Compensating Indirect Scattering for Immersive and Semi-Immersive Projection Displays

Article · January 2006

DOI: 10.1109/VR.2006.34

CITATIONS
33

READS
176

5 authors, including:

Anselm Grundhöfer
Disney Research
28 PUBLICATIONS **586** CITATIONS

SEE PROFILE

Compensating Indirect Scattering for Immersive and Semi-Immersive Projection Displays

Oliver Bimber, Anselm Grundhöfer, Thomas Zeidler, Daniel Danch and Pedro Kapakos
Bauhaus-University Weimar

ABSTRACT

We present a real-time reverse radiosity method for compensating indirect scattering effects that occur with immersive and semi-immersive projection displays. It computes a numerical solution directly on the GPU and is implemented with pixel shading and multi-pass rendering which together realizes a Jacobi solver for sparse matrix linear equation systems. Our method is validated and evaluated based on a stereoscopic two-sided wall display. The images appear more brilliant and uniform when compensating the scattering contribution.

CR Categories and Subject Descriptors: I.3.3 [Computer Graphic]: Picture/Image Generation – Display Algorithms; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism – Radiosity / Virtual Reality

Additional Keywords: Compensating Indirect Scattering, Interreflections, Projection Displays, Reverse Radiosity, Image Correction, Real-Time Rendering, GPU Programming.

1 INTRODUCTION AND MOTIVATION

Today, a variety of different immersive and semi-immersive projection displays exist to support Virtual Reality (VR) applications. The pallet reaches from single-sided walls, over single- and two-sided workbenches to surround screen displays, such as cylindrical or conical displays [26], spherical domes and multi-sided CAVEs [9].

Beside such dedicated screens, other research activities focus on using projection-optimized [27] or unmodified [3] real surfaces for realizing embedded stereoscopic visualizations in everyday environments.

Several image correction techniques, such as geometric warping, photometric mapping (chrominance and luminance) and edge blending have been developed for multi-projectors displays. While some of them are implemented on dedicated hardware, others benefit from the continuously increasing capabilities of consumer graphics cards. A recent state-of-the-art overview on such techniques was presented by Brown, et al. [5].

Many projection screens are concavely shaped to cover a large portion of the visual field with graphics for giving the user an immersive experience. The light that is projected onto these screens is not only diffused towards the direction of the observer. A fraction of it scatters also to other screen portions – causing a blending with the directly projected image. This *indirect scattering* causes the displayed image to appear partially inconsistent and washed out (cf. figure 1). Specialized screen materials can direct the light towards a particular direction only, and consequently reduce the effects of scattering on other portions. However, such screens do not provide a consistent brightness for arbitrary perspectives.

Email: {bimber,grundhoefer,zeidler,danch,kapakos}
@uni-weimar.de

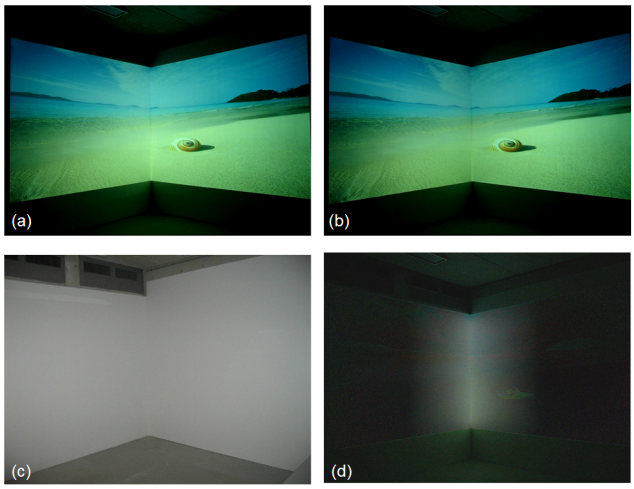

Figure 1. Front-projection onto two-sided wall display (c): Normal projection (a) and projection with compensated indirect scattering (b). The compensated scattering is visualized (d[1]) by subtracting photograph (b) from photograph (a).

In this paper, we introduce a method that minimizes these indirect scattering effects in real-time – leading to more brilliant and uniform images.

Our main contribution is the mathematical and algorithmic formulation of a *reverse radiosity* approach for compensating first-level and higher-level indirect scattering effects. We approximate the solution numerically with a Jacobi solver for sparse matrix linear equation systems. It is described how this method can be implemented directly on the GPU via pixel shading and multi-pass rendering. Efficient data structures are presented that allow packing the required information into textures which are processed by pixel shaders. Frame-buffer objects (FBOs) are used for a fast exchange of intermediate iteration results and enable computations with floating point precision. We explain quality and performance optimizations of the rendering algorithm, and discuss simplifications for the correction of stereo-pairs, as well as modifications to support light directing screen materials. Our method is finally validated and evaluated on a stereoscopic two-sided wall display. We provide information on limitations and potentials, as well as data on the performance and the quality of our correction method.

2 RELATED WORK

We want to review two main areas of related work: image correction techniques for projection systems, and hardware accelerated radiosity and inverse radiosity approaches.

[1]The resulting image is slightly enhanced in contrast.

2.1 Image Correction for Projection Displays

Many multi-projector rendering techniques exist that aim at creating consistent image geometry, intensity and color among multiple projection units. While geometric image warping (e.g., [1]) is important for curved-screen displays, such as cylinders, cones, and domes, photometric correction (e.g., [21]) and edge-blending (e.g., [28]) is essential for multi-projector configurations, such as tiled walls and tiled curved screens. Several of them benefit from programmable rendering pipelines to achieve real-time performance. Photometric mapping, as an example, has been realized with fragment programs to support color and brightness corrections for tiled wall displays and CAVEs that apply digital projection and Infitec's multi-band color filtering technology for stereo separation [18]. To discuss all of them in detail is out of the scope of this paper. Instead, we want to refer to a state-of-the-art overview that is presented in [5].

For high-dynamic range visualizations in CAVEs, Dmitiev et al. have developed a customized tone-mapping technique [10] that (in addition to other parameters) applies foveal adaptation [24] relative to the screen portion which the observer is looking at. This is done instead of executing tone-mapping independently for each projector, which would lead to inconsistencies in the corners.

Crosstalk reduction techniques for active [17,20] and passive [16] stereo displays are other correction examples that estimate the ghost images for the left and right views, and subtract them from the opposite views before displaying them. Fixed-function pipeline rendering (e.g., standard OpenGL) is used for achieving interactive frame rates [16]. To support non-linear correction functions in real-time, however, the application of programmable GPUs is obvious, but –to our knowledge– not yet put into practice.

Recent work on radiometric compensation [2,13,22,27] uses cameras in combination with projectors for measuring the surface reflectance as well as the contribution of the environment light. These parameters are then used for correcting the projected images in such a way that blending artifacts with the underlying surface are minimized. This allows using projection screens with imperfections or even arbitrary (i.e., geometric and radiometric complex) surfaces.

Our goal is to compensate *global illumination* effects in the projected images that result from indirect scattering. Previously presented correction techniques, such as the ones summarized above, correct *local illumination* effects that result from inconsistent direct projection. Although they are not immediately related, they can all be carried out as a pre-process of our method.

2.2 Hardware Accelerated Radiosity and Inverse Radiosity

In computer graphics, matrix radiosity is computed by solving a linear equation system with a finite element model [11]. Numerical techniques, such as Jacobi and Gauss-Seidel iteration have been used for this. Throughout the past decades, data structures and rendering algorithms have been optimized and adapted to the evolution of graphics hardware to reach interactive frame rates.

The hemicube method by Cohnen and Greenberg [7], for instance, was a first example for speeding up visibility and form factor computations using the depth buffer. Kelle, as another example, used hardware-accelerated OpenGL to speed-up radiosity computations without following a finite element approach [14]. He performed a quasi-Monte Carlo particle simulation on the CPU and used local illumination with point light sources at the corresponding particles' positions.

Recent methods achieve interactive frame rates by implementing the corpus of computations on modern GPUs. Only a few pre-computations are still carried out on the CPU. Jacobi iteration is particularly well suited for an implementation on GPUs' streaming-oriented SIMD architectures, but the Gauss-Seidel method requires less iterations. In addition, floating point textures allow encoding all parameters with the required precision. More information on matrix solvers and linear algebra for GPUs is provided in [4,19].

Carr et al., have realized radiosity with a Jacobi solver implemented as a pixel shader [6]. A meshed texture atlas is used for re-sampling and mapping the computed radiosity to the scene.

While Carr et al. pre-compute form factors, Coombe et al. have realized a progressive refinement radiosity algorithm that performs the majority of computations on the GPU – including visibility and form factor computations [8]. While the patch visibility is determined in a shooting-oriented manor with a vertex shader, the form factors as well as the radiosity are computed for the receiving patches in a gathering-oriented way with a pixel shader. An adaptive subdivision is realized on the CPU with texture quad-trees.

Inverse radiosity [29] is an inverse global illumination technique that approaches to recover the diffuse surface reflectance from Lambertian environment patches. It requires that the geometry, the lighting conditions and the radiance distribution are known. Photographs of the environment's surfaces with light sources, as well as high dynamic range techniques are used to capture the radiance distribution. If this is known, the diffuse albedo of all surface patches can be computed analytically.

In contrast to this, our algorithm compensates the amount of light that is indirectly scattered from other surfaces. It does so by iteratively solving a linear equation system that consciously estimates the scattering under a given direct illumination situation. The subtraction of these estimates from the direct illumination leads to a progressive compensation of the scattering. To avoid confusions with inverse radiosity, we want to refer to this process as *reverse radiosity*.

We assume that the surfaces don't change over time. Consequently, visibility and form factors can be pre-computed. Pixel shading and multi-pass rendering are applied to implement Jacobi iteration for solving a sparse matrix linear equation system that results in a new *direct illumination*. The main difference to traditional radiosity methods that compute *radiance* (or *radiosity*), or to inverse radiosity techniques that estimate *reflectance* is that we compute the direct illumination that has to be projected to compensate visible scattering.

3 Reverse Radiosity for Scattering Compensation

As mentioned earlier, concavely shaped projection screens scatter a fraction of light to other screen portions. The amount of indirect illumination adds to the directly projected image. A simple two-patch front-projection example is illustrated in figure 2 (this applies also to back-projection):

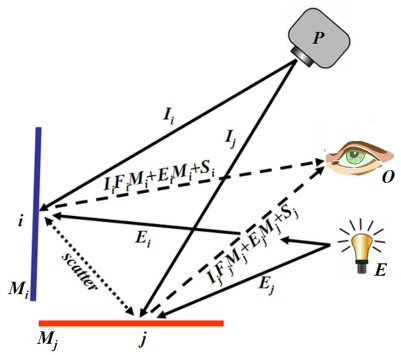

Figure 2. Scattering of light between two surface patches.

Assuming the projector to be a point light source, the energy fraction F of light I that leaves the projector and arrives at a screen patch (i or j) depends on the geometric relation between the projector and the surface. A simple representation of what is known as *projector-to-surface form factor* can be used for approximating this fraction: $F=f*dA*cos(\alpha)/(r^2*\pi)$ where α is the angular correlation between the light beam and the patch normal, dA is the patch's differential area, and r the distance (square distance attenuation) between the light source and the patch surface. The scaling factor f allows an additional fine-tuning to cope with unknown projector properties (e.g., exact intensities).

The direct illumination is reflected depending on the material reflectance M of the screen patches. In addition to the reflected environment light EM, the reflected indirect scattering S contributes to the final image that is perceived by an observer (O). Note, that in contrast to the direct illumination I, the environment light contribution E cannot be controlled. It contains stray light coming from the surrounding room (i.e., in case of semi-immersive projection displays, such as two-sided workbenches) as well as the black-level of the projector.

Problematic is the fact that the amount of scattering depends on the projected image I and vice versa. As explained earlier, the reverse situation (i.e., the global radiosity of each surface element is determined based on a known illumination I) is computed by solving the well known radiosity equation system[2] (either on the CPU or on the GPU):

$$B_i = E_i + M_i \sum_{j,j\neq i}^{P} B_j F_{ji} \tag{1}$$

where B_i is the radiosity and M_i the reflectance of patch i, and F_{ji} denotes the *patch-to-patch form factor* that describes the amount of radiosity which is transferred from patch j to patch i.

Below, we describe a *gather-oriented reverse radiosity* method which modifies I in such a way that indirect scattering effects are compensated. We want to assume projection screens with Lambertian reflectance. The form factors and the visibility can be computed off-line.

3.1 First-Level Scattering

We subdivide the entire screen surface into P discrete uniform patches (details on the data structure are presented in section 3.5. If the screen geometry is known, the patch-to-patch form factor can be computed with $F_{ji}=f*dA_i*cos(\alpha_i)*cos(\alpha_j)/(r_{ji}^2*\pi)$, where α_i and α_j are the angels between the ray connecting the two patches' centers and their normals, r_{ji} is the patch-to-patch distance, and dA_i is the differential area of patch i. The scaling factor f allows an additional fine-tuning again to cope with unknown screen properties, and to encode patch-to-patch visibility (f=0 if invisible).

The amount of radiance that scatters indirectly to patch i as a result from directly illuminated patches j can be described as (with respect to figure 2):

$$S_i = \sum_{j,j\neq i}^{P} \left(I_j F_j M_j + E_j M_j\right) F_{ji} M_i \tag{2}$$

We want to refer to this as the *first scatter level*. Physically, however, light is scattered back and forth between the patches. How to take these higher scatter levels into account is described in section 3.2.

To compensate *first-level scattering*, the following computation can be performed:

$$I_i^{l+1} = \frac{1}{F_i M_i}\left(R - E_i M_i - S_i^1\right) \tag{3}$$

where R is the original image to be displayed.

[2] Gathering-oriented, and re-formulated in our notation.

As mentioned above, the difficulty in this case is that the projected image I depends on the amount of scattering S and vice versa. A simple analytical solution to equation 3 does not exist. Rather than that, this linear equation system can be solved numerically by approximating I through several (l) iterations:

```
compute I^0 //direct illumination
l=0 //number of passes
repeat
        compute S^l (for I^l) //eqn. 2
        compute I^(l+1) with S^l //eqn. 3
        l=l+1
until exit condition is fulfilled
display I^(l+1)
```

Algorithm 1. Compensating first level scattering.

Initially, I^0 is computed for the direct case. With $I^0=(R-EM)/FM$ the environment light and projector's form factor contribution (i.e., the non-linear luminance distribution from the image center to its edges) can be compensated. Thereby, R is the original image to be displayed. If this is not necessary, $I^0=R$ assumes a constant luminance distribution and a dark environment.

The result I^0 is used in the first iteration to compute a first estimate of scattering, and to compensate the environment light and the approximated indirect illumination (equation 3). The result I^1 of this first iteration is then used in the second iteration, and so on. This process is repeated until I converges (i.e., the results of two consecutive iterations do not reveal significant differences) or the solving process exceeds a predefined maximum time slot. Details on appropriate exit conditions are explained in section 3.3.

A numerical solution to equation 3 can be computed directly on the GPU via pixel shading and multi-pass rendering that together realizes a Jacobi solver for sparse matrix linear equation systems. The shaders compute equations 2 and 3 in each pass and update I successively. After the exit condition is fulfilled, the final image is displayed. This is summarized in algorithm 1 (pixel shaders are highlighted in blue).

To efficiently exchange the intermediate results of different passes, I is rendered off-screen into an FBO that is then bound to the shaders of the following pass. This avoids time consuming read-back operations and enables computations with floating point precision.

Note that E, M, and F are constant. They do not directly appear in algorithm 1, but have to be considered when applying equations 2 and 3. They are passed as parameter textures to the corresponding shaders.

3.2 Higher-Level Scattering

Equation 4 presents an example that extends equation 2 to the first two scatter levels for a patch i (i.e., the radiance that is scattered directly from the patches j to patch i, and the radiance that is scattered from all patches k over all patches j to patch i):

$$S_i = \sum_{j,j\neq i}^{P} \left(I_j F_j M_j + E_j M_j\right) F_{ji} M_i + \sum_{j,j\neq i}^{P} \sum_{k,k\neq j}^{P} \left[\left(I_k F_k M_k + E_k M_k\right) F_{kj} M_j\right] F_{ji} M_i + \ldots \tag{4}$$

In the general case, this can be rewritten into a recursive form:

$$S_i^h = \sum_{j,j\neq i}^{P} S_j^{h-1} F_{ji} M_i \tag{5}$$

Note, that S_i^h indicates the *h*-th scatter level at patch *i* (i.e., the reflected radiance at patch *i* that can be contributed to scattering which arrives over *h-1* intermediate patches). The first scatter level (S_i^1) is defined in equation 2. With respect to equation 3 the *scattering of the first H levels* can be compensated with:

$$I_i^{l+1} = \frac{1}{F_i M_i}\left(R - E_i M_i - \sum_{h=1}^{H} S_i^h\right) \quad (6)$$

Thereby, I^0 is the initial direct illumination as explained in section 3.1.

Since equation 5 shows that higher scatter levels can be derived from lower scatter levels, all of them can be computed sequentially for the same direct illumination I^l within each render pass *l*. Having all scatter levels computed, I^{l+1} can be updated according to equation 6. Finally, if the exit condition is fulfilled, the result is displayed. This is summarized in algorithm 2.

```
compute I^0 //direct illumination
l=0 //number of passes
repeat
    compute S^{1,l} (for I^l) // eqn. 2
    for h=2 to H //scatter level
        compute S^{h,l} from S^{h-1,l} // eqn. 5
    endfor
    compute I^{l+1} with S^{1..H,l} // eqn. 6
    l=l+1 //number of passes
until exit condition is fulfilled
display I^{l+1}
```

Algorithm 2. Compensating first *H* scatter levels.

As for the implementation of algorithm 1, pixel shaders (highlighted in blue) perform the main operations for each patch, and FBOs are used to exchange intermediate textures (*I* and *S*) to avoid texture read-backs.

3.3 Exit Conditions

We have implemented two complementary exit conditions for algorithms 1 and 2:

The first exit condition terminates the solver if the images resulting from two consecutive passes do not differ by more than a predefined threshold (e.g., by more than one gray scale unit). Such an exit condition ensures a *constant quality* of scattering compensation over time and can be efficiently implemented on the GPU by using *occlusion queries*: A pixel shader can compare the difference between the two consecutive frames and discard pixels whose corresponding r,g,b differences lie below a predefined *pixel-discrepancy threshold*. All other pixels are written into an FBO[3]. After this, the query result delivers the number of diverging pixels that have been written into the FBO. This number is compared with a predefined *image-discrepancy threshold* which leads to the decision to exit the solver and display the result or to continue solving.

The second exit condition terminates the solver if the solving process exceeds an assigned time slot. This ensures a *constant solving rate* for scattering compensation. One possibility is to measure the amount of time that is required for the first solving pass initially. Since, in theory, this is constant for all passes, the maximum number of passes that fit into the provided time slot can be determined. However, due to potential hardware-dependent irregularities, such as caching operations, the time required for each pass might differ slightly. An alternative is to measure the time for each pass individually and subtract it from the time slot until it is exhausted.

Both exit conditions can be combined. Thereby the solver is determined if either the image quality does not change significantly *or* the time slot has been exhausted.

3.4 Interactive Rendering

It is clear that algorithm 1 or algorithm 2 cannot be solved in real-time if the patch resolution *P* equals the pixel resolution of the original image to be projected. Consequently, all textures (*E*, *I*, *S*, *M*, *L*, *R*, and F_i) have to be down-sampled to a resolution that can be processed at interactive rates. The final projection image can then be computed as follows:

$$I_Q^{l+1} = I_Q^0 - \uparrow\left(\downarrow I_Q^0 - I_P^{l+1}\right) \quad (7)$$

Thereby, I_Q^0 is the initial direct illumination image in the original image resolution (*Q*), I_P^{l+1} is the solved image that compensates scattering in patch resolution *P*, and $\uparrow$ and $\downarrow$ indicate image operators for up- and down-sampling (from *Q* to *P* and vice versa) with bi-linear or bi-cubic filtering. The result I_Q^{l+1} that contains the compensation of the direct illumination as well as of the scattering is finally being displayed.

Note that simply up-sampling I_P^{l+1} into the original image resolution and displaying it directly would lead to a visible patch structure in the projected image. These artifacts are avoided by subtracting the up-sampled scattering portion $\uparrow(\downarrow I_Q^0 - I_P^{l+1})$ with the direct illumination in the original resolution (I_Q^0).

Note also that only patches with patch-to-patch form factors larger than a predefined threshold are selected. This leads to additional resources and performance optimizations and does not affect the outcome's quality significantly. Thus, the patch-to-patch form factor matrix F_{ji} is a *sparse matrix* and not completely filled. This is explained in more detail below.

For stereoscopic rendering, it is sufficient to blend the stereo-pairs into a single image that is used by a single solving step, instead of solving the left and the right stereo image individually. Due to the small difference caused by the parallax shift in the down-sampled image, the variation of both solutions is neglectably small. A single solving step, however, increases the overall rendering performance. The solved result is up-sampled and applied to both stereo images, as explained above. An appropriate blending function is *(left+right)/2*.

3.5 Optimized Data Structures

The down-sampled textures are in the patch resolution *P*. They are two-dimensional because 2D textures are optimized for pixel shaders. Their corresponding values at each pixel can be addressed with the same patch index *u,v*. While F_i contains single projector-to-patch form factor values, the other textures store r,g,b values.

Especially the patch-to-patch form factor matrix F_{ji} can become as large as P^2. This does not only lead to performance issues of the above algorithms, but also to storage problems. As mentioned earlier, F_{ji} is a sparse matrix, and the number and distribution of entries depends on a pre-selected *patch-to-patch form factor threshold*. Only patches that scatter enough energy to another patch are considered. All other entries are left empty. To avoid storing the empty entries, the remaining patch-to-patch form factors are compressed into a single texture by concatenating them as illustrated in figure 3.

[3] Rendered off-screen (not displayed).

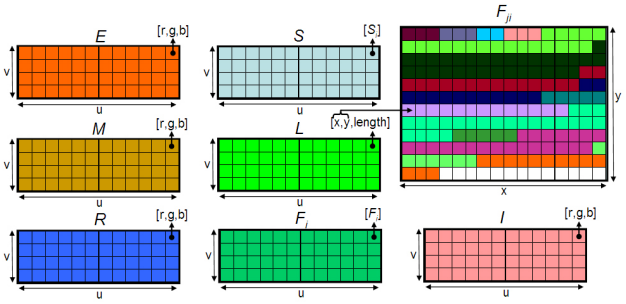

Figure 3. Data structures: The horizontal and vertical patch resolutions can be arbitrarily chosen and are parameterized over the screen surface. A look-up texture maps from *u,v*, space to the *x,y* space of a texture that stores the patch-to-patch form factors.

A look-up texture L allows mapping them back to individual patches. Thus, to find form factor components of all patches that scatter enough light to patch *u,v* the entry $L(u,v)$ gives the *x,y* coordinate of the first scattering patch as well as the number of following scattering patches (*length*). The texture F_{ji} is composed automatically during an offline procedure. Once computed, its last row can contain X-1 empty entries in the worst case, if X is F_{ji}'s horizontal resolution. The horizontal texture resolution cannot exceed the maximum supported texture size (currently X=4096 for Nvidia or X=2048 for ATI). However, it should be chosen in such a way that the number of empty entries is minimized.

Note that all data structures and data values are relative to the entire screen surface. Thus the screen surface has to be parameterized over u and v to become compatible with our matrix radiosity approach. This is simple for parametric surfaces, such as spherical or cylindrical displays. For multi-plane displays, such as two-sided workbenches, or CAVEs, the individual projection screens have to be parameterized altogether within the same *u,v* space. For our two-sided wall display, as an example, one screen surface can be parameterized with [u=0..0.5, v=0...1], and the other one with [u=0..0.5, v=0..1]. Further screen surfaces can be encoded accordingly.

The final image is displayed by texture-mapping it onto a geometric representation of the screen surface, and rendering this scene from the perspective of each projector. If multiple projectors are overlapping, proper cross-fading between the images has to be ensured. This requires an intrinsic and extrinsic registration of all projectors in 3D space. For simple surfaces, such as planes, the final image can simply be texture mapped onto an image grid and interpolated in screen space. The registration can be conducted in 2D space by manually selecting key points (i.e., grid points) directly in the screen space of each projector and assigning them to the corresponding texture coordinates manually. Note that these are common registration techniques for projection displays. While image warping is sometimes realized directly in the hardware of professional projectors or with external devices, it can also be carried out by the graphics card.

For performance reasons, we use 16bit floating point textures and FBOs instead of a slower, but more precise 32bit depth.

3.6 Light-Directing Projection Screens

Some projection screens are designed to radiate more of the projected light towards their normal direction rather than the same amount in all directions. Holographic projection screens, actively shuttered projection screens with phase dispersed liquid crystal layer (as used for the blue-c display [12]), or BARCO's diffuse and bright (DAB) screens are examples.

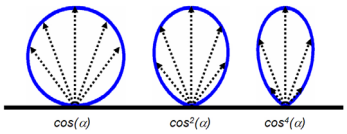

Figure 4. Angular dependency of form factors for regular projection screens (left), and exponentiated cosine function for approximating diffuse light directing projection screens (center and right).

The behavior of such screens can be approximated by our algorithm with a simple exponentiation of the cosine functions when computing the patch-to-patch and projector-to-patch form factors (cf. figure 4). The normal vector of the patches can also be tilted to simulate an alternation of the radiation's principle direction.

4 Evaluation

In this section, we want to validate our method and analyze its performance and quality parameters. The evaluation has been carried out under the following conditions:

The two-sided wall-display (figure 1c) serves as a testbed for our experiments. Two InFocus DepthQ active-stereo DLP projectors with a maximum resolution of 800x600 pixels each are applied to display stereo-pairs on each of the 3mx2m walls. A 2.8Ghz Pentium IV with 1GB RAM and a Nvidia FX3400-Quattro (PCI-Express with 256MB on-board memory) drive the projectors. For head-tracked applications, we apply an Origin DynaSight infrared tracking system.

We assume that the projection screens are perfectly white and have a Lambertian reflectance. We also ignore the projectors' black-level and consider the surrounding environment to be completely dark. Consequently, we use M=1, E=0, and all form-factors are computed with a cosine exponent of 1.

4.1 Validation of Method

To validate our method, and to show its potentials as well as its limitations, we have projected a light pattern consisting of the primary colors (red, green and blue) and their complements (magenta, yellow and cyan) onto a corner section of the walls (cf. figure 5a).

A high-resolution digital camera was used to capture a photograph of the reflected light before (U) and after (C) correction. These two photographs have been taken under exactly the same conditions (i.e., from the same perspective and under the same camera settings). When subtracting C from U the amount of correction can be visualized (cf. figure 5b).

It is worth discussing the effects that appear in figure 5b in a bit more detail:

As shown in figure 5a, the primaries are projected onto the right wall while their complements are displayed on the left wall. The projected background surrounding the color patterns is white.

We want to differentiate between the directly projected light D, and the light S that is scattered. For our interpretation of figure 5b, we want to consider only colors (not intensities) and just the first scatter level.

Scattering can be compensated efficiently on screen portions whose direct illumination D is *equal or larger* than the indirect illumination S scattered onto them in all color channels.

If this is not the case, the compensation of scattering will lead to clipping in one or more channels.

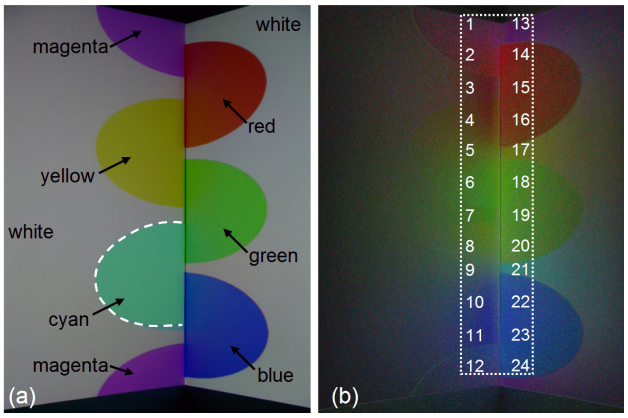

Figure 5. Projected primaries and complements (a). Subtracting a photograph of the corrected projection from a photograph of the uncorrected projection results in figure 5b[1].

Figure 5b has been divided into 24 areas (Π). The parameters D and S, as well as the result of the correction (U-C) within each area are outlined in table 1. Let us discuss an example:

The direct illumination in areas 14, 15, and 16 is red (r,g,b=1,0,0). On the opposite side of this color pattern we have a direct illumination of magenta (1,0,1) in area 2, white (1,1,1) in area 3, and yellow (1,1,0) in area 4. The scattering onto areas 2-4 is red. Since the direct illuminations of areas 2-4 are all larger or equal to the amount of scattered light, the scattering can be completely compensated. The result in U-C must consequently be the scattered light only. This is obviously the case, since the red bleeding in areas 2-4 is well visible. The scattered light can be fully compensated in all areas on the left side, except in areas 1, 5, and 9 (due to the larger white scattering from the right side). If we consider the opposite situation, area 2 scatters magenta, area 3 scatters white, and area 4 scatters yellow onto the red areas (14-16). In all these three cases, the compensation of the scattering is clipped since only the red channel can be further reduced. Consequently, U-C will reveal only a difference in the red color channel, which is also well noticeable in figure 5b. This is also the case for the other two primaries on the right side.

Table 1 presents the effects for all areas and highlights the clipping situations on both sides in bold. The framed area in table 1 corresponds to the framed area in figure 5b. It can be noticed that our theoretical thoughts are reflected in reality.

Note that the example in figure 5 is certainly an extreme one. In general the displayed scenery causes much less clipping. In many cases the colors on opposite sides are similar or even equal, and scattering can be removed efficiently (cf. figures 1 and 7, for example).

It is important to mention that compared to the uncorrected image, the visual quality of the corrected image will never degrade. In extreme situations (e.g., two different primaries bleeding onto each other), a scattering cannot be removed completely in all color channels. In such cases the corrected image does not differ much from the uncorrected one.

4.2 Performance

Figure 6 illustrates the solving performance of our method with a disabled performance-based exit condition. It was measured on the hardware described above, and by rendering the scene shown in figure 7 for the two-sided wall display. The scene is rendered four times for stereo projection, (two stereo images for two image planes). The native image resolution is 1600x600 pixels (800x600 pixels for each projector in a horizontal alignment).

Π	S	D	U-C	U-C	D	S	Π
1	1,1,1	1,0,1	**1,0,1**	1,0,1	1,1,1	1,0,1	13
2	1,0,0	1,0,1	1,0,0	**1,0,0**	1,0,0	1,0,1	14
3	1,0,0	1,1,1	1,0,0	**1,0,0**	1,0,0	1,1,1	15
4	1,0,0	1,1,0	1,0,0	**1,0,0**	1,0,0	1,1,0	16
5	1,1,1	1,1,0	**1,1,0**	1,1,0	1,1,1	1,1,0	17
6	0,1,0	1,1,0	0,1,0	**0,1,0**	0,1,0	1,1,0	18
7	0,1,0	1,1,1	0,1,0	**0,1,0**	0,1,0	1,1,1	19
8	0,1,0	0,1,1	0,1,0	**0,1,0**	0,1,0	0,1,1	20
9	1,1,1	0,1,1	**0,1,1**	0,1,1	1,1,1	0,1,1	21
10	0,0,1	0,1,1	0,0,1	**0,0,1**	0,0,1	0,1,1	22
11	0,0,1	1,1,1	0,0,1	**0,0,1**	0,0,1	1,1,1	23
12	0,0,1	1,0,1	0,0,1	**0,0,1**	0,0,1	1,0,1	24

Table 1. Analysis of effects shown in figure 5 within the 24 areas (Π): The direct illumination (D) and the first-level scattering (S) are indicated in r,g,b triplets. The effects in the framed area of figure 5b are shown in the framed section (U-C). The highlighted values indicate clipping. Only colors (not intensities) are considered.

The blue lines are the measurements for the monoscopic mode – rendering and solving one image per frame. The green lines are the measurements for the stereoscopic mode, rendering left and right stereo images, solving one image containing the blended stereo pair, and applying the result to both stereo images. The different line types illustrate the performance for varying patch resolutions.

The first entry is the frame rate (y-axis) achieved without scattering compensation. The time required to render the scene without compensation was 100fps (mono) and 55fps (stereo). The scatter level is increased over the x-axis. It can be observed that the frame rate drops with an increasing patch resolution. A patch resolution of 64x64 patches leads to a maximum solving performance of around 20fps (for both –mono and stereo). Lower resolutions, such as 32x32 and 16x16 patches led to maximal frame rates of 48fps/35fps and 63fps/42fps (mono/stereo).

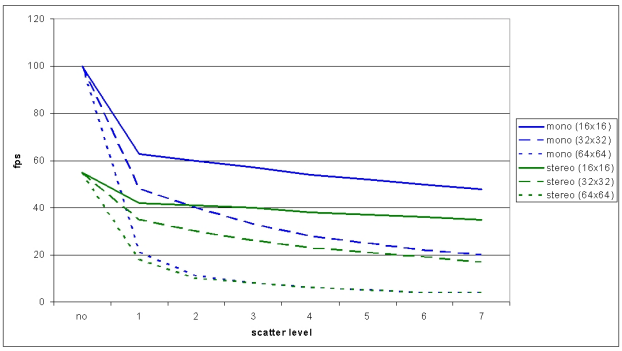

Figure 6. Solving performance measurements for monoscopic and stereoscopic two-sided projection and different patch resolutions.

For a resolution of 16x16 patches, the loss of performance was between 23.6% (1 scatter level) and 36.3% (7 scatter levels) for our example running in the compensated stereo mode (compared to the uncompensated stereo mode). For a patch resolution of 32x32, the performance loss was between 36.3% and 69% for the same mode. As it will be explained in section 4.3, a correction of scattering above level 2 does not lead to significant improvements. Note that although the performance of our method is independent of the scene complexity, the time for rendering the scene has been included in our measurements.

4.3 Quality

It is interesting to make some quantitative statements on the amount of compensated scattering. This can be done by comparing the histogram values of the difference photographs (*U-C*) over different scatter levels with the histogram values of the photograph showing the uncorrected projection.

If we do this for the examples shown in figures 1 or 7, the average amount of compensated first-level scattering (estimated over the entire image region) is approximately 9% (luminance) relative to the uncorrected projection. Note, that the scattering is regionally different. It is larger at screen intersections, or at the screens' lines or points of symmetry. Thus, approximately 28% of scattering has been compensated for the 50cm zone surrounding the screen intersection in figure 1.

If the second-level scattering is considered as well, we achieve an additional compensation of ~1% (estimated again over the entire image region). If we then perform a correction of up to level 7, we receive a further total compensation of only ~0.5% (i.e., from level 2 to level 7). This indicates that a compensation of scatter levels larger than 2 does not lead to large qualitative enhancements that would justify the corresponding loss of performance.

If the patch resolution is sufficiently high (e.g., >=16x16 in our experiments) a further increase will not change the amount of scattering compensation significantly. Instead, it will influence the precision of our method. In figure 5b, for example, a small color seam is visible directly on the left side of the screen edge. This is due to a too coarse resolution of 16x16 patches. When increasing the patch resolution it disappears. Note that the color seam is amplified in the difference photograph, and that it is barely detectable in the corrected projection.

5 Summary, Conclusion and Future Work

Indirectly scattered light that adds to directly projected light regionally washes out the displayed images. We have presented a method for compensating indirect scattering effects that occur with concavely shaped immersive or semi-immersive projection displays. The images appear more brilliant and uniform when compensating the scattering contribution (cf. figures 1 and 7). The human visual system is very sensitive to even small variations in brightness. Note, that these slight differences cannot be reproduced well with the low dynamic range photographs presented in this paper, but quantitative statements have been made in section 4.3.

An active stereo application running on a two-sided wall display at 55fps without correction drops to 40fps when our correction method is turned on. The correction, however, compensates ~10% (luminance) first- and second-level indirect scattering over the entire image area and ~30% at the 50cm zone close to the screen intersection. We believe that this is an acceptable tradeoff – especially when considering further accelerations enabled by upcoming graphics hardware. A correction of scatter levels larger than 2 did not result in significant improvements in our experiments.

The amount of compensated scattering depends on the display surface and on the presented content. A corrected image will not degrade in quality compared to the uncorrected image. If only little scattering can be compensated due to a potentially dark content, a difference between a corrected and an uncorrected case might not be detectable. By analyzing the image content first, we can extend our algorithm to automatically turn the correction off in such situations. This allows gaining additional performance resources for scene rendering.

Compared to a uniform patch resolution, an adaptive subdivision will increase the performance and enhance the precision of our method. An extension of our algorithm and data-structures towards adaptive subdivision lies ahead of us.

For displays that are driven by multiple rendering nodes, our algorithm could be distributed and solved by multiple GPUs in parallel. A pre-analysis of the surface structure allows determining which portions actually influence each other, and which don't. This makes a clustering of interacting patches possible and leads to multiple independent correction tasks that can be carried out by different rendering nodes.

Our experiments are representative for two-sided screen displays, such as two-sided walls or L-shaped workbenches. To evaluate our method for other display types, such as multi-sided CAVEs, cylindrical/conical, and spherical displays, as well as for geometrically complex surfaces belongs to our future work. Figure 8 illustrates a simulation of our method for such systems.

For a corrected projection onto geometrically and radiometrically complex display surfaces (e.g., textured everyday surfaces) [3], the surface parameters –including the patch-to-patch light transfer– have to be measured via structured light projection and camera feedback. Sen, et al. [25] presented a method that measures the entire 6D light transport through a scene by using a projector and an array of cameras. Reducing this problem to view-independent diffuse surfaces the measurements of reflectance, patch-to-patch form factors, and projector-to-patch form factors should be feasible from a single camera position and within an acceptable amount of time.

Finally, the reduction of scattering at sending patches will also be a challenging extension of our current method – which compensates scattering at receiving patches. This, however, will partially change the intensity and color of the original image. A good balance between the compensation of gathered light and the reduction of scattered light has to be found.

Several VR applications intend to simulate and visualize realistic real world lighting appearance –such as illumination engineering, interior design, or virtual prototyping– with

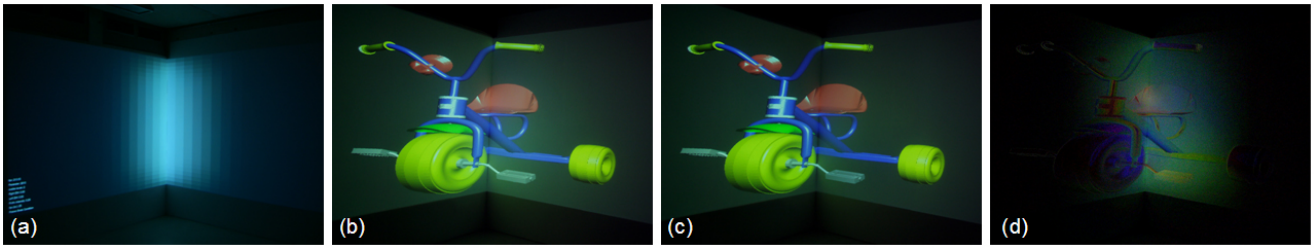

Figure 7. Computed, summed and intensity amplified form factor patches projected onto a two-sided wall display (a). Uncorrected (b) and corrected (c) three-dimensional object. The object does not contain primary colors and the background is light gray. Approximately 9% (luminance) scattering can be compensated. Difference photograph (b-c) shows the amount of correction (contrast amplified by 70%).

immersive or semi-immersive projection displays (e.g., [15]). Much effort is spent on rendering nearly photorealistic global and local illumination effects that arise in the rendered scene. Physical illumination effects that occur on the display's surfaces, however, have also to be taken into account for creating an ultimate lifelike experience.

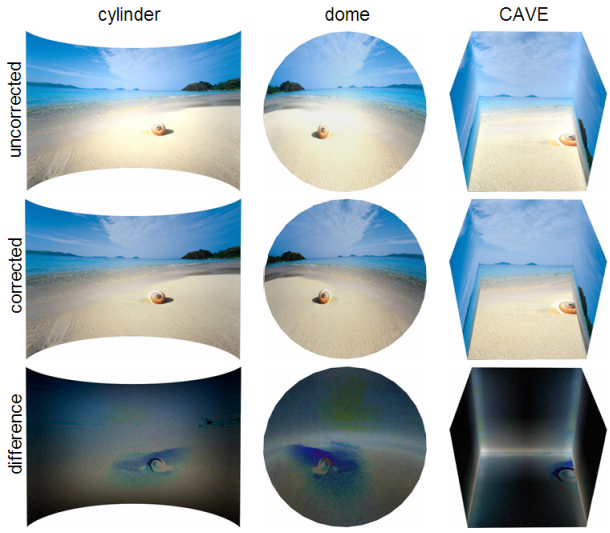

Figure 8. Simulated reverse radiosity for cylindrical display (left), dome (center), and four-sided CAVE (right). The difference images at the bottom are contrast enhanced by 70%.

REFERERNCE

[1] J. van Baar, T. Willwacher, S. Rao, and R. Raskar, Seamless Multi-Projector Display on Curved Screens, Eurographics Workshop on Virtual Environments (EGVE'03), pp. 281-286, 2003.

[2] O. Bimber, A. Emmerling, and T. Klemmer. Embedded Entertainment with Smart Projectors, IEEE Computer, pp. 56-63, vol. 38, no. 1, 2005.

[3] O. Bimber, G. Wetzstein, A. Emmerling, and C. Nitschke. Enabling View-Dependent Stereoscopic Projection in Real Environments. In proc. of IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR'05), 2005.

[4] J.I. Bolz, I. Farmer, E. Grinspun, and P. Schröder. Sparse Matrix Solvers on the GPU: Conjungate Gradiants and Multigrid. In proc. of ACM Siggraph'03, pp917-924, 2003.

[5] M. Brown, A. Majumder, and R. Yang. Camera-Based Calibration Techniques for Seamless Multi-Projector Displays, IEEE Transactions on Visualization and Computer Graphics, vol. 11, no. 2, pp. 193-206, 2005.

[6] N.A. Carr, J.D. Hall, and J.C. Hart. GPU Algorithms for Radiosity and Subsurface Scattering, In proc. of Siggraph/Eurographics Workshop on Graphics Hardware'03, pp. 51-59, 2003.

[7] M.F. Cohen and D.P. Greenberg. The hemi-cube: A radiosity solution for complex environments. In proc. of ACM Siggraph'85, pp. 31-41, 1985.

[8] G. Coombe, M.J. Harris, and A. Lastra. Radiosity on Graphics Hardware, In proc. of ACM Graphics Interface (GI'04), vol. 62, pp. 161-168, 2004.

[9] C. Cruz-Neira, D.J. Sandin, and T.A. DeFanti. Surround-screen projection-based virtual reality: The design and implementation of the cave. In proc. of ACM Siggraph'93, pp. 135–142, 1993.

[10] K. Dmitriev, T. Annen, G. Krawczyk, K. Myszkowski, and H.P. Seidel. A CAVE system for interactive modeling of global illumination in car interior. In proc. of ACM Symposium on Virtual Reality Software and Technology (VRST'04), pp. 137-145, 2004.

[11] C.M. Goral, K.E. Torrance, D.P. Greenberg, and B. Battaile. Modelling the interaction of light between diffuse surfaces. In proc. of ACM Siggraph'84, pp. 213-222, 1984.

[12] M. Gross, S. Würmlin, M. Naef, E. Lamboray, C. Spagno, A. Kunz, E. Koller-Meier, T. Svoboda, L. Van Gool, S. Lang, K. Strehlke, A. Vande Moere, and O. Staadt. blue-c: A Spatially Immersive Display and 3D Video Portal for Telepresence. In proc. of ACM Siggraph'03, pp. 819-827, 2003.

[13] M.D. Grossberg, H. Peri, S.K. Nayar, and P. Bulhumeur. Making One Object Look Like Another: Controlling Appearance Using a Projector-Camera System, In proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'04), vol. 1, pp. 452-459, 2004.

[14] A. Keller. Instance radiosity. In proc. of ACM Siggraph'97, pp. 49-56, 1997.

[15] R. Klein, J. Meseth, G. Müller, R. Sarlette, M. Guthe, and A. Balazs. RealReflect - Real-time Visualization of Complex Reflectance Behaviour in Virtual Protoyping, In proc. of Eurographics'03, Eurographics Industrial and Project Presentations, 2003.

[16] S. Klimenko, P. Frolov, L. Nikitina, and I. Nikitin. Crosstalk Reduction in Passive Stereo-Projection Systems. In proc. of Eurographics'03, pp.235-240, 2003.

[17] J., Konrad, B. Lacotte, and E. Dubois. Cancellation of image crosstalk in time-sequential displays of stereoscopic video, IEEE Transactions on Image Processing, vol. 9, no. 5, pp. 897-908, 2000.

[18] W. Kresse, D. Reiners, and C. Knöpfle. Color Consistency for Digital Multi-Projector Stereo Display Systems: The HEyeWall and The Digital CAVE, In proc. of 7th International Immersive Projection Technologies Workshop / 9th Eurographics Workshop on Virtual Environments (IPT/EGVE'03), pp. 271-335, 2003.

[19] J. Krüger and R. Westermann. Linear Algebra Operators for GPU Implementation of Numerical Algorithms. In proc. of ACM Siggraph'03, pp. 908-916, 2003.

[20] J.S. Lipscomp and W.L. Wooten, Reducing crosstalk between stereoscopic views. In proc. of SPIE'95, Stereoscopic Displays and Virtual Reality Systems II, vol. 2409, pp. 31-40, 1995.

[21] A. Majumder and R. Stevens. Perceptual Photometric Seamlessness in Tiled Projection-Based Displays, ACM Transactions on Graphics, vol. 24, no. 1, pp. 111-134, 2005.

[22] S.K. Nayar, H. Peri, M.D. Grossberg, and P.N. Belhumeur. A Projection System with Radiometric Compensation for Screen Imperfections, Proc. of International Workshop on Projector-Camera Systems, 2003.

[23] R. Raskar, G. Welch, M. Cutts, A. Lake, L. Stesin, and H. Fuchs. The office of the future: A unified approach to image-based modelling and spatially immersive displays. In proc of. ACM Siggraph'98, pp. 179–188, 1998.

[24] A. Scheel, M. Stamminger, and H.P. Seidel. Tone reproduction for interactive walkthroughs. In proc. of Eurographics'00, pp. 301-312, 2000.

[25] P. Sen, B. Chen, G. Garg, S. Marschner, M. Horowitz, M. Levoy, and H.P.A. Lensch. Dual Photography, In proc. of ACM Siggraph'05, pp. 745-755, 2005.

[26] A. Simon and m. Göbel. The i-Cone - A Panoramic Display System for Virtual Environments, In proc. of the 10th Pacific Conference on Computer Graphics and Applications (PG'02), p.3, 2002.

[27] D. Wang, I. Sato, T. Okabe, and Y. Sato. Radiometric Compensation in a Projector-Camera System Based on the Properties of Human Vision System, In proc. of IEEE International Workshop on Projector-Camera Systems (ProCams'05), 2005.

[28] R. Yang, D. Gotz, J. Hensley, and H. Towles. PixelFlex: A Reconfigurable Multi-Projector Display System, In proc. of IEEE Visualization (IEEE Vis'01), pp. 68-75, 2001.

[29] Y. Yu, P. Debevec, J. Maik, and T. Hawkins. Inverse global illumination: recovering reflectance models of real scenes from photographs. In proc. of ACM Siggraph'99, pp. 215-224, 1999.