Occlusion Shadows: Using Projected Light to Generate Realistic Occlusion Effects for View-Dependent Optical See-Through Displays

Oliver Bimber[#]
Fraunhofer Center for Research in Computer Graphics, 321 South Main St., Providence, RI 02903, USA,
oliver.bimber@medien.uni-weimar.de

Bernd Fröhlich
Bauhaus University Weimar
Bauhausstraße 11,
99423 Weimar, Germany,
bernd.froehlich@medien.uni-weimar.de

[#]now at Bauhaus University Weimar

Abstract

This paper presents projector-based illumination techniques for creating correct occlusion effects for optical see-through setups. We project view-dependent occlusion shadows onto the real surfaces that are located behind virtual objects. This results in a perfect occlusion of real objects by virtual ones. We have implemented and tested our approach in the context of the Virtual Showcase display. We describe hardware extension for projecting light into the showcase and present our rendering techniques for displaying occlusion shadows for single and multi-user environments as well as for single and multi-light-projector configurations. We also report on the limitations of our system for multi-user situations and describe our experiences with a first experimental prototype.

1. Introduction

Projection-based augmented reality systems, such as the Virtual Showcase [3], share many positive properties of projection-based virtual environments. 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 environments. One of the main challenges for projection-based AR systems as well as for head-mounted optical see-through displays is the generation of correct occlusion effects between virtual and real objects [1]. Additionally shadows of virtual objects cast onto real ones and consistent illumination of the real and virtual scenery are often difficult to achieve.

In this paper, we introduce projector-based illumination techniques for view-dependent optical see-through AR displays. This approach has the potential to solve all of the above mentioned problems. Here, we focus on using projector-based illumination for creating correct occlusion effects for mixed reality configurations (cf. figure 1).

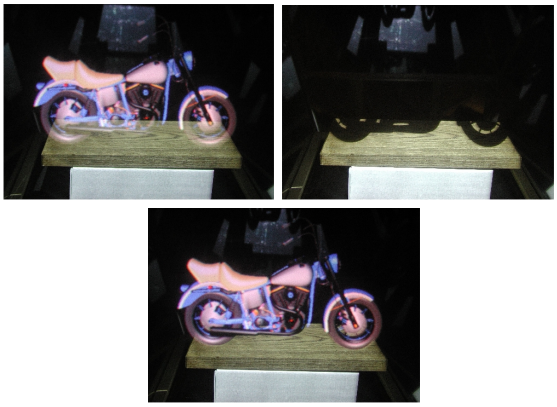

Figure 1: Wrong occlusion effects with normal illumination (left), occlusion shadow generated with projector-based illumination (right), realistic occlusion of the real object by the virtual one (center).

We have implemented and tested such a system in the context of the Virtual Showcase, which consists of a horizontal projection screen and a convex half-silvered mirror assembly (cf. figure 2). Virtual and real objects can be displayed in the same space inside the showcase.

The original Virtual Showcase used a standard light bulb to illuminate real objects. This setup does not provide very much control over the lighting situation. By using a computer-controlled video-projector as a replacement for the simple light bulb, we are able to fully control the lighting situation inside the showcase on a per-pixel basis.

Our main contribution is a solution to the problem of correct occlusion for mixed reality scenarios with view-dependent optical see-through displays. Our method produces correct occlusion effects between virtual and

real objects by projecting shadows onto real objects located behind virtual ones using projector-based illumination.

Figure 2: Our experimental prototype: The Virtual Showcase sits on top of a rear-projection screen. The lower half of the truncated pyramid configuration consists of four half-silvered mirrors. An additional set of full mirrors comprise the top half and redirects the light beam of a video projector (upper right) into the showcase center.

We describe our extended Virtual Showcase hardware for projecting these shadows into the showcase (cf. figure 2) and present rendering techniques for displaying them in single and multi-user environments as well as for single and multi-light-projector configurations. We also report on the limitations and potential extensions of our system and describe our experiences with our first setup.

2. Related Work

Kiyokawa et al. [8] present ELMO, an optical see-through head-mounted display that supports mutual occlusion. 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.

Noda et al. [10] present 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. Noda's system is strictly limited in several points. Firstly a dark surrounding environment is required, which constrains the applications possible. Secondly view-dependent rendering is not possible. 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. Lastly stereoscopic rendering is not provided.

Naemura et al. [9] proposes 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.

Head-Mounted Projective Displays, or HMPDs, (such as described by Hua et al. [7]) 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.

Raskar et al. [12] applies multiple stationary video projectors to "lift" the lighting and material properties of real objects by projecting colored images onto the real objects' surfaces. His approach provides an auto-stereoscopic behavior and does not have to deal with the occlusion problem since it is not based on the optical see-through concept. In fact, he faces an inverse problem: His method is constrained by the shape and color of the real objects. On the one hand, it is not possible to display graphics next to a real surface if another real object is not located behind the graphics that can serve as display surface. On the other hand, the real objects are required to have a bright color that diffuses the projected light. Dark objects would absorb the light. However, since a view-dependent rendering is mostly not required[1], he can

[1] Basic view-dependent illumination effects, such as specular reflection, are handled by a skillful distribution of tasks between model-view and projection transformations.

simply render a textured virtual representation of the real scene from the viewpoint of the projector(s).

3. Our Approach

Being an optical see-through display, the Virtual Showcase faces the same occlusion problem as head-mounted displays if conventional illumination is used. However, the Virtual Showcase completely encloses the contained real artifact, which offers the possibility to fully control the lighting situation. Figure 2 shows our experimental setup with an additional video-projector for illuminating the real content inside the showcase. We refer to these projectors as *light projectors*.

Our idea is similar to Noda's [10]. However, our approach supports view-dependent and stereoscopic rendering for single and multiple users and we do not require the illumination being projected from the user's point of view. This requires depth knowledge of the real scenery to support both –the occlusion of real objects by virtual ones and vice versa. In addition, the Virtual Showcase setup does not depend on a dark surrounding, since the real artifact is completely enclosed and the interior lighting of the Virtual Showcase is fully controllable.

We dynamically generate shadows directly on the real objects' surfaces wherever graphics is overlaid (figure 1). These shadows are not directly visible to the observers, since they are purposely occluded by the overlaid graphics. We call these shadows *occlusion shadows*. We additionally render phantom bodies representing real objects which occlude virtual objects behind them. The combination of occlusion shadows and phantom bodies effectively solves the occlusion problem for optical see-through displays such as the Virtual Showcase.

4. Rendering Occlusion Shadows

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 light projectors' parameters are determined only once during a calibration phase. Virtual objects can be interactively manipulated within the showcase 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

4.1. Light Projector Calibration

Before calibrating a light projector, a geometric representation of the real scene is registered to its physical counterpart. Then, the two-dimensional perspective projections of selected three-dimensional points on the real objects' surfaces are sampled within the light projector's screen space as described by Raskar [12]. The three-dimensional fiducials are highlighted on the real objects' surfaces by rendering and overlaying them with the Virtual Showcase display – given that the real objects have been registered first. A crosshair is then rendered into the light projector's frame buffer. It is aligned with the highlighted surface points to measure their 2D projections in the corresponding screen space.

Users interactively browse through the sample points, which allows the selection of reasonable calibration areas (e.g., those that are clearly visible and are not in the shadow of, or occluded by other surfaces). Once an appropriate number of samples has been taken, they are used as input for a numerical minimization which computes the light projector's intrinsic (vertical field of view and aspect ratio in our case) and extrinsic (position, optical axis, and up-vector in our case) parameters. We applied Powell's direction set method [11] to solve this perspective-n-point (PnP) problem. The result is the projector's perspective projection matrix (P) that incorporates the correct model-view transformation with respect to the scene origin.

In our case the light frustum of the projector is redirected by a planar mirror. Thus we need to incorporate the reflection transformation of the mirror. During calibration, we reflect the coordinates of the 3D fiducials over the corresponding mirror plane before passing them into the minimization routine. In this case, P needs to incorporate an additional reflection matrix that reflects the scene over the mirror plane during rendering (as described in Bimber et al. [4]).

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

4.2. Single Viewpoint

The basic algorithm below illustrates how to render occlusion shadows for a single point of view.

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 (lines 1-5) which is then perspectively mapped onto the known geometry of the real content (lines 6-7). Line 4 renders the illumination for the real content into the frame buffer. This illumination could be computed with a similar BRDF model as described in Raskar et al. [12] – producing a correct and matching radiance on real and virtual surfaces with respect to virtual light sources. Note

that this has not been implemented yet. We just project 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.

```
1: set projection matrix to V
2: render real content into depth buffer
3: render virtual content into stencil buffer
4: render illumination for real content into
   frame buffer (previously cleared to black)
5: transfer frame buffer into texture memory T
6: set projection matrix to P, set
   texture matrix to V + normalization space
   correction, clear frame buffer to black
7: render real content into frame buffer using
   projective texture T
```

Algorithm 1

Note also that the instruction in line 2 ensures a correct occlusion of virtual objects by real ones, as proposed by Breen et al. [5]. This is illustrated in figure 3. The normalization space correction in line 6 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[2].

4.3. Multiple Viewpoints

A clear limitation of our method 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. This problem cannot be solved in general with our current setup. However, we propose two approaches to reduce these effects:

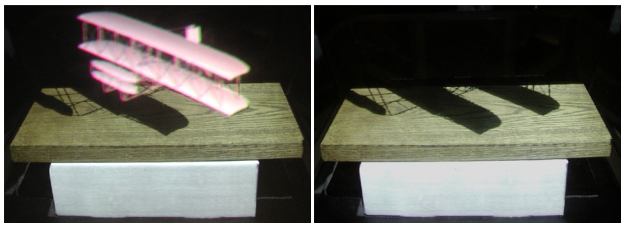

Figure 4: Occlusion shadows generated for two different viewpoints. With graphical overlay (left), and without graphical overlay (right).

[2] This applies for OpenGL.

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. We make use of this fact 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 (cf. figure 4).

Our second approach tries to minimize the interference between individual occlusion shadows 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, 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.

Figure 5 illustrates a simple example. Here, we assume two viewpoints (V1, V2), one light projector (P), and five real surfaces (1-5). For V1, surface 1,2 and 5 are completely visible, surface 3 is completely invisible, and surface 4 is partially visible. For V2, surfaces 2,3,4 and 5 are completely visible and surface 1 is completely invisible. Consequently, surfaces 2, 4 and 5 are (at least partially) visible for both viewpoints.

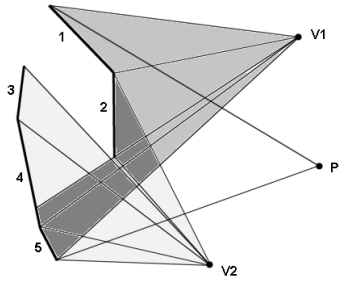

Figure 5: Visibility for different points of view.

We want to ensure that the occlusion shadows for each viewpoint are generated only on the visible portions of the surfaces. However, since different surfaces are visible from the perspective of the projector than from the perspectives of the viewpoints, the correct appearance of real scene has to be determined before it is rendered.

If algorithm 1 would be simply repeated for every viewpoint[3], the projective texture of V1 would be unnecessarily mapped onto surface 4[4] and might consequently interfere with the texture of V2. Algorithm 2 explains how we approach this problem.

Throughout lines 1-12 in algorithm 2, a shadow mask is generated for each viewpoint and stored in an

[3] And the resulting textures would be color blended appropriately.

[4] Note that surface 3 cannot be illuminated, since it isn't visible from the perspective of the projector.

individual block of the texture memory. All shadow masks are then color blended into the final image that is projected by the light projector (lines 14-20). To support a proper color blending, the first shadow map is rendered with a black shadow color and the assigned light color as background. It is used to create a base image during the first rendering iteration. All subsequent iterations generate masks with a black shadow color and a white light color. They are color blended (as sources) onto the base image (the destination) –e.g. in OpenGL– using glBlendFunc(GL_ZERO,GL_ SRC_COLOR).

```
1:  for all viewpoints i
2:      if i = 0 then LC=light color
3:      else LC=1,1,1
4:      set model-view-projection matrix to V_i
5:      render real content into depth buffer
6:      render virtual content into stencil buffer
7:      render light in LC into frame buffer
        (previously cleared in 0,0,0)
8:      transfer frame buffer into texture
        memory T_i
9:      categorize real content into fully visible
        (Δ_fi), partially visible (Δ_pi) and hidden
        (Δ_hi) triangles
10:     render Δ_fi in LC into frame buffer (depth
        test disabled)
11:     transfer frame buffer into texture
        memory T_{i+max_vp}
12: endfor
13: set model-view-projection matrix to P, clear
    frame buffer in 0,0,0
14: for all viewpoints i
15:     if i ≠ 0 then enable color blending
16:     set texture matrix to V_i +
        normalization space correction
17:     render Δ_fi using projective texture T_i
18:     render Δ_pi using projective texture T_{i+max_vp}
19:     if i = 0 then render Δ_hi in light color
20: endfor
```

Algorithm 2

In line 9, the real scene is categorized for each viewpoint into fully visible, partially visible and hidden triangles. These sets are rendered sequentially in lines 17-19. Fully visible triangles are texture mapped with the corresponding shadow mask and possibly color blended with the base image (line 17). Hidden triangles are rendered in the light color for the first viewpoint (to generate the base image), or are not rendered at all for all subsequent viewpoints (line 19). Triangles that are partially visible have to be partially texture mapped. To realize this without having to apply a time-consuming re-triangulation, a second shadow mask is generated for each viewpoint (lines 10 and 11). Thereby, the original shadow mask is modified by rendering all fully visible triangles in the current light color on top of it (with the depth test disabled). Using this new shadow mask for texture mapping the partially visible triangles (line 18) ensures that the potential shadow area appears only on the visible portions of these triangles. The remaining part is then available for occlusion shadows of other viewpoints that can see these surface areas.

An efficient categorization of the triangles is achieved by storing and reusing the depth buffer that has been produced after line 5 is executed: Several sample points on a triangle are mapped from the world coordinate system into the screen coordinate system of the viewpoint. The calculated z-values of the transformed samples are then compared with the corresponding z-values, stored in the depth buffer. The depth buffer can be indexed using the computed x/y-coordinates of the transformed sample points. If all sample points have z-values that are closer to the viewpoint than the indexed values in the depth buffer, the triangle is categorized to be fully visible. If all points are further away than the indexed depth-buffer values, then the triangle is assumed to be hidden. If some points are closer and others are further away, then the triangle is partially visible.

Note that we currently apply these approximation by considering the three corner vertices and the center point of a triangle. All computations are cached and intermediate results are reused for triangles sharing the same vertices.

4.4. Multiple Projectors

Due to self occlusion, not all portions of the real content can be lit by a single light projector (e.g., surface 3 in figure 5). 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. A set of optimal projector positions can be determined by applying Stuerzlinger's [13] hierarchical visibility algorithm. To guarantee a uniform illumination, however, surfaces should not be lit by more than one projector at the same time. Otherwise, the projected light accumulates on these surfaces and they appear brighter than others. Note that this effect is not necessarily spurious, since it reflects the natural behavior of multiple light sources (i.e., the light projectors) that illuminate the same surface. Consequently, we propose a solution to this problem that can be applied optionally.

Our method subdivides the geometry of the real content into surface portions that are assigned to, and finally rendered by an individual light projector. Since this subdivision is view-independent, and we assume that the parameters of the light projectors and the real content

do not change over time, it can be pre-computed. Algorithm 3 describes the off-line subdivision process.

Each triangle of the real content's geometry stores the following properties:

- a flag that indicates whether the triangle is fully visible (*visible*), completely hidden (*hidden*), or partially visible (*partial*) from a projector;
- the ID of the projector for which the *visible* flag applies (a triangle can be assigned to be fully visible by only one projector);
- a bit string with *n* bits for *n* projectors, indicating for which projectors the triangle is partially visible (the bit positions correspond to the projectors' IDs).

Note that a triangle's visibility from a particular view point differs from its visibility from a particular light projector. In this section, we describe only how to render triangles depending on their visibility from the projectors' perspective, while section 4.3 describes this with respect to the perspective of the view points. This should not lead to confusion.

```
1:   initialize all triangles: Δ = hidden
2:   for all projectors i
3:     for all triangles j
4:       if Δ_j is fully visible from P_i
           A = Area (P_i, Δ_j)
5:       if Δ_j = hidden or Δ_j = partial or
6:         (Δ_j = visible from P_k {k < i} and A_Δj < A )
7:         Δ_j = visible from P_i
8:         A_Δj = A
9:       endif
10:    endif
11:    if Δ_j is partially visible from P_i and Δ_j != visible
12:    then Δ_j = partial from P_i
13:  endfor
14:  endfor
```

Algorithm 3

In line 1, all triangles are initialized to be *hidden* for all projectors. Every triangle (Δ_j) is then evaluated for each projector P_i. The algorithm assigns the following priority to the triangles: full visibility overwrites partial visibility, and partial visibility overwrites no visibility. Thus, a triangle will be assigned to be fully visible from the current projector (P_i) if (lines 4-10):

(a) it is fully visible from this projector *and*
(b) it has been previously assigned to be hidden or partially visible from another projector (P_k) *or*
(c) it has been previously assigned to be fully visible from another projector, but its projected area is larger from the current one.

Whether a triangle is completely hidden, partially visible, or fully visible from a specific perspective can be computed as described in section 4.3.

If a triangle is partially visible from the current projector and not fully visible by another one (lines 11-12), then it is assigned to be partially visible. In addition, the current projector is recorded in the triangle's bit string by activating the bit that corresponds to the projector's ID.

Finally, if a triangle is completely hidden from the current projector, nothing needs to be done and it remains hidden.

After all triangles have been assigned, a static shadow mask is generated for each projector. Therefore, only the fully visible triangles are rendered in a white light color on top of a black background – leaving the partially visible areas in black. The shadow masks are then read into separate blocks of the texture memory. Note that this is also part of the off-line pre-computation and has to be done only once. However, it is not explicitly outlined in algorithm 3.

During runtime, algorithm 1 (for a single view point) or algorithm 2 (for multiple viewpoints) are executed for each light projector separately (i.e., on different rendering hosts that are connected to a single light projector). The only modification to these algorithms is to restrict them to render only those triangles that have been assigned to the corresponding projector – not the entire real content. This affects only the underlying functionality of line 7 in algorithm 1, and lines 17-19 in algorithm 2. Note that a side effect of our approach is a distributed and potentially balanced rendering of the real content between different hosts.

In general, all assigned visible or partially visible triangles are rendered as described in algorithm 1 or algorithm 2. Partially visible triangles, however, require additional treatment: After being rendered for a particular projector in the discussed way (see sections 4.2 and 4.3), some of the static shadow masks that have been pre-computed for the other projectors are combined with the currently rendered image.

Technically, this is done exactly as for the multiple view points described in algorithm 2 – using projective texture mapping (but setting the texture matrix to P instead of V) and color blending.

If the blending function described in section 4.3 is used, white texture portions of these shadow maps do not effect the current image while black portions will erase the underlying image content. Consequently, previously lit portions are erased.

Specifically, the triangles' bit strings that indicate the set of projectors from which they are partially visible are evaluated. We define the following convention: Only the static shadow maps of those projectors whose IDs are smaller than the ID of the rendering projector have to be combined with the current image. Thus, we ensure that those portions of the partially visible triangles that have already been lit by a projector will be blocked for all other projectors.

Note that if a triangle is still marked as *hidden* after the subdivision, none of the projectors can illuminate it and it remains unlit. As long as the virtual light sources are located where the light projector is located, this case is treated properly. If the virtual light sources are located in arbitrary locations, these hidden triangles will potentially appear as incorrect shadow regions.

Figure 5 illustrates a simple example with two projectors (P1 and P2), one view point (V), and six real surfaces.

Using algorithm 3, we want to assume that surfaces 1 and 2 are assigned to be fully visible from P1, while surface 3, 5 and 6 are assigned to be fully visible from P2. Surface 4 is partially visible from both projectors. Consequently, surface 1 and 2 are only rendered from P1, and surfaces 3, 5 and 6 are only rendered from P2. P1 renders surface 4 first and illuminates portion *a*. Nothing else needs to be done for P1. Then P2 renders surface 4 and portion *b* is illuminated. Since projectors exist (i.e., only P1 in our case) that have previously lit a portion of surface 4[5], the static shadow masks of these projectors are blended with the current image.

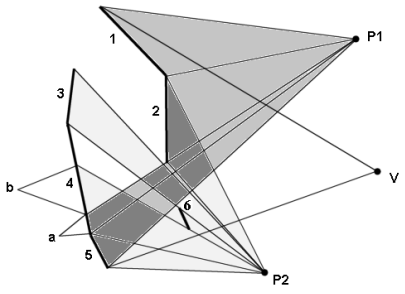

Figure 6: Visibility for different projectors.

In P1's static shadow mask, the image of portion *a* remains black while surfaces 1 and 2 are outlined in white. If mapped and blended into P2's image using the perspective texture transform of P1, portion *a* is erased in P2's image.

[5] This is determined by comparing the projector IDs in the triangle's bit string with the ID of the current projector.

4.5. Drawing Light and Shadow

Surfaces of real objects for which the geometry is not known can be illuminated by manually drawing static light and shadow effects into the frame- and stencil-buffers of the corresponding light projectors. This allows the user to paint directly on these surfaces. In figures 1,3,4 and 7, for instance, the illumination of the base's lower part is static. It has been interactively sketched using a simple mouse-based drawing tool. More advanced painting techniques (e.g., as described by Bandyopadhyay et al. [2]) can be applied to support a more artistic expression.

5. Implementation

We use multiple off-the-shelf PCs with hardware-accelerated graphics boards that are connected via a standard local area network.

Each PC is connected to a single video projector and is executing the same render client. While one client drives the CRT projector that displays the stereoscopic images, an arbitrary number of other clients can be used to control the light projectors to render the illumination. All clients store an instance of the current scene in their local memories. If multiple projectors are used, each client pre-computes the entire subdivision of the real content's geometry and generates the static shadow masks for all projectors, as described in section 4.4.

A server that continuously receives user data from a tracking device represents the beginning of the chain. This data is passed to the first client that renders the stereo overlays. The client adds information about scene changes (e.g., caused by user interaction) to the tracking data and passes it to the next client in the chain. Based on this information, the client optionally adds additional information, passes it to the next client and renders the illumination. This is continued until the end of the chain is reached.

The synchronization between the render clients is realized in software and integrated into the communication protocol. Note that rendering and communication are carried out in parallel (i.e., within individual threads created by each client) – not causing synchronization delays.

For the experimental setup displayed in figure 2, two PCs and two projectors are used. The tracking server is running on the same PC as the first render client. The first projector is a standard CRT projector rendering the left and right eye views of the virtual objects in stereoscopic mode. For the optimal setup, the light projectors need to project independent occlusion shadows for the left and right eye in sync with the scene projection. This would require stereo genlock and frame locking between the master and client render processes and graphics systems.

For our experimental setup, we use a standard DLP projector in monoscopic mode as the light projector. The occlusion shadows are therefore generated only for an idealized viewpoint in-between the left and right eye position and we use the same occlusion shadows for both eyes. In our case, the stereo disparity for the virtual objects is rather small, since the virtual projection plane (i.e., the table top surface reflected into the Virtual Showcase) is in close proximity to the actual location of the virtual objects. This small stereo disparity results in a small disparity of the occlusion shadows. In addition, we typically defocus the light projector slightly to blur the occlusion shadow boundaries. In our experience, we found the monoscopic occlusion shadows to work quite well and that the disparity errors are hardly visible. In addition, the graphics cards are not genlocked, which results in slight flicker of the illuminated areas on the real objects.

6. Summary and Conclusions

We presented a solution for the problem of generating realistic occlusion effects with optical see-through displays. We describe an illumination technique that uses video projectors to generate view-dependent shadows underneath virtual overlays, which are projected directly onto the surface of physical objects. We have shown how this method can be applied for a single viewpoint and a single light projector. We also describe the necessary extensions for multiple users and multiple light projectors. In addition we discuss the limitations of our multi-user approach and suggest two techniques for reducing the artifacts.

The precision of the calibration method described in section 4.1 depends on the resolution of the frame-buffer that is displayed by the light projector, the distance between the projector and the reference surface and its orientation with respect to the projector's image plane. Surfaces that are aligned roughly parallel to the projector's image plane (e.g., the base in figures 1,3,4) allow us to achieve sub-pixel precision since the fiducials can be referenced exactly in the frame-buffer. Surfaces that are oriented more perpendicular to the image plane (e.g., skull in figure 7) still produce an average precision of 1-3 pixels[6].

In addition, an object-based or image-based blurring can be applied to reduce the visual effects caused by slight registration errors, as described by Fuhrmann et al. [6]. Similar effects can be achieved by simply defocusing the projectors' optics – with no computational cost involved. We found that small displacements of the occlusion shadows are far less noticeable as displacements between real content and virtual overlay. This might be due to the fact that one's focus is on the visible overlays and on the real objects, rather than on the occlusion shadows that are in general invisible.

The rendering algorithm for multiple viewpoints, described in section 4.3 can be efficiently applied in combination with real objects that physically divide the viewing space shared by multiple observers (such as the skull in figure 7). In these cases, the interference between different occlusion shadows can be minimized or even avoided completely since large portions of the surface exist that are not necessarily visible for more than one observer. Objects such as the wooden base in figures 1,3,4 on the other hand, consist of large surface portions that are likely to be visible from all viewpoints. In these cases, our second algorithm becomes less efficient since a categorization of the real scene's geometry is not necessary.

Simply blending all the shadow masks together and projecting them onto the real scene results in correct occlusion effects but also causes interference between different occlusion shadows. However, as mentioned in section 4.3, the occlusion shadows that can be perceived behave exactly like umbral hard-shadows that are cast by the virtual objects onto the real scene. By computing the matching illumination on the virtual objects' surfaces using point lights at the positions of the viewpoints amplifies this illusion and reduces the interference effect.

The algorithm for multiple projectors, described in section 4.4 avoids the problem that the same portion of the real content is illuminated by more than one projector. This is achieved by subdividing the real content's geometry and assigning the resulting portions to individual projectors. Raskar et al. [12], for instance, present a cross-feathering method to merge the images of multiple projectors on a pixel-basis, which could also be used with our setup.

7. Future Work

Our current prototype has proven that occlusion shadows strongly enhance the realistic display of real and virtual objects in a shared space. Occlusion shadows might also be of use for a variety of other displays that are based on the optical see-through concept. Head-attached displays including head-mounted displays (HMDs), and stationary displays such as projection-based AR devices [4] benefit from our method. However, a light controllable real environment is a prerequisite for displaying a high quality mixed reality scenario. Such an environment is implicitly provided by the Virtual Showcase.

For the Virtual Showcase, we are working on new optical elements and rendering techniques to reduce the number of light projectors necessary to reach almost all

[6] With a frame-buffer resolution of 1024x768 pixels, and a distance between projector and reference surface of approx. 1.5m.

surfaces of real objects. Figure 8 illustrates a sketch of a potential optical configuration. We optically split up a single light frustum into multiple sub-frustums to provide a "surround illumination" with a single projector. However, this approach also splits the projector's resolution.

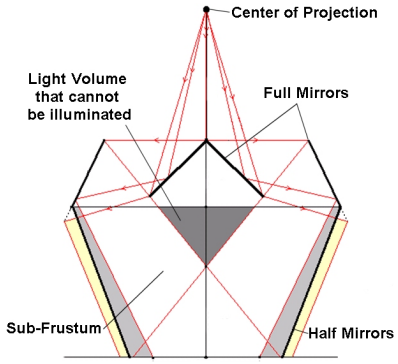

Figure 8: Sketch of a possible optical extension that splits a single light frustum into multiple sub-frustums providing a surround illumination with a single projector.

Our approach produces realistic occlusion effects between virtual and real objects. For the realistic display of mixed reality scenarios, consistent illumination of real and virtual objects is of great importance as well. As shown by Raskar et al. [12], real objects can be consistently illuminated with respect to virtual light sources using video projectors if their surface properties are known. We plan to apply this technique in the context of the Virtual Showcase to achieve a fully consistent and high quality display of mixed reality scenarios.

References

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

[2] Bandyopadhyay, D., Raskar, R., and Fuchs, H. Dynamic Shader Lamps: Painting on Real Objects. *In proceedings of IEEE and ACM International Symposium on Augmented Reality (ISAR'01)*, pp. 207-215, 2001.

[3] 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.

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

[5] 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.

[6] Fuhrmann, A., Hesina, G., Faure, F., and Gervautz, M. Occlusion in Collaborative Augmented Environments. *In proceedings of 5th Eurographics Workshop on Virtual Environments*, pp. 179-190, 1999.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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 (EGRW'01)*, 2001.

[13] Stuerzlinger, W. Imagine all Visible Surfaces. *In Proceedings of Graphics Interface' 99*, pp. 115-122, 1999.

Acknowledgements

The Virtual Showcase project is supported by the European Union, IST-2001-28610.

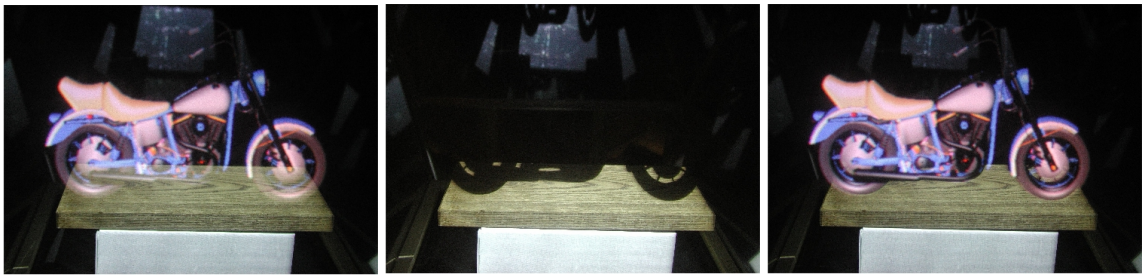

Figure 1: A normal illumination causes wrong occlusion effects between real objects and virtual overlays (left). Creating an occlusion shadow underneath the overlay prevents the light from being diffused on the real object's surface and transmitted through the optical combiner and the graphics (center). Overlaying the graphics over the shadow results in a realistic occlusion of the real object by the virtual one (left)[7].

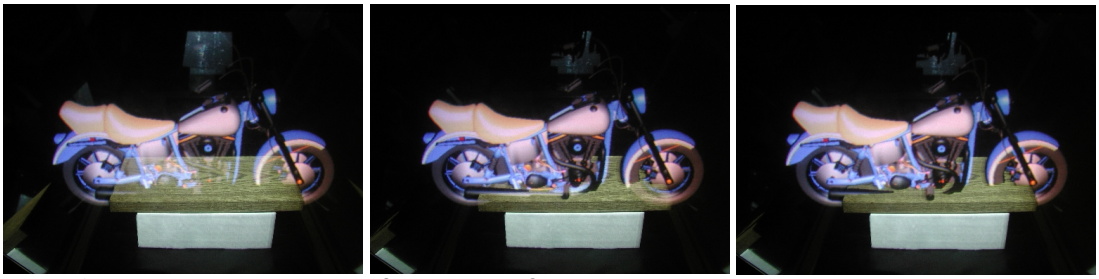

Figure 3: Knowing the depth information of the real content, also allows the occlusion of virtual objects by real ones (left). Combining this with our occlusion shadow method (center) creates correct mutual occlusion effects between both environments (right).

Figure 7: This example demonstrates our method in combination with a more complex real scene: A physical skull of a mid-cretaceous dinosaur (a *Deinonychus*) has been augmented with virtual muscles and bones (left) – generating occlusion shadows exactly underneath the virtual overlays. Covering the skull by a virtual skin (right) leaves most of the bone structure in shadow. Only the real teeth are clearly visible.

[7] Note that the photographs in this paper have not been touched up. They were taken from the observer's point of view, but were rendered monoscopically.