Modern Approaches to Augmented Reality

Oliver Bimber[1] and Ramesh Raskar[2]

[1]Bauhaus University, Weimar, Germany
T +49-3643-583724, F +49-3643-583709, Email: bimber@ieee.org, URL: http://www.uni-weimar.de/medien/AR

[2] MERL - Mitsubishi Electric Research Lab, Cambridge, USA
T +1-617-621-7533, F +1-617-621-7550, Email: raskar@merl.com, URL : http://www.merl.com/people/raskar/ , http://www.raskar.com

Abstract

This tutorial discusses the Spatial Augmented Reality (SAR) concept, its advantages and limitations. It will present examples of state-of-the-art display configurations, appropriate real-time rendering techniques, details about hardware and software implementations, and current areas of application. Specifically, it will describe techniques for optical combination using single/multiple spatially aligned mirror-beam splitters, image sources, transparent screens and optical holograms. Furthermore, it presents techniques for projector-based augmentation of geometrically complex and textured display surfaces, and (along with optical combination) methods for achieving consistent illumination and occlusion effects. Emerging technologies that have the potential of enhancing future augmented reality displays will be surveyed.

Categories and Subject Descriptors (according to ACM CSS): H.5.1 [Multimedia Information Systems]: Artificial, Augmented and Virtual Realities; I.3.1 [Hardware Architecture]: Three-dimensional Displays; I.3.2 [Computer Graphics]: Graphics Systems; I.3.3 [Computer Graphics]: Picture/Image Generation – Display Algorithms, Viewing Algorithms; I.3.7 [Computer Graphics]: Color, Shading, Shadowing, and Texture

1. Introduction and overview

Video see-through and optical see-through head-mounted displays have been the traditional output technologies for augmented reality (AR) applications for more than forty years. However, they still suffer from several technological and ergonomic drawbacks which prevent them from being used effectively in all application areas.

Novel approaches have taken augmented reality beyond traditional eye-worn or hand-held displays - enabling additional application areas. New display paradigms exploit large spatially aligned optical elements, such as mirror beam-splitters, transparent screens or holograms, as well as video-projectors. Thus, we call this technological variation "Spatial Augmented Reality (SAR)". In many situations, SAR displays are able to overcome technological and ergonomic limitations of conventional AR systems. Due to the fall in cost and availability of projection technology, personal computers and graphics hardware, there has been a considerable interest in exploiting SAR systems in universities, research laboratories, museums, industry and in the art community. Parallels to the development of virtual environments from head-attached displays to spatial projection screens can be clearly drawn. We believe that an analog evolution of augmented reality has the potential to yield a similar successful factor in many application domains. Thereby, SAR and body-attached AR are not competitive, but complementary.

Chapter 2 gives an overview over different augmented reality display techniques, from head-attached, over hand-held to spatial approaches. It will enable readers to identify parallels between virtual reality and augmented reality display technology, and stimulate them to think about alternative display approaches for AR.

Chapter 3 explains interactive rendering techniques that use fixed function and programmable rendering pipelines to support spatial optical see-through displays. They aim at neutralizing optical effects, such as reflection and refraction on planar or curved spatial optical combiners, such as transparent screen or mirror beam-splitter configurations.

Chapter 4 focuses on rendering methods for projector-based augmentation and illumination. It will be explained how a correct projection onto geometrically complex and textured screen surfaces is performed. Furthermore, it discusses projector-based illumination, and outlines

examples of how it can be used together with optical combination to create consistent illumination and occlusion effects.

In chapter 5 we summarize our tutorial and give an outlook to enabling technologies that might influence augmented reality technology in the future. The possibilities and limitations of technologies, such as autostereoscopy, video projectors, organic light emitting diodes, light emitting polymers, electronic paper, particular solid state volumetric and parallax display approaches, and holography will be outlined.

2. Augmented Reality Displays

Displays are image-forming systems that apply a set of optical, electronic and mechanical components to generate images somewhere on the *optical path* in-between the observer's eyes and the physical object to be augmented. Depending on the optics being used, the image can be formed on a plane or on a more complex non-planar surface.

Figure 2.1 illustrates the different possibilities of where the image can be formed, where the displays are located with respect to the observer and the real object, and what type of image is produced (i.e., planar or curved).

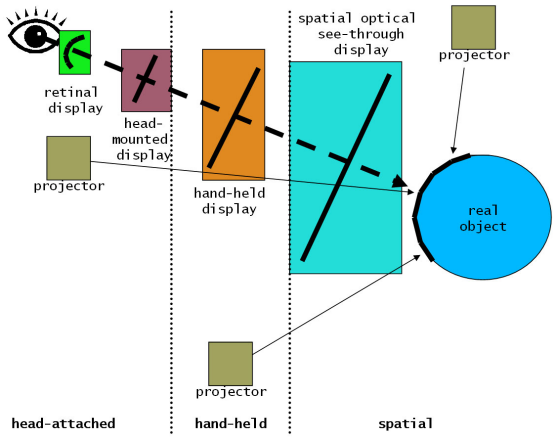


Figure 2.1: *Image-generation for augmented reality displays.*

Head-attached displays, such as retinal displays, head-mounted displays, and head-mounted projectors have to be worn by the observer. While some displays are *hand-held*, others are *spatially aligned* and completely detached from the users. Retinal displays and several projector-based approaches form curved images – either on the observer's retina or directly on the physical object. Most of the displays, however, form images on planes – called *image-planes* – that can be either head-attached or spatially aligned. Images behind real objects cannot be formed by a display that is located in front of real objects. In addition, if images are formed behind a real object, this object will occlude the image portion that is required to support augmentation.

Several pros and cons can be found by comparing the different types of displays. Most of them will be discussed within the following sections.

If *stereoscopic rendering* is used to present mixed (real and virtual) worlds, two basic fusion technologies are currently being used: video-mixing and optical combination.

While *video-mixing* merges live record video streams with computer generated graphics and displays the result on the screen, *optical combination* generates an optical image of the real screen (displaying computer graphics) which appears within the real environment (or within the viewer's visual field while observing the real environment). Both technologies entail a number of advantages and disadvantages which influence the type of application they can address.

Today, most of the stereoscopic AR displays require to wear some sort of goggles to provide stereo separation. Auto-stereoscopic approaches, however, might play a dominant role in the future of AR.

In this chapter, we discuss several types of augmented reality displays. Note that we rather present examples in each display category, than to provide a complete list of individual devices.

2.1. Head-Attached Displays

Head-attached displays require the user to wear the display system on his/her head. Depending on the image generation technology, three main types exist: Retinal displays that apply low power lasers to project images directly onto the retina of the eye, head-mounted displays that use miniature displays in front of the eyes, and head-mounted projectors that make use of miniature projectors or miniature LCD panels with backlighting and project images on the surfaces of the real environment.

2.1.1. Retinal Displays

Retinal displays [Kol93, Pry98, Lew04] utilize low-power semiconductor lasers (or –in future– special light-emitting diods) to scan modulated light directly onto the retina of the human eye, instead of providing screens in front of the eyes. This produces a much brighter and higher resolution image with a potentially wider field of view than a screen-based display.

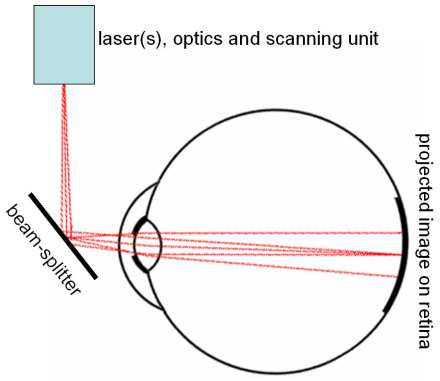


Fig 2.2: *Simplified diagram of a retinal Display.*

Current retinal displays share many shortcomings with head-munted displays (see section 2.1.2). However, some additional disadvantages can be identified for existing versions:

- Only monochrome (red) images are presented since cheap low-power blue and green lasers do not yet exist;
- The sense of ocular accommodation is not supported due to the complete bypass of the ocular motor-system by scanning directly onto the retina. Consequently, the focal length is fixed;
- Stereoscopic versions do not exist.

The main advantages of retinal displays are the high brigness and contrast, and low power consumption – which make them well suited for mobile outdoor applications. Future generations also hold the potential to provide dynamic re-focus, full-color stereoscopic images, and an extremly high resolution and large field-of-view.

2.1.2. Head-Mounted Displays

Head-mounted displays (HMDs) are currently the display devices which are mainly used for augmented reality applications.

Two different head-mounted display-technologies exist to superimpose graphics onto the user's view of the real world: *Video see-through head-mounted displays* that make use of video-mixing and display the merged images within a closed-view head-mounted display, or *optical see-through head-mounted displays* that make use of optical combiners (essentially half-silvered mirrors or transparent LCD displays). A comparison between these two general technologies can be found in [Rol94].

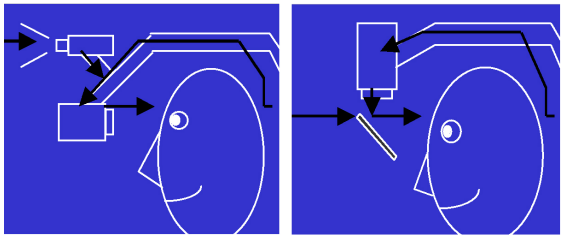


Figure 2.3: *Video see-through (left) and optical see-through (right). Courtesy: Azuma [Azu97].*

Several disadvantages can be related to the application of head-mounted displays as an augmented reality device. Note that most of these shortcomings are inherited from the general limitations of head-attached display technology:

- Lack in resolution that is due to limitations of the applied miniature displays. In the optical see-through case, only the graphical overlays suffer from a relatively low resolution, while the real environment can be perceived in the resolution of the human visual system. For video see-through devices, both – the real environment and the graphical overlays – are perceived in the resolution of the video-source or display;
- Limited field of view that is due to limitations of the applied optics;
- Imbalanced ratio between heavy optics (that results in cumbersome and uncomfortable devices) and ergonomic devices with a low image quality;
- Visual perception issues that is due to the constant image depth. For optical see-through: Since objects within the real environment and the image plane that is attached to the viewer's head are sensed at different depths, the eyes are forced to either continuously shift focus between the different depth levels, or perceive one depth level

unsharp. This is known as the *fixed focal length problem*, and is more critical for see-through than for closed-view head-mounted displays. For video see-through: Only one focal plane exists – the image-plane;

- Optical see-through devices require difficult (user and session dependent) calibration and precise head-tracking to ensure a correct graphical overlay. For video see-through, graphics can be integrated on a pixel-precise basis, but image processing for optical tracking increases the end-to-end system delay;
- Increased incidence of discomfort due to simulator sickness because of head-attached image plane (especially during fast head movements) [Pat00];
- Conventional optical see-through devices are incapable of providing consistent occlusion effects between real and virtual objects. This is due to the mirror beam splitters that reflect the light of the miniature displays which interferes with the transmitted light of the illuminated real environment. To solve this problem, Kiyokawa et al. [Kiy00] apply additional LCD panels to selectively block the incoming light with respect to the rendered graphics.

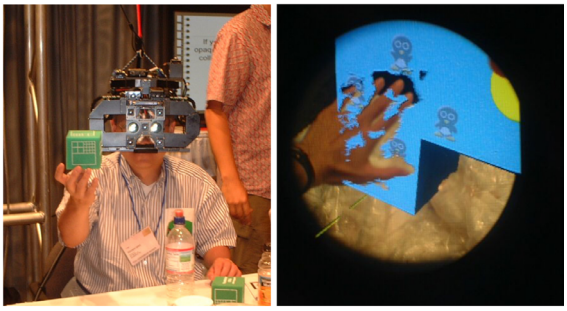


Figure 2.4: *Osaka University's ELMO – An optical see-through head-mounted display that provides mutual occlusion by using a see-through LCD panel in front of the HMD optics [Kiy00]. Courtesy: Kiyokawa.*

Head-mounted displays are currently the dominant display technology within the AR field. They support mobile applications and multi-user applications, if a large number of users need to be supported.

Some variations of head-mounted displays exist that are more attached to the real environment than to the user. Optical see-through boom-like displays (e.g., Osaka University's ELMO [Kiy00]) or video see-through, application-adopted devices (e.g., the head-mounted operating microscope [Fig02]) represent only two examples.

2.1.3. Head-Mounted Projectors

Head-mounted projective displays [Par98, Ina00, Hua01] redirect the projection frustum with a mirror beam-splitter so that the images are beamed onto *retro-reflective surfaces* that are located in front of the viewer. A retro-reflective surface is covered with many thousands of micro corner cubes. Since each micro corner cube has the unique optical property to reflect light back along its incident direction, such surfaces reflect brighter images than normal surfaces that diffuse light. Note that this is similar in spirit to the holographic films used for transparent projection screens. However, these films are back-projected while retro-reflective surfaces are front-projected.

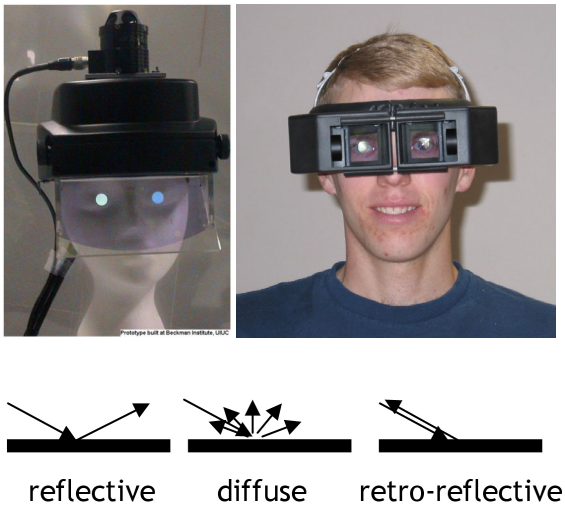


Figure 2.5: *Examples of head-mounted projector prototypes [Hua01] (top). Courtesy: Hua, Gao, Brown, Ahuja, and Rolland. Reflection-properties of retro-reflective material (bottom).*

Projective head-mounted displays [Kij97] beam the generated images onto regular ceilings, rather than onto special surfaces that face the viewer. Two half-silvered mirrors are used to integrate the projected image into the viewer's visual field so that the projectors' parameters match the viewer's parameters (i.e., projection/viewing center and frustum).

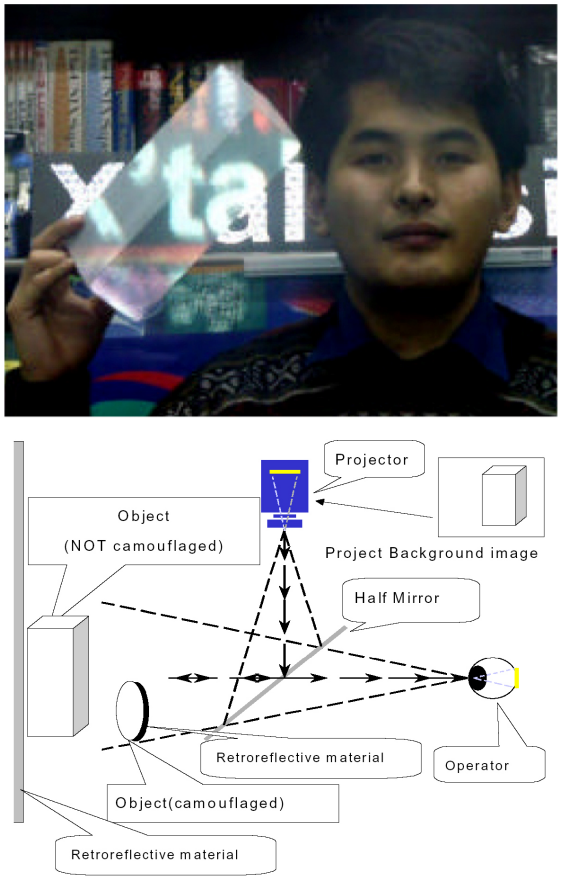


Figure 2.6: *Example of how HMPDs are used to make things transparent – Optical Camouflage [Ina01]. Courtesy: Inami, Kawakami, Sekiguchi, Yanagida, Maeda, and Tachi.*

Head-mounted projective displays decrease the effect of inconsistency of accommodation and convergence that is related to HMDs. Both, head-mounted projective displays and projective head-mounted displays also address other problems that are related to HMDs: They provide a larger field of view without the application of additional lenses that introduce distorting arbitrations. They also prevent incorrect parallax distortions caused by IPD (inter-pupil distance) mismatch that occurs if HMDs are worn incorrectly (e.g., if they slip slightly from their designed position). However, they also introduce several shortcomings:

- Both, head-mounted projective displays and projective head-mounted displays are currently cumbersome. However, new prototypes tend to be smaller and more ergonomically to wear;
- The integrated miniature projectors/LCDs offer limited resolution and brightness;
- Head-mounted projective displays might require special display surfaces (i.e., retro-reflective surfaces) to provide bright images;
- For projective head-mounted displays, the brightness of the images depends on the environmental light conditions;
- Projective head-mounted displays can only be used indoors, since they require the presence of a ceiling.

Although such displays technically tend to combine the advantages of projection displays with the advantages of traditional HMDs, their cumbersomeness currently prevents them from being applicable outside research laboratories. Like head-attached displays in general, they suffer from the imbalanced ratio between heavy optics (or projectors) that results in cumbersome and uncomfortable devices or ergonomic devices with a relatively poor image quality.

2.2. Hand-Held Displays

Conventional examples of *hand-held displays*, such as Tablet PCs, personal digital assistants (PDAs) [Fru01, Gei01, Gau03, Pas03, Wag03], or –more recently– cell-phones [Moe04] generate images at arm reach. All of these examples combine processor, memory, display, and interaction technology in one single device, and aim at supporting a wireless and unconstrained mobile handling. Video see-through is the preferred concept for such approaches. Integrated video cameras capture live video streams of the environment that are overlaid by graphical augmentations before displaying them.

Figure 2.7: *AR application running on a PocketPC [Wag03] (top, courtesy: TU-Vienna), and a first prototype on a conventional consumer cell-phone [Moe04] (bottom).*

However, optical see-through hand-held devices exist. Stetton et al [Ste01], for instance, has introduced device for overlaying real-time tomographic data over the patient. It consists of an ultrasound transducer that scans ultrasound slices of objects in front of it. The slices are displayed time-sequentially on a small flat-panel monitor and are then reflected by a planar half-silvered mirror in such a way that the virtual image is exactly aligned with the scanned slice area. Stereoscopic rendering is not required in this case, since the visualized data is two-dimensional and appears at its correct three-dimensional location.

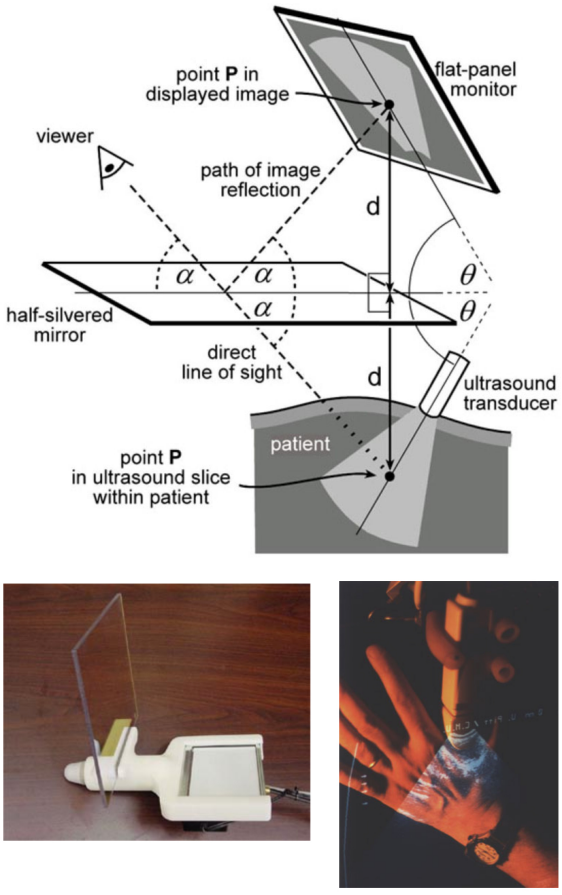


Figure 2.8: *Example of a hand-held mirror display - The Sonic Flashlight [Ste01]. Courtesy: Stetten, Chib, Hildebrand, and Bursee.*

Hand-held mirror beam splitters can be used in combination with large, semi-immersive or immersive screens to support augmented reality tasks with rear-projection systems [Bim00]. Tracked mirror beam-splitters act as optical combiners that merge the reflected graphics, which are displayed on the projection plane, with the transmitted image of the real environment.

Figure 2.9: *The Transflective Pad [Bim00] – a hand-held mirror beam-splitter in combination with a large rear-projection screen.*

Yet another interesting display concept was described in [Ras03] and proposes the application of a hand-held video projector as a real flashlight to interactively generate shadow effects of virtual objects on real surfaces. A combination of a hand-held video projector and a camera was also used by Foxlin and Naimark of Intersense to demonstrate the capabilities of their optical tracking system. This concept might represent an interesting application of AR to the fields of architecture and maintenance.

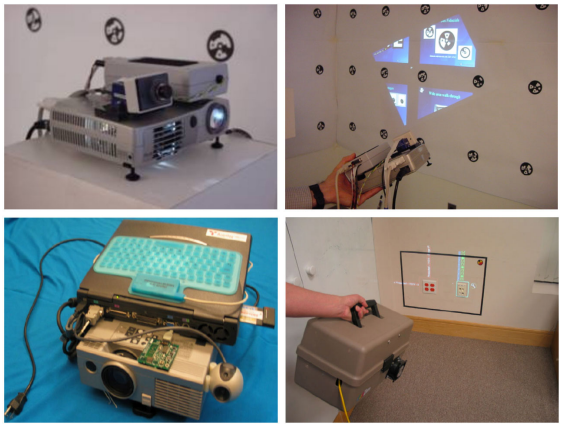


Figure 2.10: *AR Flashlight (top): augmenting the world with a tracked handheld projector. Courtesy: InterSense Inc., Foxlin and Naimark. Context aware iLamp (bottom): augmenting of an identified surface [Ras03].*

The following disadvantages can be related to the individual approaches:

- The image analysis and rendering components is processor and memory intensive. This is critical for low-end devices such as PDAs and cell-phones, and might result in a too high end-to-end system delay and low frame rates. Such devices often lack a floating point unit – which makes precise image processing and fast rendering even more difficult;
- The limited screen size of most hand-held devices restricts the covered field-of-view. However, moving the mirror to navigate3 through an information space that is essentially larger than the display device supports a visual perception phenomenon that is known as *Parks effect* [Par65]. That is, moving a scene on a fixed display is not the same as moving a display over a stationary scene because of the persistence of the image on the viewer's retina. Thus, if the display can be moved, the effective size of the virtual display can be larger than its physical size, and a larger image of the scene can be left on the retina. This effect was also pointed out by the early work of Fitzmaurice [Fiz93] on mobile VR devices;
- The optics and image sensor chips of integrated cameras in consumer hand-held devices is targeted to other applications and consequently provide a limited quality for image processing tasks (e.g., usually high barrel distortion). For example, they do not support an auto-focus. Fixed focus cameras can only be effective in a certain depth range. This also applies to the image output of hand-held projectors if they are used to augment surfaces with a certain depth variance;
- Compared to head-attached devices, hand-held devices do not provide a completely hands-free working.

Hand-held devices represent a real alternative to head-attached devices for mobile applications. Especially consumer devices, such as PDAs and cell-phones have the potential to bring AR to a mass market.

As it is the case for PCs, the graphics capabilities of cell-phones are clearly driven by the game industry. Vendors, such as ATI and nVidia, already offer the first 3D graphics acceleration chips for cell-phones. Autostereoscopic displays are available for off-the-shelf phones (e.g., Sharp) for viewing graphics in 3D. The processors of phones are becoming continuously faster and memory restrictions like today will become history.

2.3. Spatial Displays

In contrast to body-attached displays (head-attached or hand-held), *spatial displays* detach most of the technology from the user and integrate it into the environment. Three different approaches exist which mainly differ in the way they augment the environment – either using video see-through, optical see-through or direct augmentation.

2.3.1. Screen-Based Video See-Through Displays

Screen-based augmented reality has sometimes been referred to as window on the world [Fei93]. Such systems make use of video-mixing (video see-through) and display the merged images on a regular monitor.

Figure 2.11: *Example for a screen-based video see-through display.The locomotion of a dinosaur is simulated over a physical foot-print [Bim02b].*

As fish tank virtual reality systems which also apply monitors, window on the world setups provide a low degree of immersion. Within an augmented reality context the degree of immersion into an augmented real environment is frequently expressed by the amount of the observer's visual field (i.e., the field of view) that can be superimposed with graphics. In case of screen-based augmented reality, the field of view is limited and restricted to the monitor size, its spatial alignment relative to the observer, and its distance to the observer.

For screen-based augmented reality, the following disadvantages can be found:

- Small field of view that is due to relatively small monitor sizes. However, the screen-size is scalable if projection displays are applied;
- Limited resolution of the merged images (especially dissatisfying is the limited resolution of the real environment). This is a general disadvantage of video see-through [Rol94];
- Mostly provides a remote viewing, rather than supporting a see-through metaphor;
- Direct interaction with the real environment and the graphical augmentation is usually not supported. Only indirect / remote interaction techniques can be supported;

Screen-based augmentation is a quite common technique if mobile applications or optical see-through do not have to be supported. It represents probably the most cost efficient AR approach, since only off-the-shelf hardware components and standard PC equipment is required.

2.3.2. Spatial Optical See-Through Displays

In contrast to head-attached or hand-held optical see-through displays, *spatial optical see-through displays* generate images that are aligned within the physical environment. Spatial optical combiners, such as planar [Bim01a, Bim01b] or curved [Bim01a, Bim03] mirror beam splitters, transparent screens [Ogi01, Bim04b], or optical holograms [Bim04a] are essential components of such displays. This class of augmented reality displays is described in much more detail in chapter 3.

Figure 2.12: *A monitor-based Virtual Showcase variation [Bim01a] (top) and the Extended Virtual Table [Bim01b] (bottom) using a large beam-splitter and projection screen.*

The following shortcomings are related to spatial optical see-through configurations:

- They do not support mobile applications because of the spatially aligned optics and display technology;
- In most cases, the applied optics prevents a direct manipulative interaction with virtual and real objects that are located behind the optics. Exceptions are reach-in configurations – either realized with see-through LCD panels [Sch02] or mirror-beam splitters [www.arsys-tricorder.de];
- The number of observers that can be supported simultaneously is restricted by the applied optics. Multi-user displays such as the Virtual Showcase [Bim01a] and its variations support four and more users;
- For the same reasons as for optical see-through head mounted displays, a mutual occlusion between real and virtual environment is not supported. Projector-based illumination techniques which solve this problem [Bim02a] are discussed in chapter 4;
- Due to the limited size of screens and optical combiners, virtual objects outside the display area are unnaturally cropped. This effect is known as window violation and is also present for fish-tank and semi-immersive virtual reality displays.

Figure 2.13: *Transparent screen (top, courtesy: LaserMagic) and Ogi's invisible interface (bottom) [Ogi01]. Courtesy: Ogi, Yamada, Yamamoto, and Hirose.*

The general advantages of such displays are an easier eye accommodation and vergence, a higher and scalable resolution, a larger and scalable field of view, improved ergonomic factors, an easier and more stable calibration, and a better controllable environment. Chapter 3 will also discuss these issues in more detail.

Figure 2.14: *Example of a screen-based AR display using a see-through LCD panel – The AR window (upper-left and -right) [Sch02]. Courtesy: ZGDV, Schwald. Example of a mirror-based AR display using optical see-through beam splitters (lower-left and -right). Courtesy: Fraunhofer IMK (www.arsys-tricoder.de).*

2.3.3. Projection-Based Spatial Displays

Projector-based spatial displays apply front-projection to seamlessly project images directly on physical objects' surfaces instead of displaying them on an image plane (or surface) somewhere within the viewer's visual field. Single static [Ras98, Und99, Bim02b] or steerable [Pin01], and multiple [Ras99, Ras01, Ras02] projectors are applied to increase the potential display area. Chapter 4 will discuss this concept in more detail.

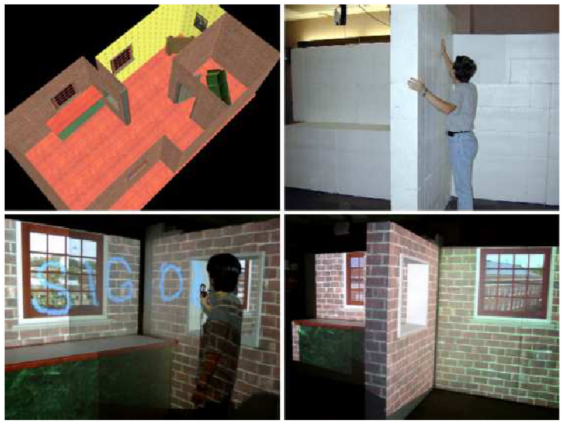


Figure 2.15: *Projector-based augmentation of a large environment [Low01]. Virtual model (upper left). Physical display environment constructed using Styrofoam blocks (upper right). Augmented display (bottom). Note the view dependent nature of the display, the perspectively correct view through the hole in the wall and the windows. Courtesy: Low.*

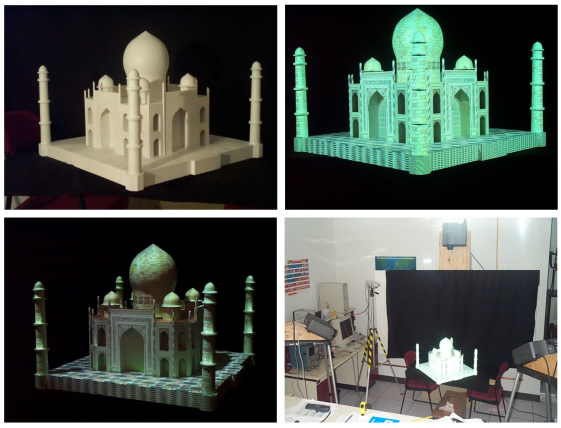


Figure 2.16: *ShaderLamps [Ras01] with TajMahal: The wooden white model (upper left) is illuminated. The scanned geometry of the TajMahal is authored to add texture and material properties. The geometry is registered to the real TajMahal and is displayed from projector's viewpoint.*

A stereoscopic projection and consequently the technology to separate stereo images is not necessarily required if only the surface properties (e.g., its color, illumination or texture) of the real objects are changed by overlaying images. In this case a correct depth perception is still provided by the physical depth of the objects' surfaces.

However, if 3D graphics are displayed in front of the object's surfaces, a view-dependent, stereoscopic projection is required as for other oblique screen displays [Ras98].

On the one hand projector-based spatial displays introduce several new problems:

- Shadow-casting of the physical objects and of interacting users that is due to the utilized front-projection;
- Restrictions of the display area that is constrained to the size, shape, and color of the physical objects' surfaces (for example, no graphics can be displayed beside the objects' surfaces);
- Restricted to a single user in case virtual objects are displayed with non-zero parallax;
- Conventional projectors can only focus on a single focal plane located at a constant distance. Projecting images onto non-planar surfaces causes blur. Exceptions are laser-projectors which do not suffer from this effect;
- The complexity of consistent geometric alignment and color calibration increases with the number of applied projectors.

On the other hand, they overcome some of the shortcomings that are related to head-attached displays: an improved ergonomics, a theoretically unlimited field of view, a scalable resolution, and an easier eye accommodation (because the virtual objects are typically rendered near their real world location).

Figure 2.17: *The Everywhere Display concepts [Pin01]. Courtesy: IBM, Pinhanez.*

References

[Azu97] Azuma, R. T., A Survey of Augmented Reality. Presence: Teleoperators and Virtual Environments, vol. 6, no. 4, pp. 355-385, 1997.

[Bim00] Bimber, O., Encarnação, L.M., and Schmalstieg, D. Augmented Reality with Back-Projection Systems using Transflective Surfaces. Computer Graphics Forum (proceedings of EUROGRAPHICS 2000 - EG'2000), vol. 19, no. 3, pp.161-168, 2000.

[Bim01a] Bimber, O., Fröhlich, B., Schmalstieg, D., and Encarnação, L.M. The Virtual Showcase. IEEE Computer Graphics & Applications, vol. 21, no.6, pp. 48-55, 2001.

[Bim01b] Bimber, O., Encarnação, L.M. and Branco, P. The Extended Virtual Table: An Optical Extension for Table-Like Projection Systems. Presence: Teleoperators and Virtual Environments, vol.10, no. 6, pp. 613-631, 2001.

[Bim02a] Bimber, O. and Fröhlich, B. Occlusion Shadows: Using Projected Light to Generate Realistic Occlusion Effects for View-Dependent Optical See-Through Displays. In proceedings of International Symposium on Mixed and Augmented Reality (ISMAR'02), pp. 186-195, 2002.

[Bim02b] Bimber, O., Gatesy, S.M., Witmer, L.M., Raskar, R. and Encarnação, L.M. Merging Fossil Specimens with Computer-Generated Information. IEEE Computer, September, pp. 45-50, 2002.

[Bim03] Bimber, O., Fröhlich, B., Schmalstieg, D., and Encarnação, L.M. Real-Time View-Dependent Image Warping to correct Non-Linear Distortion for Curved Virtual Showcase Displays. To appear in Computers and Graphics - The international Journal of Systems and Applications in Computer Graphics, vol. 27, no. 4, 2003.

[Bim04a] Bimber, O. Combining Optical Holograms with Interactive Computer Graphics. In IEEE Computer, January issue, pp. 85-91, 2004.

[Bim04b] Bimber, O., Coriand, F., Kleppe, A., Bruns, E., Zollmann, S., and Langlotz, T. Superimposing Pictorial Artwork with Projected Imagery. To appear in IEEE MultiMedia, 2004.

[Fei93] Feiner, S., MacIntyre, B., et al. Windows on the World: 2D Windows for 3D Augmented Reality. In proceedings of ACM Symposium on User Interface Software and Technology, pp. 145-155, 1993.

[Fig02] Figl, M., Birkfellner, W., Ede, C., Hummel, J., Hanel, R. Watzinger, F., Wanschitz, F., Ewers, R., and Bergmann H. The Control Unit for a Head Mounted Operating Microscope used for Augmented Reality Visualization in Computer Aided Surgery. In proceedings of International Symposium on Mixed and Augmented Reality (ISMAR'02), pp. 69-76, 2002.

[Fiz93] Fitzmaurice, G.W. Situated Information Spaces and Spatially Aware Palmtop Computer. CACM, vol. 35(7), pp. 38-49, 1993.

[Fru01] Fründ, J.; Geiger, C.; Grafe, M.; Kleinjohann, B.: The augmented reality personal digital assistant. In proceedings of International Symposium on Mixed Reality, 2001.

[Gau03] Gausemeier, J., Fruend, J., Matysczok, C., Bruederlin, B., and Beier, D. Development of a real time image based object recognition method for mobile AR-devices. In proceedings of International Conference on Computer graphics, Virtual Reality, Visualisation and Interaction in Africa, pp. 133-139, 2003.

[Gei01] Geiger, C., Kleinjohann, B., Reimann, C. Stichling, D. Mobile Ar4All. In proceedings of IEEE and ACM International Symposium on Augmented Reality, pp. 181-182, 2001.

[Hua01] Hua, H., Gao, C., Brown, L., Ahuja, N., and Rolland, J.P. Using a head-mounted projective display in interactive augmented environments. In Proceedings of

IEEE and ACM International Symposium on Augmented Reality 2001, pp. 217-223, 2001.

[Ina00] Inami, M., Kawakami, N., Sekiguchi, D., Yanagida, Y., Maeda, T. and Tachi, S., Visuo-Haptic Display Using Head-Mounted Projector, Proceedings of IEEE Virtual Reality 2000, pp.233-240, 2000.

[Kij97] Kijima, R. and Ojika, T. Transition between virtual environment and workstation environment with projective head-mounted display, In proceedings of IEEE Virtual Reality Annual International Symposium, pp.130-137, 1997.

[Kiy00] Kiyokawa, K., Kurata, Y. and Ohno, H. An Optical See-through Display for Mutual Occlusion of Real and Virtual Environments. In proceedings of IEEE & ACM ISAR 2000, pp. 60-67, 2000.

[Kol93] Kollin, J. A Retinal Display For Virtual-Environment Applications. In proceedings of SID International Symposium, Digest Of Technical Papers, pp. 827, 1993.

[Lew04] Lewis, J.R. In the eye of the beholder. IEEE Spectrum, May issue, pp. 16-20, 2004

[Low01] Low, K., Welch, G., Lastra, A., and Fuchs, H. Life-Sized Projector-Based Dioramas. Symposium on Virtual Reality Software and Technology, 2001.

[Moe04] Moehring, M., Lessig, C., and Bimber, O. Video see-through AR on consumer cell-phones. Submitted to IEEE/ACM ISMAR'04, 2004.

[Ogi01] Ogi, T., Yamada, T., Yamamoto, K. and Hirose, M. Invisible Interface for Immersive Virtual World. In proceedings of the Immersive Projection Technology Workshop (IPT'01), pp. 237-246, Stuttgart, Germany, 2001.

[Par98] Parsons, J., and Rolland, J.P., A non-intrusive display technique for providing real-time data within a surgeons critical area of interest, In proceedings of Medicine Meets Virtual Reality'98, pp. 246-251, 1998.

[Par65] Parks, T.E. Post Retinal Visual Storage. American Journal of Psychology, vol. 78, pp. 145-147, 1965.

[Pas03] Pasman, W. and Woodward, C. Implementation of an augmented reality system on a PDA. In proceedings of IEEE and ACM International Symposium on Mixed and Augmented Reality, pp. 276-277, 2003.

[Pat00] Patrick, E., Cosgrove, D., Slavkovic, A., Rode, J.A., Verratti, T., and Chiselko, G. Using a Large Projection Screen as an Alternative to Head-Mounted Displays for Virtual Environments. In proceedings of CHI' 2000, vol. 2, no. 1, pp. 479-485, 2000.

[Pin01] Pinhanez, C. The everywhere displays projector: A device to create ubiquitous graphical interfaces, In proceedings of Ubiquitous Computing, pp. 315-331, 2001.

[Pry98] Pryor, Homer L., Furness, Thomas A. and Viirre, E. The Virtual Retinal Display: A New Display Technology Using Scanned Laser Light. In proceedings of Human Factors and Ergonomics Society, 42nd Annual Meeting, pp. 1570-1574, 1998.

[Ras98] Raskar, R., Welch, G., and Fuchs, H. Spatially Augmented Reality. In proceedings of First IEEE Workshop on Augmented Reality (IWAR'98), San Francisco, CA, pp. 63-72, 1998.

[Ras99] Raskar, R., Welch, G., and Chen, W-C. Table-Top Spatially Augmented Reality: Bringing Physical Models to Life with Projected Imagery. In proceedings of Second International IEEE Workshop on Augmented Reality (IWAR'99), San Francisco, CA, pp. 64-71, 1999.

[Ras01] Raskar, R., Welch, G., Low, K.L. and Bandyopadhyay, D. Shader Lamps: Animating real objects with image-based illumination, In proceedings of Eurographics Rendering Workshop, pp. 89-102, 2001.

[Ras02] Raskar, R., Ziegler, R. and Willwacher, T. Cartoon Dioramas in Motion, In proceedings of Int. Symp. on Non-photorealistic Animation and Rendering, pp. 7-ff, 2002.

[Ras03] Raskar, R., van Baar, J., Beardsly, P., Willwacher, T., Rao, S. and Forlines, C. iLamps: Geometrically Aware and Self-Configuring Projectors, In proceedings of ACM Siggraph, pp. 809-818, 2003.

[Rol94] Rolland, J., Rich, H. and Fuchs, H. A Comparison of Optical and Video See-Through Head-Mounted Displays. In proceedings of SPIE: Telemanipulator and Telepresence Technologies, pp. 293-307, 1994.

[Sch02] Schwald, B., Seibert, H., Weller, T. A Flexible Tracking Concept Applied to Medical Scenarios Using an AR Window. In proceedings of International Symposium on Mixed and Augmented Reality (ISMAR'02), pp. 261-262, 2002.

[Ste01] Stetten, G., Chib, V., Hildebrand, D., and Bursee, J. Real Time Tomographic Reflection: Phantoms for Calibration and Biopsy, In proceedings of IEEE/ACM International Symposium on Augmented Reality (ISMAR'01), pp. 11-19, 2001.

[Und99] Underkoffler, J., Ullmer, B. and Ishii, H. Emancipated pixels: real-world graphics in the luminous room, In proceedings of ACM Siggraph, pp. 385-392, 1999.

[Wag03] Wagner, D., and Schmalstieg, D. First steps towards handheld augmented reality. In proceedings of International Conference on Wearable Computers, pp. 127-136, 2003.

3. Generating Graphical Overlays with Spatial Optical See-Through Displays

Spatial optical see-through displays overlay the real environment with computer graphics in such a way that the graphical images and the image of the real environment are visible at the same time. In contrast to head-attached or body attached optical see-through displays, spatial displays generate images that are aligned within the physical environment. They do not follow the users' movements but rather support moving around them. Consequently they are comparable with spatial projection displays – but do not share the opaque characteristic of such displays.

An essential component of optical see-through display is the *optical combiner* – an optical element that mixes the light emitted by the illuminated real environment with the light produced with an *image source* that displays the rendered graphics. Creating graphical overlays with spatial optical see-through displays is similar to rendering images for spatial projection screens for some optical combiners. For others, however, it is more complex and requires additional steps before the rendered graphics is displayed and optically combined.

While monitors, diffuse projection screens, or video projectors usually serve as light emitting image sources, two different types of optical combiners are normally being used for such displays: *transparent screens* and *half-silvered mirror beam combiners*. Rendering techniques that support creating correct graphical overlays with both types of optical combiners and with different images sources will be discussed below. Before going into details, we make the convention that all following elements, such as optical combiners, image sources, observers, virtual and real environments, etc. are defined within the same *Cartesian coordinate system*. This keeps the explanations particularly simple and allows a straight forward implementation of the techniques.

3.1. Transparent Screens

Transparent screens have two main properties: they are transparent up to a certain degree to transmit the image of the real environment, and they emit the light of the rendered graphics. Observers see directly through the screen (and through the image displayed on it) at the real environment.

In some cases, such screens are *active* and contain light emitting elements. Examples are liquid crystal displays that are modified (by removing the opaque back light source) to enable their see-through capabilities. In other cases external image sources, such as video or laser projectors

are applied to generate light that is diffused by a *transparent projection screen*. Examples include *holographic projection screens* that diffuse the light in a narrow angle to achieve an enhanced brightness for restricted viewing angles, or *transparent film screens* that diffuse the light in a wide angle to support a more flexible viewing and a higher degree of transparency.

On the one hand, the use of video projectors allows building large screens that do not necessarily have to be planar. Active screens, on the other hand, provide more flexibility since they are not constrained to a stationary configuration. In future new materials, such as *light emitting polymers* may allow building active transparent screens that are both – flexible and scalable.

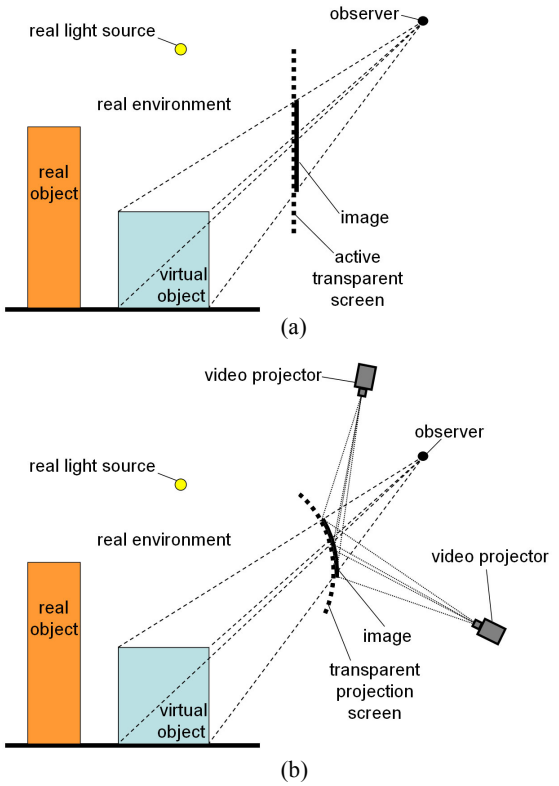


Figure 3.1: *Planar active transparent screen (a) and curved transparent projection screen with multiple projectors (b).*

Rendering for transparent screens is essentially equivalent to rendering for regular projection screens or monitors. While for planar screens an affine *off-axis projection* transformation is sufficient, *curvilinear image warping* is required for curved screens. For large or extremely curved screens the image has to be composed of the contribution displayed by multiple projectors. The different pieces have to be geometrically aligned and blended to result in a consistent final image.

Sections 3.3 through 3.5 describe a rendering framework spatial optical see-through displays that apply mirror beam combiners. Most of this framework, such as refraction correction, multi-screen and multi-plane beam combiner transformations is exactly the same for transparent screens. The main difference to mirror beam combiner is that reflection transformations do not apply in this case. They have to be ignored if the framework is used for displays with transparent screens. In addition, the outlined screen transformation is equivalent to the transformation of video projectors.

An important fact that needs to be mentioned is that the rendered image appears directly on the surface of the transparent screen. This cause *focus problems* if the real environment to be augmented is not located at the same place as the image. For transparent screens, this can never be the case since the screen itself (and therefore the image) can never take up exactly the same space within the environment. It is not possible for our eyes to focus on multiple distances simultaneously. Thus in extreme cases we can only continuously shift focus between the image and the real environment, or perceive either one unfocused.

Attention has to be paid if stereoscopic graphics needs to be displayed on transparent projection screens. Not all materials preserve the polarization of light that is produced by *passive stereoscopic projection* setups – especially when materials are bended to form curved screens. Transparent film screens, for instance, have the property to preserve polarization in any case – also if bended. Active stereo projection systems do not cause such problems, but are more expensive.

3.2. Mirror Beam Combiners

Mirror beam combiners are the most common optical combiners because of their availability and low cost. If not obtained from an optics store, they can be home made in almost any size and shape. For instance, regular float glass or Plexiglas can be coated with a *half-silvered film*, such as 3M's Scotchtint sun protection film. These materials are easily available in well-sorted hardware stores. The advantage of half-silvered film is that it can be coated onto

a flexible carrier, such as thin Plexiglas, that can easily be bended to build curved displays. The drawback of such a material is usually its non-optimal optical properties. Instead of having transmission and reflection factors of 50%, they normally provide factors of approximately 70% reflection and 30% transmission, due to their sun blocking functionality. An alternative is the so-called *spyglass* that offers better and varying transmission/reflection factors without sun blocking layers. However, the reflective film is usually impregnated into float glass – thus it is not well suited for building curved mirror displays.

Spatial optical see-through displays that apply mirror beam combiners as optical combiner have to display the rendered graphics on a *secondary screen* that is reflected by the mirror optics. The observer sees through the optical combiner and through the reflected image at the real environment. The image that appears on the secondary screen should usually not be visible. To hide it the secondary screen is often being coated with *light directing film*, such as 3M's Light Control Film. Such a film can be used to direct the displayed image only towards the mirror optics – and not to the observer. Thus only the reflection of the displayed image can be seen, and not the image itself.

In contrast to transparent screens, mirror beam combiners create an optical reflection of the secondary screen and the displayed image. There are no physical constraints in aligning the image closer to the real environment. The reflected image can even intersect with the real environment. Consequently, the focus problem of transparent screens can be improved – although not completely solved.

3.3. Planar Mirror Beam Combiners

To render a three-dimensional graphical scene on spatial optical see-through displays applying planar mirror beam combiners requires neutralizing the optical transformations that are caused by half-silvered mirrors: reflection and refraction. The goal is to render the graphical content and display it on the secondary screen in such a way that its reflection appears perspectively correct and aligned with the real environment.

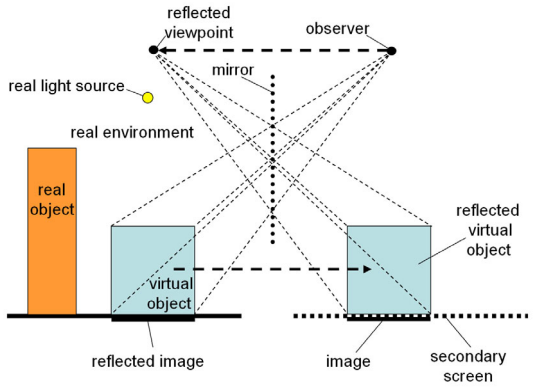
3.3.1. Reflection

A planar mirror beam combiner divides the environment into two subspaces: the one that contains the observer and the secondary screen, and the one that contains the real environment to be augmented and physical light sources that illuminate it.

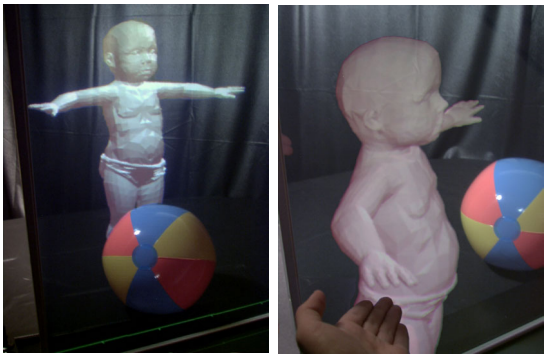
Note that from a geometric optics point of view, the real environment behind the mirror equals the *mirror's image space* (i.e., the reflection that appears behind the mirror by looking at it).

Virtual objects that consist of graphical elements (such as geometry, normal vectors, textures, clipping planes, virtual light sources, etc.) are defined within same global world coordinate system in which the real environment, the observer, the secondary screen and the mirror beam combiner are located. In contrast to conventional Virtual Reality scenarios, this coordinate system actually exceeds the boundaries of the display screen and extends into the surrounding real environment.

The virtual objects are either defined directly within the real environment, or they are transformed to it during an *object registration process*.



(a)



(b) (c)

Figure 3.2: *Affine reflection transformation for planar mirror beam combiner . A virtual baby observed through a large beam combiner reflecting a horizontal projection screen (b). Due to parallax effects virtual objects (e.g., the baby's arm) can appear in front of the mirror (c).*

We now consider the mirror and compute the reflection of the observer's physical eye locations (as well as possible virtual headlights). We then apply the inverse reflection to every graphical element that is located within the real environment. In this manner these graphical elements are transformed and can be projected and displayed on the secondary screen. The displayed image is optically reflected back by the mirror into the real environment (or optically: into the mirror's image space).

When the setup is sufficiently calibrated, the real environment and the mirror's image space overlay exactly. The virtual objects appear in the same position within the image space as they would within the real environment without the mirror (if a direct display possibility was given within the real environment).

Note that the *reflection transformation* of planar mirrors is a *rigid-body transformation*, and preserves all properties of the transformed geometry.

With known plane parameters of the mirror beam combiner within the world coordinate system, a point in 3D space can be reflected with:

$$p' = p - 2(np + d)n$$

Where p' is the reflection of p over the mirror plane $[n,d] = [a,b,c,d]$.

This is equivalent to multiplying p with the 4x4 reflection matrix:

$$R = \begin{bmatrix} 1-2a^2 & -2ab & -2ac & -2ad \\ -2ab & 1-2b^2 & -2bc & -2bd \\ -2ac & -2bc & 1-2c^2 & -2cd \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Note that $R = R^{-1}$.

The reflected viewpoint e' of the observer (which, for instance, is head-tracked) can be computed with the equation above or by multiplying the original viewpoint e with the reflection matrix.

The inverse reflection of the virtual scene that is located within the real environment is simply computed from its reflection with respect to the mirror plane. Since we assume that the real environment and the mirror's image space exactly overlay, we can also assume that the reflection of the graphical elements located within the real environment results in the inverse reflection of the image space, that is, they are transformed to their corresponding positions on the observer's side of the mirror and can be displayed on the secondary screen. Consequently, the additional model transformation (i.e., the inverse reflection of the scene) is achieved by multiplying the reflection matrix onto the current model-view matrix of the *transformation pipeline* (between scene transformation and view transformation).

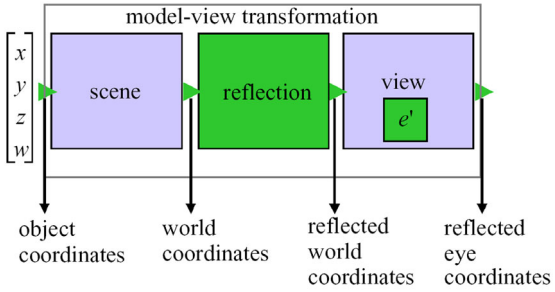


Figure 3.3: *The integration of reflection transformations into the rendering pipeline.*

Consequently, geometry that is registered to the real environment is first transformed from object coordinates into the coordinates of the world coordinate system, then into *reflected world coordinates*, and finally into *reflected eye coordinates*. After the model-view transformation has been applied, the rendering pipeline is continued in the normal way: the reflected eye coordinates are off-axis projected into clip coordinates, then –after the perspective division– transformed into normalized device coordinates, and finally (via the viewport transformation) mapped into window coordinates.

By applying the reflection matrix, every graphical element is reflected with respect to the mirror plane. A side effect of this is that the order of reflected polygons is also reversed (e.g., from counterclockwise to clockwise) which, due to the wrong front-face determination, results in a wrong rendering (e.g., lighting, culling, etc.). This can easily be solved by explicitly reversing the *polygon order*. Note that transformations and rendering have to be done for both viewpoints (left and right) if stereoscopic rendering is activated.

The reflection transformation can be entirely executed by accelerated graphics hardware that provides a *fixed function rendering pipeline*. The following OpenGL code fragment can be used to configure the rendering pipeline with reflection transformation:

```
...
// mirror plane = a,b,c,d
// viewpoint = e[0..2]
// world coordinate system = x/y-axes
// horizontal plane,
// z-axis points up
```

```
float e[3], e_[3];
float NP;
float R[16];

// compute reflected viewpoint e_ from
// original viewpoint e
NP=a*e[0]+b*e[1]+c*e[2];
e_[0]= e[0]-2.0*(NP+d)*a;
e_[1]= e[1]-2.0*(NP+d)*b;
e_[2]= e[2]-2.0*(NP+d)*c;

//set up reflection matrix
R[0]=1-2*a*a;       R[1]=-2*a*b;
R[2]=-2*a*c;        R[3]=0;
R[4]=-2*a*b;        R[5]=1-2*b*b;
R[6]=-2*b*c;        R[7]=0;
R[8]=-2*a*c;        R[9]=-2*b*c;
R[10]=1-2*c*c;      R[11]=0;
R[12]=-2*a*d;       R[13]=-2*b*d;
R[14]=-2*c*d;       R[15]=1;

//configure rendering pipeline
glViewport(…);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glFrustum(…);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(e_[0],e_[1],e_[2],e_[0],e_[1],0
,0,1,0);
glMultMatrixf(R);

//reverse polygon order
glFrontFace(GL_CW+GL_CCW-
glGetIntegerv(GL_FRONT_FACE));

//draw scene with scene transformation
…
```

An alternative to a reflection of scene geometry and viewpoint is to set up a viewing frustum that is defined by the reflected image plane instead of the image plane on the physical secondary screen. In figure 3.2a the secondary screen is reflected from the right sides of the beam combiner to its left side. The unreflected viewing frustum (right side) can be used if defined relative to the reflected image plane (left side). For this, the exact position and orientation of the reflected image plane has to be known. They can also be derived from the parameters of the mirror plane.

In addition, the displayed image has to be reflected over the on-screen axis that is perpendicular to the intersection vector (i_x, i_y) of the mirror plane and the secondary screen. In a simple example, this intersection equals the X- axis (or the Y-axis) of the screen coordinate system. In this case an additional scaling transformation can be applied after the projection transformation that causes the fragments to be reflected within the normalized device space. For instance, glScale($i_x, i_y, 0$) with $i_x = -1, i_y = 0$ causes a reflection over the Y-axis for the case that the intersection of the mirror with the screen is on the X-axis (all in normalized screen/device coordinates). For an arbitrary intersection vector, however, a scaling transformation alone is not sufficient. An additional rotation transformation is required that first aligns the intersection vector with either the X- or the Y-axis in the screen coordinate system. Then the scaling is transformed along this principle axis (for example over the X-axis as explained above). Finally the reverse rotation transformation has to be applied to produce the yield the correct effect.

It is important to apply this reflection transformation after the projection (e.g., before glFrustum() in OpenGL's reversed matrix stack notation) since it has to be the final transformation, and it must not influence other computations, such as lighting and depth culling.

An interesting optical effect can be observed by applying mirrors in combination with stereoscopic secondary screens: Convex or planar mirrors can optically only generate virtual images. However, in combination with a stereoscopic graphics and the effects caused by stereopsis, virtual objects can appear in front of the mirror optics (cf. figure 3.2c). We can refer to this effect as *pseudo real images*. In nature, real images of reflected real objects can only be generated with concave mirrors. Note that a restricted direct manipulative interaction with pseudo real images in front of the mirror optics is supported.

3.3.2. Refraction

From an optics point of view, the glass or Plexiglas carriers used for optical combiners (i.e., mirror beam combiner or transparent screens) are lenses that cause *refraction* distortion. A homogeneous medium that is bound by two plane-parallel panels is referred to as *planar lens*. The refraction distortion is small and can be neglected for thin planar lenses, but has to be corrected for thick plates. Note that the techniques described below are also applicable for transparent screens that suffer also from refraction distortion.

The following problem occurs: All virtual objects that are registered to the real environment are virtual points that are not physically located behind the optical combiner. They are images that are created by the optical combiner. Mirror beam combiners are usually *front surface mirrors* (i.e., the mirror film coated on the side of the carrier that

faces the image source and the observer) while transparent screens can be front projected – causing the same registration problem: the displayed image is not physically refracted by the optical combiner. However, the transmitted light which is emitted by the real environment and perceived by the observer is refracted. Consequently, the transmitted image of the real environment cannot be precisely registered to the reflected virtual objects, even if their geometry and alignment match exactly within our world coordinate system.

Unfortunately refraction cannot be undistorted by a rigid-body transformation, but approximations exist that are sufficient for augmented reality display types.

All optical systems that use any kind of see-through element have to deal with similar problems. For head-mounted displays, *aberrations* (optical distortion) caused by refraction of the integrated lenses are mostly assumed to be static [Azu97]. Due to the lack of eye-tracking technology as component of head-mounted displays, the rotation of the eye-balls and the exact position as well as the movement of the optics in front of the observer's eyes is not taken into account. Thus a static refraction distortion is pre-computed for a *centered on-axis* optical constellation with methods of *paraxial analysis*. The result is stored in a *two-dimensional look-up-table*. During rendering, this look-up-table is referenced to transform rendered vertices on the image plane before they are displayed. Rolland and Hopkins [Rol93], for instance, describe a polygon-warping technique that uses the lookup table to map projected vertices of the virtual objects' polygons to their pre-distorted location on the image plane. This approach requires subdividing polygons that cover large areas on the image plane. Instead of pre-distorting the polygons of projected virtual objects, the projected image itself can be pre-distorted, as described by Watson and Hodges [Wat95], to achieve a higher rendering performance.

For spatial see-through displays, however, aberrations caused by refraction are dynamic, since the optical distortion changes with a moving viewpoint that is normally off-axis and off-centered with respect to the optical combiner.

For some *near-field displays*, such as reach-in displays, the displacement caused by refraction can be estimated [Wei99]. An estimation of a constant refraction might be sufficient for near-field displays with a fixed viewpoint that apply relatively thin beam combiner. For larger spatial optical see-through setups that consider a head-tracked observer and apply a relatively thick beam combiner require, however, require more precise methods. In the following explanation, we want to consider such a display constellation:

Since we cannot pre-distort the refracted transmitted image of the real environment, we artificially refract the virtual scene before it is displayed, in order to make both images match. In contrast the static transformation that is assumed for head-mounted displays, refraction is dynamic for spatial optical see-through displays and cannot be pre-computed. In general, refraction is a complex curvilinear transformation and does not yield a *stigmatic mapping* (the intersection of multiple light rays in a single image point) in any case, we can only approximate it.

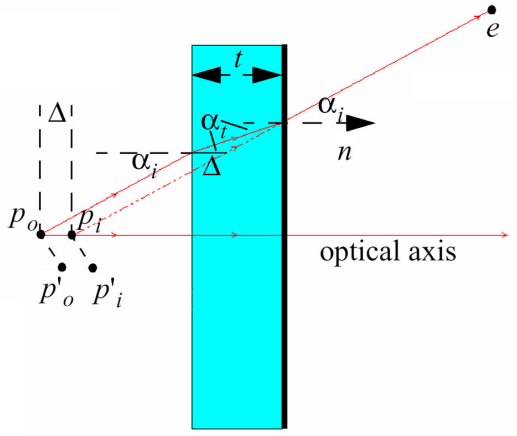


Figure 3.4: *Off-axis refraction transformation for planar lenses.*

In case of planar lenses, light rays are refracted twice – at their entrance points and at their exit points. This is referred to as in-out refraction. In case of planar lenses, the resulting out-refracted light rays have the same direction as the corresponding original rays, but they are shifted by the amount Δ parallel to their original counterparts. Due to refraction, an object that is physically located at the position p_o appears at the position p_i. To ´maintain registration between a virtual object (that will appear at position p_o) and the corresponding real object (that will appear at position p_i), the virtual object has to be re-positioned to p_i.

The offset Δ can be computed with:

$$\Delta = t\left(1 - \frac{\tan \alpha_t}{\tan \alpha_i}\right)$$

where t is the thickness of the planar lens, α_i the angle between the plane normal of the plate and the line of sight, and α_t is given by *Snellius' law of refraction*:

$$\eta_1 \sin\alpha_i = \eta_2 \sin\alpha_t$$

It is constrained to the following boundaries:

$$\lim\left(\alpha_i \to \frac{\pi}{2}\right) \Rightarrow \Delta = t \quad \text{and}$$

$$\lim(\alpha_i \to 0) \Rightarrow \Delta = t\left(1 - \frac{\sin\alpha_t}{\sin\alpha_i}\right) = t\left(1 - \frac{1}{\eta_2}\right) = const.$$

A special case of the above transformation is an *on-axis* situation where the incidence angle is perpendicular to the lens and parallel with the optical axis (i.e., $\alpha_i = 0$). In this situation the offset equation can be simplified to $\Delta = t(1 - 1/\eta_2)$.

In case of an optical combiner that is located in air, the refraction index η_1 is equal to one. The refraction index η_2 is the one of the carrier's base material (e.g., glass or Plexiglas).

A simple solution to simulate refraction is the assumption that, against its optical nature, it can be expressed as a rigid-body transformation. This is a common approximation of the 3D computer graphics community to render realistic-looking refraction effects in real time. In beam-tracing [Hec84], for instance, it was assumed that, considering only *paraxial rays* (entering light rays that are exactly or nearly perpendicular to the refracting plane), objects seen through a polygon with a refraction index of η appear to be η times their actual distance. This is because light travels slower in denser materials by precisely this factor. For this approximation, the incidence angles of the optical line of sight were not taken into account but. Instead, a constant incidence angle of $\alpha_i = 0$ to the optical axis was assumed. This, however, covers on-axis situations only. For off-axis situations that occur with spatial optical see-through displays the incident angle has to be considered. As a rigid-body transformation, refraction can be expressed by a homogenous 4x4 matrix:

$$F = \begin{bmatrix} 1 & 0 & 0 & \Delta a \\ 0 & 1 & 0 & \Delta b \\ 0 & 0 & 1 & \Delta c \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The advantage of this simple translation along the plate's optical axis is that it can be integrated into a fixed function rendering pipeline and can be carried out by graphics hardware. The drawback, however, is that this is only a rough approximation for refraction distortion, since every vertex is translated by the same amount Δ which is computed from a common incident angle (such as the angle between the viewing direction and the optical axis, for instance). The curvilinear characteristics of optical refraction and the individual incident angles of the viewing vectors to each scene vertex are not taken into account.

Programmable rendering pipelines allow per-vertex transformations directly on the graphics hardware. Thus the correct *curvilinear refraction transformation* can be implemented as *vertex shader*, rather than a rigid-body transformation expressed by a homogenous matrix.

The following CG shader fragment can be used to configure a programmable rendering pipeline with the refraction transformation:

```
…
//viewpoint = e[0..2]
//vertex = v[0..3]
//plane normal = n[0..2]
//plane thickness = t
//refraction index of material = r

float3 n,d;
float alpha_i, alpha_t, delta;

// compute viewing vector to vertex
d=e-v;
d=normalize(d);
n = normalize(n);

// compute angle between normal and
viewing vector
alpha_i = acos(dot(d,n));

// compute delta
alpha_t = asin(sin(alpha_i)/r);
delta = t*(1-
(tan(alpha_t))/tan(alpha_i));

//compute refracted vertex
v.xyz = v.xyz+n*delta;
…
```

No matter if refraction is implemented as rigid-body transformation or as per-vertex transformation, the sequence of the transformations carried out by the rendering pipeline is important for spatial optical see-through displays.

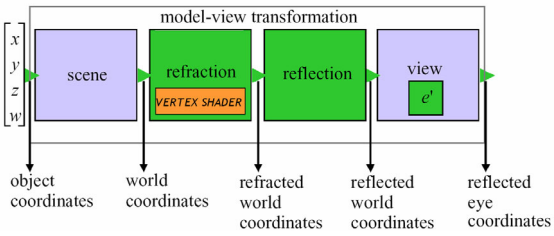


Figure 3.5: *The integration of reflection transformations into the rendering pipeline.*

Figure 3.5 illustrates the extension of figure 3.3. Refraction transformation has to be carried out after the scene transformation (either a rigid-body or as per-vertex transformation). Vertices are transformed from world coordinates to *refracted world coordinates* first. If the optical combiner is a mirror beam combiner, the refracted vertices are then reflected. If the optical combiner is a transparent screen, reflection transformation is not applied. Finally, the vertices are mapped into reflected eye coordinates (either with the reflected or with the un-reflected viewpoint – depending on the optical combiner), projected, converted into window coordinates, rastered and displayed.

The pseudo code below summarizes the rendering steps for spatial optical see-through displays that apply one planar screen and one planar beam combiner. Only one user is supported.

```
for left and right viewpoints i
    initialize transformation pipeline
    and polygon order
    compute reflected viewpoint e_i' = R e_i
    compute refraction offset Δ_i for i
    set transformation pipeline: RMF_i V_i P
    reverse polygon order
    render scene from e_i'
endfor
```

First, the polygon order and the transformation pipeline have to be set to an initial state. Then the reflected viewpoint and the view-dependent refraction offset (Δ_i) are computed. The transformation pipeline is then set to the corresponding concatenation of transformation matrices: reflection transformation (R), model transformation (M), refraction transformation (F_i, with e_i), view transformation (V_i, with e_i') and the projection transformation (P). Finally, the polygon order is reversed and the scene is rendered. Note that we assume that the screen coordinate system is equivalent to the world coordinate system. Note also that R might not be static but has to be re-computed continuously (e.g., if moving components have to be supported – see section 3.5).

3.4. Screen Transformation and Curved Screens

If the coordinate system of the secondary screen does not match with the world-coordinate system, an additional *screen transformation* has to be added to the rendering pipeline. It expresses the screen's transformation within the world coordinate system and is usually a composition of multiple translation and rotation transformations. Similar to the reflection transformation, not only the reflected geometry but also the reflected viewpoint has to be mapped into the *screen coordinate system*.

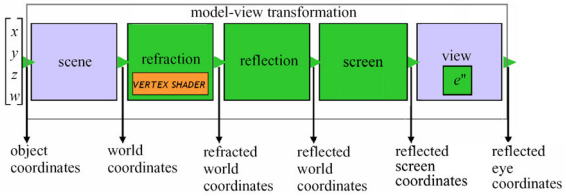


Figure 3.6: *The integration of screen transformations into the rendering pipeline.*

Important is once again the sequence in which all transformations are carried out. The screen transformation is applied after the reflection transformation and before the view transformation – mapping reflected world coordinates into *reflect screen coordinates*. The view transformation has to be performed with the reflected viewpoint in screen coordinates e'', that can be computed by simply applying the screen transformation to the reflected viewpoint e'.

To give an example, consider a 30cm x 40cm screen that is not aligned with the world coordinate system. In particular assume it is rotated by 180 degrees around the z-axis and translated in such a way that the screen's -y edge is aligned with the edge of a mirror beam combiner, located 20cm away from the world origin. In addition, the screen is tilted by 30 degrees about the edge that faces the mirror to achieve a higher contrast and a better position of the reflected image.

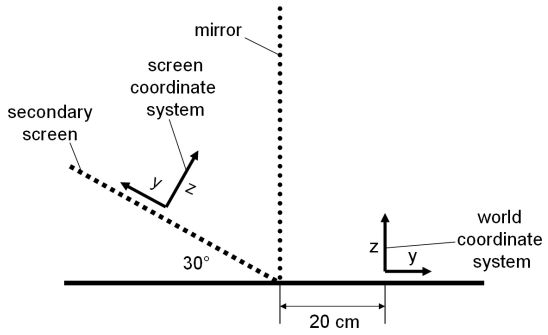


Figure 3.7: *Example of a screen transformation.*

The corresponding OpenGL screen transformation matrix can be calculated as follows:

```
…
glRotate(180,0,0,1);        //rotate
around z-axis by 180 deg
glTranlatef(0,15,0);        //translate
screen edge to origin
glRotatef(30,1,0,0);        //rotate
around screen edge by 30 deg
glTranslatef(0,20,0);       //translate
screen edge to mirror edge
…
```

Remember that the OpenGL notation has to be read from bottom-up (but it has to be implemented this way). To understand the transformation, it is helpful to think that the misaligned screen coordinate system has to be transformed back into the world coordinate system – since this corresponds to the transformation that needs to be applied to the rendered geometry: Reading the OpenGL fraction bottom-up, the screen is first translated by 20cm along the positive y-axis of the world coordinate system – aligning the adjacent edge with the world's origin. Then it is rotated around this edge by 30 degrees – fitting it to the x/y plane of the world coordinate system. Then the screen's origin position is matched with the world's origin position by translating the screen further up the positive y-axis (by exactly 15cm – half of the screen height). Finally, the screen rotated by 180 degrees to fully align both coordinate systems. A read-back from OpenGL's model-view stack (e.g., with glGetFloatv(GL_MODELVIEW_MATRIX,S);) allows to obtain this matrix and multiply it to the reflected viewpoint before the view transformation is added.

Projective texture-mapping [Seg92] can be applied in combination with *two-pass rendering* to support single or multiple front/back projections onto a multi-plane or curved display surface.

Projective textures utilize a *perspective texture matrix* to map projection-surface vertices into texture coordinates of those pixels that project onto these vertices. A first rendering pass generates an image of the virtual scene that will look perspectively correct to the user. During the second pass, this image is projected out from the user's current point of view onto a registered virtual model of the display surface -using projective texture-mapping. Finally, the textured model is rendered from the projector's point of view and the result is beamed onto the real display surface. If multiple projectors are used, the second pass have to be repeated for each projector individually. The generated images have to be geometrically aligned and color and edge blended appropriately to realize a seamless transition between them.

To support planar mirror beam combiners that require affine model and view transformations in combination with curved secondary screens, the above outlined method can be slightly modified: Instead of rendering the original scene from the observer's actual viewpoint, the reflected scene has to be rendered from the reflected viewpoint. The reflection transformations that are applied to the scene and the viewpoint depend on the optics. All other rendering passes remain unchanged. Note that if multiple planar display surfaces are used, and if one projector is assigned to one projection plane, projective textures and two-pass rendering are unnessecary. Instead, regular multi-pipeline rendering can be applied (as it is done for surround-screen projection systems, such as CAVEs or multi-sided workbenches).

3.5. Moving Components

Some spatial see-through displays do not require a static constellation of screens and optical combiners. They rather allow moving them freely within the world coordinate system during run time. With respect to rendering, it makes no difference whether the optical combiner and the screen is fixed or movable. The only modification to the transformations that are described above is that the plane parameters of the optical combiner (which influences reflection and refraction transformations), and/or the parameters of the screen transformation change continuously. This means that the rendering pipeline has to be updated every time a component has been moved – every frame in the worst case.

3.6. Multi-Plane Beam Combiners

More that one planar beam combiner can be utilized to build displays that provide views on an augmented environment from multiple –very different– angles, or to support multiple users simultaneously. This applies for mirror beam combiners as well as for transparent screens. If multiple planar beam combiners need to be supported by an optical see-through display simultaneously, the rendering pipeline that drives this display can be configured from the basic elements that have been discussed above. Convex mirror assemblies unequivocally tessellate the surrounding space into mirror individual reflection zones which –for the same point of view– do not intersect or overlap and consequently provide a definite one-to-one mapping between screen space and reflection space. The constellation of basic transformations for these displays depends on the physical constellation of optical combiners (e.g., mirror beam combiners or transparent screens) and image sources (e.g., video projectors or secondary screens). Several scenarios are discussed below.

3.6.1. Single Viewer

For the following explanation, we want to assume a *multi-plane beam combiner* that provides different perspectives onto the same scene to a single observer. We also want to assume that mirror beam combiners are applied as optical combiners, and a single secondary screen is used to display the rendered images. For example, four half-silvered mirrors can be assembled in form of an up-side-down pyramid frustum that reflects a large, horizontal projection screen. The real environment that needs to be augmented would be located inside the pyramid frustum.

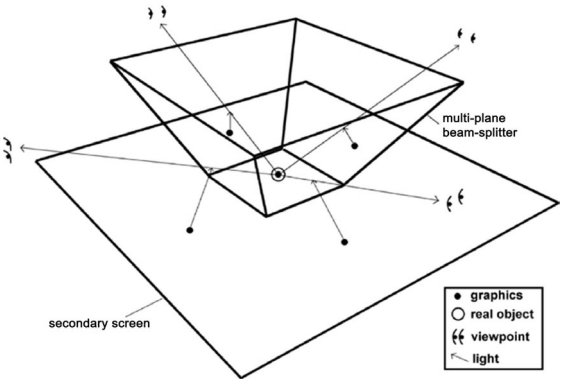


Figure 3.8: *Example for mulit-plane beam combiner constellation: four mirror beam combiners and one secondary screen that support observing the augmented real environment from 360 degrees.*

The step from single-plane to multi-plane optical see-through displays is similar to the step from single-plane projection displays (like walls or workbenches) to multi-plane projection displays (like caves or two-sided workbenches).

As outlined below, the rendering pipeline is split into several (four in our example) *sub-pipelines*. Each sub-pipeline is responsible for rendering a single image onto the secondary screen. Since only one screen is used in this example, four separate images have to be rendered into the same frame buffer that is displayed on the screen on each frame.

The images differ in their mirror-individual transformations (reflection and –optionally- refraction). The scene and the view transformation is the same for each sub-pipeline – except that individual reflections of the same viewpoint have to be applied together with the view transformation.

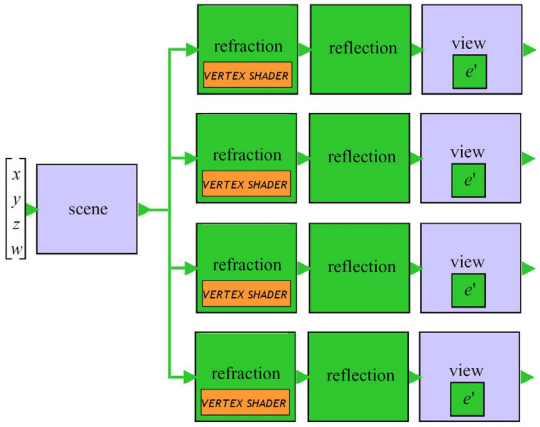


Figure 3.9: *Dependent rendering pipelines that support a multi-plane beam combiner in combination with a single screen and a single observer.*

Note that the number of sub-pipelines is doubled for stereoscopic viewing. In this case, the left and right stereo images may be rendered into two different frame buffers – the back-left and the back-right buffer if the graphics hardware provides a quad-buffer.

If the display is calibrated precisely enough (i.e., the parameters of the pipeline –such as mirror-plane

parameters, etc.– have been determined correctly), the different images merge into a single, consistent reflection from all points of view. The perpendicular mirror constellation that has been chosen in our example ensures that the different images never overlap in the frame buffer. Only two images and the corresponding two mirrors are visible from a single point of view at a time. The reflections of these images result in a single graphical scene.

Figure 3.10: *To images are reflected by two mirrors and merge into a single, consistent graphical scene.*

The sub-pipelines can be carried out sequentially on the same computer (i.e., by rendering each image into the frame buffer before swapping it). This divides the frame rate by a factor that equals the number of sub-pipelines.

Alternatively, hardware *image composition technology* (such as the Lightning-2 device [Stol01]) can be applied to distribute the sub-pipelines to multiple computers and merge the resulting images within a single frame buffer before the outcome is displayed on the screen. In this case multiple rendering nodes and the displays can be connected to a specific display subsystem. The subsystem allows the image data generated by the connected nodes to be mapped to any location of the connected displays without losing much time for transmission and composition.

To support configurations that apply multiple planar beam combiners and a single screen the following algorithm can be applied for a single user:

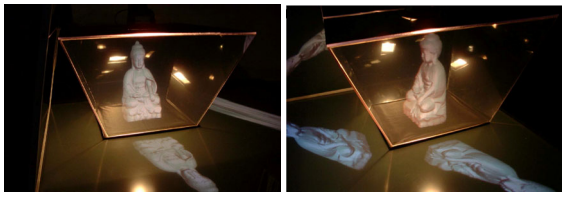
```
for left and right viewpoints i
   for each to i front-facing beam
   combiner j
      initialize transformation pipeline
      and polygon order
      compute reflected viewpoint
      e_i' = R_j  e_i
      compute refraction offset Δ_ij for
      i and j
      set transformation pipeline:
      R_j M F_ij V_i P
      reverse polygon order
      render scene from e_i'
   endfor
endfor
```

The scene has to be rendered for each viewpoint and each beam combiner. If the configuration of the optics unequivocally tessellates the surrounding space into mirror individual reflection zones which do not intersect or overlap, a single object that is displayed within the screen space appears exactly once within the reflection space.

3.6.2. Multiple Viewers

To support *multiple viewers*, each observer can be assigned to an individual beam combiner (and corresponding screen portion and rendering sub-pipeline).

Instead of using the same viewpoint for the view transformation of each sub-pipeline, different viewpoints of the assigned observe have to be applied. This means that the generated images are completely independent of each other and will no longer merge into a single, consistent three-dimensional scene. They rather represent the individual views on the augmented scene of each observer.



(a) (b)

Figure 3.11: *Individual views on the same scene generated for two different observers.*

If we use the above display example, the same scene is transformed and rendered for each observer individually, before all images are displayed on the same screen at the same time. In this case, each observer is restricted to limited viewing area that allows observing the scene through the assigned beam combiner only.

The following algorithm will support such kind of configurations:

```
for left and right viewpoints i of all
viewers
     initialize transformation pipeline
     and polygon order
     compute reflected viewpoint   e_i' = R_i  e_i
     compute refraction offset  Δ_i  for  i
     set transformation pipeline:  R_i M F_i V_i P
     reverse polygon order
     render scene from  e_i'
endfor
```

Note that each user is restricted to the viewing zone that is given by the assigned beam combiner.

3.6.3. Multiple Screens

Using a single screen for multiple beam combiners has the disadvantage that each sub-pipeline renders its image into a portion of the common frame buffer. The resolution of a single image is only a fraction of the frame buffer's total resolution. To overcome this problem, an individual screen can be applied in combination with each beam combiner, sub-pipeline and –optionally– with each viewer.

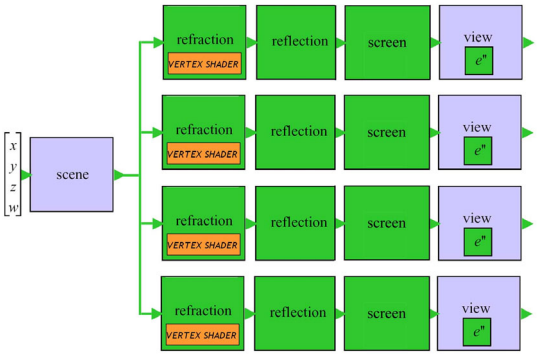


Figure 3.12: *Dependent rendering pipelines that support a multi-plane beam combiner in combination with multiple screens and a single or multiple observer(s).*

This implies that each image is rendered in full resolution into separate frame buffers that are displayed on different screens. Each sub-pipeline has to be extended by a screen transformation that copes with the screens' relative position and orientation within the global world coordinate system. Note, that single and multiple viewer scenarios are supported by this frame work – depending on the viewpoints that are passed to the view transformations.

- In general such a scenario can only be realized with *multi-channel rendering pipelines* that offer multiple frame buffers and the possibility to connect to more than one screen. One possibility is to use a large *graphics workstation* that provides these features. An alternative option that is becoming more and more popular is to apply multiple networked personal computers – so-called *PC clusters*. Nowadays, PC clusters are much more efficient than graphics workstations, both – from a performance and from a financial point of view.
- If PC clusters are used, it is important that the entire rendering process is synchronized correctly. The generated images have to be displayed at exactly the same time. This means that not only scene information have to be updated on all PCs continuously and at an acceptable speed, but also the signal that causes swapping the rendered images from back buffers to front buffers might have to be distributed in addition. This swapping synchronization is called *GenLocking* and is an essential feature for synchronizing multiple active stereoscopic screens. While some newer graphics cards can be connected to distribute the GenLock signal directly, older graphics cards do not provide this feature. However, open source software solutions (called *SoftGenLocking*) exist [All03] that offer an acceptable workaround. The scene information is usually distributed over the network, using distributed scene graph frameworks or other distributed graphics concepts. This ensures that the same scene state is displayed on each screen at a time. This scene synchronization is referred to as *frame-locking*.

Display configurations that assign an individual screen and beam combiner to each user benefit from a higher resolution those configurations that split the screen (and its resolution) into different portions. To support such configurations, an additional screen transformation (S) has to be introduces (see section 3.4). This matrix represents a concatenation of several translations, rotations and scaling transformations, that map coordinates from the world coordinate system into an individual coordinate system of the corresponding screen. One can also think of the inverse transformation of the screen itself within the world coordinate system.

The following algorithm can be applied in combination with such configurations:

```
for left and right viewpoints i of all
viewers
     initialize transformation pipeline
     and polygon order
     compute reflected viewpoint e_i' = R_i e_i
     compute refraction offset Δ_i for i
     set transformation pipeline:
     R_i M F_i V_i S_i P
     reverse polygon order
     render scene from e_i'
endfor
```

Note that by keeping track of individual model transformations (M) and by rendering individual scenes for each user, the display configuration practically acts as multiple user-individual displays. Otherwise, multiple users share and interact with the same scene. To use individual model transformations but to render the same scene for each user has been proven to be an effective way of precisely registering virtual objects to real ones – cancelling out slight physical misalignments of the screens and mirrors (whose measured/calculated parameters that are stored in R_i and S_i reflect these errors) within each M. Thereby, the object transformation that is used to align real and virtual objects is stored in each M. This also applies for the algorithm that is presented in section 3.6.2 – although only one screen is used.

3.6.4. Individual Scenes

For some multi-user scenarios, a common scene is not required. Rather than that, each user observers and interacts with his or her own scene representation on the same display. The actions that are carried out by a user do not effect the scene state of the other users.

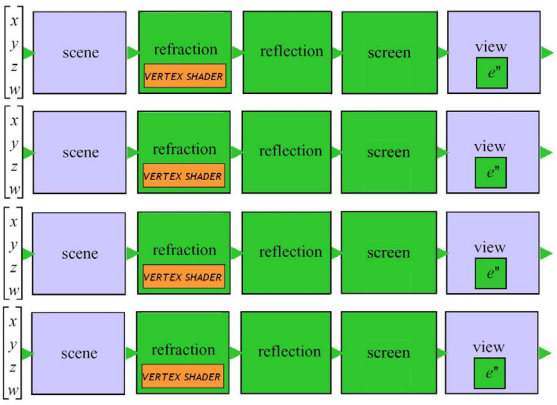


Figure 3.13: *Independent rendering pipelines that support individual scenes.*

In this case, the scene transformation becomes individual to each user and is carried out within each sub-pipeline. This makes all sub-pipelines completely independent from each other and allows distributing them (together with copies of the scene description) on separate rendering notes without the need of any synchronization mechanism. Now every user is assigned to completely independent components that are integrated into a single spatial optical see-through display.

3.7. Curved Mirror Beam-Splitters

Optical see-through displays can be assembled from many planar optical elements (mirror beam combiners or transparent screens). Every element requires its own rendering sub-pipeline. However, if many small optical elements are used to approximate curved surfaces, the above rendering concept becomes inefficient quickly.

Compared to calibrating multiple optical elements, the calibration effort for a single curved element can be reduced and optical aberrations that result from miscalibration can be decreased by several orders of magnitude. In addition, a curved optics can provide an *edge-free view* onto the augmented scene.

Like for curved projection screens the rendering for spatial optical see-through displays that apply *curved optics* (mirror beam combiners or transparent screens) has to be *warped* before being displayed. This transformation is view-dependent and curvilinear rather than a rigid-body transformation, and requires a *per-vertex viewpoint and model transformation*. A single accumulation of simple

homogenous transformation matrices (as described for planar optics) cannot express such a complex deformation for the entire scene. Consequently, fixed function rendering pipelines cannot be fully used for rendering graphics on curved optical see-through displays.

(a) (b)

Figure 3.14: *A warped and projected image reflected in a curved (conical) mirror beam combiner. A virtual cartridge is being placed in a physical printer.*

Note that since convex mirrors map a larger portion of the screen space into a smaller portion within the image space inside the optics, a high density of pixels can be compressed into a small reflection (cf. figure 3.14). Consequently, the *spatial resolution* within the reflection is regional higher than the spatial resolution of the display device!

Warping the scene geometry itself requires highly tessellated graphical models to approximate the *curvilinear optical transformation* well enough. Instead *multi-pass rendering* is frequently being applied. Rather that warping geometry, images of the geometry are warped and displayed. This causes the same effect, but the image transformation does not depend on the scene geometry (e.g., its type, its complexity, its shading, etc.).

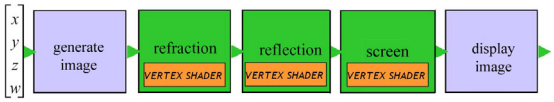
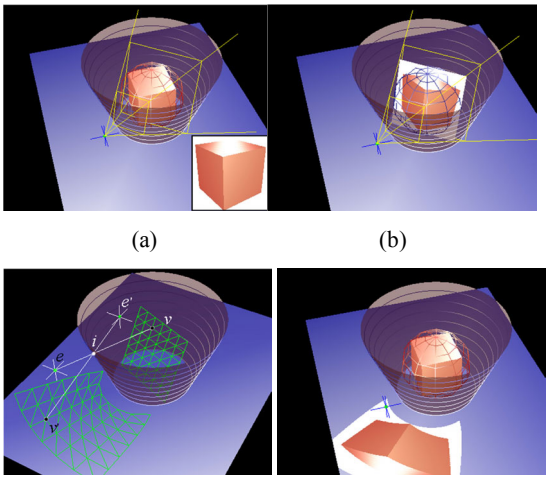


Figure 3.15: *Two-pass image warping pipeline for neutralizing curvilinear optical transformations.*

As outlined above, the *first rendering pass* generates an image of the graphical overlay – perspectively correct from the viewpoint of the observer. This image contains all visual information that is expected to appear within the real environment (e.g., shading, shadows, occlusion effects, etc.), and can be rendered completely in hardware. It is geometrically approximated by a *two-dimensional tessellated grid* that is transformed into the current viewing frustum in such a way that it is positioned perpendicular to the optical axis.

The grid vertices and the image's texture coordinates are transformed (warped) with respect to the viewpoint, the optics and the image source in the following steps. These transformations are applied on a *per-vertex level* and neutralize the image deformations that are caused by the optics (such as reflection and refraction of mirror beam combiners and transparent screens) and the image source (such as lens distortion of video projectors, or projections onto curved screens). As indicated, per-vertex transformations can be implemented as *vertex shaders* on programmable rendering pipelines.



(a) (b)

(c) (d)

Figure 3.16: *Different steps of image warping process: image generation, transformation of image grid, texturing and displaying deformed image grid.*

Finally, the image that was generated during the first pass is mapped onto the warped grid using *texture mapping* and *bi- or tri-linear texture* , and is displayed during the *second pass*. This process is repeated for multiple individual viewpoints (e.g., for stereoscopic viewing and for multi-user scenarios).

The following sections will describe all steps of the *two-pass image warping pipeline* in more detail.

3.7.1. Generating Images

The first rendering pass generates the un-deformed graphical overlay with all information that is expected to be visible within the real environment. Conventional graphics techniques that are provided by fixed function rendering pipelines are sufficient for this step.

To capture the entire virtual scene in the image, the *scene's bounding sphere* is used to define an *on-axis viewing frustum* from the viewpoint of the observer. The *apex* of the frustum is chosen in such a way that it encloses the bounding sphere exactly. This ensures that the complete scene is visible in the image and that it covers a large portion of the image. Finally the scene is rendered through this frustum into a texture memory block – rather than directly into the frame buffer. Older graphics cards do not allow drawing an image into the on-board texture memory directly. In this case the scene has to be rendered into the back frame buffer, and then copied into the texture memory of the graphics card. This step is referred to as *read-back*.

New graphics cards do allow rendering the image directly into an auxiliary on-board memory block (usually called *P-buffer*) that can be referenced during texture mapping. This step is referred to as *render-to-texture* and avoids the time-consuming operation of transferring the image information from the frame buffer into the texture memory.

The following OpenGL code fragment can be used to configure the correct on-axis frustum for generating the un-deformed image of the virtual scene:

```
…
//scene's center = p[0..2]
//scene's bounding sphere radius = r
//viewpoint = e[0..2], up-vector =
u[0..2]
//texture height, texture width = th, tw

float l, d;
float left, right, bottom, top, near,
far;

//compute parameters of on-axis frustum
l=sqrt((p[0]-e[0])*(p[0]-e[0])+
       (p[1]-e[1])*(p[1]-e[1])+
       (p[2]-e[2])*(p[2]-e[2]));
d=r*(l-r)/l;
left=-d; right=d; bottom=-d; top=d;
near=l-r; far=l+r;

//configure rendering pipeline
glViewport(0,0,tw,th);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glFrustum(left,right,bottom,top,near,far)
;
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(e[0],e[1],e[2],p[0],p[1],p[2],u
[0],u[1],[u2]);

//draw scene with scene transformation
into texture memory
…
```

The big advantage of this image-based method is that it is completely independent on the scene's content and the way this content is rendered. Instead of using a geometric renderer, other techniques (such as image-based and non-photo-realistic rendering, interactive ray-tracing, volume rendering, or point-based rendering, etc.) can be employed to generate the image.

The image that has been generated during the first rendering pass has to be transformed in such as way that is perceived undistorted while observing it through or with the display's combiner optics. These transformations are described below in detail. To support these image deformations, a geometric representation of the image plane is generated initially. This *image geometry* consists of a *tessellated grid* (e.g., represented by an *indexed triangle mesh*) which is transformed into the current viewing frustum in such a way that, if the image is mapped onto the grid each line of sight intersects its corresponding pixel. Thus, the image grid is perpendicular to the optical axis and centered with the scene geometry. The following deformation steps transform the positions and the texture coordinates of every grid vertex individually.

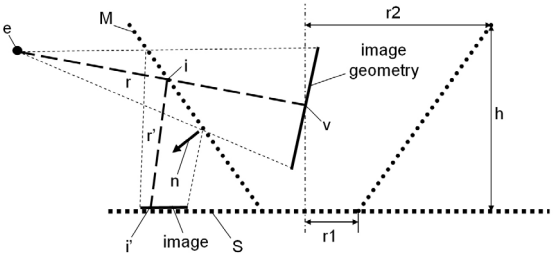
3.7.2. Reflection on Convex Mirrors

As discussed in section 3.6, convex multi-plane mirror assemblies unequivocally tessellate the surrounding space into mirror individual reflection zones which –for the same point of view– do not intersect or overlap. Consequently they provide a definite one-to-one mapping between screen space and reflection space. This is also true for curved convex mirrors. Curved convex mirrors cannot provide true stigmatism between all object-image pairs, but rather a close approximation which introduces small optical aberrations. Since due to the limited resolution of the eyes, human observers can usually not perceive these aberrations, we want to disregard them for the subsequent sections.

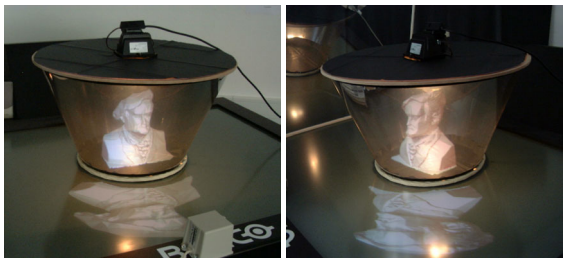
Images formed by convex mirrors appear to be reduced and deformed versions of the reflected objects. Hence, the image of the scene that results from the first rendering pass has to be stretched before displaying it on the secondary display. This results in the original (unscaled) image after the physical reflection.

Several curved mirror displays exist that generally don't pre-distort the graphics before they are displayed. Yet, some systems apply additional optics (such as lenses) to stretch or undistort the reflected image (e.g. [Mck99a, Mck99b]). But these devices constrain the observer to a single point of view or to very restricted viewing zones. However, if a view-dependent rendering is required to support freely moving observers, interactive rendering and real-time image warping techniques are needed which provide appropriate error metrics.

The reflection transformation for displays that apply convexly curved mirror beam combiners will be explained below based on an example of the conical mirror display illustrated in the figures above. Any other surface type can be used in combination with the described techniques.



(a)



(b) (c)

Figure 3.17: *Reflection transformation on conical mirror.*

Each grid point v of the image geometry has to be transformed with respect to the mirror surface M, the current viewpoint e, and the secondary screen S and texture each grid point with the image that we generated during the v, the intersection i of the geometric line of sight with the mirror surface has to be computed (that is, the ray r that's spanned by the eye e and the vertex v). Next, the normal vector n at the intersection i needs to be determined. The intersection point, together with the normal vector, gives the tangential plane at the intersection. Thus, they deliver the individual plane parameters for the per-vertex re

reflection ray r' and the intersection i' of r' with the secondary screen surface S.

Having a geometric representation (e.g., a triangle mesh) to approximate the mirror's and the screen's surfaces, M and S, supports a flexible way of describing the dimensions of arbitrary shapes. However, the computational cost of the per-vertex transformations increases with a higher resolution surface geometry. A high resolution surface description, however, is required to neutralize the optical reflection distortion effectively. For triangle meshes, a fast ray-triangle intersection method (such as [Moe97]) is required that automatically delivers the barycentric coordinates of the intersection within a triangle. The barycentric coordinates can then be used to interpolate between the three vertex normals of a triangle to approximate the normal vector at the intersection.

A more efficient way of describing surface shapes for this purpose is to apply explicit functions in the form of $F(x,y,z)=0$. Explicit functions to describe mirror and screen surfaces ($M(x,y,z)=0$ and $S(x,y,z)=0$) can be used to calculate the intersections and the normal vectors (using its 1st order derivatives) with an unlimited resolution by solving simple equations. However, not all shapes can be expressed by explicit functions.

Obviously, we have the choice between a numerical and an analytical approach, for describing mirror and screen surfaces. If an analytical solution is given, it should be preferred over the numerical variant. Higher order curved surfaces, however, require the application of numerical approximations.

The explicit function for the conical mirror example and its 1st order derivatives are:

$$M(x,y,z)=\frac{x^2}{r1^2}+\frac{y^2}{r2^2}-\frac{z^2}{h^2}=0 \qquad \text{and}$$

$$n=\left[\frac{M}{\delta x},\frac{M}{\delta y},\frac{M}{\delta z}\right]=2\left[\frac{x}{r1^2},\frac{y}{r2^2},-\frac{z}{h^2}\right]$$

where $r1, r2$ are the cone's radii with its center located at the world-coordinate-system's origin, and h is the cone's height along the z-axis.

To intersect the ray r with the mirror surface, it has to be transformed from the world-coordinate-system into the coordinate-system that is used by the explicit function. Then it can be intersected easily with the surface by

solving a (in our example quadratic) equation created by inserting a parametric ray representation of $r(x,y,z)$ into the mirror equation $M(x,y,z)$.

Given the surface intersection i and the normal n at i, the specular reflection ray can be computed with:

$$r' = r - 2n(nr)$$

Finally, r' has to be transformed from the coordinate system of M into the coordinate system of S to compute the intersection i' with S by solving another equation created by inserting r' into S.

The following CG vertex shader fragment can be used to compute the reflection transformation for conical mirror surfaces (this example applies to the truncated conical mirror surface illustrated in figure 3.17):

```
…
// viewpoint = e[0..2]
// vertex = v[0..3]
// truncated cone parameters: lower
// radius = r1, upper radius = r2,

height = h

float3 i,i_,n,r,r_;
float  a,b,c,r,s,t,u,l1,l2,l3;

// compute cone parameters
a=r2;b=r2; c=(r2*h)/(r2-r1);

// transformation to cone-coordinat
system // (r=v-e)
v.z=v.z+(c-h);      e.z=e.z+(c-h);
//compute viewping direction (r)
r=v-e; r=normalize(r);

//compute cone intersection

s=(e.x*e.x)/(a*a)+(e.y*e.y)/(b*b)-
(e.z*e.z)/(c*c);

t=2.0*((e.x*r.x)/(a*a)+(e.y*r.y)/(b*b)-
(e.z*r.z)/(c*c));

u=(r.x*r.x)/(a*a)+(r.y*r.y)/(b*b)-
(r.z*r.z)/(c*c);

l1=(-t+sqrt(t*t-4.0*u*s))/(2.0*u);
l2=(-t-sqrt(t*t-4.0*u*s))/(2.0*u);
i=e+r*l2;

// back-transformation to world
// coordinate-system
i.z=i.z-(c-h);

// compute cone normal

n.x=i.x/(a*a);
n.y=i.y/(b*b);
n.z=i.z/(c*c);
n=normalize(n);

// compute incident ray (i-e)
r=i-e; r=normalize(r);

// compute reflector
r_=reflect(r,n);

// compute intersection with screen (x/y
// plane in this example)
l1=-dot(float3(0,0,1),i);

l2=dot(float3(0,0,1),r_);
l3=l1/l2;
i_=i+l3*r_;

// compute projected vertex
v.xyz=i_.xyz; v.w=1.0;
…
```

Note, that the same shader can be used in combination with any other explicit functions that describe the mirror and the screen surfaces. Only the surface-equation specific parts have to be replaced.

3.7.3. Refraction

A vertex transformation is applied to the rendered geometric model between scene and reflection transformation. To cope with the curvilinearity of refraction, vertex shaders are preferred over a rigid-body approximation if programmable function pipelines are available.

For the image-based method that is discussed above, a per-pixel transformation on the image plane would be more efficient than a per-vertex transformation of the image grid. Pixel shaders of programmable function pipelines, however, do not yet support the geometric transformation of pixels. But pixel transformations can be approximated with vertex shaders by computing new texture coordinates for each vertex that offsets the corresponding pixel positions on the image plane. The positions of pixels in between vertices (i.e., within patches of the image grid) are then approximated via linear interpolation of texture mapping. As for the reflection transformation, this introduces optical errors that can be high if the resolution of image grid is too low.

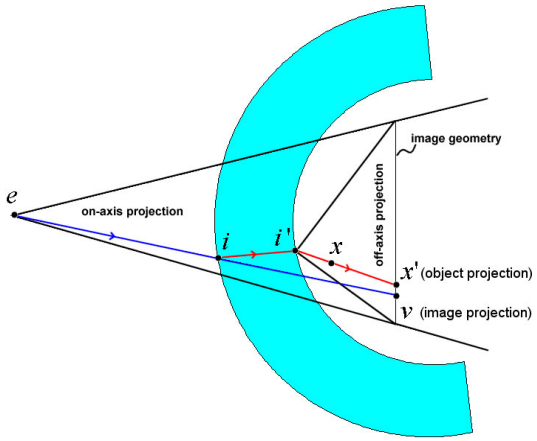


Figure 3.18: *Image-based in-/out-refraction transformation on curved lens.*

As illustrated above, two surface descriptions (M_o and M_i) are required to compute in- and out-refracted rays for such optical combiners since they represent curved, surface-parallel lenses. As for the reflection transformation, explicit functions in the form of $F(x,y,z)=0$ are again most efficient for computing the required intersections i, i' and the corresponding parameters of the tangent planes.

Similar to the reflection case, each ray from the viewpoint e to all grid vertices v have to be processed. Each ray has to be transformed from the world-coordinate-system into the coordinate-system that is used by the explicit function, and then intersected with the outer in-refracting surface M_i. The computed intersection i and its tangential plane parameters are used to determine the in-refractor, which is intersected again with the inner out-refracting surface M_o. The second intersection i' and its tangential plane parameters allow computing the out-refractor. Any point on this ray (x) can be projected onto the image plane from i'. This results in position x' in projected and normalized device coordinates (e.g., [-1,1]). After a conversion from device coordinates into normalized texture-coordinate (e.g.,[0,1]), the texture coordinates of the pixel that needs to appear at grid vertex v has been determined. To simulate refraction, these texture coordinates are simply assigned to v. Note, that x' can also be computed by intersecting the out-refractor directly with the image plane. This may requires a different conversion from world coordinate systems into normalized texture coordinates.

The composition of an appropriate texture matrix that computes new texture coordinates for the image vertex is summarized in the algorithm below:

```
compute texture normalization correction*:
```
$$S = scale(0.5,0.5,0.5), s = s \cdot translate(1,1,0)$$

```
compute off-axis projection
transformation:
```
$$\phi = \frac{(i'_z - 1)}{i'_z},$$
$$left = -\phi(1 + i'_x),$$
$$right = \phi(1 - i'_x)$$
$$bottom = -\phi(1 + i'_y), \quad top = \phi(1 + i'_y)$$
$$near = i'_z - 1, \quad far = i'_z + 1$$
$$P = frustum(left, right, bottom, top, near, far)$$

```
compute view transformation:
```
$$V = lookat(i'_x, i'_y, i'_z, i'_x, i'_y, 0,0,1,0)$$

```
compute new texture coordinate x' for
particular x(v), including perspective
division:
```
$$x' = S \cdot \frac{(P \cdot V \cdot x)}{w}$$

As illustrated in figure 3.18, an off-axis projection transformation is applied, where the center of projection is i'. Multiplying x by the resulting texture matrix and performing the perspective division projects x to the correct location within the normalized texture space of the image. Finally, the resulting texture coordinate x' has to be assigned to v'.

Note that if the image size is smaller than the size of the allocated texture memory, this difference has to be considered for the normalization correction. In this case, the image's texture coordinates are not bound by the value range of [0,1].

As mentioned above, the matrix operations can be replaced by an explicit ray-casting.

*This applies for OpenGL-like definitions of the texture and normalized device coordinates.

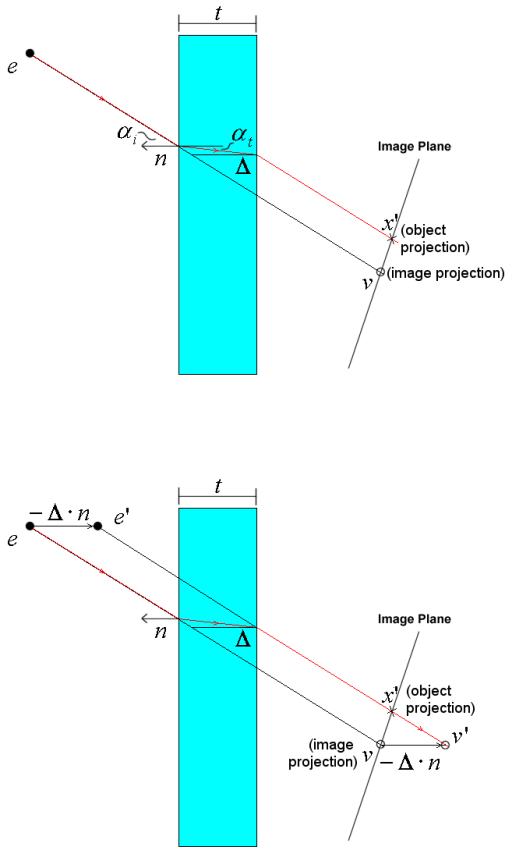


Figure 3.19: *Image-based in-/out-refraction transformation on planar lens.*

For plane parallel lenses, the in-refracting and out-refracting surfaces are the same – but offset by the thickness t of the lens. Explicit in- and out-refractors do not have to be computed in this case. Rather than that, the offset Δ can be computed for every geometric line of sigh $(v - e)$ as described in section 3.3.2. The new projection on the image plane can be computed by translating the viewpoint e and the corresponding grid vertex v by the amount Δ along the normal of the lens. The new texture coordinates can be computed by projecting v' onto the image plane from e' and converting the result into the normalized texture space.

Nevertheless, both refraction methods face the following problems for outer areas on the image:

- Given a geometric line of sight to an outer grid vertex, its corresponding optical line of sight does not intersect the image. Thus, a grid vertex exists but its new texture coordinate cannot be computed. This results in vertices with no, or wrong texture information;
- Given an optical line of sight to an outer pixel on the image, its corresponding geometric line of sight does not intersect the image. Thus, a texture coordinate can be found but an assignable grid vertex does not exist. Consequently, the portion surrounding this pixel cannot be transformed. This results in image portions that aren't mapped onto the image geometry.

A simple solution to address these problems does not avoid them, but ensures that they do not occur for image portions which contain visible information: As described in section 3.7.1 the image size depends on the radius of the scene's bounding sphere. Its radius can simply be increase by a constant amount before carrying out the first rendering pass. An enlarged image does not affect the image content, but subjoins additional outer image space that does not contain any visible information (i.e., just black pixels). In this way, we ensure that the above mentioned problems emerge only at the new (black) regions. Yet, these regions will not be visible in the optical combiner.

Note that in contrast to the image-based reflection transformation (section 3.7.2) which transforms grid vertices, the refracted image transform re-maps texture coordinates. However, all image transformations have to be applied before the final image is displayed during the second rendering pass.

3.7.4. Screen Transformation and Non-Planar Screens

In cases where the screen coordinate system is not equivalent to the world coordinate system in which beam combiners, user(s) and scene(s) are defined, an additional screen transformation has to be applied to project the reflected/refracted image-geometry vertices to their correct places. This can be achieved with additional affine transformation operations that are applied to the image-geometry itself, as described in section 3.4, or by intersecting the traced rays with the geometry of the transformed screen plane (e.g., if explicit ray-casting is used inside a vertex shader).

If the secondary screen is not planar, an explicit ray-casting approach still leads to a correct result. The traced

rays simply have to be intersected with a geometric representation of the curved screen.

Note that an approach that applies projective texture-mapping [Seg92] for planar mirror beam-spitters fails for a combination of curved mirror optics and curved screens. Although the first rendering pass is similar to the first pass that generates the image (see section 3.7.1), the second pass cannot be applied, because multiple centers of projection exist (one for each transformed image vertex).

3.7.5. Displaying Images

During the second rendering pass, the transformed image geometry is
mapping the outcome of the
onto its surface.

Note, that if the re
the previous steps deliver device coordinates, but the secondary screen and the mirror optics are defined within the world-coordinate system, a second projection transformation (such as glFrustum) and the corresponding perspective divisions and viewpoint transformation (such as gluLookAt) aren't required. If a plane secondary screen is used, a simple scale transformation suf
the device coordinates – for example, glScale(1/device_width/2),1/device_height/2,1). A subsequent view-port transformation
into the window-coordinate system – for example, glViewport(0,0,window_ width, window_height).

Time-consuming rendering operations that aren't required to display the 2D image (such as illumination computations, back-face culling, depth buffering, and so on) should be disabled to increase the rendering performance.

In this case, the polygon order doesn't need to be reversed before rendering, as we noted previously.

Obviously, one can choose between numerical and analytical approaches to represent curved mirror surfaces. Simple shapes can be expressed as parametric functions, but higher order curved mirrors require numerical approximations. In addition, the grid resolution that's required for the image geometry also depends on the mirror's shape. Pixels between the triangles of the deformed image mesh are linearly approximated during rasterization (that is, after the second rendering pass). Thus, some image portions stretch the texture while others compress it. This results in different regional image resolutions. However, because of the symmetry of simple mirror setups (such as cones and cylinders), a regular grid resolution and a uniform image resolution achieve acceptable image quality. In section 3.7.8 a selective refinement method is described that generates a non-uniform image geometry to minimize the displacement error of the image portions, the complexity of the image geometry and consequently the number of vertex transformations and triangles to be rendered.

Since a primitive-based (or fragment-based) antialiasing doesn't apply in deformed texture cases, bilinear or trilinear texture
with antialiasing, texture
the graphics hardware.

Note that the image's background and the empty area on the secondary screen must be rendered in black, because black doesn't emit light and therefore won't be re
into the reflection space.

3.7.6. Multiple Viewers

To support multi-user applications, the viewer-individual images must be composed and the black background must be *color-blended* appropriately. Since for convex beam combiners the images are stretched within the screen space to appear correctly within the reflection space (cf. figure 3.20), multiple images for different observers might intersect.

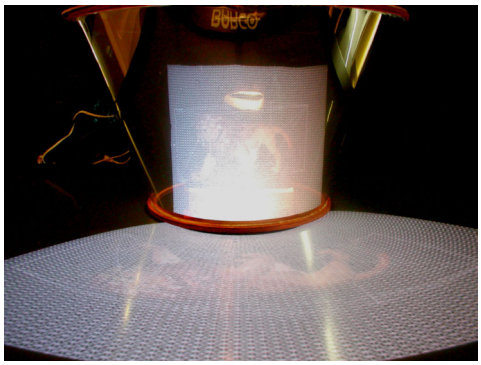


Figure 3.20: *Stretched image geometry projected on secondary screen and reflected in conical mirror beam combiner.*

In these cases, individual observers can perceive the (perspectively wrong) images of other users in addition to their own images. The amount of intersection depends on the size of the graphical scene, the positions of the observers and the parameters of the mirror optics and the

secondary screen. The larger the graphical scene, for instance, the more space on the secondary screen is required and conflicting situations become more likely. Such conflicts also arise for multi-plane beam combiner configurations if multiple users would be allowed to move freely around the display (as it has been described for a single user in section 3.6.1). Although this is technically possible, it has not been discussed in section 3.6.2.

3.7.7. Concave Beam-Splitters

Concave mirrors can generate both -real images and virtual images. Light rays which are reflected off convex mirror assemblies do not intersect. In contrast to this, light rays that are reflected off a *parabolic concave mirror* do intersect exactly within its *focal point*. For concave mirrors that deviate from a parabolic shape (e.g., *spherical mirrors*) the light rays do not intersect within a single point, but bundle within an *area of rejuvenation*.

The rendering techniques for curved mirror beam combiners explained above, are described on the basis of convex mirrors. However, they can be applied for concave mirrors without or with only minor modifications to the algorithms.

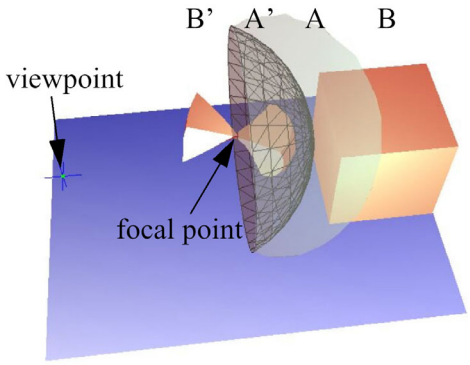


Figure 3.21: *Reflected geometry at a concave mirror.*

Figure 3.21 illustrates how, given the viewpoint and the location of the mirror, the geometry of a cube is transformed by the algorithm described in section 3.7.2 (without projection onto the screen plane). Note, that the vertices of a highly tessellated surface approximation of the cube are reflected, instead of the vertices of the image geometry.

The portion of the geometry that is located within area A is mapped to area A' (behind the mirror's focal point -as seen from the viewpoint). This mapping has a similar behavior as for convex mirrors. The portion of the geometry that is located within area B, is mapped to area B' (in front of the mirror's focal point -as seen from the viewpoint). In this case, the mapped geometry is flipped and the polygon order is changed. The geometry that is located on the intersecting surface of A and B is mapped to the focal point.

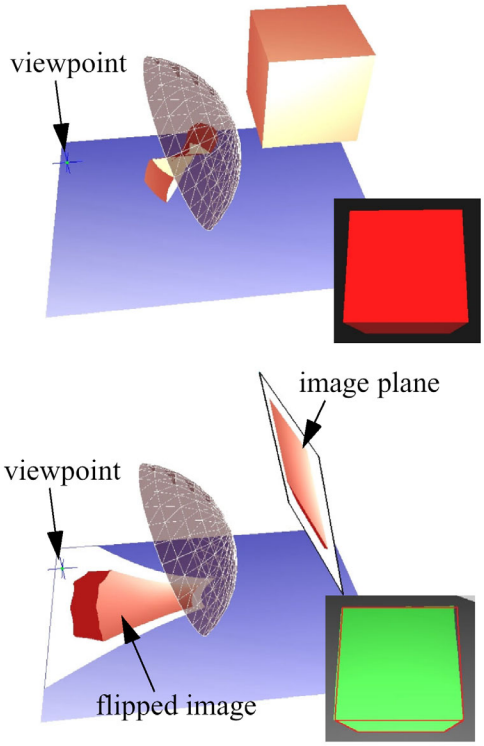


Figure 3.22: *Reflecting scene geometry (left) and reflecting image geometry (right) at a concave mirror.*

Figure 3.22 shows that concave mirrors can be treated just like convex mirrors. The left image indicates the reflection of the scene geometry itself (again, without projection onto the screen plane). The transformed geometry and the mirror model have been exported to a ray-tracer and the image that is seen from the viewpoint has been raytraced. The result is outlined at the lower right of the left image (red cube).

It can be seen, that the transformed geometry appears untransformed as reflection in the mirror. Note that due to

the different orders of polygons that are located in front of and behind the focal point, polygon order related options offered by the applied rendering package should be disabled. Otherwise, additional actions have to be taken to determine the location of each transformed polygon and its order.

The same experiment has been carried out reflecting the image geometry (including the projection onto the screen plane). The image geometry was transformed and the image that has been generated during the first rendering pass was mapped onto the transformed image grid. The right part of figure 3.22 illustrates that the projected image is a flipped version of the original image. If ray-traced from the given viewpoint, this deformed image appears as correct reflection in the mirror (see lower right of the right image). Note that the contours of the ray-traced result have been highlighted, since the planar projection of the deformed image does not provide sufficient normal or depth information to generate correct shading effects with the ray-tracer. Note also, that reflection the scene geometry and projecting the result onto the screen plane yields the same results as reflecting and projecting the image geometry.

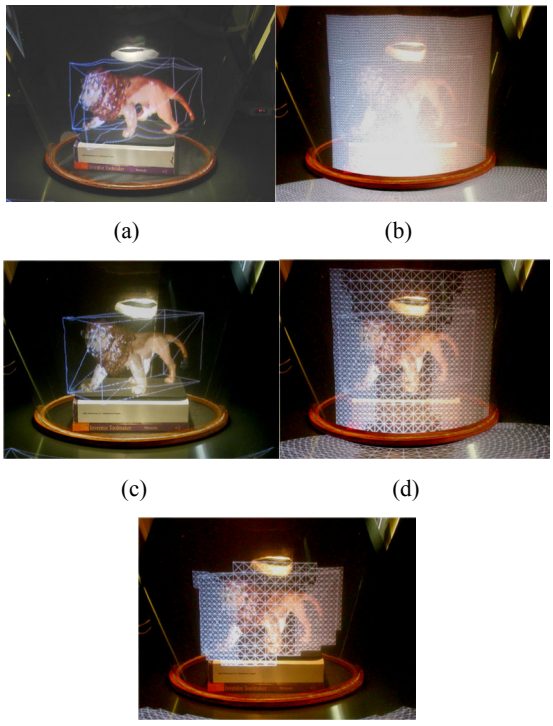
Mirrors with a *mixed convexity* (i.e., simultaneously convex and concave mirrors) can cause multiple images of the same object, or they can reflect multiple objects to the same location within the image space. In such cases, the transformations from screen space to reflection space and vice versa are not definite and are represented by possible many-to-many mappings. Such types of mirrors should be decomposed into convex and concave parts (as done by Ofek [Ofe98, Ofe99]) to ensure a correct functioning of our algorithms. Although for many surfaces this can be done fully atomically [Spa92], spatial optical see-through configurations do not make use of mirrors with mixed convexities. This is because one of the initial aim of such displays -namely to unequivocally overlay real environments with computer graphics in an AR manner- is physically not supported by such mirrors. Consequently, mirrors with mixed convexities are not considered for spatial augmented reality.

3.7.8. Non-Uniform Image Geometry

If image warping is applied for a *non-linear predistortion* (such as described above), the required grid resolution of the underlying image geometry depends on the *degree of curvilinearity* that is introduced by the display (e.g. caused by the properties of a mirror, a lens, or a projection surface).

Pixels within the triangles of the warped image mesh are linearly approximated during rasterization (i.e. after the second rendering pass). Thus, some image portions stretch the texture while others compress it. This results in different local image resolutions. If, on the one hand, the geometric resolution of an applied uniform grid is too coarse, texture artifacts are generated during rasterization (cf. figure 3.23a). This happens because a piecewise bi- or tri-linear interpolation within large triangles is only a crude approximation to neutralize the curvilinear optical transformations.

If on the other hand, the grid is oversampled to make the interpolation sections smaller (cf. figure 3.23b), interactive frame rates cannot be achieved. In addition to the speed of the
view-dependent multi-pass methods depends mainly on the complexity of the image geometry that in
geometric transformation times and on the image resolution that in *texture transfer* and *times*.



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

Figure 3.23: *Selectively refined image geometry: (a) distorted image created with an undersampled uniform image grid, (b) oversampled uniform image grid, (c) undistored image, (d) selectively refined image grid, (e) grid portion that projects onto bounding container.*

This section describes a selective re
[Bim03] that generates image grids with appropriate local grid resolutions on the
oversampling and the occurrence of artifacts within the

The main dif
that supports curved mirror beam combiners is that in contrast to simpler screens, a displacement error that de
the image space of the mirror optics, rather than in the screen space of the display. For convexly curved mirror beam combiners, this requires fast and precise numerical methods. For displays that do not contain view-dependent optical components (e.g. curved screens), these computations are much simpler because analytical methods or look-up tables can be used to determine the error within the object space (e.g. the screen space).

Recent advances in level-of-detail (LOD) rendering take advantage of *temporal coherence* to *adaptively re* geometry between subsequent frames. Especially, terrain-rendering algorithms locally enhance terrain models by considering viewing parameters.

Hoppe introduced *progressive meshes* [Hop96] and a later developed a view-dependent re
progressive meshes [Hop97, Hop98]. Given a complex triangle mesh, Hoppe -generates a coarse representation called *base mesh* by applying a series of *edge collapse operations*. A sequence of pre-computed *vertex split operations* that are inverse to the corresponding edge collapse operations can then be applied to the base mesh's regions of interest to successively re

The selection of the appropriate vertex split operations is based on his re
describes a method that generates a view-dependent and continuous LOD of height -time, instead of precomputing a coarse base mesh and a sequence of re
height *quad-tree* of *discrete grid blocks* with individual LODs. Beginning with the highest LOD, Lindstrom locally applies a two-step surface simpli method: he
for a particular region by applying a coarse *block-based simpli* , and then performs a *-grained re-triangulation* of each LOD model in which vertices can be removed. To satisfy continuity among the different LODs, Lindstrom considers vertex dependencies at the block boundaries.

The main difference between both methods is that Lindstrom performs a dynamic simpli highresolution height ^{2} during rendering. Lindstrom's mesh de *implicit hierarchical LOD structure*. Hoppe applies re steps to low-resolution LODs of arbitrary meshes during rendering. His mesh de
require an implicit hierarchical LOD structure. Consequently, the re
base mesh have to be precomputed. In addition, he applies *triangulated irregular networks* (TINs) for triangulation, rather than regular grids. Note that these two types of re
techniques. Since the image geometry for spatial optical see-through displays can also be parameterized in R^2 and provides an implicit hierarchical LOD structure, multiple LODs or appropriate re
pre-computed but can be ef
This is similar to Lindstrom's approach. However, simplifying a high-resolution mesh instead of re
low-resolution mesh would require to re-transform all grid vertices of the highest LOD after a change of the viewpoint occurred. This is very inef
displays that we consider, viewpoint changes normally happen at each frame.

In contrast to the static geometry that is assumed in Lindstrom's and Hoppe's case the image geometry generated for spatial optical see-through displays is not constant but dynamically deforms with a moving viewpoint. Consequently, the geometry within all LODs dynamically changes as well.

Therefore, this section describes a method that dynamically deforms the image geometry within the required LODs while *selectively re* the lowest-resolution base mesh during rendering. The method aims at minimizing the *displacement error* of the image portions, the complexity of the image geometry and consequently the number of vertex transformations and triangles to be rendered.

Note that this method cannot be implemented with the limited capabilities of vertex shaders, yet. Consequently it has to be executed on the CPU rather than on the GPU. This implies that rendering a high-resolution uniform image grid on a GPU might be faster that creating a selectively refined image grid on the CPU. Nevertheless, the LOD method is explained to impart knowledge about displacement errors that are generated by approximating a curvilinear optical transformation with discretized image grids.

3.7.8.1. Image Triangulation

Instead of transforming and rendering a uniform high-resolution mesh, one can start from the coarsest geometry representation and successively refine it locally until certain *refinement criteria* are satisfied. Due to the well-defined structure of the image grid, all possible *global or discrete LODs* can be computed at runtime – including the highest, which is the uniform high-resolution representation of the mesh itself.

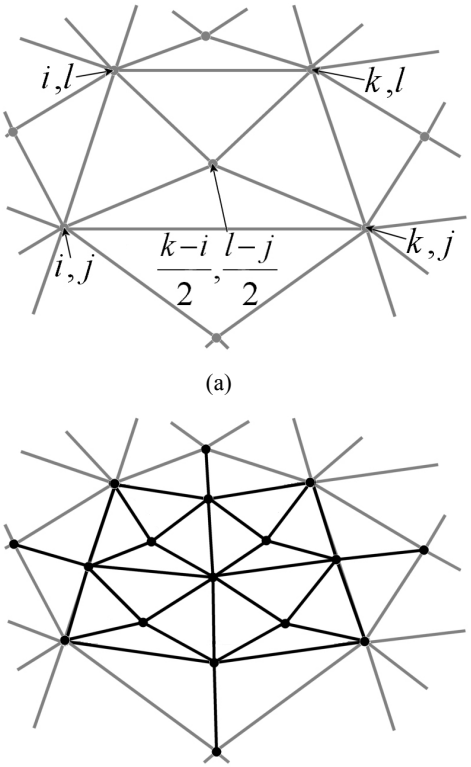


Figure 3.24: *Triangulation of unrefined patch at LOD n (a), and triangulation of refined patch at LOD n+1 with resolution transitions (b).*

Figure 3.24 illustrates a quadtree-based image triangulation, which is similar to Lindstrom's triangulation method for height fields [Lin96]. While figure 3.24a shows an unrefined patch at LOD n, figure 3.24b shows the same patch at LOD $n+1$, with lower LOD neighbors. Given a highest LOD of m, an indexed $(2^m+1)\times(2^m+1)$ matrix structure can be chosen to store the grid vertices.

For illustration purposes the following types of patch vertices can be differentiated:

- *L-vertices* are vertices at the corners of a patch (e.g., at indices [*i*,*j*],[*i*,*l*],[*k*,*l*] and [*k*,*j*] in figure 3.24a);
- *X-vertices* are vertices at the center of a patch (e.g., at index [(*k*-*i*)/2,(*l*-*j*)/2] in figure 3.24a);
- *T-vertices* are vertices that split the patch edges after refinement (e.g., at indices [*i*,(*l*-*j*)/2], [*k*,(*l*-*j*)/2], [(*k*-*i*)/2,*l*], and [(*k*-*i*)/2,*j*] in figure 3.24b)

To refine a patch, it is divided into four sub-patches by computing the corresponding four T-vertices, as well as the four X-Vertices that lie inside the sub-patches. Note that the matrix structure is in this particular case equivalent to a quad-tree data structure. To ensure consistency during rasterization, the T-vertices have to be connected to their neighboring X-vertices wherever a LOD transition occurs (e.g., at all four neighbors of the refined patch, as shown in the example illustrated in figure 3.24b).

Due to the well-defined matrix structure that contains the image grid, the following conditions are given:

- A clear relationship between the X-vertices and T-vertices exists: X-vertices can never be T-vertices, and vice versa.
- Each patch has definite L-vertices, T-vertices and X-vertices, whose indices can always be computed.
- Each X-vertex can be explicitly assigned to a single patch at a specific LOD.
- Each T-Vertex can be explicitly assigned to exactly one or two adjacent patches at the same LOD.

The triangulation methods described above require *continuous* level-of-detail transitions [Mit92]. This implies that neighboring patches do not differ by more than one LOD.

3.7.8.2. Recursive Grid Refinement

The objective of this step is to generate an image grid that provides a sufficient local grid resolution (i.e. appropriate discrete LODs) to avoid artifacts within the rasterized texture that would result from undersampling, as well as oversampling.

The following pseudo code illustrates an approach to recursively refine a grid patch, which initially is equivalent to the lowest LOD (i.e. the patch at the lowest LOD is outlined by the L-vertices at the four corners of the image geometry):

```
RecursiveGridRefinement(i,j,k,l )
1: begin
2:   a=(k-i)/2 , b=(l-i)/2
3:   if
     GeneratePatch([i,j],[i,l],[k,l],[k,j
     ],[a,b])
4:     begin
5:
       TransformPatchVertices([i,j],[i,l]
       ,[k,l],[k,j],[a,b])
6:     P = P ∪ {[i,j,k,l]}
7:     if
       RefineFurther([i,j],[i,l],[k,l],[k
       ,j],[a,b])
8:     begin
9:       RecursiveGridRefinement(i,j,a,b)
10:      RecursiveGridRefinement(a,j,k,b)
11:      RecursiveGridRefinement(i,b,a,l)
12:      RecursiveGridRefinement(a,b,k,l)
13:      if j < 2^m + 1  TC[a,j]+=1
14:      if k < 2^m + 1  TC[k,b]+=1
15:      if l > 1  TC[a,l]+=1
16:      if i > 1  TC[i,b]+=1
17:    end
18:  end
19: end
```

The patch that has to be refined is stored at indices *i*,*j*,*k*,*l* within a matrix structure. Condition (ii) allows to locate the position of the patch-individual X-vertex at indices *a*,*b* (line 2). First, it is evaluated whether or not a patch has to be generated at all (line 3). The conditions that are implemented within the *GeneratePatch* function will be discussed below. The four L-vertices and the X-vertex are transformed from the image plane to the display surface (line 5) –as described in detail in sections 3.7.2 and 3.7.3. Normally the image plane is located within the image space of the mirror optics. In this case, these mappings composite the vertex individual model-view transformations to neutralize reflection and refraction, as well as the projection transformation that maps a vertex onto the display surface. Note that vertices are only transformed once – even if the recursive refinement function addresses them multiple times. This is realized by attaching a marker flag to each vertex. A reference to the transformed patch is stored in patch set *P* by adding the patch's indices to *P* (line 6). In line 7, a function is called that evaluates the transformed patch based on pre-defined refinement criteria and decides whether or not this patch has to be further refined. The main refinement criterion is described below. If this decision is positive, the patch is divided into four equal sub-patches and the refinement function is recursively called for all of these sub-patches (lines 9-12). Note that condition (ii) also allows to determine the indices of the patch's four T-vertices, which become L-vertices of the sub-patches in the next LOD. Consequently, the *GeneratePatch* and the *RefineFurther* functions represent the exit conditions for the recursion.

It was mentioned that T-vertices have to be connected to their neighboring X-vertices whenever an LOD transition occurs to ensure consistency during rasterization. To detect LOD transitions, a counter (TC) is attached to each T-vertex. This counter is incremented by 1, each time the corresponding T-vertex is addressed during the recursive refinement (lines 13-16). Note that the if-conditions ensure a correct behavior of the counter at the image geometry's boundaries. Due to condition (iv) each counter can have one of the following three values:

- *0*: indicates that the T-vertex is located at a boundary edge of the image geometry or it is contained by a discrete LOD that is higher than the required one for the corresponding region.
- *1*: indicates a LOD transition between the two neighboring patches that –with respect to condition (iv)– belong to the T-vertex.
- *2*: indicates no resolution transition between the two neighboring patches that belong to the T-vertex.

After the image grid has been completely generated, all patches that are referred to in *P* are rendered with appropriate texture coordinates during the second rendering pass. Thereby, the counters of the patch's four T-vertices are evaluated. Depending on their values, either one or two triangles are rendered for each counter. These triangles form the final patch. Counter values of 0 or 2 indicate no LOD transition between adjacent patches. Consequently, a single triangle can be rendered which is spanned by the T-vertex's neighboring two L-vertices and the patch's X-vertex (this is illustrated in figure 3.24b). A counter value of 1, however, indicates a LOD transition. Two triangles have to be rendered that are spanned by the T-vertex itself,

the two neighboring L-vertices and the X-vertex of the adjacent patch (this is illustrated in figure 3.24a). A selectively refined image geometry is shown in figure 3.23d.

3.7.8.3. Generation and Refinement Criteria

This section discusses the patch generation and refinement criteria that are implemented within the *GeneratePatch* and *RefineFurther* functions. The input for these functions is the four L-vertices, as well as the X-vertex of a patch. They deliver the Boolean value *true* if the patch has to be generated, transformed, rendered or further refined, or *false* if this is not the case.

The *GeneratePatch* function, that supports an appropriate image clipping, and an evaluation criterion that considers the scene's convex container is described in part 1.

In general, the *RefineFurther* function can represent a Boolean concatenation of multiple refinement criteria (such as maximal patch size, angular deformation, etc.). An important refinement criterion for LOD methods is the *screen space error*. Since the computations of this displacement error are display specific, an important variation of the screen space error that can be applied for convex mirror beam combiners is described -the *image space error*. This error is explained in greater detail in parts 2 through 4.

1.) Spatial Limits

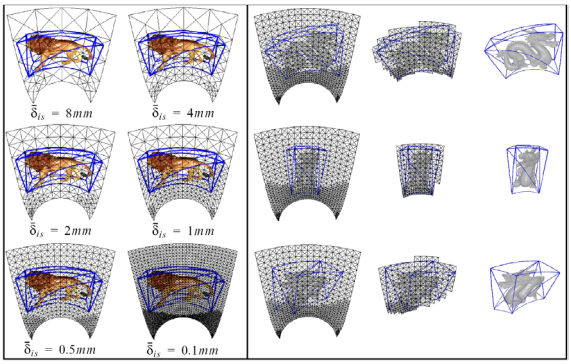
The multi-pass rendering method that is described in sections 3.7.1-3.7.7 uses the scene's *bounding sphere* to determine the parameters of the symmetric viewing frustum and the image size (cf. figure 3.16). Since all image generation methods assume a rectangular image shape that is adapted to today's screen shapes, the bounding sphere provides enough information to determine the rectangular image size.

Bounding spheres, however, are only rough approximations of the scene's extensions and consequently cover a fairly large amount of void space. This void space results in grid patches on the image plane whose texture does not contain visible color information.

To achieve a speed-up, these patches are avoided while creating the image grid. This implies that they are not transformed and refined during the *RecursiveGridRefinement* algorithm and that they are not rendered during the second rendering pass. As a result an image grid is generated and renderd that is not rectangular but dynamically approximates the silhouette of the scene as perceived from the observer's perspective (cf. Figure 3.23e). A condition that causes the recursion to exit in these cases is implemented within the *GeneratePatch* function:

A tighter convex container (such as an oriented convex hull or a bounding box) of the scene is evaluated that is generated either in a pre-process for static objects, or at runtime for animated scenes. For each untransformed patch that is passed recursively into the *RecursiveGridRefinement* algorithm, it has to be determined whether the container is visible on that patch – partially or as a whole. The approximation is twofold: First, the geometric lines of sight from e to all four L-vertices of the patch are intersected with the front-facing portion of the container. Second, the geometric lines of sight from e to front-facing container vertices are intersected with the patch. If at least one of the resulting rays causes an intersection, the patch might contain visible color information and it will be further processed. If, however, none of the rays cause intersections, the patch is not treated further (i.e. it won't be transformed nor refined or rendered). If the convex container is represented as a triangle mesh, a fast *ray-triangle intersection* method [MOE97] is applied together with the front-facing container triangles. Note that as for vertex transformations, the intersection information are buffered and looked up in the memory, rather than re-computing them multiple times while evaluating adjacent patches.

Oriented convex hulls can be used as containers. It is obvious that the complexity of the container influences the performance of this method. Although a precise container can eliminate a maximum number of patches, the number of intersection tests increases with the container's number of vertices and polygons. Experiments have shown that the highest speedups are reached if the container is as simple as an oriented bounding box or a very coarse but tighter convex hull (cf. figure 3.25). However, the complexity of the convex hull that maximizes the speedup and balances intersection tests with patch generations depends on the scene and the required rendering precision.



(a) (b)

Figure 3.25: *Selective grid refinement for different image space error thresholds (a). Spatially limited grids for different perspectives (b).*

2.) Image Space Error

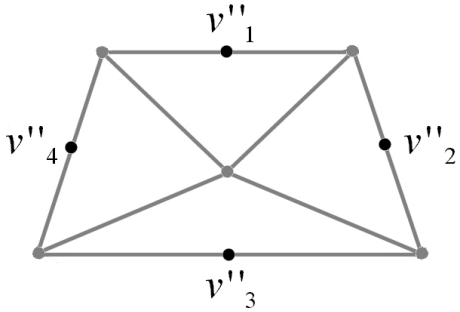
The consideration of the *screen space error* that is determined relative to a display surface is a common measure for many computer graphics methods (e.g., [Hop97, Hop98, Lin96]). In contrast to traditional displays, curved mirror beam combiners create a view-dependent, non-planar image surface which is located inside the mirror optics. Consequently, an appropriate error function has to consider this optical transformation. The error is determined by first warping the curved image surface into an image plane, and then computing the screen space error on this plane. This error is referred to as *image space error* (δ_{is}).

The image space error is a variation of a screen space error that can be computed for convex mirror beam compiners which present the image plane within the image space of their optics, rather than on a display surface. The image space error is defined as the geometric distance between the desired position (v_d) and the actual appearance (v_a) of a point on the image plane. Consequently, the image space error is given by:

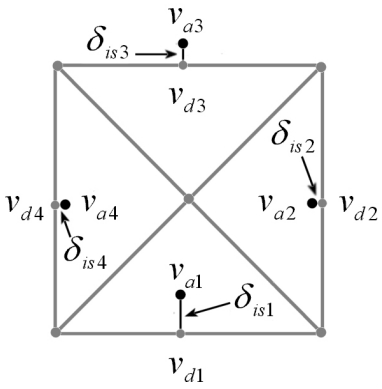
$$\delta_{is} = |v_d - v_a|$$

It delivers results in image space coordinates (e.g., *mm* in this case).

In case of convexly curved mirror optics, the *image space* is the reflection of the *object space* (i.e. the secondary screen in front of the mirror optics) that optically overlays the real environment inside the mirror optics. In addition, the optically deformed pixels do not maintain a uniform size within the image space. They are deformed in exactly the same way as the entire image after being reflected from the object space into the image space - although on a smaller scale. Consequently, the Euclidean distance between geometric points can be chosen as an error metric, rather than expressing the image space error with a uniform pixel size.



(a)



(b)

Figure 3.26: *Samples on the transformed patch (a) The distance between desired and actual appearance of samples near the untransformed patch results in the image space error (b).*

For any given pixel on the transformed patch with texture coordinates *u*,*v*, δ_{is} can be computed as follows (cf. figure 3.26):

First, the pixel's world coordinate v'' at *u*,*v* within the object space (i.e. on the secondary display surface) is determined. Note that the pixels, which are mapped onto the patch's transformed vertices, optically appear at their correct locations on the image plane inside the image space. This is because their exact mappings have been determined during the patch's transformation. This transformation considers the laws of geometric optics (i.e., reflection and refraction laws). Note also that the pixels that are displayed anywhere else (i.e. inside one of a patch's triangle) do not necessarily appear at their correct locations on the image plane. This is because their positions on the display surface are approximated by a *linear texture interpolation*, rather than by optical laws.

The second step is to determine the position of the optical image (v') of v'' within the image space of the mirror optics. The projection of v' onto the image plane results in v_a. The transformation from v'' to v_a will be discussed in detail in part 3.

In an ideal case, v_a is located at the position that also maps to the texture coordinates *u*,*v* within the untransformed patch. We can identify the location which does this as the desired image position v_d. However, if $v_a \neq v_d$, the image space error δ_{is} for this pixel is non-zero.

The image space errors for the four points on the transformed patch (figure 3.26a) can be computed that should map to the patch's T-vertices on the untransformed patch (figure 3.26b) as if the image space errors were zero for these positions. Obviously, this is not the case in the example, shown in figure figure 3.26b.

Since the untransformed patch is a *rectangular quad*, small image space errors suggest that the optical mapping between the transformed and untransformed patch is linear. Furthermore, it can then be concluded that a linear texture interpolation within the displayed transformed patch produces approximately correct results while being mapped (i.e., reflected and refracted) into image space. Consequently, it can be assumed that the resulting image space errors describe a patch's curvilinearity at the representative pixel locations.

To decide whether or not a patch has to be further refined, the largest of the four image space errors is determined. If it is above a pre-defined threshold value $\bar{\delta}_{is}$ the patch has to be further refined and the *RefineFurther* returns *true*.

3.) Computing Object-Image Reflections

To compute v_a from a given viewpoint e, the object point v'' and the optic's geometry is equivalent to finding the *extremal Fermat path* from v'' to e via the optics. In general, this would be a difficult problem of *variational calculus*.

Beside ray- and beam-tracing approaches, several methods have been proposed that approximate reflection on curved mirrors to simulate global illumination phenomena within rendered 3D scenes. All of these methods face the above mentioned problem in one or the other way.

Mitchell and Hanrahan [Mit92], for instance, solve a multidimensional non-linear optimization problem for explicitly defined mirror objects ($g(x) = 0$) with *interval techniques*. To compute reflected illumination from curved mirror surfaces, they seek the *osculation ellipsoid* that is tangent to the mirror surface, whereby its two focal points match the light source and the object point.

For a given viewpoint e, Ofek and Rappoport [Ofe98] spatially subdivide the object space into *truncated tri-pyramid shaped cells*. In contrast to solving an *optimization problem*, they apply *accelerated search techniques* to find the corresponding cell that contains the object v'' at interactive rates.

While Mitchell's *multidimensional optimization* approach is far from being applied at interactive rates, Ofek's search method offers a good approximation for rendering global illumination effects (such as reflections), but does not provide the precision required by an optical display.

In the following, a numerical minimization method to compute the object-image reflection for specific parametric mirror surfaces (such as cones and cylinders) is described. For such surfaces, the optimization problem be reduced to only one dimension. Consequently, the method provides an appropriate precision at interactive rates.

For the subsequent example the same cone-shaped mirror surface is chosen as used for the examples in sections 3.7.1-3.7.8.

Cones and similar *bodies of revolution* have the property that multiple surface points lie on a common plane. For example, all points on the straight line spanned

by a cone's peak and an arbitrary point on its bottom circle lie on the same plane. Consequently, individual plane parameters can be determined for all angles around the cone's principle axis.

To determine an object's (v''_d) image for a given viewpoint e and mirror surface $g(x)=0$, an arbitrary angle α around the cone's principle axis is assumed. Then the surface's tangent plane TP_1 is determined at α by computing a surface point and the surface normal at α. Since $g(x)$ is an explicit function, the surface normal can be computed by using its first-order derivatives $g/(\partial x)$. Next, v''_d is reflected over TP_1 to its corresponding position within the image space and it is projected onto the image plane. In figure 3.27, the projected image point is outlined by v.

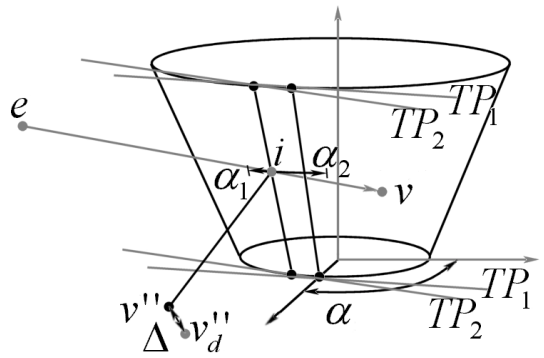


Figure 3.27: *Object-image reflection via nummerical minimization.*

To verify the quality of this assumption, v is reflected back into the object space and project it onto the display surface. For a given v, $g(x)$ and e, a simple *analytical solution* exists to determine this *image-object transformation*: The ray spanned by e and v is intersected with $g(x)$ by solving a simple quadratic equation. In the case of a conical beam combiner, the *quadratic equation* is given by inserting the linear ray equation into the quadratic equation of a cone, and solving for x. The surface intersection i, together with the normal vector at i (again determined using the surface's first-order derivatives) gives the tangent plane TP_2 at i. Reflecting v and e over TP_2 and projecting the reflection of v onto the display surface using the reflection of e as center of projection, results in point v''. The image-object transformation is illustrated in figure 3.16c. Note that for simplicity, the image-object transformation and the *object-image transformation* have been described as simple reflection/projection transformations. Normally, they incorporate refraction as well, as described in section 3.7.3.

If the tangent plane at α produces the *extremal Fermat path* between v''_d and e, the geometric distance Δ between v''_d and v'' is zero, $TP_1 = TP_2$, and $v_a = v$. Otherwise Δ is non-zero.

To approximate the correct α, the above-described function is minimized for Δ. Since this function depends only on α, a fast numerical optimizers for one dimension can be applied. However, because its first order derivatives cannot easily be derived but it appears to be nicely parabolic near its minimum, Brent's inverse parabolic interpolation [Bre73] with bracketing is applied.

To speed up the minimization process (i.e. to reduce the number of function iterations), the function range for a particular v''_d can be constrained. As illustrated in figure 3.27, the minimization is restricted to find α between α_1 and α_2. These boundaries can be determined as follows: Given e and the untransformed patch that belongs to v''_d, the projections on the mirror surface of the untransformed patch at the next lower LOD are evaluated. The two angles α_1 and α_2 at the horizontal extrema of these projections are the corresponding *boundary angles*. Note that these projections are determined while transforming the patches (i.e., within *TransformPatchVertices*). Thus, for efficiency reasons, they are stored and looked up in the memory, rather than re-computing them again.

Experiments showed that sufficiently precise solutions can be found after a small number of iterations. Typically, average errors of $\Delta = 0{,}002mm$ are achieved with an average of 3 iterations.

4.) Error Direction Propagation

Although the number of iterations is relatively small, the computational expenses of four minimization steps per patch result in a noticeable loss of performance. Especially while evaluating the large number of higher LOD patches, such an approach might not produce a speedup.

It can be noticed that the direction in which the highest image space error (i.e. the one of the four patch sides where δ_{is} is maximal) propagates is the same for higher LOD patches that are derived from the same parent patch.

However, from which level of detail on this is the case depends on the display's properties. If, for instance, the optics produce well-behaved image deformations, the propagation direction of the highest error is consistent for relatively high LODs. If, on the other hand, the optics produces noisy images, the error direction alternates as randomly.

To benefit from this situation, we specify a LOD Λ depending on the optic's and the display surface's properties.

For patches that are below Λ, the image space error is determined as described above: δ_{is} is compute at all four edges and the highest value is found. In addition, the error direction (i.e. the edge where δ_{is} is maximal) is recorded for each patch.

For patches that are above Λ the error direction of the corresponding parent patch can be reused and δ_{is} can be computed only for the edge in this direction, rather than at all four edges. By doing this, it is assumed that the largest error will occur in the same direction as for the parent patch. Consequently, the number of minimization steps is reduced from four to one for all patches that are above Λ - i.e., for the majority of all relevant grid patches.

Experiments have shown that this heuristic leads to a significant speedup while producing the same visual final output.

Note that although the examples above deal with a simple mirror shape (i.e., a cone), an algorithm that uses a pre-computed look-up table (e.g., such as the one described by Ofek [Ofe98]) instead of dynamic numerical minimizations would either result in fairly large data-structures that cannot be efficiently searched, or in a precision that is not adequate for an optical display.

3.7.8.4. Display Specific Components

While the described algorithm is valid for other displays that require non-linear pre-distortion, its display specific components have been explained based on a particular mirror display -a cone-shaped beam bombiner. The nature of the additional mirror optics makes the transformation of the grid patches and the computation of the resulting displacement error fairly complex. In fact, the implementation of the *TransformPatchVertices* and the *RefineFurther* functions have been explained with an emphasize on cone-shaped mirror surfaces. These two functions represent the display specific components of our algorithm. While for a mirror display, *TransformPatchVertices* and *RefineFurther* have to consider laws of geometric optics (such as reflection and refraction transformations) to map grid vertices from the image plane onto the projection surface and vice versa, they can be generalized to do the same for other displays without modifying the general algorithm.

If, for instance, the algorithm is used to project images onto curved screens (e.g., a cylindrical or spherical projection device), *TransformPatchVertices* would incorporate only projection transformations (i.e., it would only determine intersections of vertex-individual geometric lines of sight with the display surface). The resulting displacement error that is computed by *RefineFurther* can then be determined within the object space (e.g., the screen space), rather than within an image space. Compared to our numerical approach for convex mirror displays, this would be less complex since it involves only simple intersection computations for which analytical solutions exist. If a view-dependence is not required, *TransformPatchVertices* and *RefineFurther* could also retrieve pre-computed values from a look-up table.

For curved mirror beam combiners that require a correction of a non-linear distortion by applying multi-pass rendering, generating appropriate local levels of detail instead of applying uniform grids or constant image geometries allows to consider the error that is caused from a piecewise linear texture interpolation and to minimize it by adapting the underlying geometry.

The method described above prevents oversampling and texture artifacts that result from undersampling. If executed on CPUs, a significant speed up can be achieved by rendering a selectively refined image geometry instead of a high-resolution uniform image grid. However, the capability of per-vertex operations of today's GPUs beats the computational overhad that is required for recusions and minimizations with a faster transformation of high-resolution uniform image geometries. This is described in sections 3.7.2 nd 3.7.3.

3.8. Summary and Discussion

The basic principles of geometric optics represent the mathematical, physical and physiological foundations for this chapter. Planar mirror beam combiners that produce true stigmatic reflections between all object-image pairs, and convexly curved mirror beam combiners that represent non-absolute optical systems represent the main components for the creation of optical overlays in combination with spatial secondary screens.

If such beam combiners are integrated into the optical path between observer and secondary screen, the perceived graphical images are transformed and distorted by the optical elements. We have discussed real-time rendering techniques that are capable of neutralizing these effects for different display configurations in such a way that orthoscopic, stereoscopically and perspectively correct and undistorted images can be seen from any viewing position. Transparent primary screens, such as active or passive screens which can be utilized for spatial optical combination have been outlined briefly. Their rendering techniques are the same as for planar opaque projection screens.

Mirror beam combiners that consist of a half-silvered mirror layer coated or impregnated onto or into a transparent base carrier represent a combination of lens and mirror. Consequently, reflection transformation and refraction distortion have to be considered.

Since planar mirrors are absolute optical systems, affine model and viewpoint transformations can be applied for such elements. These transformations can be integrated into traditional fixed function rendering pipelines and consequently can be fully supported by hardware acceleration without causing additional computational costs.

In contrast to this, elements that cause significant refraction and warped reflection distortion require curvilinear image transformations. A multi-pass image-warping method avoids a direct access to the scene geometry, and consequently prevents the time-consuming transformations of many scene vertices. In addition, it is not restricted to a geometry-based first rendering pass, but rather supports any perspective image generating rendering method (such as point-based rendering, image-based rendering, interactive raytracing and radiosity, or non-photorealisitc rendering). It can be easily configured and extended to support new optics and display technology, and it is flexible enough to be smoothly integrated into existing software frameworks. Additionally, it takes as much advantage of hardware implemented rendering concepts (such as vertex-shaders, and render-to-texture capability) as currently possible. Such optics also cause aberrations of non-stigmatic mappings that cannot be corrected in software. However, due to the limited retinal resolution of the eyes small aberrations are not detected.

To efficiently support simultaneous rendering for multiple viewers on cost-effective rendering platforms (e.g., PCs), a networked cluster of rendering nodes could be applied. Each node would carry out the image generation and deformation for a single observer. However, in terms of displaying the final images on arbitrary locations of a single projection screen, they need to be composed within one image. One possibility for doing this is to utilize coding and streaming technology to transmit the images to a central node that is responsible for their composition and the presentation of the resulting image. However, the transmission of the images over an

all-purpose network (e.g., an Ethernet) as well as their explicit composition might represent bottlenecks that slow down the overall rendering performance significantly. Recent advances in hardware *image composition technology* (such as [Stol01]) solve this problem. Instead of connecting the rendering nodes via an all-purpose network, they can be connected to a specific display subsystem. These subsystems allow the image data generated by the connected nodes to be mapped to any location of the connected displays without losing much time for transmission and composition. Some devices (such as the Lighning-2 device [Stol01]) are able to scale in both dimensions -the number of rendering nodes and the number of displays. Such technology also supports to efficiently drive high-resolution displays that are built from a collection of multiple display units.

In general several advantages of spatial optical combination over head-attached optical see-through exist:

- *Easier eye accommodation and vergence*: using mirror beam combiners, for instance, the reflected image plane can be brought very close the physical location that has to be overlaid. Consequently multiple focal planes do not differ much. In addition, the reflected image remains at a static position in space if neither mirror combiner nor secondary screen is not moved. This position is invariant of the user's location (note: the presented content is not). Moving closer to or further away from the physical location and the reflected image yields the same natural accommodation and vergence effects. This is not the case for head-attached displays that present the image at a constant distance in front of the observer. In this case the image's position and it's focal plane do not chance for a moving user. Consequently multiple focal planes (i.e., in the physical world and on the image plane) make it impossible to focus at real and virtual content at the same time. Either real or virtual environment have to be perceived unfocused. This requires that the observer continuously switches focus;
- *High and scalable resolution*: spatial screens, such as projection display or monitors provide a high resolution. If, however, a single display does not offer the resolution that is required for a particular application, multiple image sources (such as video beamer) can be combined. Packing issues that are related to the miniature displays which have to be integrated into the limited space of for

head-attached devices are not a problem for spatial configurations. In addition, the compression characteristics of convex mirror that map a larger screen portion into a smaller reflection area causes a high density of pixels. Consequently the spatial resolution of such a display is higher than the resolution of the secondary screen;

- *Large and scalable field of view*: the same argumentation as for the resolution applies to the field of view. Screens and optics are theoretically not constrained in their dimensions. Spatial surround screen display, for instance, can fill out the entire field of view – a characteristic that cannot be achieved with today's head-attached devices;
- *Improved ergonomic factors*: today's head-attached displays suffer from an unbalanced ratio between cumbersome and high quality optics. Spatial displays do not require the user to wear much technology. While stereoscopic displays imply the application light-weight passive or active glasses, autostereoscopic displays detach any kind of technology form the user;
- *Easier and more stable calibration*: while some head-attached displays can have up to 12 degrees of freedom, spatial optical see-through configurations generally have significantly less. This implies an easier and user/session independent calibration, and consequently yields more precise overlays;
- *Better controllable environment*: a limited indoor space can be easier controlled than large scale outdoor environments. This applies to tracking and illumination conditions, and leads to higher precision and consistency, as well as to better quality.

Beside these advantages, several disadvantages can be found with respect to head-attached approaches:

- *Mutual occlusion*: Optical beam combiners cannot present mutual occlusion of real and virtual environments. Consequently, bright real surfaces overlaid with dark virtual objects optically emphasize the real environment and let the virtual objects disappear or appear semi-transparent. A general solution to this problem for spatial optical see-through displays is presented in chapter 4;
- *Window violation*: the window violation or clipping problem is linked to fish-tank-sized and semi-immersive screens. Graphics that –due to a disadvantageous user position– is projected outside the display area is unnaturally cropped. This effect also appears with the mirror beam combiners: displayed images that –due to a disadvantageous user position– cannot be reflected are cropped at he mirror edges as well.
- *Multi-user viewing*: the number of observers that can be supported simultaneously is restricted by the applied optics and screen configuration;
- *Support mobile of applications*: Spatial displays do not support mobile applications because of the spatially aligned optics and display technology;
- *Direct interaction*: In most cases, the applied optics prevents a direct manipulative interaction. Real objects might not be touchable because they are out of arm reach or because the additional optics represents a physical barrier. Indirect interaction methods have to be applied in these cases.

In general, it has to be annotated that upcoming and new technology will not only open new possibilities for the spatial optical see-through concept, but also for other display concepts, such as head-attached displays. *Organic Light Emitting Diodes* (OLEDs), for instance, may replace the crystalline LEDs that are currently being used to build the miniature displays for HMDs.

OLEDs promise to produce cheap and very high-resolution full-color matrix displays that can give head-attached displays a technological push. A variant of OLEDs are *Light Emitting Polymers* (LEPs) that provide the opportunity for the fabrication of large, flexible, full-color emissive displays with a high resolution, a wide viewing angle and a high durability. Large-scale transparent LEPs may present a future variation of optical combination with an active screen. However, LEPs have not yet left the basic research stages and will not be applicable to build stereoscopic AR displays in the near future. But in combination with autostereoscopic or holographic techniques this problem may be solved.

In the short run, especially high-resolution and bright display devices, high-performance and cost-efficient rendering hardware, reliable and precise tracking technology, and advanced interaction techniques and devices will pave the way for forthcoming spatial optical see-through configurations. However, the underlying technology must be robust, flexible and the technology that directly interfaces to users should adapt to humans, rather than forcing users to adapt to the technology. Therefore, human-centered and seamless technologies, devices and techniques will play a major role for augmented reality in future.

References

[All03] Allard, J., Gouranton, V., Lamarque, G., Melin, E., and Raffin, B. Softgenlock: active stereo and GenLock for PC clusters. In proceedings of Immersive Projection Technology and Eurographics Virtual Environments Workshop 2003 (IPT/EGVE'03), pp. 255-260, 2003.

[Azu97] Azuma, R. T., A Survey of Augmented Reality. Presence: Teleoperators and Virtual Environments, vol. 6, no. 4, pp. 355-385, 1997.

[Bim00] Bimber, O., Encarnação, L.M., and Schmalstieg, D. Augmented Reality with Back-Projection Systems using Transflective Surfaces. Computer Graphics Forum (proceedings of EUROGRAPHICS 2000 - EG'2000), vol. 19, no. 3, pp.161-168, 2000.

[Bim01a] Bimber, O., Fröhlich, B., Schmalstieg, D., and Encarnação, L.M. The Virtual Showcase. IEEE Computer Graphics & Applications, vol. 21, no.6, pp. 48-55, 2001.

[Bim01b] Bimber, O., Encarnação, L.M. and Branco, P. The Extended Virtual Table: An Optical Extension for Table-Like Projection Systems. Presence: Teleoperators and Virtual Environments, vol.10, no. 6, pp. 613-631, 2001.

[Bim02] Bimber, O. Interactive rendering for Projection-Based Augmented Reality Displays. Ph.D. Dissertation, University of Technology Darmstadt, http://elib.tu-darmstadt.de/diss/000270/, 2002.

[Bim03] Bimber, O., Fröhlich, B., Schmalstieg, D., and Encarnação, L.M. Real-Time View-Dependent Image Warping to correct Non-Linear Distortion for Curved Virtual Showcase Displays. To appear in Computers and Graphics - The international Journal of Systems and Applications in Computer Graphics, vol. 27, no. 4, 2003.

[Bre73] Brent, R.P. Algorithms for minimization without derivatives. Prentice-Hall, Engelwood Cliffs, NJ, 1973.

[Hec84] Heckbert, P. and Hanrahan, P. Beam tracing polygonal objects. Compute Graphics (proceedings of SIGGRAPH'84), vol. 18, no. 3, pp. 119-127, 1984.

[Hop96] Hoppe H. Progressive meshes. Computer Graphics (Proceedings of SIGGRAPH'96), New Orleans, LA, p. 99–108, 1996.

[Hop97] Hoppe H. View-dependent re
progressive meshes. Computer Graphics (Proceedings of SIGGRAPH'97), Los Angeles, CA, p. 189–97, 1997.

[Hop98] Hoppe H. Smooth view-dependent level-of-detail control and its application to terrain rendering. Proceedings of IEEE Visualization'98, Research Triangle Park, NC, p. 35–42, 1998.

[Lin96] Lindstrom P, Koller D, Ribarsky W, Hughes L, Faust N, Turner G. Realtime, continuous level of detail rendering for height
(Proceedings of SIGGRAPH'96), New Orleans, LA, p. 109–18,1996.

[Mck99a] McKay, S., Mason, S., Mair, L. S., Waddell, P., and Fraser, M. Membrane Mirror-Based Display For Viewing 2D and 3D Images. In proceedings of SPIE, vol. 3634, pp. 144-155, 1999.

[Mit92] Mitchell, D. and Hanrahan, P. Illumination from Curved Reflectors. Computer Graphics (proceedings of SIGGRAPH'92), pp. 283-291, 1992.

[Moe97] Möller, T., and Trumbore, B. Fast, Minimum Storage Ray-Triangle Intersection. Journal of graphics tools. vol. 2, no. 1, pp. 21-28, 1997.

[Mck99b] McKay, S., Mason, S., Mair, L. S., Waddell, P., and Fraser, M. Stereoscopic Display using a 1.2-M Diameter Stretchable Membrane Mirror. In proceedings of SPIE, vol. 3639, pp. 122-131, 1999.

[Ofe98] Ofek, E. and Rappoport A. Interactive reflections on curved objects. Computer Graphics (proceedings of SIGGRAPH'98), pp. 333-342, 1998.

[Ofe99] Ofek. E. Interactive Rendering of View-Dependent Global Lighting Phenomena. Ph.D. Dissertation, Hebrew University (Israel), 1999.

[Rol93] Rolland, J.P., and Hopkins, T. A Method of Computational Correction for Optical Distortion in Head-Mounted Displays. (Tech. Rep. No. TR93-045). UNC Chapel Hill, Department of Computer Science, 1993.

[Seg92] Segal, M., Korobkin, C., van Widenfelt, R. Foran, J., and Haeberli, P. Fast Shaddows and Lighting Effects Using Texture Mapping. Computer Graphics (proceedings of SIGGRAPH'92), vol. 26, no. 2, pp. 249-252, 1992.

[Spa92] Spanguolo, M. Polyhedral surface decomposition based on curvature analysis. Modern Geometric Computing for Visualization, Springer Verlag, 1992.

[Stol01] Stoll, G., Eldridge, M., Patterson, D., Webb, A., Berman, S., Levy, R., Caywood, C., Taveira, M., Hunt, S., Hanrahan, P. Lightning-2: A high-performance subsystem for PC clusters. Computer Graphics (proceedings of SIGGRAPH'01), 2001, pp. 141-148.

[Wat95] Watson, B., and Hodges, L. Using Texture Maps to Correct for Optical Distortion in Head-Mounted Displays. In proceedings of IEEE VRAIS'95, pp. 172-178, 1995.

[Wei99] Wiegand, T. E., von Schloerb, D. W., and Sachtler, W. L. Virtual Workbench: Near-Field Virtual Environment System with Applications. Presence: Teleoperators and Virtual Environments, vol. 8, no. 5, pp. 492-519, 1999.

4. Projector-Based Illumination and Augmentation

Their increasing capabilities and declining cost make video projectors become widespread and established presentation tools. Being able to generate images that are larger than the actual display device virtually anywhere is an interesting feature for many applications that cannot be provided by desktop screens. Several research groups discover this potential by applying projectors in unconventional ways to develop new and innovative information displays that go beyond simple screen presentations.

The *Luminous Room* [Und99] for instance, describes an early concept for providing graphical display and interaction at each of an interior architecture space's surface. Co-located two-way optical transducers –called *I/O bulbs*– that consist of projector-camera pairs capture the user interactions and display the corresponding output. With the *Everywhere Displays projector* [Pin01] this concept has been extended technically by allowing a steerable projection using a pan/tilt mirror. Recenty, it was demonstrates how context-aware hand-held projectors –so-called *iLamps*– can be used as mobile information displays and interaction devices [Ras03].

Another concept called *Shader Lamps* [Ras01] approaches to lift the visual properties of neutral diffuse objects that serve as projection screen. The computed radiance at a point of a non-trivial physical surface is mimicked by changing the BRDF and illuminating the point appropriately with projector pixels. Animating the projected images allows creating the perception of motion without physical displacement of the real object [Ras02]. This type of spatial augmentation is also possible for large, human-sized environments, as demonstrated in [Low01].

Projector-based illumination has become an effective technique in augmented reality to achieve *consistent occlusion* [Nod99, Bim02a] and *illumination* [Bim03] effects between real artifacts and optically overlaid graphics. Video projectors instead of simple light bulbs are used to illuminated physical objects with arbitrary diffuse reflectance. The per-pixel illumination is controllable and can be synchronized with the rendering of the graphical overlays. It also makes the combination of high-quality *optical holograms* with interactive graphical elements possible [Bim04a]. Using a video projector to produce a controlled reference wave allows reconstructing the hologram's object wave partially – not at those portions that are overlaid by integrated graphical elements.

New optical combiners together with real-time color correction methods allow to effectively *superimposing arbitrarily textured surfaces*, such as paintings [Bim04b]. This is enabled by thin, transparent film materials that

diffuse a fraction of the light projected onto them. Such a film can be seamlessly integrated into the frame that holds the artwork. Any kind of visual information, such as animations or interactive graphical elements can be overlaid in correct colors.

A virtual retouching technique that applies video projectors for reconstructing and enhancing the faded colors of paintings by projecting light directly onto the canvas is described in [Yos03]. An affine correlation between projection and the result captured by a camera is established manually for each pixel. Users are then able to retouch the original painting interactively via a desktop GUI.

4.1. Projector-based augmentation

In projector-based augmentation, the user's physical environment is augmented with images that are integrated directly in the user's environment, not simply in their visual field [Ras98c, Ras99]. For example, the images could be projected onto real objects using digital light projectors, or embedded directly in the environment with flat panel displays. For the purpose of this discussion we will focus on projector-based augmented reality. While the approach has certain restrictions, it offers an interesting new method to realizing compelling illusions of virtual objects coexisting with the real world. The images could appear in 2D, aligned on a flat display surface, or they could be 3D and floating above a planar or even a non-planar surface. In the most basic applications, projector-based augmentation combines the benefits of traditional spatially immersive displays and augmented reality displays.

The idea of projector-based augmentation is best used when virtual objects are close to the physical objects on which they are displayed. For example, an architect can augment a tabletop scaled model of a house or building using a projector. She can start with a very simple neutral colored cardboard model and its geometric CAD representation. Then it is easy to add virtual objects such as doors, windows, chimneys. She can also visualize underground water pipes or support structure inside the building. A compelling example of projector-based augmentation is the application aimed at walk-through of virtual human-sized environments built by Kok-lim Low et. al. in the Being There project at UNC [Low01]. Instead of building an exact detailed physical replica for projection, the display is made of simplified versions. For example, primary structures of building interiors and mid-sized architectural objects (walls, columns, cupboards, tables, etc.), can usually be approximated with simple components (boxes, cylinders, etc.). As seen in the Figure 2.1.5, the display is made of construction Styrofoam blocks. The main architectural features that match the simplified physical model retain 3D auto-stereo, but the other details must be presented by projecting view-dependent images. Nevertheless, the experiment to simulate a building interior is convincing and provides a stronger sense of immersion when compared to spatual immersive displays, as the user is allowed to really walk around in the virtual environment. However, strategic placement of projectors to allow complete illumination and avoiding user shadows is critical.

4.1.1. Shader Lamps

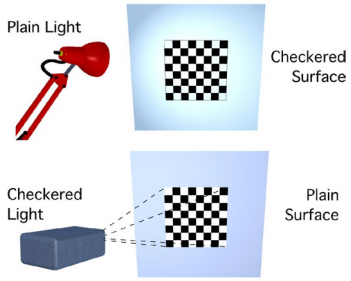


Figure 4.1: *The idea of ShaderLamps*

In this section, we describe a special case of projector-based augmentation. The idea is to replace a physical object with its inherent color, texture, and material properties with a neutral object and projected imagery, reproducing the original appearance directly on the object. Furthermore the projected imagery can be used to reproduce alternative appearances, including alternate shading, lighting, and even animation. The approach is to effectively lift the visual properties of the object into the projector and then re-project onto a neutral surface (Figure 4.1). We use the phrase shader lamps to describe this mode of operation for projectors [Ras01]. Consider the effect shown in Figure 4.2. The underlying physical object is a white diffuse vase. (The other objects such as the book and flowers are also real objects.) Can we make this white vase appear to look like it is made of marble, plastic or metal? Can we change the color or texture? The pictures show that the vase can be effectively 'painted' by projecting an image with view-independent diffuse shading, textures and intensity correction. The view-dependent effects such as specular highlights are generated for a given user location by modifying reflectance properties of the graphics model. The figure shows appearance of a red plastic and a green metallic material on the clay vase. Although, there have

been other attempts at augmenting appearances of objects by projecting color or texture, those effects are very limited and have been achieved for only specific applications. In this section, we show that the real challenges to realizing this as a new medium for computer graphics lies in addressing the problems related to complete illumination of non-trivial physical objects. The approach presented here offers a compelling method of visualization for a variety of applications including dynamic mechanical and architectural models, animated or living dioramas, artistic effects, entertainment, and even general visualization for problems that have meaningful physical shape representations. We present and demonstrate methods for using multiple shader lamps to animate physical objects of varying complexity, from a flower vase (Figure 4.2), to some wooden blocks, to a model of the Taj Mahal (Figure 2.16).

Figure 4.2: *(Left) The underlying physical object is a white diffuse vase. (Middle and right) View-dependent effects, such as specular highlights, can be generated by tracking the user's location and projecting images on the vase.*

4.1.1.1. Illumination Process

We introduce the idea of rearranging the terms in the relationship between illumination and reflectance to reproduce equivalent radiance at a surface. As shown in flatland in Figure 4.3, the radiance in a certain direction at point (x), which has a given BRDF in the physical world (left), can be mimicked by changing the BRDF and illuminating the point with a appropriately chosen light source, e.g. a projector pixel (right). Below we identify a radiance adjustment equation for determining the necessary intensity of a projector pixel, given the position and orientation of the viewer and the virtual scene. For a more systematic rendering scheme, we describe the notion of separating the *rendering view*—the traditional virtual camera view, from the *shading view*—the position of the viewer for lighting calculations.

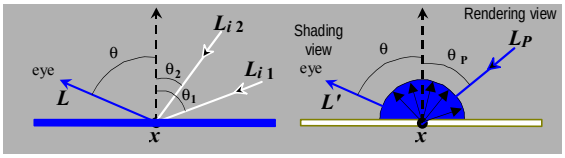


Figure 4.3: *(Left) The radiance at a point in the direction (θ, ϕ). (Right) The radiance as a result of illumination from a projector lamp. By rearranging the parameters in the optical path, the two can be made equal.*

First, let us consider the rendering equation, which is essentially a geometrical optics approximation as explained in [Kaj86]. The radiance at a visible surface point (x) in the direction (θ,ϕ) that would reach the observer of a physical realization of the scene is

$$L(x,\theta,\phi) = g(x,\theta,\phi)(L_e(x,\theta,\phi) + h(x,\theta,\phi))$$

where

$$h(x,\theta,\phi) = \int_i F_r(x,\theta,\phi,\theta_i,\phi_i) L_i(x,\theta_i,\phi_i)\cos(\theta_i) d\omega_i$$

and $g(x,\theta,\phi)$ is the geometry term (visibility and distance), $L_e(x,\theta,\phi)$ is the emitted radiance at the point (non-zero only for light sources), and $F_r(x,\theta,\phi,\theta_i,\phi_i)$ is the BRDF of the point. The integral in $h(x,\theta,\phi)$ accounts for all reflection of incident radiance $L_i(x,\theta_i,\phi_i)$ from solid angles $d\omega_i$. Radiance has dimensions of energy per unit time, area and solid angle.

Treating the projector lamp as a point emitter, the radiance due to direct projector illumination at the same surface point at distance $d(x)$ but with diffuse reflectance $k_u(x)$ is given by

$$L'(x,\theta,\phi) = g(x,\theta,\phi) k_u(x) I_p(x,\theta_p,\phi_p)\cos(\theta_p)/d(x)^2$$

where $I_p(x,\theta_p,\phi_p)$ = radiant intensity of projector in the direction (θ_p,ϕ_p) and is related to a discretized pixel value via filtering and tone representation.

We can reproduce radiance $L'(x,\theta,\phi)$ equivalent to $L(x,\theta,\phi)$ for a given viewer location, by solving Equation (3) for I_p:

$$I_p(x,\theta_p,\phi_p) = \frac{L(x,\theta,\phi)d(x)^2}{k_u(x)\cos(\theta_p)} \quad \text{for } k_u(x) > 0.$$

Thus, as long as the diffuse reflectance $k_u(x)$ is nonzero for all the wavelengths represented in $L(x,\theta,\phi)$, we can effectively represent the surface attribute with appropriate pixel intensities. In practice, however, the range of values we can display are limited by the brightness, dynamic range and pixel resolution of the projector.

The rendering process here involves two viewpoints: the user's and the projector's. A simple approach would be to first render the image as seen by the user, which is represented by $L(x,\theta,\phi)$, and then use traditional image-based rendering techniques to warp this image to generate the intensity-corrected projected image, represented by $I_p(x,\theta_p,\phi_p)$ [Che93, McM95]. For a changing viewer location, view-dependent shading under static lighting conditions can also be implemented [Deb98, Lev96, Gortler96]. However, the warping can be avoided in the case where the display medium is the same as the virtual object. For a single-pass rendering, we treat the moving user's viewpoint as the shading view. Then, the image synthesis process involves rendering the scene from the projector's view, by using a perspective projection matrix that matches the projector's intrinsic and extrinsic parameters, followed by radiance adjustment. The separation of the two views offers an interesting topic of study. For example, for a static projector, the visibility and view-independent shading calculations can be performed just once even when the user's viewpoint is changing.

To realize a real-time interactive implementation we use conventional 3D rendering APIs, which only approximate the general rendering equation. The BRDF computation is divided into view-dependent specular, and view-independent diffuse and ambient components. View-independent shading calculations can be performed by assuming the rendering and shading view are the same. (The virtual shadows, also view-independent, are computed using the traditional two-pass shadow-buffer technique.) For view-dependent shading, such as specular highlights (Figure 4.3), however, there is no existing support to separate the two views. A note in the appendix describes the required modification.

4.1.1.2. Secondary Scattering

Shader lamps are limited in the type of surface attributes that can be reproduced. In addition, since we are using neutral surfaces with (presumed) diffuse characteristics, secondary scattering is unavoidable and can potentially affect the quality of the results. When the underlying virtual object is purely diffuse, sometimes the secondary scattering can be used to our advantage. The geometric relationships, also known as form factors, among parts of the physical objects, are naturally the same as those among parts of the virtual object. Consider the radiosity solution for a patch *i* in a virtual scene with *m* light sources and *n* patches:

$$B_{i\text{-intended}} = k_{d_i}\sum_j B_j F_{i,j} = k_{d_i}\left(\sum_m B_m F_{i,m} + \sum_n B_n F_{i,n}\right)$$

Here k_d is the diffuse reflectance, B_j is the radiance of patch j, and $F_{i,j}$ is the form factor between patches. Using shader lamps to reproduce simply the effect of direct illumination (after radiance adjustment), we are able to generate the effect of *m* light sources:

$$B_{i\text{-direct}} = k_{d_i}\sum_m B_m F_{i,m}.$$

However, due to secondary scattering, if the neutral surfaces have diffuse reflectance k_u, the perceived radiance also includes the secondary scattering due to the *n* patches, and that gives us

$$B_{i\text{-actual}} = B_{i\text{-direct}} + B_{i\text{-secondary}} = k_{d_i}\sum_m B_m F_{i,m} + k_u\sum_n B_n F_{i,n}.$$

The difference between the desired and perceived radiance is

$$\left|(k_{d_i} - k_u)\sum_n B_n F_{i,n}\right|.$$

Thus, in scenarios where k_d and k_u are similar, we get approximate radiosity for "free"—projection of even a simple direct illumination rendering produces believable "spilling" of colors on neighboring parts of the physical objects. From the equation above, the secondary contribution from the neutral surfaces is certainly not accurate, even if we reproduce the first bounce exactly. The difference is even larger when the virtual object has non-lambertian reflectance properties. We are currently investigating *inverse global illumination* methods so that the projected image can more accurately deliver the desired global illumination effect. Figure 4.4 shows a green and a white paper with spill over from natural white and projected green illumination. In this special case, the secondary scattering off the horizontal white surface below is similar for both parts.

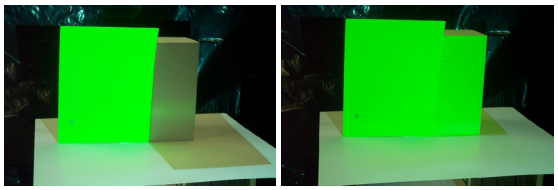


Figure 4.4: *(Left) A green paper illuminated with white light. (Right) The white diffuse surface on the right is illuminated with green light.*

4.1.2. Changing surface appearance of non-trivial projection screens

The image-based illumination of physical objects has been explored by many. But, we believe, two main challenges have kept the previous efforts to only expensive, large scale, or one-off implementations. (a) First, the geometric registration problem, which is cast as matching the projection of a single 2D image with an object. The projection of a perspective device has up to 11 degrees of freedom (6 external and 5 internal) [Fau93], therefore, any effort to manually achieve the registration is likely to be extremely tedious. We propose a new simple technique below. (b) The second problem, which appears to be unexplored, is the complete illumination of non-trivial physical objects in presence of shadows due to self occlusion. With the advent of digitally-fed projectors and real-time 3D graphics rendering, a new approach for image-based illumination is now possible. We approach these problems by creating a 3D geometric understanding of the display setup. We describe an important intensity correction step and our solution for dealing with shadows.

4.1.2.1. Authoring and Alignment

One of the important tasks in achieving compelling visualization is to create the association between the physical objects and the graphics primitives that will enhance those objects when projected. For example, how do we specify which texture image should be used for the face of a building model, or what color distribution will look better for a physical object? We need the physical object as well as its geometric 3D representation, and real or desired surface attributes. As mentioned earlier, many hardware and software solutions are now available to scan/print 3D objects and capture/create highly detailed, textured graphics models. The authoring can also be done interactively by "painting" directly on top of the physical objects. The result of the user interaction can be projected on the objects and also stored on the computer. Ideally, a more sophisticated user interface would be used to create and edit graphics primitives of different shape, color and texture.

To align a projector, first we *approximately* position the projector and then *adapt* to its geometric relationship with respect to the physical object. That relationship is computed by finding projector's intrinsic parameters and the rigid transformation between the two coordinate systems. This is a classical computer vision problem [Fau93]. As seen in Figure 4.5, we take a set of fiducials with known 3D locations on the physical object and find the corresponding projector pixels that illuminate them. This allows us to compute a 3×4 perspective projection matrix up to scale, which is decomposed to find the intrinsic and the extrinsic parameters of the projector. The rendering process uses the same internal and external parameters, so that the projected images are registered with the physical objects.

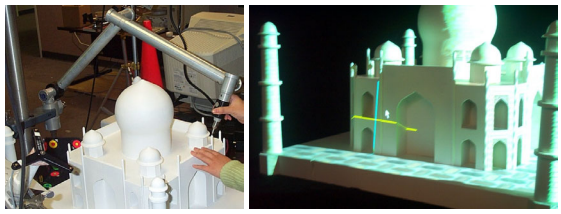


Figure 4.5: *(Left) We use a 3D touch probe scanner to create a 3D model of the real object. (Right) The projectors are calibrated with respect to the model by finding which pixels (center of cross) illuminate the known 3D fiducials.*

4.1.2.2. Intensity Correction

The intensity of the rendered image is modified on a per-pixel basis to take into account the reflectance of the neutral surface, the local orientation and distance with respect to the projector using the intensity equation (see section 4.1.1.1). Since the surface normals used to compute the $1/\cos(\theta_P)$ correction are available only at the vertices in polygonal graphics models, we exploit the rendering pipeline for approximate interpolation. We illuminate a white diffuse version of the graphics model (or a model matching appropriate $k_u(x)$ of the physical model) with a virtual white light placed at the location of the projector lamp and render it with squared distance attenuation. The resultant intensities are smooth across curved surfaces due to shading interpolation and inversely proportional to $(d(x)^2/k_u(x)\cos(\theta_P))$ factor. To use the limited dynamic range of the projectors more efficiently, we do not illuminate surfaces with $\theta_P > 60$ (since $1/\cos(\theta)$ ranges from 2 to infinity). This avoids the low sampling rate of the

projected pixels on oblique surfaces and also minimizes the misregistration artifacts due to any errors in geometric calibration. During the calculations to find the overlap regions (described below), highly oblique surfaces are considered not to be illuminated by that projector.

```
During pre-processing:
    Create 3D graphics model, G, of
    physical object
    Create 3D graphics model, B, of
    background
    Approximately position the projector
    Find perspective pose, P, of the
    projector wrt the physical object

During run-time:
    Get user location, U
    Get animation transformation, T
    Modify G's surface attributes
    Render G using the pose P, and user
    location U
    Transform B using T^-1, B'
    Render B' using the pose P, and user
    location U
    Modify image intensity to compensate
    for surface orientation
```

Let is look at the steps in detail. The user's location, U, can be tracked using magnetic or optical tracking technology. To reproduce purely view-independent surface appearance (diffuse reflectance), user location is not required. For view-dependent effects such as specular highlights, approximate user location is necessary. We may assume the user is at a sweet-spot and need not track the user. The projector projection matrix, P, is obtained using an off-line calibration process similar to the technique used for finding internal and external parameters of a camera [Fau93]. We take a set of fiducials with known 3D locations on the physical object and find the corresponding projector pixels that illuminate them. This allows us to compute a 3x4 perspective projection matrix up to scale, which is decomposed to find the internal and the external parameters of the projector. The rendering process uses the same internal and external parameters, so that the projected images are registered with the physical objects. During run-time, instead of the object, G, the background, B, is transformed to create the apparent motion. At each frame, an intensity correction stage pre-multiplies the projected image with intensity weights that compensate for the local surface orientation. Otherwise, surfaces normal to the incident light will appear brighter than surfaces illuminated obliquely due to the cosine fall-off.

4.1.2.3. Complete illumination of complex 3D shapes

For complete illumination that avoids shadows due to self-occlusion, using additional projectors is an obvious choice. This leads to the more difficult problem of seamlessly merging images from multiple projectors. A naïve solution may involve letting only a single projector illuminate any given surface patch. But, there are two main issues when dealing with overlapping CRT, LCD or DLP projectors, which compel the use of feathering (or cross-fading) of intensities. The first is the lack of color equivalence between neighboring projectors [Maj00], due to manufacturing process and temperature color drift during their use. The second is our desire to minimize the sensitivity to small errors in the estimated geometric calibration parameters or mechanical variations.

Feathering is commonly used to generate seamless panoramic photomosaics by combining several views from a single location [Sze97]. Similar techniques are exploited in multi-projector wide-field-of-view displays [Pan, Ras99], and two-dimensional arrays of flat projections. In such cases, the overlap region is typically a (well-defined) contiguous region on the display surface as well as in each projector's frame buffer. In the algorithm used in [Sze97, Ras99] the intensity of a pixel is weighted proportional to the Euclidean distance to the nearest boundary (zero contribution) pixel of the (projected) image. The per-pixel weights are in the range [0, 1]. They are multiplied to the pixel intensities in the final rendered image. The pixels weights near the boundary of a source image are near zero and the pixels contribute very little, so that there is a smooth transition to the next source image. This leads to the commonly seen intensity roll-off as shown in Figure 4.6(a). Under ideal conditions and assuming color equivalence, the weight contribution of both projectors A+B adds up to 1. Even when projector B's color response is different than that of A (say, attenuated—shown as B'), the resultant A+B' (shown in blue) transitions smoothly in the overlap region.

This weight assignment strategy works well only when the target image illuminates a smooth continuous surface at and around the overlap. In our case, the physical model is usually made up of non-convex objects or a collection of disjoint objects resulting in shadows, fragmented overlap

regions and, more importantly, overlap regions containing surfaces with depth discontinuities, as shown in Figure 4.6(c) with a simple occluder. Now, with unequal color response, the resultant weight distribution $A+B'$ has offending sharp changes, e.g. at points f and g. This situation is analogous to image-based rendering (IBR), where warping a single depth-enhanced image creates dis-occlusion artifacts. When multiple source images are warped to the target image, the color assigned to a pixel needs to be derived (from either a single image where they overwrite each other or) as a weighted combination of corresponding pixels from source images. The feathering, which actually blurs the result, is usually necessary to overcome (minor) color difference in corresponding pixels in input images and to hide ghosting effects (due to small mis-registration errors). One of the few solutions to this is proposed by [Deb98], in which they scale the intensities by weights proportional to the angles between the target view and the source views. As mentioned in their paper, "it does not guarantee that the weights will transition smoothly across surfaces of the scene. As a result, seams can appear in the renderings where neighboring polygons are rendered with very different combinations of images.

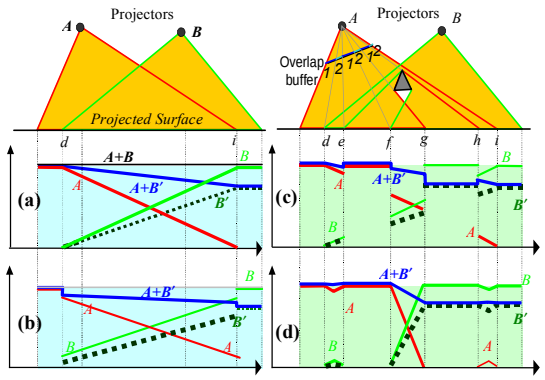


Figure 4.6: *Intensity weights using feathering methods. The plots show the contribution of projectors A, B and B′ and the resultant accumulation A+B and A+B′ along the lit planar surface. Our technique, shown in (d), creates smooth weight transitions.*

The plots in Figure 4.6(b) show a sample weighting scheme based on a similar idea and the corresponding problems. Below, we present a global solution using a new feathering algorithm that suits IBR as well as shader lamps. The algorithm is based on the following guidelines:

1. The sum of the intensity weights of the corresponding projector pixels is one so that the intensities are normalized;
2. The weights for pixels of a projector along a physical surface change smoothly in and near overlaps so that the inter-projector color differences do not create visible discontinuity in displayed images; and
3. The distribution of intensity weights for a projector within its framebuffer is smooth so that small errors in calibration or mechanical variations do not result in sharp edges.

In practice, it is easier to achieve (or maintain) precise geometric calibration than to ensure color equality among a set of projectors over a period of time [Maj00]. This makes condition 2 more important than 3. But, it is not always possible to satisfy condition 2 or 3 (e.g. if the occluder moves closer to the plane so that $f = g$ in figure 4.6) and hence they remain as guidelines rather than rules.

The three guidelines suggest solving the feathering problem, without violating the weight constraints at depth discontinuities and shadow boundaries. Traditional feathering methods use the distance to the nearest boundary pixel to find the weight [Sze97, Ras99]. Instead, we first find pixels corresponding to regions illuminated by a single projector and assign them an intensity weight of 1. Then, for each remaining pixel, the basic idea behind our technique is to find the shortest Euclidean distance to a pixel with weight 1, ignoring paths that cross depth discontinuities. The assigned weight is inversely proportional to this distance. Figure 4.6(d) shows the result of the new feathering algorithm in flatland for two projectors. Even under different color responses, the algorithm generates smooth transitions (see $A+B'$) on the planar surface in presence of shadows and fragmented overlaps. The algorithm can be used for 3 or more projectors without modification.

For a practical implementation, we use two buffers—an overlap buffer and a depth buffer. The depth buffer is updated by rendering the graphics model. The overlap buffer contains integer values to indicate the number of overlapping projectors for each pixel. The overlap regions (i.e. overlap count of two or more) are computed using the traditional shadow-buffer technique. The algorithm follows:

```
At each projector:

    Compute boundaries between regions of
    overlap count 1 and  > 1

    Compute depth discontinuities using
    edge detection in depth buffer
```

```
        For each pixel in overlap region:

            update shortest distance to
            overlap count==1 region ignoring
            paths crossing depth discontinuity

    At each projector:

        For each pixel in overlap region

        Find all corresponding pixels in
        other projectors

        Assign weights inversely proportional
        to the shortest distance
```

For some pixels in the overlap region, such as region [*h,i*] for projector *A*, no nearest pixel with overlap count of 1 can be found, and so the shortest distance is set to a large value. This elegantly reduces the weight in isolated regions and also cuts down unnecessary transition zones.

4.1.3. Enhancing moving objects

We can illuminate objects so that the surface textures appear glued to the objects even as they move. In this case, we can display updated specular highlights even for a static viewer. For example, in showroom windows or on exhibition floors, one can show a rotating model of the product in changing colors or with different features enhanced. In an experimental system, a tracked "paintbrush" was used to paint on a tracked moving cuboid held by the user (Figure 4.7). The presence of the physical model allows natural haptic feedback. The need to attach a tracker and dynamic mis-registration due to tracker latency are the two main problems [Ban01].

Figure 4.7: *A tracked "paintbrush" painting on a tracked cuboid.*

4.1.3.1. Vehicle simulation, computer aided engineering and design

Let us consider the problem of creating the perception of motion without corresponding physical displacement in space. This perception of motion is known as *apparent motion*. Here we describe it in the context of simulation of motion for a car.

The motion analysis by humans is divided into three levels of processing: retinal motion sensing, 2D integration and 3D interpretation. The need to sense retinal motion, and analyze it as quickly as possible, places great demands on the visual system. A great deal of perceptual and physiological research has been conducted to discover the properties of these mechanisms. With several (retinal and other) cues, some potentially conflicting, the visual system attempts to integrate into a meaningful 'best' solution, which may be actually incorrect [Mat98]. The resulting illusions provide us an opportunity.

Using 3D computer graphics, we exploit the induced motion effect and the 3D interpretation error for some very constrained cases. For induced motion, we segment a continuous static physical scene into sub-parts illuminated with images that have consistent rate of relative displacement. We also create temporary and invalid 3D interpretation using shadows, lighting and texture movements. We describe a system to show some limited effects on a static car model and present techniques that can be used in similar setups.

The scene is made up of a toy-car model, with a simple horizontal surface and a vertical surface. The horizontal surface coarsely represents the ground surface, either a road or rough terrain. The vertical surface is the backdrop which represents everything else including bushes, trees and sky. The background can be made arbitrarily complex (but must have constant profile along any plane perpendicular to the direction of apparent motion) to improve the visual fidelity.

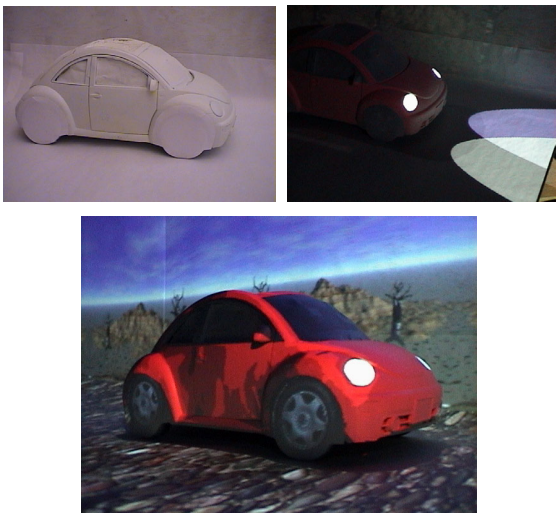


Figure 4.8: *(Top-left) The diorama of a car and approximate background. (Bottom) A cartoon-like environment created by image projection in which the car appears to move (Top-right) A night-time simulation using virtual headlights.*

4.1.3.2. Motion Effects

The car is simulated as driving along a road, on a rough surface, or in various other environments. To create apparent motion, we illuminate wheels with images of rotating wheels. The images of the background (made up of the backdrop and ground) move in a direction opposite to the intended car movement. In the simplest case, as seen in the video, the car appears to move forward i.e. left to right. To create this effect, the wheels rotate clockwise and the background moves right to left (Figure 4.9(i)). The crucial task for a believable movement is maintaining consistency between the angular wheel movement and corresponding translation of the ground (and the backdrop). For any duration,

$$\int \text{wheel perimeter arclength} = \left| \text{displacement of the background} \right|$$

This ensures that the wheels are not sliding while the car is in motion. A small amount of motion blur is added to the wheels. Along the direction of the motion, the background geometry with the associated texture maps is infinitely long, and is implemented using a simple sink and a source.

We experimented with many types of cartoon or non-realistic motions. Two important factors that add to the effect are sound and removal of (physical) sharp edges in the background.

4.1.3.3. Wobbly Motion

Slow moving cartoon cars usually have a wavy or bumpy movement resulting in a small periodic or random displacement along the vertical direction. This sometimes is emphasized by non-circular wheels. In our case, the car is static. Hence, to create the desired apparent vertical shift while the car is in motion, we instead translate the background in the opposite (vertical) direction (Figure 4.9(ii)). During rendering, the 3D model of the backdrop as well as the ground is translated vertically. The amount of translation, in the case of non-circular wheels, is determined by distance between the point of contact of the wheels from wheel axis (Figure 4.9(ii)). The distance traveled is again consistent with integration of the arc length of wheel perimeter. Both types of motions can be seen in the video available on our website. (The video does not do justice and it is difficult to feel the transitions, the effect has to be experienced in person with all the three dimensional cues. Please see the subsection on user reactions.)

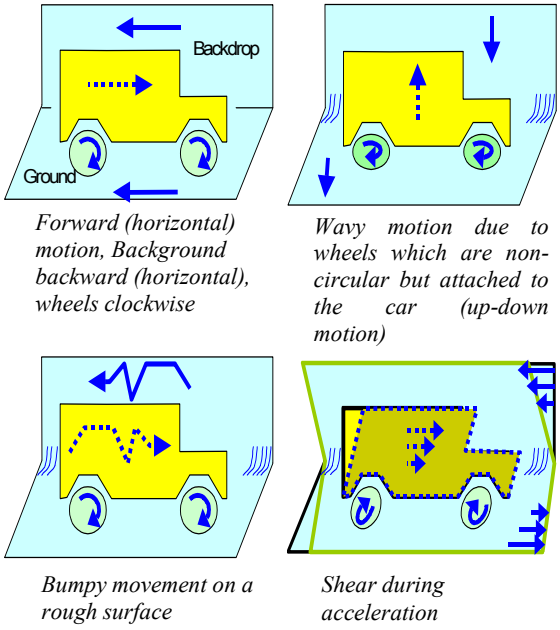


Figure 4.9: *Types of apparent motion for a car. Rendered motion is shown with blue arrow. The resultant apparent motion is shown with dashed blue arrow.*

4.1.3.4. Lights and Shadows

Shadows provide a very important cue in apparent motion. For example, in a two-dimensional image sequence (i.e. without any other depth cues), a ball moving diagonally upwards across the screen can be made to appear as if it is gradually moving away from the viewer by rendering a shadow that appears to stay with and under the ball. The cue overrides contradictory information regarding the unchanging size of the ball. The strength of the effect does depend on the properties of the shadow region. For example, shadows with a penumbra, and which fall below the object, work best.

We enhance the wobbly motion effect by rendering virtual shadows from directional light source (sun) or local lights (street lights). The change in shadow position creates the illusion that the change is a result of changing vertical distance between the car and the ground. We noticed that, it is not necessary to use the same light position to calculate shading and shadows ! Further, perceptually, the movement of virtual shadows (Figure 4.10) is not affected by the fixed real shadows of the physical car on the background.

For night time simulation, the only cues are headlight beams and shading due to the street lights. The spread of the parabolic projection of the headlights and change in overlap between the two beams indicates the vertical shift. The color of the two beams is intentionally chosen slightly different, so that the lateral shift as the car moves up and down is clearly seen. We also exploit spot lights from the street lights (Figure 4.11).

Figure 4.10: *Vertical displacement by shifting the background and shadows. These two pictures are taken from the same camera position, so the car in both images is at the same location. Note the change in position of the virtual shadows and the parallax for the edge between ground and backdrop.*

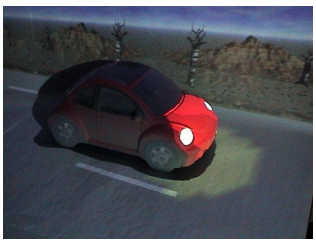


Figure 4.11: *Shading due to street lights as spot lights.*

4.1.3.5. Acceleration dependent modifications

Cartoon animators emphasize acceleration, such as a sudden start or a screeching halt, by geometric deformation of the object. For example, a car appears to 'work hard' to move forward while starting when the top of the car is sheared in the direction of the acceleration. Similarly a hard brake and stop is indicated by shearing the top backwards. Since we cannot shear the physical model, we enhance this effect using two tricks.

First, we implement a shear in the background that is opposite of the shear expected in the car (figure 4.9). The shear is along one dimension, along the vertical axis. Hence, for example, during a sudden start, the background shears backward and the shear at a point is proportional to the vertical distance from the center of the car. The points above the vertical center of the car translate backwards while points below translate forward. Without a loss of generality, lets assume that the car center is at the origin, the vertical direction is parallel to the z-axis, and the forward direction is parallel to the x-axis, Then the shear at a given frame is achieved using a simple transformation matrix [1, 0, -*a*, 0; 0, 1, 0, 0; 0, 0, 1, 0; 0, 0, 0, 1]. Where *a* is proportional to the acceleration at that frame.

Since the acceleration is positive during starting, negative during braking, and zero during constant speed and velocity, the same shear equation can be used throughout the animation.

For the second trick, we observe that rapid acceleration also means relative slide between the ground and the wheels. Hence, for example, a sudden brake results in halt in rotation of the wheels, but the background continues to move (Please see the video available on our website).

Figure 4.12: *Shearing of the background during acceleration*

Figure 4.13: *Photo-realistic diorama (left column) vs. cartoon diorama (right column) appearance based on shading and motion.*

4.1.3.6. Tracked Illumination

We synchronize and update the rendering of the objects so that the surface textures appear glued to the objects even as they move. In this case, we rotate the car (along with the background) on a turntable. Keeping the virtual light sources fixed, we see corresponding shading changes.

4.2. Creating Consistent Occlusion

Spatial augmented reality systems share many positive properties of spatial virtual reality systems. These displays provide high resolution, improved consistency of eye accommodation and convergence, little motion sickness potential, and the possibility of an integration into common working and living environments. One of the main challenges for spatial AR systems as well as for head-mounted optical see-through displays is the generation of *correct occlusion effects* between virtual and real objects.

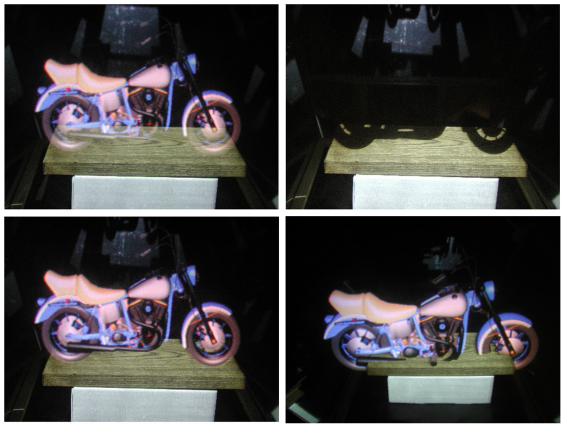


Figure 4.14: *Wrong occlusion effects with normal illumination (upper-left). Occlusion shadow generated with projector-based illumination (upper-right). Realistic occlusion of the real object by the virtual one (lower-left). Knowing depth information of real objects allows the occlusion of virtual objects by real ones (lower-right).*

The basic problem is that in most cases, the existing environment illumination is applied to lighten the physical scenery. However, light that is reflected off the real objects' surface interferes with the optically overlaid graphics – that appear as semi-transparent ghosts which are unrealistically floating in midair (cf. figure 4.14). For indoor situations, the environment illumination can be controlled and synchronized with the rendering process if the simple light bulbs are replaced by video projectors (sometimes referred to as *light projectors*). This concept is called *projector-based illumination*. It requires a physical environment that is initially not illuminated. While this condition can be man-made for many stationary display situations, some setups already provide a dark surrounding by default – like the Virtual Showcase [Bim01].

For optical see-through head-mounted displays, a special optics has been developed by Kiyokawa et al. [Kiy00] that supports *mutual occlusion* – called ELMO. ELMO uses half-silvered mirrors as optical combiners and an additional semi-transparent LCD panel in front of the conventional optics. The LCD panel is used to selectively block the incoming light on a per-pixel basis. This enables virtual objects to occlude real ones. A head-attached depth

sensor allows them to acquire depth maps of the real environment in real time. This makes the occlusion of virtual objects by real ones possible. ELMO faces a number of problems that are linked to the LCD panel: light attenuation caused by the LCD panel, and low response time and resolution of the LCD panel. However, as the first functioning system of its kind, it effectively addresses the occlusion problem of optical see-through head-mounted displays.

Head-Mounted Projective Displays, or HMPDs, (such as described by Hua et al. [Hua01]) require the observer to wear miniature projectors. The projectors beam the synthetic images directly onto the surfaces of the real objects that are within the user's field of view. Since the observer's viewing frustum can be optically matched with the projection frustum, view-dependent rendering is possible while benefiting from a view-independent projection (i.e., depth information for real objects is not required). However, the real objects' surfaces have to be coated with a retro-reflective material in terms of providing stereoscopic rendering, multi-user applications, and the usage of such displays within uncontrolled illuminated environments. The occlusion problem of optical see-through displays is not an issue for HMPDs, since the retro-reflective material avoids the problem of environment light interfering with the graphical overlays.

Video-projectors have been applied to address the occlusion problem for spatial AR configurations: Noda et al. [Nod99], for instance, has presented a stationary optical see-through display that uses a video projector to illuminate real objects selectively – not lighting those areas that are overlaid by graphics. However, view-dependent rendering is not possible in this case. The observer's viewpoint has to match with the center of projection of the video projector since the illumination pattern is rendered from this point using a normal on-axis projection. In this special case no depth information of the real environment is required for a correct rendering. In addition, stereoscopic rendering is not provided. Later, Naemura et al. [Nae02] has proposed an approach that is technically similar to Noda's. The conceptual difference, however, is that he applies a hand-held video projector as a real flashlight to interactively generate shadow effects of virtual objects on real surfaces. He does not address the occlusion problem of optical see-through displays, but focuses on enhancing such interactive mixed reality applications by providing additional visual cues through shadows. As in Noda's case no depth information of the real objects are needed.

In the following sections, general projector-based illumination techniques are described that can be applied in combination with spatial optical see-through AR displays. Off-axis and on-axis situations are treaded in exactly the same way. By using computer-controlled video-projectors as replacements for simple light bulbs, the lighting situation can be fully controlled on a per-pixel basis. This allows producing correct occlusion effects between virtual and real objects by projecting view-dependent shadows – called *occlusion shadows* [Bim02a]- directly onto real objects located behind virtual ones using projector-based illumination (cf. figure 4.14).

4.2.1. Single Users

For rendering occlusion shadows the viewpoints of each user, the intrinsic and extrinsic parameters of each light projector, as well as the virtual and the real scene must be known.

The viewpoints are continuously measured with head-tracking technology, while the projectors' parameters are determined only once during a calibration phase. Virtual objects can be interactively manipulated during runtime.

Knowing the scene and the view transformation lets us compute the perspective projection matrix (V) of the corresponding viewpoint that incorporates the model-view transformation with respect to the scene's origin.

The projectors can be calibrated to a registered geometric representation of the real scene that is registered to its physical counterpart. To do this a semi-manual calibration routine can be applied: The two-dimensional projections of known three-dimensional fiducials can be marked on the projector's image plane. Using these mappings, a numerical minimization method (such as Powell's direction set method [Pre92]) is applied to solve a *perspective N-point problem*. This results in the perspective 4x4 projection matrices P of the projector that incorporates the correct model-view transformation with respect to the scene origin.

If multiple projectors are used, the calibration process has to be repeated for each projector separately.

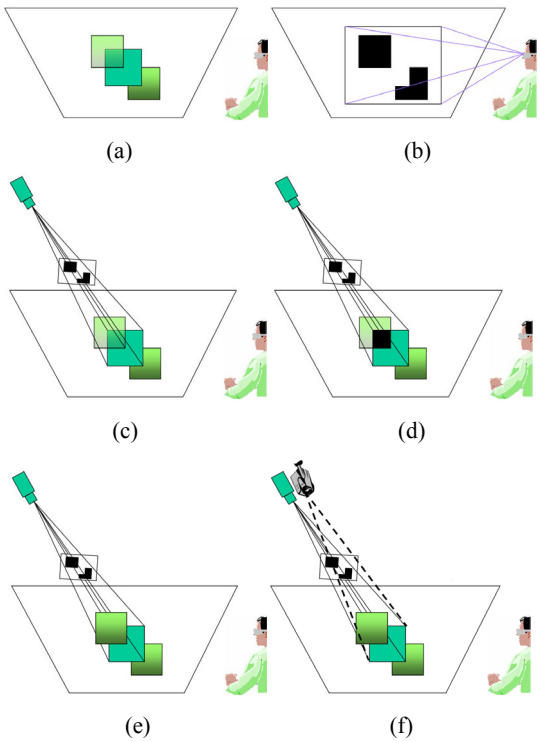


Figure 4.15: *Multi-pass rendering and perspective texture mapping for creating occlusion shadows.*

The basic algorithm below illustrates how to generate occlusion shadows for a single point of view with the aid of *mulit-pass rendering*. The depth information of both – the real and the virtual content have to be known. A *shadow mask* that contains the silhouette of the virtual content is generated in the first pass (cf. figure 4.15a-b) which is then mapped onto the known geometry of the real content in the second pass via *perspective texture mapping* (cf. 4.15c-d). Then the illumination for the real content is rendered into the frame buffer.

For now we want to assume that we project only uniformly colored light onto the real surfaces from the light projector's point of view while virtual objects are illuminated from the positions of the virtual light sources. This illumination, however, could be computed with a more an advanced BRDF model – producing a correct and matching radiance on real and virtual surfaces with respect to virtual light sources. We will describe this in section 4.6.

```
generate shadow mask (first pass):
    set projection matrix to V
    render real content into depth buffer
    render virtual content into stencil
    buffer
    render illumination for real content
    into …
    …frame buffer (previously cleared to
    black)

read-back:
    transfer frame buffer into texture
    memory T

render shadow mask (second pass):
    set projection matrix to P
    set texture matrix to V +
    normalization space correction
    clear frame buffer to black
    render real content into frame buffer
    using …
    …projective texture T
```

Rendering the real content into the depth buffer ensures a correct occlusion of virtual objects by real ones. The normalization space correction consists of a scaling by [0.5,0.5,1.0], followed by a translation of [0.5,0.5,0.5] to map from normalized screen space to normalized texture space*.

Note, that if the graphics card provides a render-to-texture option, the read-back operation from the frame buffer into texture memory can be bypassed.

*This applies for OpenGL-like definitions of the texture and normalized device coordinates.

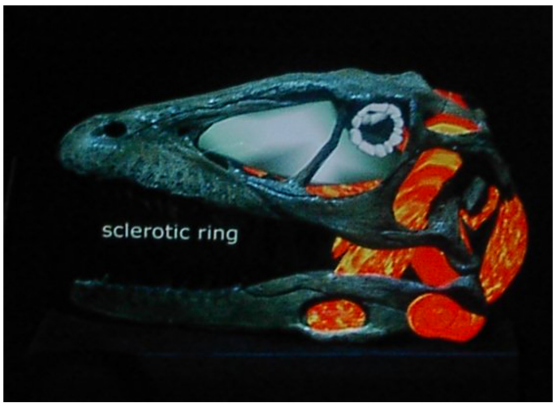


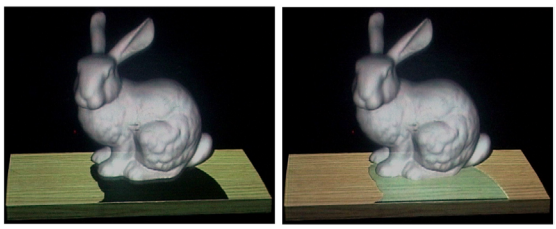
Figure 4.16: *Correct occlusion effects for a complex scene: a real dinosaur with integrated soft-tissue [Bim02b].*

While figure 4.14 illustrates the entire process on a trivial real scene, figure 4.16 shows that the same process is also efficient for complex objects.

4.2.2. Multiple Users

A clear limitation of the basic method described in section 4.2.1 is the following fact: If the same real surfaces are simultaneously visible from multiple points of view (e.g. for different observers), individual occlusion shadows that project onto these surfaces are also visible from different viewpoints at the same time.

Considering two observers, for instance, observer A might be able to see the occlusion shadow that is generated for observer B and vice versa. In addition, the shadows move if the viewers are moving, which might be confusing (cf. figure 4.17a).



(a) (b)

Figure 4.17: *(a) Occlusion shadow of second observer is clearly visible. (b) Wrongly visible occlusion shadow is covered by optically overlaying the corresponding part of the reflectance map.*

One can think of several methods to reduce or avoid these effects:

- *Method I* [Bim02a]: Occlusion shadows generated for other viewpoints are the *umbral hard-shadows* that are cast by the virtual scene with a light source positioned at the other viewpoints' locations. This fact can be made use of by attaching a point light to each viewpoint. This generates correct lighting effects on the virtual scene's surfaces – in addition to matching hard-shadows on the real scene's surfaces.
- *Method II* [Bim02a]: The *interference* between individual occlusion shadows can be *minimized* by ensuring that they are generated only on those real surfaces that are visible from the corresponding viewpoint. However, since the occlusion shadows are finally rendered from the viewpoint of the projector(s), all view-dependent computations (e.g., back-face culling and depth buffering) are done for this perspective – not for the perspectives of the actual viewpoints.
- *Method III* [Bim03]: Each projector is complemented by a video camera that is used to dynamically scan *reflectance* information from the surfaces of the real objects (cf. figure 4.15f). Knowing this reflectance information, however, leads to an effective and general solution: In addition to the virtual scene, the portions of the real scene (i.e., its registered reflectance map) that are covered by the occlusion shadows of all other observers are rendered. If this is done well, a seamless transitions between the real and the virtual portions we can create (cf. figure 4.17b). For each observer the occlusion shadows of all other observers are rendered into the stencil buffer first. This is done by rendering the real scene's geometry from each observer's perspective and adding the corresponding occlusion shadows via projective texture mapping. The stencil buffer has to be filled in such a way that the area surrounding the occlusion shadows will be blanked out in the final image. Then the real scene's reflectance map is rendered into the frame buffer (also from the perspective of the observer) and is shaded under the virtual lighting situation. After stenciling has been disabled, the virtual objects can be added to the observer's view. A possibility of obtaining the reflectance map of diffuse real scenes with projector-camera combinations is described in section 4.3.

Due to *self occlusion*, not all portions of the real content can be lit by a single light projector. A solution to this problem is to increase the number of projectors and place them in such a way that the projected light is distributed over the real content. To guarantee a *uniform*

illumination, however, surfaces should not be lit by more than one projector at the same time or with the same intensity. Otherwise, the projected light accumulates on these surfaces and they appear brighter than others. A *crossfeathering* method can be applied that balances the contribution of each projector equally.

4.3. Creating Consisting Illumination

To achieve a *consistent lighting* situation between real and virtual environments is important for convincing augmented reality (AR) applications.

A rich pallet of algorithms and techniques have been developed that match illumination for video- or image-based augmented reality. However, very little work has been done in this area for optical see-through AR.

Inspired by the pioneering work of Nakamae et al. [Nak01] and –later– Fournier et al. [For93], many researchers have approached to create consistent illumination effects while integrating synthetic objects into a real environment. Most of these approaches represent the real environment in form of images or videos. Consequently, mainly image processing, inverse rendering, inverse global illumination, image-based and photo-realistic rendering techniques are applied to solve this problem. Due to the lack of real-time processing, these approaches are only applicable in combination with desktop screens and an unresponsive* user interaction. Devices that require interactive frame-rates, such as head-tracked personal or spatial displays, cannot be supported. Representative for the large body of literature that exists in this area, several more recent achievements can be highlighted:

Boivin et al. [Boi01] present an interactive and hierarchical algorithm for reflectance recovery from a single image. They assume that the geometric model of the scenery and the lighting conditions within the image are known. Making assumptions about the scene's photometric model, a virtual image is generated with global illumination techniques (i.e., ray-tracing and radiosity). This synthetic image is then compared to the photograph and a photometric error is estimated. If this error is too large, their algorithm will use a more complex BRDF model (step by step – using diffuse, specular, isotropic, and finally anisotropic terms) in the following iterations, until the deviation between synthetic image and photograph is satisfactory. Once the reflectance of the real scenery is recovered, virtual objects can be integrated and the scene must be re-rendered. They report that the analysis and re-rendering of the sample images takes between 30 minutes to several hours – depending on the quality required and the scene's complexity.

Yu et al. [Yu99] present a robust iterative approach that uses global illumination and inverse global illumination techniques. They estimate diffuse and specular reflectance, as well as radiance and irradiance from a sparse set of photographs and the given geometry model of the real scenery. Their method is applied to the insertion of virtual objects, the modification of illumination conditions and to the re-rendering of the scenery from novel viewpoints. As for Boivin's approach, BRDF recovery and re-rendering are not supported at interactive frame-rates.

Loscos et al. [Los00] estimate only the diffuse reflectance from a set of photographs with different but controlled real world illumination conditions. They are able to insert and remove real and virtual objects and shadows, and to modify the lighting conditions. To provide an interactive manipulation of the scenery, they separate the calculation of the direct and indirect illumination. While the direct illumination is computed on a per-pixel basis, indirect illumination is generated with a hierarchical radiosity system that is optimized for dynamic updates [Dre97]. While the reflectance analysis is done during an offline preprocessing step, interactive frame-rates can be achieved during re-rendering. Depending on the performed task and the complexity of the scenery, they report re-rendering times for their examples between 1-3 seconds on a SGI R10000. Although these results are quite remarkable, the update rates are still too low to satisfy the high response requirements of stereoscopic displays that support head-tracking (and possibly multiple users).

Gibson and Murta [Gib00] present another interactive image composition method to merge synthetic objects into a single background photograph of a real environment. A geometric model of the real scenery is also assumed to be known. In contrast to the techniques described above, their approach does not consider global illumination effects to benefit from hardware accelerated multi-pass rendering. Consequently, a reflectance analysis of the real surfaces is not required, indirect illumination effects are ignored, and a modification of the lighting conditions is not possible. The illumination of the real environment is first captured in form of an omni-directional image. Then a series of high dynamic basis radiance maps are pre-computed. They are used during runtime to simulate a matching direct illumination of the synthetic objects using sphere mapping. Shadow casting between real and virtual objects is approximated with standard shadow mapping. With their method, convincing images can be rendered at frame rates

* Not real-time.

up to 10fps on an SGI Onyx 2. However, it is restricted to a static viewpoint.

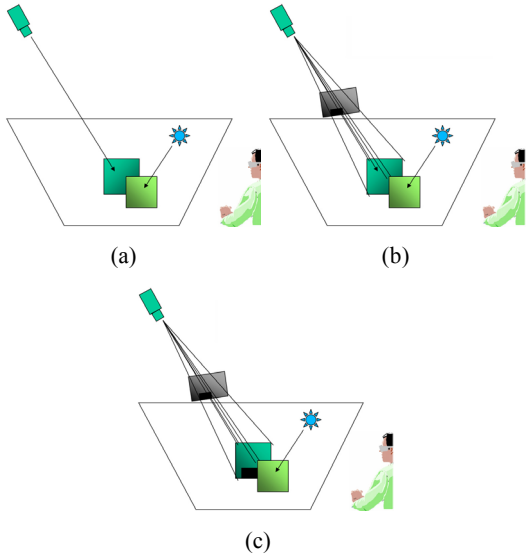


Figure 4.18: *Multi-pass rendering and perspective texture mapping for creating occlusion shadows.*

The main problem of a consistent illumination for optical see-through approaches is that the real environment is illuminated by physical light sources while the virtual environment is illuminated based on synthetic light sources (cf. figure 4.18a). This results in inconsistent shading and shadow effects unless the virtual light sources approximate the properties (such as position, direction, intensities, color, etc.) of the physical ones. This, however, is a very inflexible. In contrast to video see-through, the pixel appearance of the real environment in the video image cannot be modified to achieve a matching illumination. Consequently, the physical contribution of the real light sources has to be neutralized to illuminate the entire environment based on the virtual lighting situation (cf. figures 4.18a and 4.18c). This is only possible with controllable physical light sources, such as video projectors.

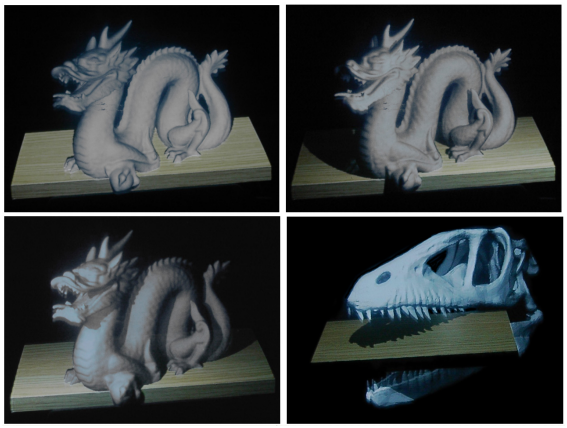


Figure 4.19: *Unrealistic illumination with correct occlusion (upper-left). Realistic illumination under varying virtual lighting conditions with matching shading and shadows (upper-right, lower-left and -right).*

This section describes methods which create a consistent illumination between real and virtual components within an optical see-through environment [Bim03]. Combinations of video projectors and cameras can be applied to capture *reflectance information* from diffuse real objects and to illuminate them under new *synthetic lighting* conditions (cf. figure 4.19). For diffuse objects, the capturing process can also benefit from *hardware acceleration* – supporting dynamic update rates. To handle *indirect lighting* effects (like *color bleeding*) an off-line *radiosity* procedure is outlined that consists of multiple rendering passes. For *direct lighting* effects (such as simple shading, shadows and reflections) hardware accelerated techniques are described which allow to achieve interactive frame rates. The reflectance information can be used in addition to solve multi-user occlusion problem discussed in section 4.2.

4.3.1. Diffuse Reflectance Analysis

As in section 4.2, we want to assume that the geometry of both object types –real and virtual– has been modeled or scanned in advance. While the material properties of virtual objects are also defined during their modeling process, the diffuse reflectance of physical objects is captured on- or off-line with a set of video projectors and cameras and a structured light analysis. This sort of analysis is standard practice for many range scanner setups. But since only diffuse real objects can be considered (a projector-based illumination will generally fail for any specular surface), a

simple diffuse reflectance analysis can benefit from hardware accelerated rendering techniques. In contrast to conventional scanning approaches, this leads to dynamic update rates. Figure 4.20 illustrates an example.

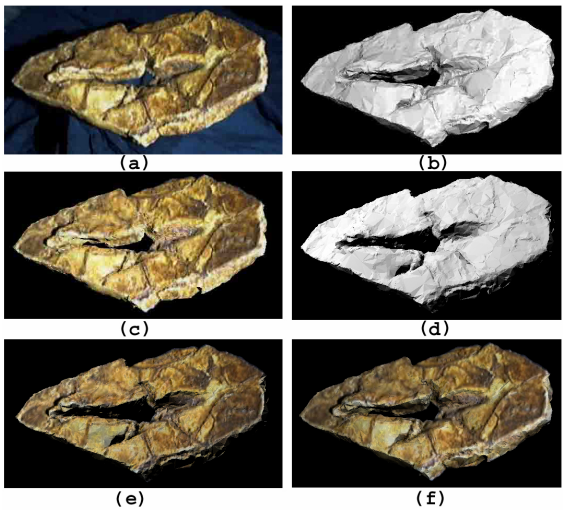


Figure 4.20: *(a) Captured radiance map of a fossilized dinosaur footprint; (b) Intensity image rendered for calibrated projector from (a); (c) Computed reflectance map; (d) Novel illumination situation; (e) Reflectance map under novel illumination from (d); (f) Reflectance map under virtual illumination from (b).*

Again, the intrinsic and extrinsic parameters of projectors and cameras within the world coordinate system have to be estimated first. Each device has to be calibrated separately. As described in section 4.2, two-dimensional projections of known three-dimensional fiducials can be interactively marked on a projector's/camera's image plane. Using these mappings, a minimization method is used to solve a perspective n-point problem for each device. This results in the perspective projection matrices P, C of a projector and a camera. Both matrices incorporate the correct model-view transformations with respect to the origin of the world coordinate system.

Once calibrated, a projector/camera combination can be used to perform the diffuse reflectance analysis as follows:

4.3.1.1. Capturing a Radiance Map

The video projector is used to send structured light samples to the diffuse physical object and illuminate it with a predefined color C_p and an estimated intensity η. Synchronized to the illumination, the video camera captures an input image. Since this image contains the diffuse reflectance of the object's surface under known lighting conditions it represents a *radiance map*. *White-balancing* and other *dynamic correction functions* have been disabled in advance. The parameters of the camera's *response function* can be adjusted manually in such a way that the recorded images approximate the real world situation as close as possible.

Some types of video projectors (such as digital light projectors, DLPs) display a single image within sequential, time-multiplexed light intervals to achieve different intensity levels per color. If such projectors are used, a single snapshot of the illuminated scene would capture only a slice of the entire display period. Consequently, this image would contain incomplete color fragments instead of a full-color image. The width of this slice depends on the exposure time of the camera. To overcome this problem, and to be independent of the camera's *exposure* capabilities, we capture a sequence of images over a predefined period of time. These images are then combined and averaged to create the final diffuse radiance map I_{rad} (cf. figure 4.20a).

4.3.1.2. Creating an Intensity Image

To extract the diffuse material reflectance out of I_{rad} the lighting conditions that have been created by the projector have to be neutralized. OpenGL's diffuse lighting component, for instance, is given by [Nei96]:

$$I_i = \frac{1}{r_i^2}\cos(\theta_i)(D_l D_m)_i$$

where I_i is the final intensity (color) of a vertex i, D_l is the diffuse color of the light, D_m is the diffuse material property, the angle θ_i is spanned by the vertex's normal and the direction vector to the light source, and the factor $1/r_j^2$ represents a square distance attenuation.

Similar as described in section 4.1, an intensity image I_{int} that contains only the diffuse illumination can be created by rendering the object's geometry (with $D_m = 1$) from the perspective of the video camera, illuminated by a

point light source (with $D_l = C_p \eta$) that is virtually located at the position of the projector (cf. figure 4.20b).

In addition, hard shadows can be added to the intensity image by applying standard shadow mapping techniques. Consequently, the background pixels of I_{int}, as well as pixels of regions that are occluded from the perspective of the light source are blanked out ($I_{\text{int}}(x,y) = 0$), while all other pixels are shaded under consideration of the above diffuse lighting equation.

4.3.1.3. Extracting and Re-Rendering Diffuse Reflectance

Given the captured radiance map I_{rad} and the rendered intensity image I_{int}, the diffuse reflectance for each surface that is visible to the camera can be computed by:

$$I_{ref}(x,y) = \frac{I_{rad}(x,y)}{I_{\text{int}}(x,y)} \quad \text{for} \quad I_{\text{int}}(x,y) > 0,$$

$$I_{ref}(x,y) = 0 \quad \text{for} \quad I_{\text{int}}(x,y) = 0$$

Note that the division of the two images can be implemented by a pixel shader, and can consequently be executed in real-time on the graphics hardware.

The reflectance image I_{ref} is stored, together with the matrix C and the real object's world-transformation O_c that is active during the capturing process within the same data-structure. This data structure is referred to as *reflectance map* (cf. figure 4.20c).

The captured reflectance map can be re-rendered together with the real object's geometric representation from any perspective with an arbitrary world-transformation O_a. Thereby, I_{ref} is applied as projective texture map with the texture matrix* set to $O_a^{-1} O_c C$. Enabling texture modulation, it can then be re-lit virtually under novel illumination conditions (cf. figures 4.20d, 4.20e and 4.20f).

The basic diffuse reflectance analysis method as described above faces the following problems:

- due to under-sampling, surfaces which span a large angle ϕ_i between their normal vectors and the direction vectors to the camera can cause texture artifacts if I_{ref} is re-mapped from a different perspective;
- a single reflectance map covers only the surface portion that is visible from the perspective of the camera;
- the radiance map can contain indirect illumination effects caused by light fractions that are diffused off other surfaces (so-called *secondary scattering*). The intensity image I_{int}, however, does not contain secondary scattering effects since a global illumination solution is not computed. Consequently, the extracted reflectance is incorrect at those areas that are indirectly illuminated by secondary scattering;
- the projector intensity η has to be estimated correctly;

To overcome the under-sampling problem, the definition can be made that only surfaces with $\phi_i \leq \phi_{\max}$ are analyzed. All other surfaces will be blanked-out in I_{ref} (i.e., $I_{ref}(x,y) = 0$). Experiments have shown that $\phi_{\max} = 60°$ is an appropriate value.

Multiple reflectance maps that cover different surface portions can be captured under varying transformations O_c or C. They are merged and *alpha blended* during re-mapping them via *multi-texturing* onto the object's geometric representation. This ensures that regions which are blanked out in one reflectance map can be covered by other reflectance maps. To generate seamless transitions between the different texture maps, *bi- or tri-linear texture filtering* can be enabled.

Illuminating the entire scene can cause an extreme secondary scattering of the light. To minimize the appearance of secondary scattering in I_{rad}, the scene can be divided into discrete pieces and their reflectance can be captured one after the other. For this, the same algorithm as described above can be applied. The difference, however, is that only one piece at a time is illuminated and rendered which then appears in I_{rad} and I_{int}. By evaluating the blanked out background information provided in I_{int}, the selected piece we can be effectively segmented in I_{int} and

* Including the corresponding mapping transformation from normalized device coordinates to normalized texture coordinates.

its reflectance can be computed. This is repeated for each front-facing piece, until I_{ref} is complete.

The projector's intensity η can be estimated as follows: First, a reflectance map is generated with an initial guess of η. This reflectance map is then re-mapped onto the object's geometric representation, which is rendered from the perspective of the camera and illuminated by a virtual point light source with η located at the projector. The rendered radiance map I_{radv} is then compared to the captured radiance map I_{rad} by determining the average square distance error Δ among all corresponding pixels. Finally, an approximation for η can be found by minimizing the error function f_Δ. For this, Brent's inverse parabolic minimization method with bracketing [Bre73] can be applied as one possibility. By estimating η, the constant black-level of the projector can also be incorporated.

4.3.2. Augmenting Radiance

In computer graphics, the *radiosity method* [Gor84] is used to approximate a *global illumination solution* by solving an *energy-flow* equation. Indirect illumination effects, such as *secondary scattering* can be simulated with *radiosity*. The general radiosity equation for *n* surface patches is given by:

$$B_i = E_i + \rho_i \sum_{j=1}^{n} B_j F_{ij}$$

where B_i is the radiance of surface i, E_i is the emitted energy per unit area of surface i, ρ_i is the reflectance of surface i, and F_{ij} represents the fraction of energy that is exchanged between surface i and surface j (the *form-factor*).

The simulation of radiosity effects within an optical see-through environment that consists of diffuse physical and virtual objects, is facing the following challenges and problems:

- light energy has to flow between all surfaces – real ones and virtual ones;
- physical objects are illuminated with physical light sources (i.e., video projectors in our case) which do not share the geometric and radiometric properties of the virtual light sources;
- no physical light energy flows from virtual objects to real ones (and vice versa). Consequently, the illuminated physical environment causes (due to the absence of the virtual objects) a different radiometric behavior than the entire environment (i.e., real and virtual objects together).

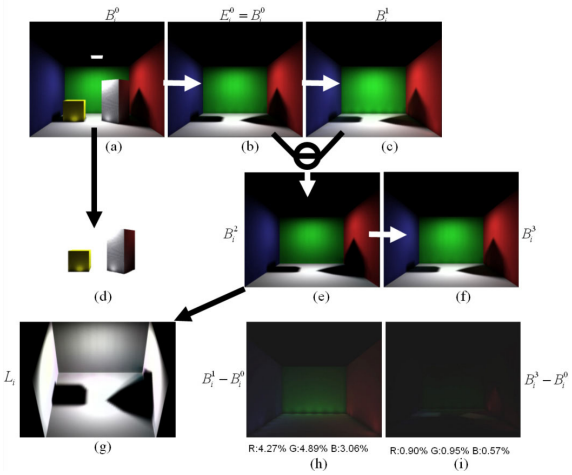


Figure 4.21: *Multi-Pass radiosity for augmenting synthetic radiance onto a real environment.*

An example is illustrated in figure 4.21a*. The entire environment consists of three walls, a floor, two boxes and a surface light source on the ceiling. We want to assume that the walls and the floor are the geometric representations of the physical environment, and the boxes as well as the light source belong to the virtual environment. While the diffuse reflectance ρ_i of the physical environment can be automatically captured (as described in section 4.3.1), it has to be defined for the virtual environment. After a radiosity simulation† of the entire environment the radiance values B_i^0 for all surfaces have been computed‡. *Color-bleeding* and *shadow-casting* are clearly visible.

For virtual objects, the computed radiance values are already correct (cf. figure 4.21d). The rendered image represents a radiance map that is generated from one

* A physical mock-up of the Cornell room has been chosen since it is used in many other examples as a reference to evaluate radiosity techniques.

† A hemi-cube-based radiosity implementation has been applied with progressive refinement, adaptive subdivision and interpolated rendering for our simulations.

‡ Note, that the upper index represents the radiosity pass.

specific perspective. Rendering the virtual objects from multiple perspectives results in multiple radiance maps that can be merged and alpha blended during re-mapping them via multi-texturing onto the virtual geometry (as described for reflectance maps in section 4.3.1). In this case, our radiance maps are equivalent to light maps that are often being applied during pre-lighting steps to speed up the online rendering process.

The pre-lit virtual objects can simply be rendered together with their light maps and can be optically overlaid over the physical environment.

The physical surfaces, however, have to emit the energy that was computed in B_i^0 (cf. figure 4.21b). To approximate this, it can be assumed first that every physical surface patch directly emits an energy E_i^0 that is equivalent to B_i^0. If this is the case, fractions of this energy will radiate to other surfaces and illuminate them in addition. This can be simulated by a second radiosity-pass (cf. figure 4.21c), which computes new reflectance values B_i^1 for all the physical surfaces, by assuming that $E_i^0 = B_i^0$, and not considering the direct influence of the virtual light source.

If we subtract the radiance values that have been computed in both passes we receive the scattered light only. That is, the light energy radiated between the physical surfaces $B_i^1 - B_i^0$ (cf. figure 4.21h).

Consequently,

$$B_i^2 = B_i^0 - \left(B_i^1 - B_i^0\right)$$

approximates the energy that has to be created physically on every real surface patch. To prove this a third radiosity pass can be applied to simulate the energy flow between the patches (cf. figure 4.21f). It can bee see that the remaining energy $B_i^1 - B_i^0$ will be nearly added, and we receive:

$$B_i^3 = B_i^2 + \left(B_i^1 - B_i^0\right) \approx B_i^0$$

By removing the virtual objects from the environment and simulating the second radiosity pass, light energy will also be radiated onto surfaces which were originally blocked or covered by the virtual objects (either completely or partially). Examples are the shadow areas that have been cast by the virtual objects. This can be observed in figure 4.21h and figure 4.21i. Consequently, negative radiance values are possible for such areas. To avoid this, the resulting values have to be clipped to a valid range.

The average deviations between B_i^0 and B_i^1, as well as between B_i^0 and B_i^3, within the three spectral samples red (R), green (G), and blue (B) are presented below figures 4.21h and 4.21i, respectively. Treating a video projector as a point light source B_i^2 can be expressed as follows:

$$B_i^2 = \rho_i L_i F_i$$

where L_i is the irradiance that has to be projected onto surface i by the projector, and F_i is the form-factor for surface i, which is given by:

$$F_i = \frac{\cos(\theta_i)}{r_i^2} h_i$$

where θ_i is the angle between a surface patch's normal and the direction vector to the projector, r_i is the distance between a surface patch and the projector, and h_i is the visibility term of the surface patch, seen from the projector's perspective.

Extending and solving the above equations for L_i, we receive (cf. figure 4.21g):

$$L_i = \frac{B_i^2}{\rho_i F_i} \eta$$

To cope with the individual brightness of a video projector, the intensity factor η can be added. How to estimate η for a specific projector was described in section 4.3.1. To be consistent with our previously used terminology, we call L_i the *irradiance map*.

The computed radiance and irradiance values are view-independent. Consequently, irradiance maps for the real objects and radiance maps for the virtual objects can be pre-computed offline.

The real objects are illuminated with projected light during runtime by rendering the generated irradiance map from the viewpoint of the projector (e.g., as illustrated in figure 4.21g). Virtual objects are rendered with the computed light maps (e.g., as illustrated in figure 4.21d) and are then optically overlaid over the real environment. Due to the view-independence of the method, the augmented scene can be observed from any perspective (i.e., head-tracking and stereoscopic rendering are possible). However, the scene has to remain static, since any modification would require to re-compute new radiance and irradiance maps throughout multiple radiosity passes. This is not yet possible at interactive rates.

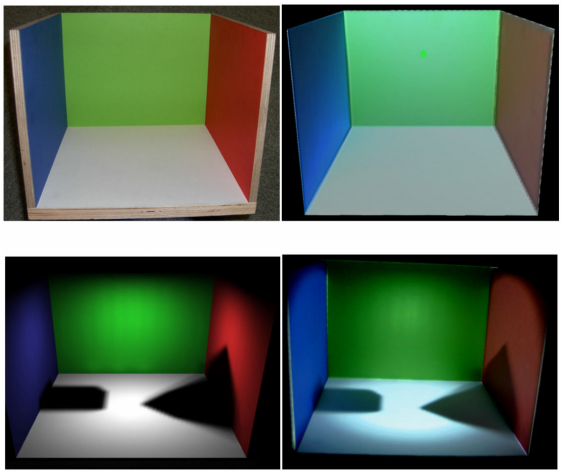


Figure 4.22: *(a) Photograph of original object under room illumination; (b) Screen-shot of captured reflectance re-lit with virtual point light source and Phong shading; (c) Screen-shot of simulated radiosity solution with captured reflectance, virtual surface light source (shown in figure 4.21), and two virtual objects (show in figure 4.21); (d) Photograph of original object illuminated with the computed irradiance.*

Figure 4.22 shows a photograph of (a) the physical object under room illumination, (b) a screen-shot of captured reflectance maps that have been re-rendered under novel lighting conditions, (c) a screen-shot of the simulated radiance situation B_i^0, and (d) a photograph of a physical object that has been illuminated with L_i. Note, that small deviations between the images can be contributed to the responds of the digital camera that was used to take the photograph, as well as to the high black-level of the projector that, for instance, makes it impossible to create completely black shadow areas.

4.3.3. Interactive Approximations: Shading and Shadows

In the following sections several *interactive rendering* methods are described that make use of *hardware acceleration*. In particular how to create matching shading, shadow and reflection effects on real and virtual objects are discussed. Indirect lighting effects such as color-bleeding, however, cannot be created with these techniques. Yet, they create acceptable results at *interactive frame rates* for multiple head-tracked users and stereoscopic viewing on conventional PCs.

The generation of *direct illumination* effects on virtual surfaces caused by virtual light sources is a standard task of today's hardware accelerated computer graphics technology. Real-time algorithms, such as Gouraud shading or Phong shading are often implemented on graphics boards.

Consistent and matching shading effects on real surfaces from virtual light sources can be achieved by using video projectors that project appropriate irradiance maps onto the real objects. In section 4.1 it is shown how to compute an irradiance map to lift the radiance properties of neutral diffuse objects with uniform white surfaces into a pre-computed radiance map of a virtual scene illuminated by virtual light sources. An irradiance map that creates virtual illumination effects on diffuse real objects with *arbitrary reflectance properties* (color and texture) can be computed as follows:

First, the real objects' captured reflectance map (I_{ref}) is rendered from the viewpoint of the projector and is shaded with all virtual light sources in the scene. This results in the radiance map I_{rad_1}. Then I_{ref} is rendered again from the viewpoint of the projector. This time, however, it is illuminated by a single point light source (with $D_l = 1 \cdot \eta$) which is located at the position of the projector. This results in the radiance map I_{rad_2}. Finally, the correct irradiance map is computed by:

$$L = \frac{I_{rad\_1}}{I_{rad\_2}}$$

Note that in general this method correlates to the method described in section 4.3.2. The difference is the applied illumination model. While in section 4.3.2 an indirect global illumination model (radiosity) is used, here a hardware accelerated direct models (such as Phong or Gouraud shading) is applied. It is easy to see that I_{rad_1} is the opponent to B_i^2 and that I_{rad_2} corresponds to $\rho_i F \eta$.

Note also that this method is actually completely independent of the real objects' reflectance. This can be shown by equalizing I_{rad_1} with I_{rad_2}. In this case the diffuse material property D_m (i.e., the reflectance) is canceled out. Consequently, I_{rad_1} and I_{rad_2} can be rendered with a constant (but equal) reflectance (D_m). If $D_m = 1$ is chosen then the irradiance map is simply the

quotient between the two intensity images I_{int_1} and I_{int_2} that result from the two different lighting conditions – the virtual one and the real one.

The irradiance map L should also contain consistent shadow information. How to achieve this is outlined below. Figure 4.19 illustrates examples with matching shading effects*.

Six types of shadows can be identified within an optical see-through environment:

- shadows on real objects created by real objects and real light sources;
- shadows on virtual objects created by virtual objects and virtual light sources;
- shadows on virtual objects created by real objects and virtual light sources;
- shadows on real objects created by real objects and virtual light sources;
- shadows on real objects created by virtual objects and virtual light sources;
- occlusion shadows;

The first type of shadow is the result of occlusions and self-occlusions of the physical environment that is illuminated by a physical light source (e.g., a video projector). Since it is focused on controlling the illumination conditions within the entire environment via virtual light sources these shadows have to be removed. This can be achieved by using multiple synchronized projectors that are able to illuminate all visible real surfaces.

The second and third shadow types can be created with standard *shadow mapping* or *shadow buffering* techniques. To cast shadows from real objects onto virtual ones, the registered geometric representations of the real objects have to be rendered together with the virtual objects when the shadow map is created (i.e., during the first shadow pass). Such geometric real world representations (sometimes called *phantoms* [Bre96]) are often rendered continuously to generate a realistic occlusion of virtual objects by real ones. Note that these hardware accelerated techniques create hard shadows while global illumination methods (such as radiosity) can create soft shadows. Texture blending, however, allows ambient light to be added to the shadow regions. This results in dark shadow regions that are blended with the underlying surface texture, instead of creating unrealistic black shadows.

Shadow types number 4 and 5 can also be created via shadow mapping. However, they are projected on the surface of the real object together with the irradiance map L, as discussed in section 4.1. Therefore, I_{rad_1} has to contain the black (non-blended) shadows of the virtual and the real objects. This is achieved by rendering all virtual objects and all phantoms during the first shadow pass to create a shadow map. During the second pass the shaded reflectance texture and the generated shadow texture are blended and mapped onto the objects' phantoms. A division of the black shadow regions by I_{rad_2} preserves these regions. Note that a blending of the projected shadows with the texture of the real objects occurs physically if the corresponding surface portions are illuminated (e.g., by a relatively small amount of projected ambient light).

Occlusion shadows have been described in section 4.2. They are special view-dependent shadows created by the projectors on the real objects' surfaces to achieve a realistic occlusion of real objects by virtual ones within optical see-through augmented environments. They are normally not visible from the perspective of the observer, since they are displayed exactly underneath the graphical overlays. Occlusion shadow-maps, however, have also to be blended to the irradiance map L before it is projected.

The entire process can be summarized in form of a three-pass rendering algorithm:

```
create an intensity image I_rad_2 of the
real object (first pass):

    render real object from perspective
    of light projector…

    …having a white diffuse material…

    …illuminated by a white virtual point
    light source…

    …located at the projector


create a shading image I_rad_1 of real and
virtual objects (second pass):
```

* Note that a simple wooden plate has been chosen to demonstrate and to compare the different effects. However, all techniques that are explained in these sections can be applied to arbitrary object shapes.

```
generate shadow map for real and
virtual objects

render real objects from perspective
of light projector…

…having a white diffuse material…

…illuminated by the virtual light
sources…

…with shadow mapping enabled
```

compute irradiance image $L = \frac{I_{rad\_1}}{I_{rad\_2}}$

```
render occlusion shadows from perspective
of projector…

…and blend with L (third pass)
```

Note that the implementation of the image division to compute the irradiance map L can be a pixel shader, to achieve real-time performance.

4.3.4. Interactive Approximations: Reflections

Using hardware accelerated *cube mapping* techniques, the virtual representation of the real environment (i.e., the objects' geometry together with the correctly illuminated reflectance map) can be reflected by virtual objects (cf. figure 4.23).

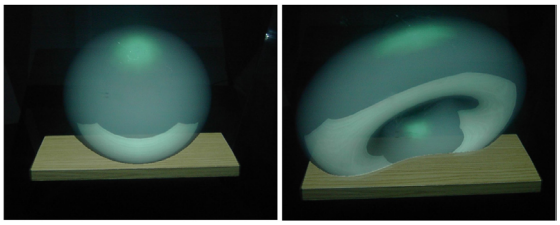


Figure 4.23: *Virtual objects reflecting and occluding the real object (wooden plate).*

Therefore, only the registered virtual representation of the real environment has to be rendered during the generation step of the cube map. Virtual objects are then simply rendered with cube mapping enabled. Note, that for conventional cube mapping, reflection effects on a virtual object are physically correct for only a single point – the center of the cube map frusta. To create convincing approximations this center has to be matched with the virtual object's center of gravity, and the cube map has to be updated every time the scene changes.

4.4. Augmenting Optical Holograms

A hologram is a *photometric emulsion* that records *interference patterns* of *coherent light*. The recording itself stores *amplitude*, *wavelength* and *phase* information of light waves. In contrast to simple photographs (which can record only amplitude and wavelength information), holograms have the ability to reconstruct complete *optical wavefronts*. This results in a three-dimensional appearance of the captured scenery, which is observable from different perspectives.

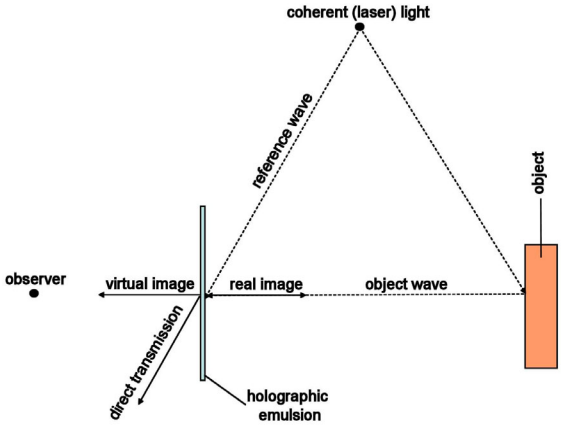


Figure 4.24: *Recording and reconstruction of optical holograms.*

Optical holograms are reconstructed by illuminating them with *monochromatic* (purity of color) light. Thereby the light has to hit the emulsion at the same angle as the reference laser beam that was used to record the hologram.

Figure 4.24 illustrates the basics of optical holographic recording and reconstruction: A laser beam is split into two identical beams. While one beam (called *reference wave*) illuminates the holographic emulsion directly, the other beam illuminates the object to be recorded. The light that is reflected from the object (called the *object wave*) shines on the emulsion and creates the fringe pattern together with the reference wave.

If the emulsion is illuminated with a copy of the reference wave, it interacts with the recorded *interference fringes* and reconstructs the object wave which is visible to

the observer. The amplitude of the reconstructed object wave is proportional to the intensity of the reference wave. Its reconstruction can be controlled with a selective illumination.

There are two basic types of optical holograms: *transmission* and *reflection holograms*.

A transmission hologram is viewed with the light source and the observer on opposite sides of the holographic plate. The light is transmitted through the plate before it reaches the observer's eyes. Portions of the emulsion that are not recorded or not illuminated remain transparent.

Reflection holograms are viewed with the light source and the observer on the same side of the holographic plate. The light is reflected from the plate towards the eyes of the observer. As for transmission holograms, not recorded or not illuminated portions of the emulsion remain transparent (without opaque backing layer).

Among these two basic types of holograms, a large pallet of different variations exists. While some holograms can only be reconstructed with laser light, others can be viewed under white light. They are called *white-light holograms*.

One of the most popular white-light transmission holograms are *rainbow holograms*. With rainbow holograms, each wavelength of the light is diffracted through a different angle. This allows observing the recorded scene from different horizontal viewing positions, but also makes the scene appear in different colors when observed from different viewing positions.

In contrast to rainbow holograms, *white-light reflection holograms* can provide full parallax and display the recorded scene in a consistent –but in most cases monochrome– color for different viewing positions.

Color white-light holograms (both: transmission and reflection types) can also be produced. Usually the same content is recorded on several emulsion layers. However, each layer is exposed to laser light with a different wavelength. When reconstructed, each object wave from each layer contributes with its individual wavelength. Together, they merge into a colored image.

Today, many applications for optical holograms exist. Examples include interferometry, copy protections, data storage and holographic optical elements.

Due to their unique capability to present three-dimensional objects to many observers with almost no loss in visual quality, optical holograms are often being used in museums. They can display artefacts that are not physically present without the need to build real replicas of the originals.

In addition, medical, dental, archaeological and other holographic records can be made – both for teaching and for documentation.

Optical holograms, however, are static and lack in interactivity. Exceptions are *multiplex holograms* that are built from multiple narrow (vertical) strip-holograms that contain recordings of the same scenery at different time intervals. While moving around the hologram (or spinning a cylindrically-shaped version around its principle axis), observers can perceive the recorded scene in motion. Yet, multiplex holograms are not interactive.

Three-dimensional computer graphics in combination with stereoscopic presentation techniques represents an alternative that allows interactivity. State of the art rendering methods and graphics hardware can produce realistically looking images at interactive rates. However, they do not nearly reach the quality and realism of holographic recordings. Autostereoscopic displays allow for a glass-free observation of computer-generated scenes. Several autostereoscopic displays exist that can present multiple perspective views at a time – thus supporting multiple users simultaneously. Resolution and rendering speed, however, decreases with the number of generated views. Holographic images, in contrast, can provide all depth-cues (perspective, binocular disparity, motion parallax, convergence and accommodation) and can be viewed from a theoretically unlimited number of perspectives at the same time.

Parallax displays are display screens (e.g., CRT or LCD displays) that are overlaid with an array of light-directing or light-blocking elements. Using these elements, the emitted light is directed to both eyes differently – allowing them to see individual portions of the displayed image. The observer's visual system interprets corresponding light rays to be emitted by the same spatial point. Dividing the screen space into left-right image portions allows for a glass free separation of stereo pairs into two or more viewing zones.

Some displays control the parallax array mechanically or electronically depending on the viewpoint of the observer to direct the viewing zones more precisely towards the eyes. Others generate many dense viewing zones – each of them showing a slightly different perspective of the rendered scene at the same time. Such displays support multiple users simultaneously, but do not yet allow to reach high frame rates.

Examples of parallax displays are *parallax barrier displays* that apply an array of light-blocking elements

(e.g., a light blocking film or liquid crystal barriers) in front of a screen. The light-blocking elements are used to cover portions of the screen for one eye that are visible from the other eye.

Other examples are *lenticular sheet displays* that utilize refraction of a lens array (e.g., small cylindrical, prism-like or spherical lenses) to direct the light into the different viewing-zones. Images that are generated with lenticular sheet displays appear brighter than the ones displayed on barrier displays. While prisms and cylinders provide a horizontal parallax only, spherical lenses support a full parallax.

Several research groups are working on computer generated holograms. There are two main types: *digital holography* and *electro-holography*.

In digital holography [Yam90], holographic printers are used to sequentially expose small fractions of the photometric emulsion with a computer generated image. Conventional holograms which display a computer-generated content are the result. This technique can also be used for the construction of large-scale, tiled holograms [Klu02]. Although digital holograms can be multiplexed to display scenes in motion, they remain non-interactive.

Electro-holography aims at the computer-based generation and display of holograms in real time [Kol89, Luc97]. Holographic fringes can be computed by rendering multiple perspective images that are combined into a stereogram [Luc95], or by simulating the optical interference and calculating the interference pattern [Luc93]. Once computed, the fringes are dynamically visualized with a holographic display. Since a massive amount of data has to be processed, transmitted and stored to create holograms, today's computer technology still sets the limits of electro-holography. To overcome some of the performance issues, advanced reduction and compression methods have been developed. This results in electro-holograms that are interactive, but small, low-resolution and pure in color. Recent advances on consumer graphics hardware may reveal potential acceleration possibilities [Pet03].

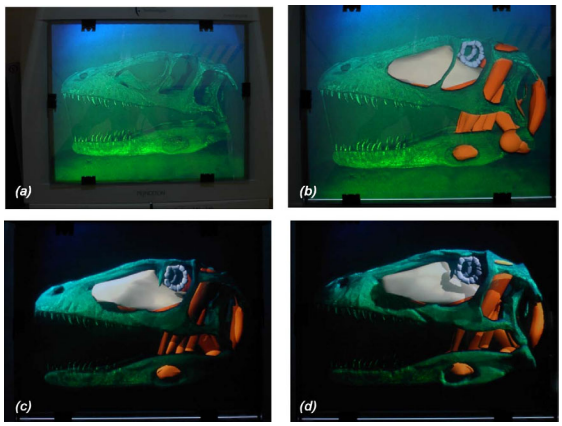


Figure 4.25: *A rainbow hologram of a dinosaur skull combined with three dimensional graphical elements and synthetic occlusion and shading effects. (a) The hologram only. (b) Optically integrated graphical elements (muscles and other soft-tissue). (c and d) Consistent illumination effects between holographic and graphical content.*

Combining optical holograms with 2D or 3D graphical elements can be an acceptable trade-off between quality and interactivity (cf. figure 4.25). While the holographic content provides a high quality but remains static, additional graphical information can be generated, inserted, modified and animated at interactive rates.

Technically, optical combiners such as mirror beam combiners or semi-transparent screens (see chapter 3) can be used to visually overlay the output rendered on a screen over a holographic plate. In this case, however, the reconstructed light of the hologram will interfere with the overlaid light of the rendered graphics and an effective combination is impossible. However, the holographic emulsion can be used as optical combiner itself, since it is transparent if not illuminated in the correct way. Consequently, it provides see-through capabilities.

This section describes how holograms can be optically combined with interactive, stereoscopic or autostereoscopic computer graphics [Bim04a]. While section 4.4.1 explains how correct occlusion effects between holographic and graphical content can be achieved, section 4.4.2 outlines how a consistent illumination is created for such chases.

4.4.1. Partially Reconstructing Wavefronts

A key solution to this problem is to reconstruct the object wave only partially – not at those places where graphical elements have been inserted.

This requires a point light source capable of selectively emitting light in different directions – creating an "incomplete" reference wave. Conventional video projectors are such light sources. They are also well suited for viewing white-light reflection or transmission holograms, since today's high-intensity discharge (HDI) lamps can produce a very bright light.

If autostereoscopic displays, such as parallax displays are used to render 3D graphics registered to the hologram, then both holographic and graphical content appear three-dimensional within the same space. This is also the case if stereoscopic displays with special glasses that separate the stereo images are applied.

Reflection holograms without opaque backing layer and transmission holograms both remain transparent if not illuminated. Thus they can serve as optical combiners themselves – leading to very compact displays. The illumination and rendering techniques that are described in these sections are the same for both hologram types.

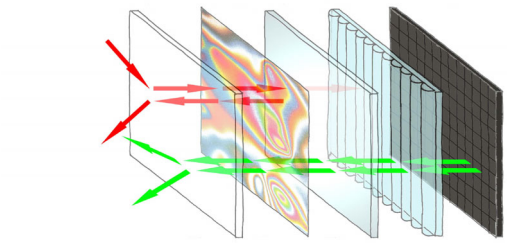


Figure 4.26: *An explosion model of the stacked structure of optical layers. From left to right: glass protection, holographic emulsion, mirror beam combiner (only for transmission holograms), lenticular lens sheet, LCD array. The light-rays (red arrows) are reflected and reconstruct the object wave on their way back through the emulsion. The stereoscopic images (green arrows) pass through all layers until they are merged with the hologram.*

Figure 4.26 illustrates an example of how a transmission hologram can be combined effectively with a flat-panel lenticular lens sheet display (a variation of a parallax display that utilizes refraction of a lens array to direct the light into the different viewing-zones).

Placing a transmission hologram in front of a mirror beam combiner allows to illuminate it from the front and to augment it with graphics from the back. For reflection holograms, this beam combiner is not necessary.

A thin glass plate protects the emulsion from being damaged and keeps it flat to prevent optical distortion. The lenticular lens sheet directs the light emitted from the LCD array through all layers towards the eyes of the observer. The projected light is transmitted through the first two layers, and is partially reflected back (either by the beam combiner in combination with a transmission hologram or by a reflection hologram) – reconstructing the recorded content. The remaining portion of light that is transmitted through all layers is mostly absorbed by the screen.

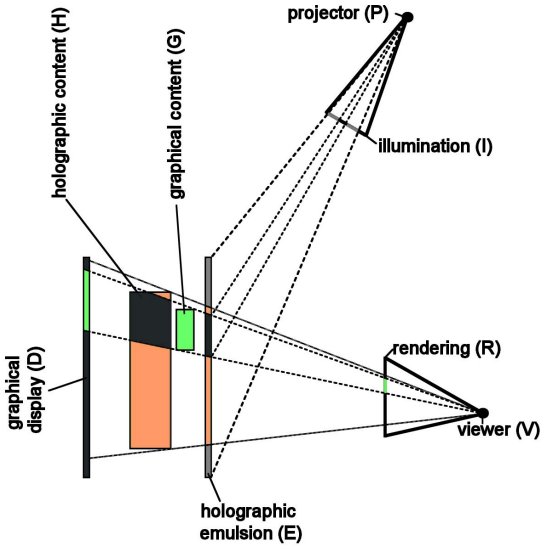


Figure 4.27: *A conceptual sketch of the display constellation. The colored areas on the graphical display and on the holographic emulsion illustrate which portion of the visible image is hologram (red) and which is graphics (green).*

Figure 4.27 illustrates how the selective illumination on the holographic plate is computed to reconstruct the portion of the hologram that is not occluded by graphics.

The following outlines how rendering and illumination can be realized with conventional graphics hardware. It is assumed that depth information of the holographic content (H), as well as a scene description of the graphical content (G) are available. Both contents are geometrically aligned during an offline registered step. If optical markers are recorded in the hologram together with the actual content,

cameras can be used to perform this registration automatically. In addition, the extrinsic and intrinsic parameters of the video projector (P) with respect to the holographic emulsion (E) have to be known. They are also determined during an offline calibration step. If it is mechanically possible to mount the holographic emulsion close to the graphical display (i.e., the distance between E and D is small), then E and D can be approximated to be identical.

First, an intermediate texture image (T) is created from V over E by rendering H into the graphics card's depth buffer and filling the card's frame buffer entirely with predefined light color values. In addition, G is rendered into depth and stencil buffers. The stencilled areas in the frame buffer are cleared in black and the result is copied into the memory block allocated for T.

Note that if a render-to-texture option is provided by the graphics card, the read-back operation from frame buffer into the texture memory is not necessary. The final illumination image (I) is rendered from P by drawing E into the frame buffer and texturing E's geometry with T. The illumination image (I) is beamed onto the holographic emulsion (E) with the projector (P).

Second, a rendering image (R) is generated from V over D (off-axis) by rendering H into the depth buffer, and G into depth and frame buffers. The rendering image (R) is displayed on the graphical display (D).

This can be summarizes with the following algorithm:

```
create intermediate texture T from V over
E (first pass):
    render H into depth buffer
    fill frame buffer with light color
    render G into depth and stencil
    buffer
    fill stenciled areas in frame buffer
    with black

render E from P textured with T and
display on projector (second pass)

create rendering image R from V over D
(optical inlay):
    render H into depth buffer
    render G into depth buffer and frame
    buffer
    display G on projector
```

Figure 4.25a shows a photograph of the entire reconstructed hologram while it is illuminated with a projected uniform light.

By applying the presented techniques, illumination and stereoscopic images are generated in such a way that graphical and holographic content can be merged within the same space (cf. figure 4.25b).

4.4.2. Modifying Amplitude Information

The amplitude of the reconstructed object wave is proportional to the intensity of the reference wave. Beside using an incomplete reference wave for reconstructing a fraction of the hologram, intensity variations of the projected light allow to locally modify the recorded object-wave's amplitude.

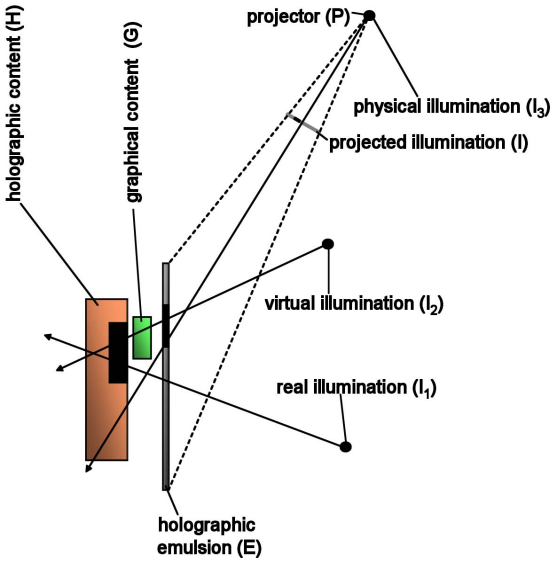


Figure 4.28: *Light-interaction between hologram and graphics: To simulate virtual shading and shadow effects on the holographic content, the recorded and the physical illumination effects have to be neutralized.*

Practically this means that for creating the illumination image (I), shading and shadowing techniques are used to render the holographic content, instead of rendering it with a uniform intensity.

To do this, the shading effects caused by the real light sources that were used for illumination while the hologram was recorded, as well as the physical lighting effects caused by the video projector on the holographic plate have to be neutralized. Then the influence of a synthetic illumination has to be simulated. This can also be done with conventional graphics hardware (cf. figure 4.28). Three intensity images have to be rendered:

For the first image (I_1), H is rendered from V over E with a white diffuse material factor and graphical light sources that generate approximately the same shading and shadow effects on H as the real light sources that were used during the holographic recording process. This results in the intermediate texture T_1. I_1 is generated by rendering E from P and texturing it with T_1. It simulates the intensity of the recorded object wave. The same process is repeated to create the second image (I_2) – but this time graphical light sources are used to shade H under the new, virtual lighting situation. The ratio I_2/I_1 represents the required intensity of the reference wave at holographic plate E.

For the third image (I_3), E is rendered from P with a white diffuse material factor and a virtual point light source located at the projector's position. This intensity image represents the geometric relationship between the video projector as a physical point light source and the holographic plate:

$$F = \frac{\cos(\theta)}{r^2}$$

It contains form factor components, such as the square distance attenuation (r^2) and the angular correlation ($\cos(\theta)$) of the projected light onto the holographic plate and allows neutralizing the physical effects of the projector itself.

The final illumination image (I) can be computed in real time with $I=I_2/I_1/I_3$ via pixel shaders. The projection of I onto E will neutralize the physical and the recorded illumination effects as good as possible, and will create new shadings and shadows based on the virtual illumination. Note that as described previously the graphical content has to be stencilled out in I before displaying it.

During all illumination and rendering steps, hardware-accelerated shadow mapping techniques are used to simulate real and virtual shadow effects on H and on G. Finally, synthetic shadows can be cast correctly from all elements (holographic and graphical) onto all other elements.

The following algorithm summarized this approach:

```
create intensity image I1 (first pass):
     render H from V over E (white diffuse
     factor)…
     …and graphical light sources that
     simulate real…
     …shading on H → T1
     render E from P textured with T1

create intensity image I2 (second pass):
     render H from V over E (white diffuse
     factor)…
     …and graphical light sources that
     simulate virtual…
     …shading on H → T2
     render E from P textured with T1

create intensity image I3 (third pass):
     render E from P (white diffuse
     factor)…
     …and graphical point light source
     attached to P

create and display illumination image I
from P (fourth pass):
     I=I2/I1/I3  (via pixel shader)
```

The capabilities of this technique are clearly limited. It produces acceptable results if the recorded scenery has been illuminated well while making the hologram. Recorded shadows and extreme shading differences cannot be neutralized. Furthermore, recorded color, reflections and higher-order optical effects cannot be cancelled out either.

Projecting an intensity image that contains new shading and shadow effects instead of a uniform illumination allows neutralizing most of the diffuse shading that is recorded in the hologram and produced by the projector. The holographic and graphical content can then be consistently illuminated (creating matching shading and shadow effects) under a novel lighting.

Figures 4.25c and 4.25d illustrate the synthetic shading effects that are caused by a virtual light source. In addition,

virtual shadows are cast correctly between hologram and graphical elements. Although figure 4.25 illustrates the results with a monochrome transmission hologram, the same effects are achieved with reflection holograms and color holograms.

Since all the discussed rendering techniques (like shadow mapping and shading) are supported by hardware-accelerated consumer graphics cards, interactive frame rates are easily achieved.

Using the concept described in this section a pallet of different display variations can be developed. With only minor changes of the presented techniques, for example, arbitrarily curved shapes (such as cylindrical shapes being used for multiplex holograms) can be supported instead of simple planar plates. Even without graphical augmentations, the projector-based illumination alone holds several potentials. In combination with optical or digital holograms it can be used to create visual effects. Certain portions of a hologram, for instance, can be made temporarily invisible while others can be highlighted. Emerging large-scale autostereoscopic displays and existing stereoscopic projection screens allow to up-scale the proposed concept. Not only the display, but also the holograms can be composed from multiple smaller tiles to reach large dimensions and high resolutions.

4.5. Superimposing Pictorial Artwork with Projected Imagery

A seamless and space efficient way for integrating visual information directly into *pictorial artwork* is to use the artwork itself as *information display* [Bim04b]. It can serve as diffuse projection screen and conventional video projectors can be applied to display computer graphics together with the painted content (cf. figure 4.29).

The main difference of this approach to all the methods that are described above is, that an arbitrarily textured surface has to be augmented with colored information. To perceive the projected imagery in the correct colors and intensities, however, requires that the influence of the underlying physical *color pigments* is neutralized. In most situations, this is not possible if untreated images are simply projected directly onto arbitrary colored surfaces. The problem is that the projected light interacts with the color pigments on the canvas and is partially absorbed if the pigment's color isn't fully white.

Figure 4.29: *Results of color correction process with a single projector on real drawing: (a) Real drawing (64x48cm) under environment light. (b) Output image emitted onto drawing. (c) Partially augmented drawing. (d) Output image on a white piece of paper. (e-h) Close-ups. While the upper body part coincides in drawing and painting, Michelangelo modified the lower body part. The arrows indicate the displaced knee and belly sections. They point at the same spot on the drawing.*

A solution to this problem is provided by a new film material which has two properties: first, it is completely transparent and second, it diffuses a fraction of the light that is projected onto it. The film consists of an even deposition of fine particles on both sides of a polyester base with no visible artifacts. It was used for the creation of special effects in Hollywood movies, such as Minority Report (20th Century Fox, 2002) and Paycheck (Paramount Pictures, 2003), and sells for $350 per square foot. Initial measurements have revealed that in average *20%* (*+/- 1%*)

of the light that strikes the film is diffused while the remaining fraction is transmitted towards the canvas (with or without direct contact). This *0.1mm* thin *transparent film* can be seamlessly overlaid over the canvas by integrating it into the frame that holds the artwork. Off-the-shelf 1100 ANSI lumen XGA digital light projectors have been used to display images on film and canvas.

4.5.1. Technical Approach and Mathematical Model

If a light beam with incident radiance L is projected onto the transparent film material that is located on top of the original artwork, a portion d of L is directly diffused from the film while the remaining portion t of L is transmitted through the film. The transmitted light tL interacts with the underlying pigment's diffuse reflectance M on the canvas, and a color blended light fraction tLM is diffused. The portion $tLMt$ is then transmitted through the film, while the remaining part $tLMd$ is reflected back towards the canvas where it is color blended and diffused from the same pigment again. This ping-pong effect between film material and canvas is repeated infinitely while for every pass a continuously decreasing amount of light is transmitted through the film that contributes to the resulting radiance R. Mathematically, this can be expressed as an infinite geometric series that converges towards a finite value. The same is true for the environment light with incident radiance E that is emitted from uncontrollable light sources. Since these light sources also illuminate the canvas and the film material, the environment light's contribution to R has to be considered as well.

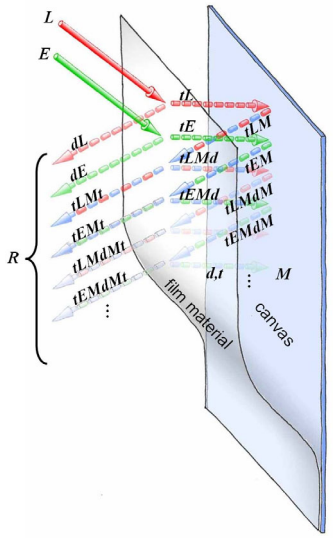


Figure 4.30: *Interaction of projected light and environment light with the canvas and the film material (sequence diagram).*

Figure 4.30 describes this process in form of a sequence diagram. Note that in contrast to this conceptual illustration, there is no physical gap between film material and canvas, and that the light interaction occurs at the same spot.

If all parameters (L, E, M, t, and d) are known we can compute the resulting radiance R that is visible to an observer in front of the canvas:

$$R = (L+E)d + (L+E)t^2 M \sum_{i=0}^{\infty} (Md)^i =$$

$$(L+E)\left(d + \frac{t^2 M}{1-Md}\right)$$

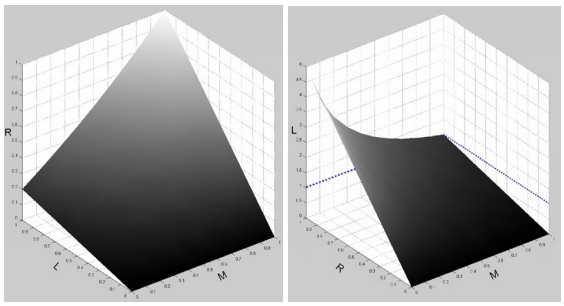
Since R forms the image we expect to see is known, we need to solve the above equation for L:

$$L = \frac{R}{\left(d + \frac{t^2 M}{1-Md}\right)} - E$$

This allows computing the incident radiance L that needs to be projected onto the film and the canvas to create the known result R. The radiant intensity I of the projector to create L is related to a discretized pixel value and is given by:

$$I = L\frac{r^2}{\cos\alpha}s$$

where $r^2/\cos\alpha$ are the form factor components: square distance attenuation and angular correlation of the projected light onto the canvas. The additional factor s allows scaling the intensity to avoid clipping and to consider the simultaneous contributions of multiple projectors.



(a) (b)

Figure 4.31: *Visualization of equation R (a) and equation L (b) over R, L, and M (without E, t=80% and d=20%).*

This approach has clear limitations, which are illustrated in figures 4.31a and 4.31b. Not all radiances R can be produced under every condition. If M is dark, most of L and E are absorbed. In an extreme case, the corresponding pigment is black (M=0). In this case the right term of the equation that computes R is canceled out. The remaining left term –which depends on the diffusion factor d of the film material– sets the boundaries of the final result that can be produced. The intersection of the surface with the RL-plane in figure 4.31a illustrates these limitations. Consequently, in the worst case of this example only 20% of R can be generated. This situation is also reflected in figure 4.31b as the intersection of the surface with the LR-plane. Here we want to assume that $sr^2/\cos\alpha=1$, which results in $L=I$. For a single video beamer, the projected radiance L and consequently the radiant intensity I cannot exceed the normalized intensity value of 1 (dotted line). But for creating most of the resulting radiance values, L and I must be larger. This situation worsens for $r^2/\cos\alpha>1$ and for $E\rightarrow 1$ or $M\rightarrow 0$.

However, the contributions of multiple (n) projectors allow displacing this boundary with:

$$L = \sum_{i=1}^{n} L_i \quad , \quad I_i = L_i \frac{r_i^2}{\cos\alpha_i} s_i$$

If all projectors provide linear transfer functions (e.g., after gamma correction) and identical brightness, $s_i=1/n$ balances the load among them equally. However, s_i might be decreased further to avoid clipping and to adapt for differently aged bulbs.

For both illustrations in figures 4.31a and 4.31b, the environment light E is not considered and is set to zero. Additional environment light would simply shift the surface in figure 4.31a up on the R-axis, and the surface in figure 4.31b down on the L-axis. Note that the above discussed mathematical model has to be applied to all color channels (e.g., red, green, and blue for projected graphical images) separately.

4.5.2. Real-Time Color Correction

The equations described in section 4.5.1 can be implemented as a pixel shader to support a color correction in real time. Figure 4.32 illustrates the rendering process based on an example of Michelangelo's masterpiece *Creation of Adam*. Although we will use this example to explain the process, it is universal and can be applied with arbitrary background images.

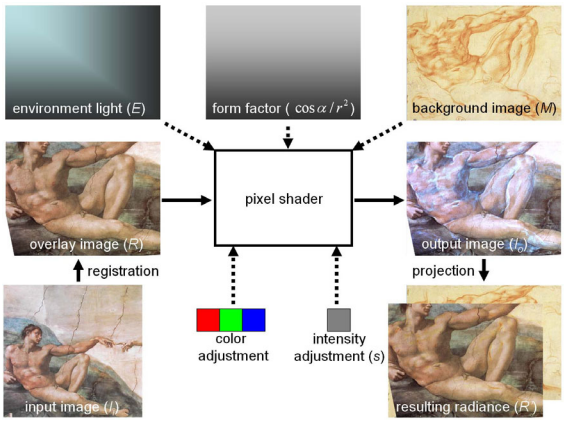


Figure 4.32: *Real-time color-correction process with pixel shader.*

In this example, a copy of an early sanguine sketch of the Adam scene (now being displayed in the London British Museum) serves as real background image M. Our goal is to overlay it entirely with a registered photograph R of the actual ceiling fresco painted the Sistine Chapel.

The first step of the rendering process is to create an input image I_i, which needs to be overlaid. This image can be either dynamically rendered (e.g., as part of a real-time animation or an interactive experience), it can be played back (e.g., frames of a prerecorded movie) or it can be static (e.g., a photograph of the corresponding ceiling portion – as it is the case in our example above).

The input image has to be registered to the physical drawing. Registration is achieved by texture mapping I_i onto a pre-distorted image geometry that is precisely aligned with the physical drawing. The amount of distortion depends on the geometric relation between the video projector and the canvas. It reaches from simple *keystone deformations* to more complex *curvilinear warping* effects (e.g., if *lens distortion* of the projector has to be neutralized). Several automatic approaches have been described that apply video cameras and image analysis to align multiple images of tiled projection screens [Che00, Yan01]. A structured light registration benefits from controllable features, such as projected grid edges or Gaussian matchpoints that can be easily detected in the camera views with a sub-pixel precision. In the case described above, however, a digital representation of the artistic content has to be registered against its physical representation on the canvas, rather than registering one projected structured light image against another one. To detect non-structured artistic features, such as fine lines, in the artwork and register them automatically against the corresponding features in the digital content represents a no-trivial task of computer vision – especially if projected pixels and physical pigments have to be aligned very precisely on the canvas. One can be critical about the feasibility and precision of an automatic method for this problem. I can be solved with a manual registration process that benefits from the resolution of the human visual system. Since the following steps have to be performed only once, they represent an acceptable solution.

An off-line registration process allows to interactively identify 2D correspondences between artistic features in the background image M within the image space –that is an image of the drawing displayed on a control screen– and the physical drawing on the wall. This is done using a 2D input device – such as a conventional mouse whose pointer is visible on the control screen and as projection on the canvas. A dual output graphics card and an additional video signal splitter are used to drive the control screen and one or two projectors. The result is a set of 2D vertex fiducials with their corresponding texture coordinates within the image space. The fiducials are Delauny triangulated and the texture coordinates are used to map the correct image portions of I onto the image geometry. This results in the overlay image R. It has to be stressed, that a precise correspondence between R and M is important to achieve qualitatively good results. The measurement of 50 to 70 fiducials proved to be sufficient for medium sized canvases. In contrast to *uniform grid methods* normally applied for projector alignment, this general geometric registration allows correlating arbitrary features in the physical drawing with the corresponding pixels of M in the image space. Thus, it provides an accurate matching which can be regionally improved further if linear interpolation within single grid triangles fails to be precise enough. The registration results do not change if projector and background image are fixed. Before R is rendered the color-correction pixel-shader has to be enabled. Five parameters are passed to it to ensure that equations discussed in section 4.5.1 can be computed.

The first parameter is the *environment light* E in form of intensity texture that has the same size as R. It contains intensity values that represent the uncontrollable lighting situation on the canvas. The intensity values can be determined by measuring the irradiance of the environment light with a lightmeter for a discrete number of sample spots on the canvas' surface – resulting in the lux values E'. To be processed by the shader, these values have to be normalized to an intensity space that ranges from 0 to 1. To do this, the same spots are measured again, but this time the highest intensity possible (i.e., a white image) is projected onto the lightmeter which is measured in addition to the environment light. These measurements are equivalent to the total irradiance $T'=L'+E'$, and also carry the unit lux. Since we know that $L'=T'-E'$ is equivalent to the scaled intensity value $\cos\alpha/r^2$, we can convert the measured radiance of the environment light from lux into the normalized intensity space with $E=E'/(T'-E')\ \cos\alpha/r^2$. To approximate the intensity values for the entire image area in E, all the measured spots are mapped into the image space, are Delauny triangulated and the values for the remaining pixels are linearly interpolated by the graphics pipeline. The values for the remaining pixels are linearly interpolated by the graphics card while rendering the Delauny mesh. Note that E is constant if the environment light does not change. For the reasons that are described below, we can assume that $\cos\alpha/r^2$ is constant and equals 1.

The second parameter is the *form factor* that represents the geometric relation between the video projector as a point light source and the canvas. Since it does not change for a fixed relation between projector and canvas, it can be precomputed and passed to the pixel shader in form of an intensity texture with the same dimensions as E and R. Similar as for the techniques described in the previous sections this texture can be produced by the graphics pipeline: A geometric model of the canvas is rendered with a white diffuse reflectance from the viewpoint of the

projector. Attaching a virtual point light source (also with a white diffuse light component) to the position of the projector and enabling square distance attenuation produces intensities that are proportional to $cos\alpha/r^2$. The required reciprocal can be computed by the pixel shader. Practically (i.e., for normal-sized canvases and non-extreme projector configurations), the form factor can be assumed to be constant over all pixels. It is then contained by the intensity adjustment parameter *s*.

The third parameter is the *background image M*. It also has the same dimensions as *E* and *R*. This image can be generated by, for example, scanning the color values or taking a photograph of the original drawing under uniform illumination.

The fourth and fifth parameters contain *color and intensity adjustment values* that allow fine-tuning the video projector's individual color response and prevent from intensity clipping. They also allow adopting for color drifts that can be introduced during capturing the background image and allow considering the contributions of multiple projectors. Note that gamma correction has to be applied in advance. This regards projectors with non-linear transfer functions as well as projectors with linear transfer functions that apply a de-gamma mapping on the video input. Gamma correction is usually supported by the graphics hardware and the video driver, but can also be carried out by the pixel shader. These values are adjusted manually, but the support of automated methods for color-matching multiple projectors [Maj00] is imaginable.

The output image I_o is the final result of this rendering process and will be displayed by the video projector. If projected geometrically correct onto the drawing, the result *R'* will be visible. Both images, *R* and *R'* are mostly identical, except for slight artifacts that are due to the limitations discussed above. Figure 4.29 shows the result of our example projected onto the real drawing with a single video projector.

Apparently, the underlying drawing can be made partially or completely invisible to display the graphical overlay in the correct colors on top of it. Close-ups are illustrated in figures 4.29e-h, in which diverging body parts (such as belly and knee) are overdrawn and displaced by the projection.

Figure 4.33: *Results of color correction process with two projectors: (top).The limited intensity capabilities of a single projector result in visible artifacts. (bottom) The contribution of a second projector reduces these effects.*

Some intensities and colors that are required to neutralize the underlying color pigments cannot be achieved by a single video projector. The worst case is to turn a black pigment on the canvas into a white color spot. Figures 4.31a and 4.31b illustrate that in such a case the required intensity can easily exceed the boundary of 1 in our normalized intensity space. The pixel shader clips these values to 1 which results in visible artifacts.

The simultaneous contributions of multiple projectors can reduce or even eliminate these effects. Figure 4.33 shows the extreme case of an input image that has no geometric correspondences to the underlying background image. In addition, it approaches to create bright colors (the sky) on top of dark color pigments on the canvas. In figure 4.33a, a single projector is used. Intensity values that are too large are clipped and result in visible artifacts. Balancing the intensity load between two projectors reduces these artifacts clearly (cf. figure 4.33b).

Due to hardware acceleration of today's graphics cards, the color correction process can be easily performed in real time. Note that none of the photographs in this article have been retouched. Slight variations of color and brightness are due to different camera responds.

As discussed above, the color correction method has limitations that are mainly defined by the capabilities of the applied hardware. For example, the restricted *resolution*,

brightness, *contrast*, *minimum focus distance* and *black-level* of video projectors are issues that will certainly be improved by future generations. The XGA resolution and the brightness of 1100 ANSI lumen of the low-cost projectors that were used to create the results for this section was appropriate for small and medium sized paintings in a normally lit environment. An up-scaling is possible by using more projectors, but downscaling would either result in a loss of effective resolution or in focus problems.

Black, for instance, is a color that cannot be projected. Instead the environment light together with the *black level* of the projectors illuminates areas that need to appear black. However, the human vision system adjusts well to *local contrast effects* – which makes these areas appear much darker than they actually are. Even with little environment light, the high black-level of video projectors causes this illusion to fail in extreme situations, such as the one shown in figure 4.34. The development of video projectors indicates that a decrease of the black-level and an increase of the contrast ratio can be expected in future.

(a) (b)

(c) (d)

Figure 4.34: *Rembrandt's self-portrait: (a) copy of original painting as it looks today (illuminated under environment light), (b)–(d) various cleaning stages to remove the overpainted layers form 1935(d), 1950(c) and 1980(b) are projected onto (a). Only black and white photographs of these stages are available. The high black-level of the video projectors prevents from creating a totally black color on the canvas. Extreme regions, such as overlaid hair and hat cannot appear completely black for this reason.*

Light can damage the artwork. Especially ultra violet (UV) and infrared (IR) radiation produced by the lamps of video projectors is critical. Commercially available UV/IR blocking filters can be mounted in front of the projectors' lenses to remove most of these unwanted rays while transmitting visible wavelengths. For the remaining visible light portion a rule of thumb advises to illuminate valuable and delicate pieces permanently with no more than 100lx-150lx. The potential damage caused by light is cumulative (e.g. 1 hour with 1000lx equals 1000 hour with 1lx) and depends on the material and color of the painting and the wavelength (i.e. the color) of the light. A temporary illumination of a higher light intensity is not critical. During a 2-3 minute presentation, an increased lighting is only temporary and such highlight situations usually only appear (if at all) for a short period of time (e.g., a few seconds) at varying locations on the canvas. Nevertheless, using the intensity adjustment described above, the maximum light level can be constrained to be below an upper threshold. However, this might cause visible artifacts depending on the presented content, the paining, and the environment light (as described in figure 4.31). Thus it is important to reach a good balance between total illumination (projected light and environment light) over time and convincing presentation effects.

The method described above currently considers only the intensity of the environment light E. This is adequate for regular white light sources but will result in artifacts if visible color shading is created on the canvas by the environment illumination. Without a modification to either the mathematical model or the rendering process the environment light's color can be compensated by determining it with aid of *colorimeters*, encoding this information in E, and passing it to the pixel shader.

The presented concept and techniques are applicable in combination with diffuse pictorial artwork, such as watercolor paintings, pen or ink drawings, sanguine sketches, or matt oil paintings. Extreme light and view-dependent effects, such as *non-Lambertian specular reflections*, *self-shadows*, *sub-surface scattering* and *interreflections* that are created by brush strokes, paint material or canvas textures cannot be handled with this method.

References

[Ban01] D Bandyopadhyay, R Raskar, A State, H Fuchs, Dynamic Spatially Augmented 3D Painting. UNC Chapel Hill Tech Report TR01-006, 2001.

[Bim01] Bimber, O., Fröhlich, B., Schmalstieg, D., and Encarnação, L.M. The Virtual Showcase. IEEE Computer Graphics & Applications, vol. 21, no.6, pp. 48-55, 2001.

[Bim02b] Bimber, O., Gatesy, S.M., Witmer, L.M., Raskar, R. and Encarnação, L.M. Merging Fossil Specimens with Computer-Generated Information. IEEE Computer, September, pp. 45-50, 2002.

[Bim02a] Bimber, O. and Fröhlich, B. Occlusion Shadows: Using Projected Light to Generate Realistic Occlusion Effects for View-Dependent Optical See-Through Displays. In proceedings of International Symposium on Mixed and Augmented Reality (ISMAR'02), pp. 186-195, 2002.

[Bim03] Bimber, O. Grundhöfer, A., Wetzstein, G., and Knödel, S. Consistent Illumination within Optical See-Through Augmented Environments, In proceedings of IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR'03), , pp. 198-207, 2003.

[Bim04a] Bimber, O. Combining Optical Holograms with Interactive Computer Graphics. In IEEE Computer, January issue, pp. 85-91, 2004.

[Bim04b] Bimber, O., Coriand, F., Kleppe, A., Bruns, E., Zollmann, S., and Langlotz, T. Superimposing Pictorial Artwork with Projected Imagery. Submitted to IEEE MultiMedia, 2004.

[Boi01] Boivin, S. and Gagalowicz, A. Image-based rendering of diffuse, specular and glossy surfaces from a single image. In proceedings of ACM Siggraph, pp. 107-116, 2001.

[Bre73] Brent, R.P. Algorithms for minimization without derivatives. Prentice-Hall, Engelwood Cliffs, NJ, 1973.

[Bre96] Breen, D.E., Whitaker, R. T., Rose, E. and Tuceryan, M. Interactive Occlusion and Automatic Object Placement for Augmented Reality. Computer and Graphics Forum (proceedings of EUROGRAPHICS'96), vol. 15, no. 3, pp. C11-C22, 1996.

[Che00] Chen, Y., D. Clark, Finkelstein, A., Housel, T. and Li., K., Automatic Alignment of high resolution Multi-Projector Displays using an un-calibrated Camera, In proceedings of IEEE Visualization, pp. 125-130, 2000.

[Che93] S. E. Chen, and L. Williams. View Interpolation from Image Synthesis. SIGGRAPH '93, pp. 279-288, July 1993.

[Deb96] P. Debevec, C. J. Taylor, and J. Malik. Modeling and Rendering Architecture from Photographs. SIGGRAPH '96, August 1996.

[Deb98] P. Debevec, Y. Yu, and G. Borshukov. Efficient View-Dependent Image-Based Rendering with Projective Texture-Mapping. Proc. of 9th Eurographics Workshop on Rendering, June 1998.

[Dre97] Dreesen, F., von Bally, G. Color holography in a single layer for documentation and analysis of cultural heritage. In Optics within Life Sciences (OWLS IV), Springer Verlag Berlin, Heidelberg, New York, pp. 79-82, 1997.

[Dre00] Dreesen, F., Deleré, H., von Bally, G. High Resolution Color-Holography for Archeological and Medical Applications. In Optics within Life Sciences (OWLS V), Springer Verlag Berlin, Heidelberg, New York, pp. 349-352, 2000.

[Dre97] Drettakis, G. and Sillion, F. Interactive update of global illumination using a line-space hierarchy. In proceedings of ACM Siggraph'97, pp- 57-64, 1997.

[Fau93] O. Faugeras. Three-Dimensional Computer Vision: A Geometric Viewpoint. MIT Press, Cambridge, Massachusetts, 1993.

[For93] Fournier, A. Gunawan, A.S., and Romanzin, C. Common illumination between real and computer generated scenes. In proceedings of Graphics Interface'93, pp. 254-262, 1993.

[Gib00] Gibson, S. and Murta, A. Interactive rendering with real-world illumination. In proceedings of 11th Eurographics Workshop on Rendering, pp. 365-376, 2000.

[Gor84] Goral, C.M., Torrance, K.E., Greenberg, D.P., and Battaile, B. Modeling the interaction of light between diffuse surfaces. Computer In proceedings of ACM Siggraph'84, vol. 18, no. 3, pp. 212-222, 1984.

[Gor96] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The Lumigraph. SIGGRAPH '96, August 1996.

[Hua01] Hua, H., Gao, C., Brown, L., Ahuja, N., and Rolland, J.P. Using a head-mounted projective display in interactive augmented environments. In proceedings of IEEE and ACM International Symposium on Augmented Reality (ISAR'01), pp. 217-223, 2001.

[Kaj86] J. T. Kajiya. The Rendering Equation. Computer Graphics 20(4) (1986), pp. 143-151.

[Kiy00] Kiyokawa, K., Kurata, Y. and Ohno, H. An Optical See-through Display for Mutual Occlusion of Real and Virtual Environments. In proceedings of IEEE & ACM ISAR 2000, pp. 60-67, 2000.

[Klu02] Klug, M.A., Scalable digital holographic displays, IS&T PICS 2001: Image Processing, Image Quality, Image Capture Systems Conference, Montreal, Quebec, Canada, pp. 26-32, 2001.

[Kol89] Kollin, J.S., Benton, S.A., and Jepsen, M.L. Real-Time Dispaly of 3-D Computer Holograms by Scanning the Image of an Acousto-Optic Mudulator. In Proceedings of SPIE, vol. 1136, Holographic Optics II: Principle and Applications, 1989.

[Lev96] M. Levoy, and P. Hanrahan. Light Field Rendering. SIGGRAPH '96, August 1996.

[Los00] Loscos, C., Drettakis, G., and Robert, L. Interactive virtual relighting of real scenes. IEEE Transactions on Visualization and Computer Graphics, vol, 6, no. 3, pp. 289-305, 2000.

[Low01] Low, K., Welch, G., Lastra, A. and Fuchs, H. Life-Sized Projector-Based Dioramas, In proceedings of Symposium on Virtual Reality Software and Technology, pp-93-101, 2001.

[Luc93] Lucente, M. Interactive computation of holograms using a look-up table. Journal of Electronic Imaging, vol. 2, no. 1, pp. 28-34, 1993.

[Luc95] Lucente, M. and Tinsley, A. Rendering interactive images. Computer Graphics (proceedings of SIGGRAPH'95), pp. 387-394, 1995.

[Luc97] Lucente, M. Interactive three-dimensional holographic displays: seeing the future in depth. SIGGRAPH Computer Graphics, special issue on Current, New, and Emerging Display Systems, 1997.

[Mat98] Mather G, Verstraten F, and Anstis S The Motion Aftereffect: a Modern Perspective. Cambridge, Mass: MIT Press, 1998. (http://www.biols.susx.ac.uk/home/George_Mather/Motion/)

[Maj00] Majumder, A., He, Z., Towles, H., and Welch, G., Achieving Color Uniformity Across Multi-Projector Displays, In proceedings of IEEE Visualization, pp. 117–124, 2000.

[McM96] L. McMillan, and G. Bishop. Plenoptic Modeling. SIGGRAPH '95, August 1995.

[Nae02] Naemura, T., Nitta, T., Mimura, A., Harashima, H. Virtual Shadows – Enhanced Interaction in Mixed Reality Environments. In proceedings of IEEE Virtual Reality (IEEE VR'02), pp. 293-294, 2002.

[Nak01] Nakamae, E., Harada, K., Ishizaki, T., and Nishita, T.A montage method: The overlaying of computer generated images onto background photographs. In proceedings of ACM Siggraph'86, pp. 207-214, 1986.

[Nei96] Neider, J., Davis, T., and Woo, M. OpenGL Programming Guide. Release 1, Addison Wesley, ISBN 0-201-63274-8, pp.157-194, 1996.

[Nod99] Noda, S., Ban, Y., Sato, K., and Chihara, K. An Optical See-Through Mixed Reality Display with Realtime Rangefinder and an Active Pattern Light Source. Transactions of the Virtual Reality Society of Japan, vol. 4, no. 4, pp. 665-670, 1999.

[Pan] Panoram Technology. http://www.panoramtech.com

[Pet03] Petz, C. and Magnor, M. Fast Hologram Synthesis for 3D Geometry Models using Graphics Hardware. In proceedings of SPIE'03, Practical Holography XVII and Holographic Materials IX, vol. 5005, pp 266-275, 2003.

[Pin01] Pinhanez, C. The everywhere displays projector: A device to create ubiquitous graphical interfaces, In proceedings of Ubiquitous Computing, pp. 315-331, 2001.

[Pre92] Press, W.H., Teukolsky, S.A., Vetterling, W.T. and Flannery, B.P. Numerical Recipes in C - The Art of Scientific Computing (2nd edition), Cambridge University Press, ISBN 0-521-43108-5, pp. 412-420,1992.

[Ras98] R. Raskar, G. Welch, M. Cutts, A. Lake, L. Stesin, and H. Fuchs. The Office of the Future: A Unified Approach to Image-Based Modeling and Spatially Immersive Displays. SIGGRAPH '98, July 1998

[Ras99] R. Raskar, M. Brown, R. Yang, W. Chen, G. Welch, H. Towles, B. Seales, H. Fuchs. Multi-Projector Displays Using Camera-Based Registration. IEEE Visualization 99, October 1999.

[Ras99b] R. Raskar, G. Welch, W. Chen. Tabletop Spatially Augmented Reality: Bringing Physical Models to Life using Projected Imagery. Second Int Workshop on Augmented Reality (IWAR'99), October 1999, San Francisco, CA

[Ras01] Raskar, R., Welch, G., Low, K.L. and Bandyopadhyay, D. Shader Lamps: Animating real objects with image-based illumination, In proceedings of Eurographics Rendering Workshop, pp. 89-102, 2001.

[Ras02] Raskar, R., Ziegler, R. and Willwacher, T. Cartoon Dioramas in Motion, In proceedings of Int. Symp on Non-photorealistic Animation and Rendering, pp. 7-ff, 2002.

[Ras03] Raskar, R., van Baar, J., Beardsly, P., Willwacher, T., Rao, S. and Forlines, C. iLamps: Geometrically Aware and Self-Configuring Projectors, In proceedings of ACM Siggraph, pp. 809-818, 2003.

[Ras04] R. Raskar, P Bearsdley, J VanBaar, Y Wang, P Dietz, J Lee, D Leigh, T Willwacher. RFIG Lamps: Interacting with a Self-Describing World via Photosensing Wireless Tags and Projectors. SIGGRAPH '04, 2004.

[Sze97] R. Szeliski and H. Shum. Creating Full View Panoramic Mosaics and Environment Maps. SIGGRAPH '97, August 1997.

[Und99] Underkoffler, J., Ullmer, B. and Ishii, H. Emancipated pixels: real-world graphics in the luminous room, In proceedings of ACM Siggraph, pp. 385-392, 1999.

[Yan01] Yang, R., Gotz, D., Hensley, J., Towels., H., and Borwn, M., PixelFlex: A Reconfigurable Multi-Projector Display System, In proceedings of IEEE Visualization, 2001.

[Yam90] Yamaguchi, M, Ohyama, N., and Honda, T. Holographic 3-D Printer. In Proceedings of SPIE, V. 1212, p. 84, 1990.

[Yos03] Yoshida, T., Horii, C. and Sato, K. A Virtual Color reconstruction System for Real Heritage with Light Projection, In proceedings of Virtual Systems and Multimedia, pp. 158-164, 2003.

[Yu99] Yu, Y., Debevec, P., Malik, J., and Hawkins, T. Inverse global illumination: Recovering reflectance models of real scenes from photographs. In proceedings of ACM Siggraph' 99, pp. 215-224, 1999.

5. Summary and enabling technologies

In this tutorial we discussed the Spatial Augmented Reality (SAR) concept. For specific applications, SAR allows to overcome some of the limitations linked to conventional AR displays. In addition it holds the potential of opening doors to new application domains.

First, we gave an overview over different AR display techniques that may enable readers to identify parallels between virtual reality and augmented reality, and stimulate them to think about alternative display approaches for AR.

One of our focuses was on spatial optical see-through technology: We described different display configurations using transparent screens and mirror beam-splitters, as well as interactive rendering techniques that support such configurations and neutralize optical distortion.

The other focus was on projector-based augmentation and illumination. Rendering techniques for non-trivial (geometry and texture) projection screens were described. Finally, the projector-based illumination concept was explained, and examples were outlined how it can be used to create consistent illumination and occlusion effects.

We want to annotate that upcoming and new technology will not only open new possibilities for SAR, but also for other display concepts, such as hand-held and head-attached displays.

Projectors of the near future, for instance, will be compact, portable, and with the built-in awareness which will enable them to automatically create satisfactory displays on many of the surfaces in the everyday environment. Alongside the advantages, there are limitations, but we anticipate projectors being complementary to other modes of display for everyday personal use in the future, and to have new application areas for which they are especially suited. LEDs are replacing lamps and reflective instead of transmissive displays (DLPs, LCOS) are becoming popular. Both lead to improved efficiency requiring less power and less cooling. DLP and LCOS projectors can display images at extremely high frame rates, currently 180Hz and 540 Hz respectively, but lack video bandwidth. Several efforts are already in the making and are very promising. For example Symbol Technologies [Sym02] has demonstrated a small laser projector (two tiny steering mirrors for vertical and horizontal deflection) and has even built a handheld 3D scanner based on such a projector. Siemens has built a 'mini-beamer' attachment for mobile-phones [Sie02]. Cam3D has built a 'Wedge' display where a projector can be converted into a 'flat panel' display by projecting images at the bottom of a wedge shaped glass [Tra02]. A

future mobile projector may double up as 'flat panel' when there is no appropriate surface to illuminate, or ambient light is problematic. Super bright, sharp infinite focus laser projectors are also becoming widespread [Jen02] which may allow shape-adaptive projection without focus and ambient lighting problems. In addition suitable input devices are also appearing; e.g., Canesta [Cas02] has built a projected laser pattern on which one can type. The finger movement is detected by IR sensing. Finally novel lamp designs, especially those based on LEDs or lasers are creating smaller, lighter, efficient and long-life solutions. Two important problems in Augmented Reality, object identification and determining the pose of the displayed image with respect to the physical object, can be solved by using photosensing RFID tags. This is being explored as a Radio Frequency Identification and Geometry (RFIG) discovery method [Ras04]. Figure 6.12 shows a current application in warehouse scenario where two employees can locate products that are about to expire using RF channel to send the query but optical channel to locate the corresponding wireless tags. The handheld projector finally displayed the appropriate information overlaid on those objects.

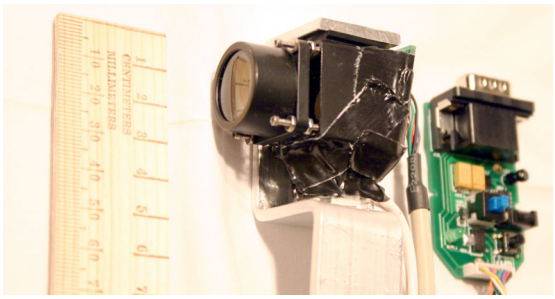


Figure 5.1: *A micro-projector prototype we have built -- to indicate the viability of designing a projector for truly handheld use. It is a 1-inch cube using an LED and LCOS imager to create a color animated projection at a distance of about 12-inches.*

Figure 5.2: *Warehouse scenario using RFIG (Ras04). A user directs a handheld projector at tagged inventory, with communication mediated by two channels -- RF and photo-sensing on the tags. The user sees a projection of the retrieved tag information collocated with the physical objects, and performs a desktop-like interaction with the projection. A second user performs similar operations, without conflict in the interaction because the projector beams do not overlap.*

Organic Light Emitting Diodes (OLEDs) [How01], for instance, may replace the crystalline LEDs that are currently being used to build the miniature displays for HMDs. OLEDs promise to produce cheap and very high-resolution full-color matrix displays that can give head-attached displays a technological push. In contrast to normal LEDs, OLEDs are made from plastic compounds instead from semi-conducting elements, such as silicon or gallium, etc. Like LEDs, OLEDs glow when voltage is applied. Two main classes exist today: small molecule OLEDs and polymer OLEDs. While small molecule OLEDs are built up by depositing molecules of the compound onto the display itself under very low pressure, polymer OLEDs have the active molecules suspended in a liquid-like pigment in paint. It can be printed onto displays using ink jets, screen printing or any of the various contact techniques used for ordinary inks. Small molecule OLED displays are limited in size, but they may be suitable for head-mounted displays. Polymer OLEDs can be used to build large scale displays – such as 500 inch displays or larger. Resolution approaching 300 dpi is also possible, approaching the quality of ink on paper. They may become more interesting for spatial AR approaches.

The general advantages of OLEDs are:

- Because OLED displays are fluorescent and don't require backlights they will need far less power than LCD screens (currently LCDs require three times as much power than OLEDs);
- OLED layers can be made much thinner than LCD layers (about a thousand times thinner than a human hair);
- In contrast to LCDs, OLEDs don't use polarized light filters. The displayed images on OLEDs can be viewed from a much wider angle;
- OLEDs have a much wider working temperature range than LCDs;
- Can be "printed" onto large-scale and flexible display surfaces of any shape and (almost) any material.

Today's reality, however, is that the OLED compounds degrade over time (especially when they get in contact with oxygen or water) – limiting the maximum lifetime of a display. Different colors degrade at different rates – making the color balance change. These problems may be solved in future.

A variation of OLEDs are *Light Emitting Polymers* (LEPs) [Bur90] that provide the opportunity for the fabrication of large, flexible, full-color, fast emissive displays with a high resolution, a wide viewing angle and a high durability. In LEP technology, a thin film of light-emitting polymer is applied onto a glass or plastic substrate coated with a transparent electrode. A metal electrode is evaporated on top of the polymer. The polymer emits light when the electric field between the two electrodes is activated. The response time of LEPs is ultra-fast (sub-microsecond) and is unaffected by temperature. Consequently they may support high enough frame rates for active stereoscopic rendering. Light emission occurs at low voltage, and (unlike LCD or plasma displays) it can be fabricated on a single sheet of glass. Also, because it can be made of flexible plastic substrates, it is not only extremely difficult to break, but can also be molded into different shapes and contours.

The advantages of LEPs can be summarized as follows:

- Since a low voltage is required LEPs need little power;
- The response time of LEPs is very high – potentially allowing active stereoscopic rendering;
- Can be fabricated on transparent glass or plastic surfaces of any shape;
- LEPs provide a high contrast (currently between 3-5 times higher than LCDs).

Transparent LEPs may present other future alternatives for spatial AR configurations.

Electronic paper (or E-Ink) [Xer03] also has potential as an AR display technology of the future. Here an electric charge moves magnetic colored capsules within the "paper" either toward or away from the surface in order to form an image. The capsules retain their positions until another charge is applied. The ink simply resides on the display while an image is not changing; therefore, it consumes no power. Philips and other companies in the field are working on color as well as bendable applications. The main advantage of electronic paper is that it does not require power at all, as long as the displayed image does not change. The current generation of electronic paper, however, is a black-on-white technology where the ink is pumped onto a white background, giving maximum readability.

Several approaches exist that allow a projection of 2D images onto a screen composed of fog [Fog04] or condensed air [IO204]. Such *fog/air displays* [Pau04] suffer from distortion caused by turbulences but allow through-the-screen interactions since they lack a physical screen boundary.

Solid-state displays [Dow96] generate visible photons (i.e. light) within a transparent host material by exciting optically active ions with special energy sources. Ions at a known three-dimensional position within the host materials can be excited by crossing energy beams, such as infrared lasers, ultraviolet sources of radiation, or electron beams.

Examples of suitable host materials are various gases, crystals and electro-active polymers. The host material must provide several properties:

- In its initial state it must be transparent;
- It must emit visible light in its excited state;
- Its inner structure must be homogenous;
- It must have a refraction index similar to air to avoid distortion.

If it was possible to use air as the host material, then this approach would represent the Holy Grail – not only for spatial augmented reality displays, but for 3D display technology in general. Unfortunately this is not yet feasible. In addition, conceptual problems, such as the "ghost voxel problem" (when energy beams that are used to create voxels intersect with other beams and create unwanted voxels) have to be solved.

Parallax displays are display screens (e.g., CRT or LCD displays) that are overlaid with an array of light-directing elements. Depending on the observer's location, the emitted light that is presented by the display is directed in a way that it appears to originate from different parts of the display while changing the viewpoint. If the light is directed to both eyes individually, the observer's visual system interprets the different light information to be emitted by the same spatial point. Examples of parallax displays are parallax barrier displays (cf. figure 5.3-top) that apply a controllable array of light-blocking elements (e.g., a light blocking film or liquid crystal barriers [Per00]) in front of a CRT screen. Depending on the observer's viewpoint, these light-blocking elements are used to direct the displayed stereo-images to the corresponding eyes. Other examples are lenticular sheet displays (cf. figure 5.3-bottom) that apply an array of optical elements (e.g., small cylindrical or spherical lenses) to direct the light for a limited number of defined viewing-zones.

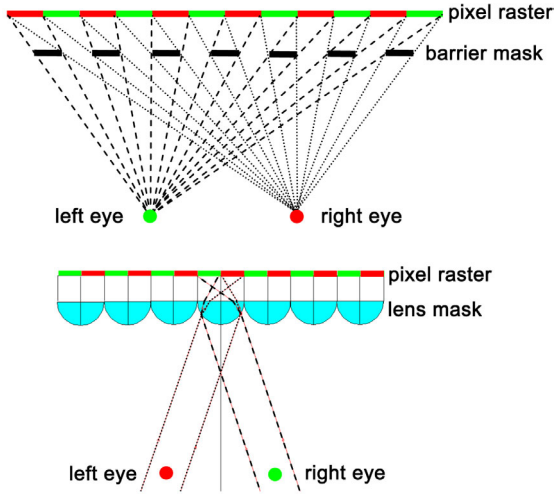


Figure 5.3: *Basic parallax display concepts: (top) parallax barrier display and (bottom) lenticular sheet display.*

Parallax displays can be published and mass-produced in a wide range of sizes, and can be used to display photo-realistic images. Many different types are commercially available and used from desktop screens to cell-phone displays. We believe that the application of parallax displays will be the next logical and feasible step for SAR solutions towards autostereoscopy.

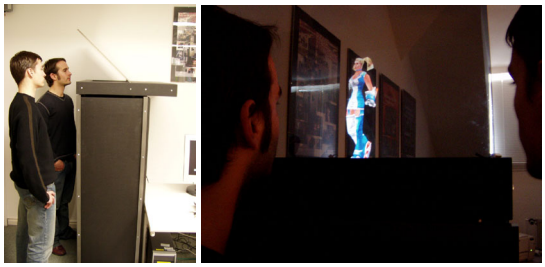


Figure 5.4: *A first autostereoscopic, multi-user capable SAR display, (information available through www.uni-weimar.de/medien/AR).*

Figure 5.4 shows a first running autostereoscopic SAR display. It uses a special barrier mask to generate multiple viewing zones simultaneously and is able to support many surrounding observes at the same time. This allows detaching the display technology from the users completely.

In the future, a stronger combination of computer graphics and holography (as explained in chapter 4) can also be expected. Any three-dimensionally programmed computer image can, in principle, be transformed into a hologram basically by way of Fourier transformations. These, however, presuppose large computational, network and storage capacities. For instance, a pre-rendered or recorded movie-length holographic video could require a petabyte (1 million gigabytes) of storage. Holographic storage itself, which uses lasers to record data in the three dimensions of a clear crystalline medium may be a future solution to the problem that non-static holograms require a massive amount of data. While computer generated static images can already be transformed into large digital holograms using holographic printers [Klu01], interactive or real-time electronic holography [Luc97] with an acceptable quality (size, resolution and colors) would still require the invention of more advanced light modulators, faster computers with a higher bandwidth, and better compression techniques. However, the combination of high quality optical holograms and interactive computer graphics is already feasible with off-the-shelf hardware today [Bim04].

In the short run, especially high-resolution bright and flexible projection devices, high-performance and cost-efficient rendering hardware, reliable, precise and wireless tracking technology, and advanced interaction techniques and devices will pave the way for forthcoming alternative AR configurations. In the long run, new display concepts, such as autostereoscopy and holography will replace goggle-bound stereoscopic displays (at least for non-

mobile applications). However, the underlying technology must be robust, flexible and the technology that directly interfaces to users should adapt to humans, rather than forcing users to adapt to the technology. Therefore, human-centered and seamless technologies, devices and techniques will play a major role for augmented reality of the future. That this is feasible can be demonstrated based on the example of one early SAR display – the Virtual Showcase [Bim01]. Since its invention in 2000, several prototypes have been exhibited to an approximated total number of more than 25,000 – 30,000 visitors/users (user-study and picture gallery are available through www.uni-weimar.de/AR). Some events were technically supervised short-term exhibitions (1-7 days) at conferences and trade shows, others have been non-supervised long-term exhibitions (3-4 month) in museums. During these years, we have made the experience that SAR technology can be robust, attractive and efficient enough to be applied successfully outside research laboratories.

References

[Bim01] Bimber, O., Fröhlich, B., Schmalstieg, D., and Encarnação, L.M. The Virtual Showcase. IEEE Computer Graphics & Applications, vol. 21, no.6, pp. 48-55, 2001.

[Bim04] Bimber, O. Combining Optical Holograms with Interactive Computer Graphics. In IEEE Computer, January issue, pp. 85-91, 2004.

[Bur90] Burroughes, J.H. Light Emitting Polymers. Nature, pp. 347, 1990.

[Cas02] Canesta. Miniature Laser Projector Keyboard. Retrieved from the World Wide Web: http://www.canesta.com. 2002.

[Dow96] Downing, E.A., Hesselink, L., Ralston, J. and Macfarlane, R. A three-color, solid-state three-dimensional display. Science, vol. 273, pp. 1185-1189, 1996.

[Fog04] Fogscreen Inc. Projecting Images in thin Air. Retrieved from the World Wide Web: http://www.fogscreen.com/ , 2004.

[How01] Howard, W. E. Organic Light Emitting Diodes (OLED). OLED Technology Primer, Retrieved from the World Wide Web: http://www.emagin.com/oledpri.htm, 2001.

[IO204] IO2 Technology, The HelioDisplay. Retrieved from the World Wide Web: http://www.io2technology.com/ , 2004.

[Jen02] Jenoptik. Laser Projector. Retrieved from the World Wide Web: http://www.jenoptik.com, 2002.

[Klu01] Klug, M.A., Scalable digital holographic displays, IS&T PICS 2001: Image Processing, Image Quality, Image Capture Systems Conference, Montreal, Quebec, Canada, pp. 26-32, 2001.

[Luc97] Lucente, M. Interactive three-dimensional holographic displays: seeing the future in depth. SIGGRAPH Computer Graphics, special issue on "Current, New, and Emerging Display Systems", 1997.

[Pau04] Paulson, L.D. Displaying Data in Thin Air. IEEE Computer, March issue, p. 19, 2004.

[Per00] Perlin, K., Paxia, S., and Kollin, J.S. An autostereoscopic display. Computer Graphics (proceedings of SIGGRAPH'00), pp. 319-326, 2000.

[Ras04] R. Raskar, P Bearsdley, J VanBaar, Y Wang, P Dietz, J Lee, D Leigh, T Willwacher. RFIG Lamps: Interacting with a Self-Describing World via Photosensing Wireless Tags and Projectors. SIGGRAPH '04, 2004.

[Sie02] Siemens. Siemens Mini Beamer. Retrieved from the World Wide Web: http://w4.siemens.de/en2/html/press//newsdesk archive/2002/-foe02121b.html, Cited December 2002.

[Sym02] Symbol. Laser Projection Display. Retrieved from the World Wide Web: http://www.symbol.com/products/oem/lpd.html, December, 2002.

[Tra02] Travis, A., Payne, F., Zhong, J., and More, J. Flat panel display using projection within a wedge-shaped waveguide. Retrieved from the World Wide Web: http://ds.dial.pipex.com/cam3d/technology/technology01.html, 2002.

[Xer03] Xerox PARC, Electronic Reusable Paper. Retrieved from the World Wide Web: http://www2.parc.com/dhl/projects/gyricon/ , 2003.

Acknowledgements

We would like to acknowledge and thank all contributors of this tutorial: M. Agrawala, R. T. Azuma, J. van Baar, D. Bandyopadhyay, P. Beardsley, E. Foxlin, G. Goebbels, H. Hua, M. Inami, N. Kawakami, Y. Kitamura, K. Kiyokawa, K. Low, J. Newman, T. Ogi, C. Pinhanez, A. Preston, B. Schwald, G. Stetten, B. Kaelin, G. Wetzstein, T. Willwacher and R. Ziegler.

The Virtual Showcase project is supported by the European Union (IST-2001-28610).

The HoloGraphics project is supported by the Deutsche Forschungsgemeinschaft (DFG BI 835/1-1).