 11(6):290–296.

[43] Elmqvist, N. and Tsigas, P. (2007). A taxonomy of 3D occlusion management techniques. In *VR '07: Proceedings of the IEEE Virtual Reality Conference*, pages 51–58.

[44] Everitt, C. (2001). Interactive order-independent transparency. Whitepaper, Nvidia.

[45] Feiner, S. and Duncan Seligmann, D. (1992). Cutaways and ghosting: Satisfying visibility constraints in dynamic 3d illustrations. *VC*, 8:292–302.

[46] Feiner, S., Macintyre, B., and Seligmann, D. (1993). Knowledge-based augmented reality. *Commun. ACM*, 36(7):53–62.

[47] Foxlin, E. (1996). Inertial head-tracker sensor fusion by a complementary separate-bias kalman filter. In *Proceedings of the IEEE Virtual Reality*, pages 185–194.

[48] Furmanski, C., Azuma, R., and Daily, M. (2002). Augmented Reality visualizations guided by cognition: Perceptual heuristics for combining visible and obscured information. In *ISMAR '02: Proceedings of the International Symposium on Mixed and Augmented Reality*, pages 215–320, Washington, DC, USA. IEEE Computer Society.

[49] Goebbels, G., Troche, K., Braun, M., Ivanovic, A., Grab, A., von Lübtow, K., Sader, R., Zeilhofer, F., Albrecht, K., and Praxmarer, K. (2003). Arsystricorder, development of an augmented reality system for intraoperative navigation in maxillofacial surgery. In *ISMAR '03: Proceedings of the International Symposium on Mixed and Augmented Reality*. IEEE Computer Society.

[50] Goldstein, E. B. (1996). *Sensation and Perception*. Brooks/Cole, Pacific Grove, CA.

[51] Gooch, A., Gooch, B., Shirley, P., and Cohen, E. (1998). A non-photorealistic lighting model for automatic technical illustration. In *Proceedings of ACM SIGGRAPH*, pages 447–452, New York, NY, USA. ACM.

[52] Gooch, A. A. and Gooch, B. (2001). *Non-Photorealistic Rendering*. AK Peters, Ltd. ISBN: 1568811330.

[53] Gooch, B., Reinhard, E., and Gooch, A. (2004). Human facial illustrations: Creation and psychophysical evaluation. *ACM Transactions on Graphics*, 23(1):27–44.

[54] Grabler, F., Agrawala, M., Li, W., Dontcheva, M., and Igarashi, T. (2009). Generating photo manipulation tutorials by demonstration. *ACM Transactions on Graphics*, 28(3):1–9.

[55] Gray, H. (1918). *Anatomy of the Human Body*. LEA & FEBIGER.

[56] Grosch, T. (2007). *Augmentierte Bildsynthese (Dissertation)*. Universität Koblenz-Landau, Der Andere Verlag.

[57] Hamel, J., Schlechtweg, S., and Strothotte, T. (1998). An approach to visualizing transparency in computer-generated line drawings. In *Proceedings of the IEEE Conference on Information Visualization*, pages 151–156.

[58] Harrison, B. L. and Vicente, K. J. (1996). An experimental evaluation of transparent menu usage. In *CHI '96: Proceedings of the SIGCHI conference on human factors in computing systems*, pages 391–398, New York, NY, USA. ACM.

[59] Hartmann, K., Ali, K., and Strothotte, T. (2004). Floating labels: Applying dynamic potential fields for label layout. In *Smart Graphics*, pages 101–113.

[60] Hauser, H., Mroz, L., Bischi, G. I., and Gröller, M. E. (2001). Two-level volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 7(3):242–252.

[61] Heiser, J., Phan, D., Agrawala, M., Tversky, B., and Hanrahan, P. (2004). Identification and Validation of Cognitive Design Principles for Automated Generation of Assembly Instructions. In *AVI '04: Proceedings of the working conference on Advanced visual interfaces*, pages 311–319, New York, NY, USA. ACM Press.

[62] Hertzmann, A. (1999). Introduction to 3D Non-Photorealistic Rendering: Silhouettes and Outlines. In Green, S., editor, *ACM SIGGRAPH 99 Course Notes. Course on Non-Photorelistic Rendering*, chapter 7. ACM Press/ACM SIGGRAPH, New York.

[63] Hodges, E. R. S., editor (2003). *The Guild Handbook of Scientific Illustration*. John Wiley & Sons, Hoboken, NJ, 2nd edition.

[64] Holloway, R. L. (1997). Registration error analysis for augmented reality. *Presence*, 6(4):413–432.

[65] Homem de Mello, L. and Sanderson, A. (1991). A Correct and Complete Algorithm for the Generation of Mechanical Assembly Sequences. In *IEEE Transaction on Robotics and Automation*, volume 7, pages 228–240.

[66] Interrante, V., Fuchs, H., and Pizer, S. (1995). Enhancing transparent skin surfaces with ridge and valley lines. In *Proceedings of IEEE Visualization*, pages 52–59.

[67] Interrante, V., Fuchs, H., and Pizer, S. (1996). Illustrating transparent surfaces with curvature-directed strokes. In *In IEEE Visualization '96. IEEE*, pages 211–218.

[68] Interrante, V., Fuchs, H., Pizer, S. M., and Member, S. (1997). Conveying the 3d shape of smoothly curving transparent surfaces via texture. *IEEE Transactions on Visualization and Computer Graphics*, 3:98–117.

[69] Isenberg, T. and Brennecke, A. (2006). G-strokes: A concept for simplifying line stylization. *Computers & Graphics*, 30(5):754–766.

[70] Isenberg, T., Brennecke, A., Sousa, M. C., and Carpendale, S. (2006). Beyond Pixels: Illustration with Vector Graphics. Technical report, Departement of Computer Science, University of Calgary.

[71] Islam, S., Silver, D., and Chen, M. (2007). Volume splitting and its applications. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):193–203.

[72] Johansson, R. S., Westling, G., Backstrom, A., and Flanagan, J. R. (2001). Eye-hand coordination in object manipulation. *J. Neurosci.*, 21(17):6917–6932.

[73] Julier, S., Baillot, Y., Brown, D., and Lanzagorta, M. (2002). Information filtering for mobile augmented reality. *IEEE Computer Graphics and Applications*, 22(5):12–15.

[74] Kato, H. and Billinghurst, M. (1999). Marker tracking and HMD calibration for a video-based augmented reality conferencing system. *International Workshop on Augmented Reality*, 0:85–94.

[75] Kato, H., Billinghurst, M., Poupyrev, I., Imamoto, K., and Tachibana, K. (2000). Virtual object manipulation on a table-top ar environment. *ISAR '00: Proceedings of the IEEE/ACM International Symposium on Augmented Reality*, 0:111–119.

[76] Keahey, T. A. and Robertson, E. L. (1997). Nonlinear magnification fields. In *IEEE Symposium on Information Visualization*, pages 51–58. Press.

[77] Kim, S., Hagh-Shenas, H., and Interrante, V. (2004). Conveying three-dimensional shape with texture. In *APGV '04: Proceedings of the 1st Symposium on Applied Perception in Graphics and Visualization*, pages 119–122, New York, NY, USA. ACM.

[78] Kim, Y. and Varshney, A. (2006). Saliency-guided enhancement for volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):925–932.

[79] Klein, G. and Murray, D. (2007). Parallel tracking and mapping for small AR workspaces. In *ISMAR '07: Proceedings of the IEEE / ACM International Symposium on Mixed and Augmented Reality*, pages 1–10, Nara, Japan.

[80] Knödel, S., Hachet, M., and Guitton, P. (2009). Interactive generation and modification of cutaway illustrations for polygonal models. In *Proceedings of the 10th International Symposium on Smart Graphics*, pages 140–151, Berlin, Heidelberg. Springer-Verlag.

[81] Kobayashi, S. and Nomizu, K. (1996). *Foundations of Differential Geometry*. Wiley-Interscience.

[82] Kosara, R., Miksch, S., and Hauser, H. (2001). Semantic depth of field. In *Proceedings of the IEEE Symposium on Information Visualization 2001*, pages 97–104, Washington, DC, USA. IEEE Computer Society.

[83] Kruger, J., Schneider, J., and Westermann, R. (2006). Clearview: An interactive context preserving hotspot visualization technique. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):941–948.

[84] Lake, A., Marshall, C., Harris, M., and Blackstein, M. (2000). Stylized rendering techniques for scalable real-time 3d animation. In *NPAR '00: Proceedings of the International Symposium on Non-Photorealistic Animation and Rendering*, pages 13–20, New York, NY, USA. ACM.

[85] Leung, Y. K. and Apperley, M. D. (1994). A review and taxonomy of distortion-oriented presentation techniques. *ACM Transactions of Computer-Human Interaction*, 1(2):126–160.

[86] Li, W., Agrawala, M., Curless, B., and Salesin, D. (2008). Automated generation of interactive 3d exploded view diagrams. *ACM Transactions on Graphics*, 27(3):1–7.

[87] Li, W., Agrawala, M., and Salesin, D. (2004). Interactive image-based exploded view diagrams. In *GI '04: Proceedings of Graphics Interface*, pages 203–212.

[88] Li, W., Ritter, L., Agrawala, M., Curless, B., and Salesin, D. (2007). Interactive cutaway illustrations of complex 3d models. In *Proceedings of ACM SIGGRAPH*, pages 31–39, New York, NY, USA. ACM.

[89] Livingston, M. A., Swan II, J. E., Gabbard, J. L., Höllerer, T. H., Hix, D., Julier, S. J., Baillot, Y., and Brown, D. (2003). Resolving multiple occluded layers in augmented reality. In *ISMAR '03: Proceedings of the IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 56–65.

[90] Looser, J. (2007). *AR Magic Lenses: Addressing the Challenge of Focus and Context in Augmented Reality*. PhD thesis, University of Canterbury. Computer Science and Software Engineering.

[91] Macintyre, B., Coelho, E. M., and Julier, S. J. (2002). Estimating and adapting to registration errors in augmented reality systems. In *VR 2002: Proceedings of IEEE Virtual Reality Conference*, pages 73–80.

[92] McGuffin, M. J., Tancau, L., and Balakrishnan, R. (2003). Using deformations for browsing volumetric data. In *VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, page 53, Washington, DC, USA. IEEE Computer Society.

[93] Mendez, E., Kalkofen, D., and Schmalstieg, D. (2006). Interactive context-driven visualization tools for Augmented Reality. In *ISMAR'06: Proceedings of the IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 209–218.

[94] Milgram, P. and Kishino, F. (1994). A taxonomy of mixed reality visual displays. *IEICE Transactions on Information Systems*, E77-D(12).

[95] Neumann, U. and You, S. (1999). Natural feature tracking for augmented-reality. *IEEE Transactions on Multimedia*, 1(1):53–64.

[96] Niederauer, C., Houston, M., Agrawala, M., and Humphreys, G. (2003). Non-Invasive Interactive Visualization of Dynamic Architectural Environments. In *Proceedings of the Symposium on Interactive 3D Graphics*, pages 55–58.

[97] Nienhaus, M. and Döllner, J. (2005). Blueprint rendering and sketchy drawings. In Pharr, M., editor, *GPU Gems II: Programming Techniques for High Performance Graphics and General-Purpose Computation*, pages 235–252. Addison-Wesley Professional.

[98] Praun, E., Finkelstein, A., and Hoppe, H. (2000). Lapped textures. In *Proceedings of ACM SIGGRAPH*, pages 465–470.

[99] Praun, E., Hoppe, H., Webb, M., and Finkelstein, A. (2001). Real-time hatching. In *Proceedings of ACM SIGGRAPH*, pages 579–584, New York, NY, USA. ACM.

[100] Raab, A. and Rüger, M. (1996). 3D-Zoom: Interactive Visualisation of Structures and Relations in Complex Graphics. In B. Girod, H. Niemann, H.-P. S., editor, *3D Image Analysis and Synthesis*, pages 125–132. infix-Verlag.

[101] Raab, F., Blood, E. B., Steiner, T. O., and Jones, H. R. (1979). Magnetic position and orientation tracking system. *IEEE Transactions on Aerospace and Electronic Systems*, 5:709–718.

[102] Raskar, R., Welch, G., Low, K.-L., and Bandyopadhyay, D. (2001). Shader lamps: Animating real objects with image-based illumination. In *Rendering Techniques*, pages 89–102.

[103] Reitmayr, G. and Schmalstieg, D. (2001). Opentracker-an open software architecture for reconfigurable tracking based on xml. In *Proceedings of IEEE Virtual Reality*, volume 0, pages 285–292, Los Alamitos, CA, USA. IEEE Computer Society.

[104] Reitmayr, G. and Schmalstieg, D. (2005). Flexible parameterization of scene graphs. In *Proceedings IEEE Virtual Reality*, pages 51–58.

[105] Ritter, F., Preim, B., Deussen, O., and Strothotte, T. (2000). Using a 3D Puzzle as a Metaphor for Learning Spatial Relations. In *Graphics Interface*, pages 171–178. Morgan Kaufmann Publishers.

[106] Robertson, C. M., MacIntyre, B., and Walker, B. N. (2009). 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.

[107] Romney, B. (1997). *On the Concurrent Design of Assembly Sequences and Fixture*. PhD thesis, Stanford University, Stanford, CA, USA.

[108] Ropinski, T. and Hinrichs, K. (2004). Real-time rendering of 3d magic lenses having arbitrary convex shapes. In *In Journal of the International Winter School of Computer Graphics (WSCG04*, pages 379–386. Science Press.

[109] Rosenthal, M., State, A., Lee, J., Hirota, G., Ackerman, J., Keller, K., Pisano, E. D., Jiroutek, M., Muller, K., and Fuchs, H. (2001). Augmented Reality guidance for needle biopsies: A randomized, controlled trial in phantoms. In *MICCAI '01: Proceedings of the 4th International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 240–248, London, UK. Springer-Verlag.

[110] Saito, T. and Takahashi, T. (1990). Comprehensible rendering of 3-d shapes. volume 24, pages 197–206, New York, NY, USA. ACM.

[111] Santella, A. and DeCarlo, D. (2004). Visual interest and NPR: an evaluation and manifesto. In *NPAR '04: Proceedings of the International Symposium on Non-Photorealistic Animation and Rendering*, pages 71–150.

[112] Schwerdtfeger, B., , and Klinker, G. (2008). Supporting order picking with augmented reality. In *ISMAR'08: Proceedings of IEEE and ACM International Symposium on Mixed and Augmented reality*, pages 91–94.

[113] Seligmann, D. D. and Feiner, S. (1991). Automated generation of intent-based 3d illustrations. In *Proceedings of ACM SIGGRAPH*, pages 123–132, New York, NY, USA. ACM Press.

[114] Sielhorst, T., Bichleier, C., Heining, S., and Navab, N. (2006). Depth Perception A Major Issue in Medical AR: Evaluation Study by Twenty Surgeons. In *Medical Image Computing and Computer-Assisted Intervention*, pages 364–372.

[115] Sonnet, H., Carpendale, S., and Strothotte, T. (2004). Integrating Expanding Annotations with a 3D Explosion Probe. In *Advanced Visual Interfaces*, pages 63–70, New York, NY, USA. ACM Press.

[116] Streit, M., Kalkusch, M., Kashofer, K., and Schmalstieg, D. (2008). Navigation and Exploration of Interconnected Pathways. *Comput. Graph. Forum*, 27(3):951–958.

[117] Strothotte, T. and Schlechtweg, S. (2002). *Non Photorealistic Computer Graphics: Modeling, Rendering and Animation*. Morgan Kaufmann.

[118] Tatzgern, M. (2008). A framework for automatically creating 3d explosion diagrams. Master's thesis, Graz University of Technology.

[119] Thiel, W. (2002). *Photographischer Atlas der praktischen Anatomie*. Springer.

[120] Tietjen, C., Isenberg, T., and Preim, B. (2005). Combining silhouettes, surface, and volume rendering for surgery education and planning. In *Proceedings of EuroVis*, pages 303–310.

[121] Tjan, B., Braje, W., Legge, G., and Kersten, D. (1995). Human efficiency for recognizing 3-d objects in luminance noise. *Vision Research*, 35(21):3053–3069.

[122] Tweedie, L., Spence, B., Williams, D., and Bhogal, R. (1994). The attribute explorer. In *CHI '94: Conference companion on Human factors in computing systems*, pages 435–436, New York, NY, USA. ACM.

[123] Vicki Bruce, P. R. G. (1985). *Visual Perception : Physiology, Psychology and Ecology*. Erlbaum, second edition.

[124] Viola, I., Kanitsar, A., and Gröller, M. E. (2004a). Importance-driven volume rendering. In *Proceedings of IEEE Visualization'04*, pages 139–145.

[125] Viola, I., Kanitsar, A., and Groller, M. E. (2004b). Importance-driven volume rendering. In *VIS '04: Proceedings of the IEEE Visualization*, pages 139–146, Washington, DC, USA. IEEE Computer Society.

[126] Wagner, D., Pintaric, T., Ledermann, F., and Schmalstieg, D. (2005). Towards massively multi-user augmented reality on handheld devices. In *Pervasive '05: Proceedings of the International Conference on Pervasive Computing*, pages 208–219, Munich, Germany.

[127] Ware, C. (2004). *Information Visualization: Perception for Design (Morgan Kaufmann Interactive Technologies Series)*. Morgan Kaufmann Publishers, second edition.

[128] Wernecke, J. (1993). *The Inventor Mentor: Programming Object-Oriented 3d Graphics with Open Inventor, Release 2*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[129] Wither, J., Diverdi, S., and Hollerer, T. (2006). Using aerial photographs for improved mobile ar annotation. In *ISMAR '06: Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 159–162, Washington, DC, USA. IEEE Computer Society.

[130] Wither, J. and Hollerer, T. (2005). Pictorial depth cues for outdoor augmented reality. In *ISWC '05: Proceedings of the ninth IEEE International Symposium on Wearable Computers*, pages 92–99, Washington, DC, USA. IEEE Computer Society.

[131] Wolfe, J. M., Kluender, K. R., Levi, D. M., and et al (2004). *Sensation And Perception*. Sinauer Associates Inc, second edition.

[132] Zander, J., Isenberg, T., Schlechtweg, S., and Strothotte, T. (2004). High quality hatching. *Computer Graphics Forum (EG'04)*, 23(3):421–430.

[133] Zhai, S., Buxton, W., and Milgram, P. (1996). The partial-occlusion effect: utilizing semi-transparency in 3d human-computer interaction. *ACM Transactions on Computer-Human Interaction*, 3:254–284.