Three-Dimensional Display Rendering Acceleration Using Occlusion Camera Reference Images

Voicu Popescu, Paul Rosen, and Dan Aliaga

Abstract—**volumetric 3D displays allow the user to explore a 3D scene free of joysticks, keyboards, goggles, or trackers. For non-trivial scenes, computing and transferring a 3D image to the display takes hundreds of seconds, which is a serious bottleneck for many applications. We propose to represent the 3D scene with an occlusion camera reference image (OCRI). The OCRI is a compact scene representation that stores only and all scene samples that are visible from a viewing volume centered at a reference viewpoint. The OCRI enables computing and transferring the 3D image an order of magnitude faster than when the entire scene is processed. The OCRI approach can be readily applied to several volumetric display technologies; we have tested the OCRI approach with good results on a volumetric display that creates a 3D image by projecting 2D scene slices onto a rotating screen.**

***Index Terms*—Three-Dimensional Displays, computer graphics, image-based rendering, rendering acceleration.**

I. Introduction

Conventional 3D computer graphics applications present the scene to the user on a 2D display. The approach has at least two fundamental disadvantages. First, the system needs to know the view desired by the user. Interfaces that rely on trackers or on input devices (e.g. joysticks and keyboards) provide only a crude and non-intuitive way for the user to select the desired view. Second, the output image is flat, which deprives the user from the important depth cues of binocular stereo vision. Special goggles or displays can be used to present each eye with a different image, but stereo display technologies suffer from disadvantages such as limited range of motion, need for strenuous image fusing, and uncomfortable eyewear.

Volumetric 3D displays hold the promise to overcome these disadvantages. A sculpture of light provides a truly three dimensional replica of the scene of interest. The user naturally selects the desired view by gaze, by head motion, and by walking around the 3D image. There is no need for encumbering eyewear, and the processes of accommodation and vergence occur naturally. Although the advantages of volumetric 3D displays have been known for a long time, 3D display technology continues to suffer from fundamental challenges. One challenge is creating an adequate 3D array of pixels. The requirements are small pixel volume for good spatial resolution, and wide range of intensities, colors, and opacities. A second challenge is achieving satisfactory performance. Computing and transferring the 3D image to the display presently takes hundreds of seconds, which is unacceptable for many applications.

This paper describes a method to accelerate rendering on volumetric 3D displays, based on adapting the scene level of detail before the 3D image is computed, and on reducing the number of 3D image samples that are computed and transferred. For example, if the 3D scene represents Manhattan, a view that maps the entire island to the volume of the 3D display can be safely computed from a coarser representation than a view that only shows Times Square. Moreover, for a single user that is seated or stands in one place, many of the background buildings are completely occluded and do not become visible for normal gaze changes and head motions. The hidden buildings can be ignored when computing the 3D image.

In the case of complex scenes with numerous occlusions, the number of samples that remain hidden despite the interpupilary distance and despite the translational component of head motions is particularly large. These scenes are also the ones that presently require the largest rendering times, so the gain obtained by not processing hidden samples is substantial. Level of detail adaptation and occlusion culling are classic problems in 3D computer graphics. Many algorithms have been developed to simplify geometry and to eliminate primitives that lie in the shadow of occluders. However, quickly establishing a small set of primitives that is sufficient for a given view remains an open problem.

A relatively recent research path in computer graphics is image-based rendering (IBR), where the scene is rendered from pre-computed or pre-acquired reference images. In one variant, the scene is modeled with *depth images* (DIs), which are images enhanced with per-pixel depth [24]. The depth information allows reprojecting (3D-warping) the reference samples to any novel desired view. A DI provides a good level-of-detail solution, which holds for nearby views. Unfortunately, the occlusion culling solution of the reference image cannot be applied to nearby views. Even small translations of the viewpoint produce disocclusion errors, which are artifacts due to lack of samples for surfaces that become visible but were not sampled by the reference DI. In our context, representing the scene with a DI computed from the left eye's viewpoint produces disocclusion errors in the image seen by the right eye.

We have recently introduced *occlusion cameras* [25, 32], a class of non-pinhole cameras which sample not only surfaces visible in the reference view, but also surfaces that are likely to become visible in nearby views. The resulting occlusion camera reference image (OCRI) stores samples that are hidden in the reference view but are needed to alleviate disocclusion errors when the view translates. We represent the scene with an OCRI computed for the user's reference view, which is the average of the left and right eye views in the normal head

SIMULATOR IMAGES

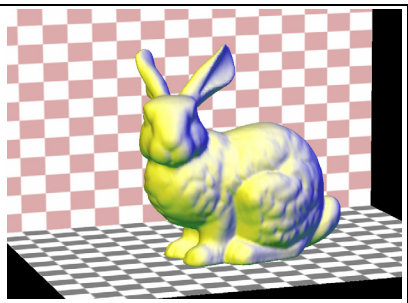
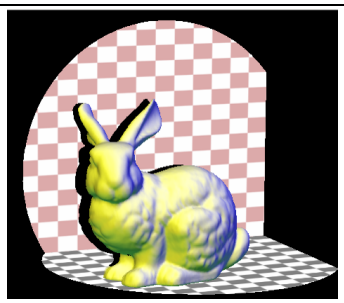


Figure 1 Depth image (DI).



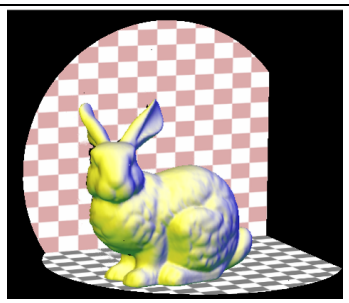


Figure 2 Images rendered from DI and OCRI, viewpoint 4" left of reference viewpoint.

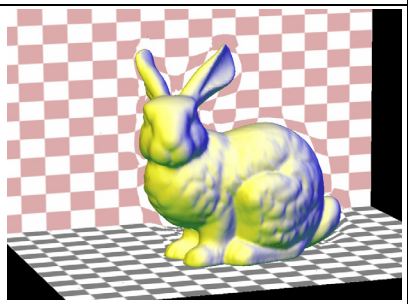
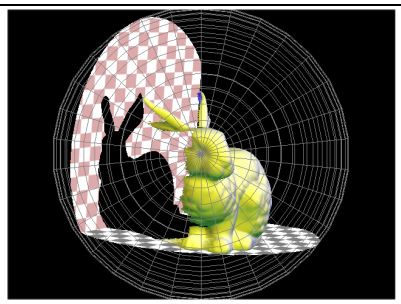


Figure 3 OCRI.



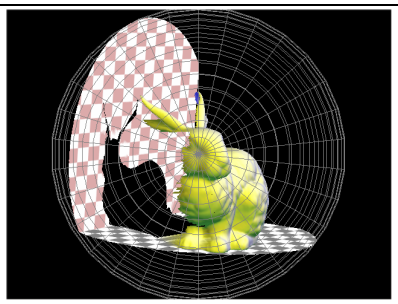
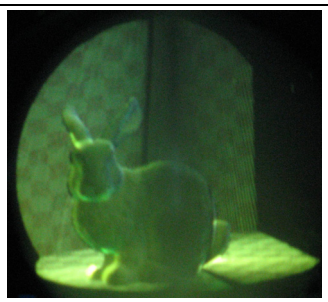
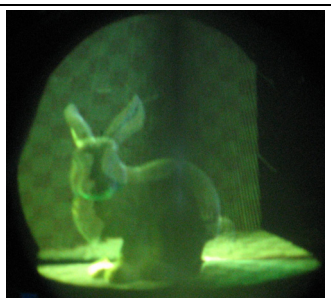


Figure 4 Images rendered from DI and OCRI. Wireframe shows spherical display volume.

PHOTOGRAPHS OF 3D DISPLAY





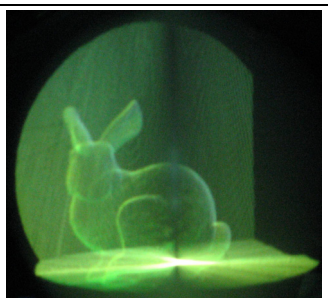
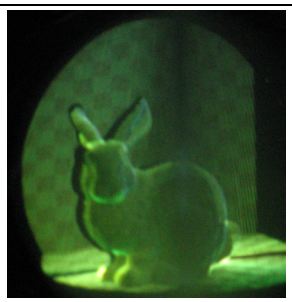


Figure 5 3D images rendered from DI (*left*), OCRI (*middle*), and original geometric model (*right*), all photographed from reference view.



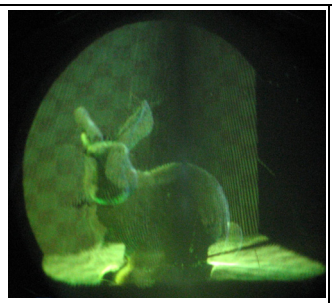
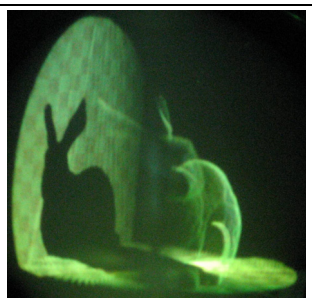


Figure 6 DI and OCRI 3D images from viewpoint translated 4" left.



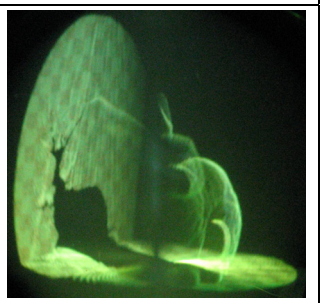


Figure 7 DI and OCRI 3D images from side view.

position. Like a regular DI, the OCRI is a single layer representation with the advantages of bounded number of samples, implicit connectivity, and efficient incremental processing. Another advantage shared with regular DIs is that OCRIs adapt the scene's level of detail to the reference view. Unlike a regular DI however, the OCRI has all samples needed for a continuum of views centered at the reference view. Interpupilary distance and normal head motion do not produce disocclusion errors.

Figures 1-7 illustrate our approach. Figures 1-4 show images computed with our volumetric 3D display simulator, and Figures 5-7 show actual photographs of our volumetric 3D display. Both simulated and real 3D displays produce spherical images with a diameter of 10". Figures 1 and 3 show a depth image (DI) and an OCRI constructed from the same viewpoint. Figure 2 shows the DI and OCRI from a viewpoint 4" left of the reference viewpoint. The severe disocclusion errors that occur for the DI are alleviated by the OCRI. Figure 4 shows the DI and OCRI from a side view. The OCRI does not sample all surfaces in the scene, nor should it. The OCRI provides occlusion culling by safely discarding the samples that are not needed in nearby views. The OCRI shrinks the "shadow" of the bunny. Figure 5 shows reference view photographs of the 3D images rendered from the DI, OCRI, and geometric model. Figures 6 and 7 correspond to Figures 2 and 4.

II. Prior Work

We describe a method to accelerate rendering on 3D displays based on a novel non-pinhole camera model that produces reference images less prone to disocclusion errors. We limit the discussion of previous work to a brief review of 3D display technologies, to prior methods for alleviating disocclusion errors, and to previous non-pinhole camera models.

A. Three-Dimensional Displays

Several technologies attempt to go beyond a flat 2D image. One approach is to use special eyewear to present each eye with a different image. Polarizing glasses, dynamic shutter glasses, or head mounted displays make the image appear 3D by providing the required parallax between the left and right eye images. These technologies are popular with virtual reality applications since the synthetic image covers the entire field of view of the user, which conveys a sense of immersion. The important limitation is the need of special eyewear.

Autostereoscopic displays [15] produce a 3D image without the need of special eyewear. Parallax autostereoscopic displays provide different images for the left and right eyes using slits [14, 30] or lenslets [5, 13, 21]. The disadvantages are reduced resolution and reduced range of supported viewpoints.

Volumetric displays produce a truly three dimensional image. One approach is to fill space, for example with a stack of transparent LCDs [17]. The approach has the disadvantage of limited z resolution. Another approach is to use a varifocal mirror whose oscillations are synchronized with a 2D display it reflects [41]; the difficulty with such a display is building the varifocal mirror.

Another type of volumetric display technology is based on sweeping the display volume. 2D slices of the scene are displayed in rapid succession and the eye integrates them into a 3D image [1, 7]. The greatest challenge is the mechanical scanning, which is noisy, imprecise, and fragile.

Several emerging technologies show potential for producing 3D images. Electroholography [18] produces an interference pattern (holographic fringe) which is then illuminated to produce a 3D image by diffraction (modulation of holographic fringe). The approach is hampered by the enormous amount of data resulting from the requirement of sampling the fringe with very high spatial frequency. A different technology uses a pair of laser beams that excite voxels inside a transparent cube of heavy metal fluoride glass [6]. Attempts to replace the heavy and expensive medium have not been successful so far. Another experimental volumetric display [23] has 76,000 voxels that are lit using optical fibers as waveguide.

To the best of our knowledge, the only volumetric displays available commercially are those produced by Actuality Systems [1] and LightSpace Technologies [17]. All volumetric displays convert a 3D scene description into a 3D image. Our method produces a simplified description of the scene which is then used to compute the 3D image. Therefore, in principle, the method can be applied to other volumetric display technologies. We demonstrate the effectiveness of our method on the Perspecta volumetric display [31], which we characterize in detail in Section VII.

B. Disocclusion errors

A brute force solution to the problem of disocclusion errors is to reconstruct the desired image by warping several depth images. The approach has the obvious disadvantage of high cost. Disocclusion errors are small groups of missing samples, scattered throughout the scene. No single additional depth image captures them all. The additional depth images contribute only a few new samples. Another important disadvantage is that the cost of rendering the desired image varies with the number of depth images that have to be considered to avoid all disocclusion errors. Such an unpredictable cost is a severe limitation for applications that rely on a guaranteed minimum frame rate. In a technique called post-rendering warping [20], conventional rendering is accelerated by warping *two* reference images. Even when the viewpoints of the two reference images are very close to the desired viewpoint, disocclusion errors persist.

Several techniques for alleviating disocclusion errors have been developed based on the idea of pre-combining several depth images into a layered representation that accommodates more than one sample along a ray. Redundant samples are detected and discarded. One example is the multi-layered z-buffer (MLZB) [22]. The approach traces the ray beyond the first surface and collects up to a maximum number of k samples for each ray. MLZBs can be inefficient since the depth complexity can be unnecessarily large at some pixels. In other words, some of the samples in the MLZB never become visible in any nearby view.

Layered depth images (LDIs) [37] address this issue: the layered representation is built from depth images constructed from nearby views. This way each sample in the resulting LDI is known to be visible in at least one nearby view. LDIs have been used to accelerate architectural walkthroughs [35], and as building blocks for hierarchical sample-based scene representations [4]. One disadvantage of LDIs is the lengthy construction time which limits their applicability to dynamic scenes, where the reference images have to be updated frequently. Another shortcoming is their hardware-unfriendly irregular structure, with an unbounded number of samples. Lastly, LDIs do not have sample connectivity, and the desired image is typically rendered by splatting, a low-quality reconstruction technique borrowed from volume rendering [42].

None of the methods discussed so far for addressing the problem of disocclusion errors is conservative. In the case of LDIs for example, it can happen that the desired image sees a surface sample that is not visible in any of the construction depth images and is therefore not present in the LDI. The vacuum buffer [33] is a conservative method for deciding whether a set of depth images is sufficient to avoid disocclusion errors in a desired image. The method keeps track of the sub-volumes of the desired view frustum which

are yet to be covered by any depth image. The disadvantages of the approach are high per-frame cost—since the algorithm needs the desired view it needs to run in real time, for every frame, and unbounded number of samples—additional depth images are needed to eliminate all disocclusion errors.

The advantages of representing and rendering a 3D scene with depth images rather than with a traditional polygonal model have been recognized by researchers and developers of interactive 3D video technologies (see [38] for a comprehensive overview of the state of the art). The depth image and the layered depth image have been adopted by the MPEG-4 standard via its Animation Framework eXtension (AFX), part of the Depth Image-Based Representation (DIBR) family [19]. Of course, the DIBR representations inherit the disocclusion errors of depth images.

All previous solutions to the problem of disocclusion errors attempt to fill in disocclusion errors once they occur. Instead, we take the approach of *preventing* disocclusion errors. A reference image is asked to provide the necessary samples for rendering the scene from a continuous range of viewpoints, centered at the reference viewpoint. Therefore a reference image also needs to store samples that are not visible in the reference view. The challenge is to find an efficient method for including in the reference image samples that are "about to become visible". Our method is based on a non-pinhole camera whose rays go around occluders to gather samples which cannot be reached by the rays of a pinhole camera. Several non-pinhole cameras have been developed by computer vision and computer graphics researchers.

C. Non-pinhole cameras

Much of the computer vision arsenal for extracting information from images is based on the single viewpoint constraint. The main reason for this is that such single viewpoint images can be trivially re-sampled to a familiar, human-vision-like planar pinhole camera image. Recently, researchers began considering camera models whose rays do not pass through a common point. The general linear camera (GLC) [45] captures all rays that are a linear combination of three given construction rays, which are not necessarily concurrent. The GLC generalizes two previously studied cameras: the pushbroom camera [12], and the two-slit camera [29]. The GLC is not sufficiently powerful to address disocclusion errors in complex scenes.

Computer graphics researchers have also studied non-pinhole cameras. In computer graphics the cameras are virtual, so camera design is free of the constraint that the novel camera be physically realizable using actual refractive, reflective, and sensing elements. The light field [10, 16] is an important non-pinhole camera which shows that a 3D scene can be rendered without knowledge of its geometry. A light field is a 4D database of rays, parameterized using two parallel planes. The rays of the desired view are looked up in the database. Light fields do not suffer from disocclusion errors, however, they are expensive to construct and scale poorly with the scene size.

The multiple-center-of-projection camera [36] samples the scene along a user chosen path. For every viewpoint a single column of rays (pixels) is collected. The disadvantage is the need for user interaction, and the high construction cost: the scene needs to be rendered for every viewpoint along the path. We have developed a class of non-pinhole cameras specifically for addressing the problem of disocclusion errors.

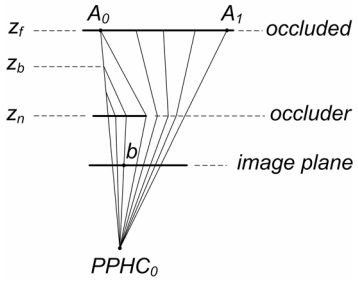


Figure 8 Illustration of the effect of the distortion on the rays of $PPHC_0$.

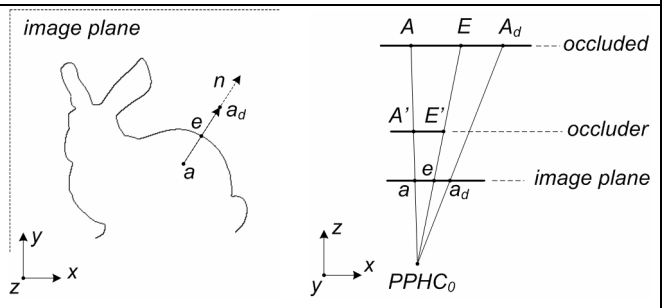


Figure 9 Illustration of distortion at depth discontinuities. Image plane view (*left*), and view in plane defined by $PPHC_0$, *a*, and *e* (*right*).

III. Algorithm Overview

Given a 3D scene *S* and a reference view expressed as a planar pinhole camera $PPHC_0$, our algorithm proceeds in the following main steps:

1. Construct an occlusion camera OC_0 from $PPHC_0$ and *S*.
2. Build a reference image $OCRI_0$ from OC_0 and *S*.
3. Produce 3D image $I3D_0$ from $OCRI_0$.

The occlusion camera depends on the reference view *and* the scene geometry it encompasses. Once OC_0 is known, *S* is replaced with $OCRI_0$, which provides a view-optimized, bounded-cost approximation of the scene. The next three sections describe each of the three main steps of the algorithm.

IV. Occlusion Camera

A. Occlusion camera class

An occlusion camera is constructed for a given scene and a given reference view, and has the following properties:

a. *Disocclusion.* Some rays of the camera sample surfaces that are not visible in the reference view, but are likely to become visible in nearby views.
b. *Single layer.* The camera acquires a 2D image; at each pixel, the image stores the depth and color of the closest surface sample along the ray at that pixel.

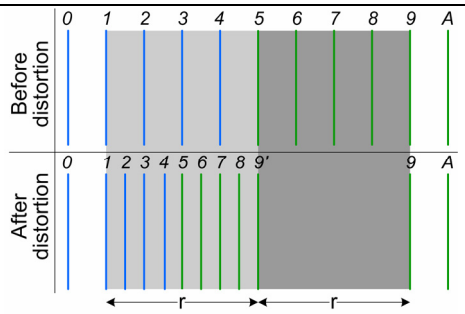


Figure 10 Visualization of distortion magnitude variation.

c. *Unambiguous projection*. A 3D point projects to at most a single image location (no two rays intersect).
d. *Efficient projection*. The projection of a 3D point is computed in a constant number of steps.

The first property ensures that the OCRI is less prone to disocclusion errors than a regular depth image. Because of the second property, the OCRI has a bounded number of samples. The depth and color samples can be trivially connected in a regular mesh by connecting each sample to its neighbors.

The last two properties ensure that the OCRI can be constructed efficiently with the feed-forward graphics pipeline (FFGP). The FFGP has two main stages: projection, when the geometric primitive is projected onto the image plane, and

rasterization, when pixels covered by the primitive are identified and set to appropriate values. The FFGP is efficient because, unlike the ray tracing pipeline [43, 9], it only considers pixel/primitive pairs that are likely to yield an intersection (a color sample). The FFGP is the approach of choice in interactive computer graphics and it is supported in hardware [28, 26, 27, 2].

If the occlusion camera provides fast, unambiguous projection, the OCRI can be constructed efficiently with the FFGP. Assuming that the scene is modeled with triangles, each triangle is projected by projecting its vertices, and then the projected triangle is rasterized to produce the reference image samples.

We demonstrated the occlusion camera concept with the single-pole occlusion camera (SPOC) [25], which is limited to a single, relatively simple occluder. To overcome this limitation we introduce a second member of the occlusion camera class.

B. *Depth discontinuity occlusion camera*

1) *Overview*

Given a 3D scene S and a reference view $PPHC_0$, the goal is to devise a camera that sees slightly more than what $PPHC_0$ sees; in other words, hidden samples that are close to the boundary of their occluder should be part of the image gathered by the camera. We achieve this by redirecting (distorting) the rays of the $PPHC_0$ that pass close to a depth discontinuity. The problems of the SPOC are avoided by defining the distortion at a fine level, using a *distortion map*. A distortion map pixel (location) stores distortion information for the $PPHC_0$ ray defined by that pixel.

Let A be a hidden point of the background that is close to the silhouette of the bunny as seen in the depth image in Figure 1. The distortion changes the projection of A from the undistorted location a given by $PPHC_0$ to a_d (Figure 9). The distortion moves the sample perpendicularly to the depth discontinuity, and away from the occluder. In Figure 9—*left*, the depth discontinuity has normal n at pixel e. The distortion does not change the projection of the bunny sample A' that is seen along the same $PPHC_0$ ray as A. This way the sample A clears the occluder and remains visible in the final OCRI.

Figure 8 illustrates the distortion by visualizing the rays of the resulting occlusion camera. The original rays of $PPHC_0$ are unaffected until the depth of the occluder, z_n. The rays close to the depth discontinuity are moved in a direction normal to the depth discontinuity, *towards* the occluder (which causes the samples to move away from the occluder). The distortion increases linearly in $1/z$ from z_n to the depth z_f of the occluded object, which makes that the rays of the occlusion camera are line segments between z_n and z_f. Rays to the left of A_0 and to the right of A_1 are not affected by the distortion. Some distorted rays are implicitly clipped by the ray of A_0—this simply means that a sample at OCRI location b cannot be farther than z_b. The entire view frustum of $PPHC_0$ is sampled by the rays of the occlusion camera.

2) *Distortion map construction*

The occlusion camera is defined by the reference view $PPHC_0$ and a distortion map $DMAP_0$ that distorts its rays. Each distortion map locations stores a distortion sample specified with a five-tuple $(d_u, d_v, z_n, z_f, d_f)$. The 2D unit vector (d_u, d_v) gives the direction of the distortion, and the distortion magnitude increases from 0 at depth z_n to d_f at z_f. The distortion map $DMAP_0$ is constructed as follows.

1. Render S with $PPHC_0$, producing z-buffer ZB.
2. Detect depth discontinuities in ZB.
3. For each depth discontinuity pixel e, splat e in $DMAP_0$.
4. For each depth discontinuity pixel e, adjust splat size.
5. For each $DMAP_0$ location, set distortion five-tuple.

At step one, the scene is rendered in hardware and the z-buffer is read back. At step two, depth discontinuity pixels are detected as pixels where the second order depth variation exceeds a threshold [34].

At step three, a first pass over the depth discontinuity pixels is taken to set the neighborhood of the depth discontinuities

Figure 11 Asymmetrical splats. The scene consists of two rectangular occluding floating in front of a background. Symmetrical splats (*left*) cannot entirely disocclude the background since the space between the two occluders is too small. Asymmetrical splats (*right*) need less image area for the same disocclusion effect and completely disocclude the background.

where the distortion acts. For each depth discontinuity pixel e, a circular splat of radius D is written into $DMAP_0$. D is a user chosen parameter that specifies how far behind the occluder the occlusion camera should reach. This value might be later decreased for some depth discontinuities to accommodate conflicting distortion requirements, as described later.

When a splat sample lands at a $DMAP_0$ location p which is already occupied, the splat whose center is closest to p wins. During the construction phase, the distortion map stores at every location 3 more scalars, in addition to the 5 needed to specify the distortion sample. Two of these additional values specify the coordinates c_u and c_v of the splat that owns the location, and are used in the splat arbitration described above. The third additional value specifies the current radius of the splat, which starts out as D.

During step four, a second and last pass over the depth discontinuity pixels reduces the radii of the splats to avoid overlap with conflicting splats. Two splats conflict if they affect the same $DMAP_0$ location and if they have distortion directions that form an angle larger than a user chosen threshold. We use in practice threshold of 90^0.

Reducing the splat size is necessary in order to avoid losing visible samples. Consider the case of a thin gap. The left edge of the gap moves samples towards the right, and the right edge towards the left. The distortion directions form an angle of 180^o. The gap is smaller than D and not adjusting the size of the splat causes the samples to compete for the same OCRI location and to lose some of them to z-buffering. Once the radius r has been determined, all distortion samples owned by the splat and located farther than r are deleted (reset).

In the last step five, a pass over $DMAP_0$ sets the distortion samples for each location that is under the influence of a depth discontinuity pixel, as indicated by valid c_u and c_v values. The direction (d_u, d_v) of the distortion at $DMAP_0$ location p is given by the depth discontinuity normal at (c_u, c_v). The depth discontinuity direction is approximated by least squares fitting a line to a neighborhood of depth discontinuity pixels, centered at (c_u, c_v). The normal points away from the occluder, towards the samples with larger z's. The near and far depths z_n and z_f between which the distortion acts are given by the depths of the two samples creating the depth discontinuity.

The distortion magnitude depends on the distance from p to the depth discontinuity pixel (c_u, c_v). If the radius of the splat at (c_u, c_v) is r, and the signed distance from p to (c_u, c_v) is x, d_f is set as $(r\text{-}x)/2$. The distortion magnitude starts out as r for $x=\text{-}r/2$, and tapers off linearly to 0 at $x=+r/2$. Figure 10 shows the effect of the distortion in the image plane of $PPHC_0$ in the neighborhood of a vertical depth discontinuity. The $+\text{-}r$ neighborhood is shown shaded in grey. The depth discontinuity separates the neighborhood in two equal parts, shaded in light and dark grey. The occluder covers the darker right half. Before the distortion, vertical bars 5-9 are hidden. The distortion compresses and shifts them to the right half of the light grey region. In order to make room, the originally visible samples between bars 1-5 are compressed and shifted to the left half of the light grey region.

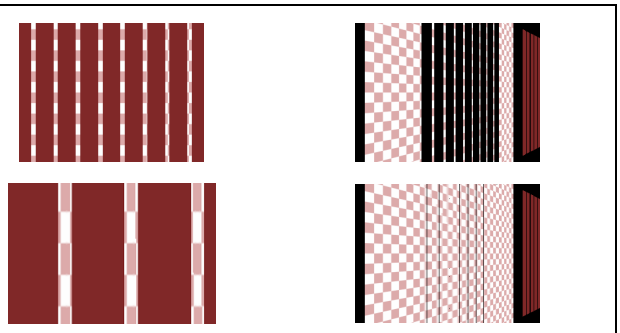


Figure 12 (*Top*) Depth image (*left*) used to render the scene from a side view (*right*). (*Bottom*) OCRI constructed with asymmetrical splats (magnified fragment, *left*) and same side view (*right*).

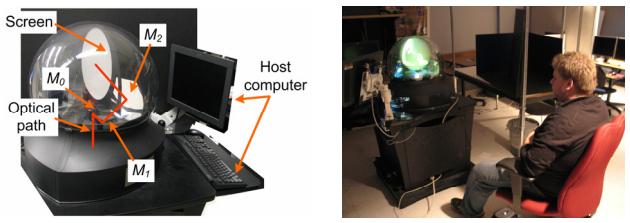


Figure 13 Volumetric 3D display used to validate the OCRI approach (*left*) and typical viewing distance of 50" (*right*).

The resulting occlusion camera trades (u, v) resolution for resolution along the same reference view ray. The hidden samples are accommodated in the single layer OCRI by compressing the image close to the depth discontinuities. In Figure 10 the sampling rate is half that in the original image.

3) Asymmetrical splats

For complex scenes, numerous conflicting splats have centers closely located from one another, which reduces the effective splat radius r, and with it, the disocclusion capability of the resulting occlusion camera. There just isn't enough room in the image to accommodate the hidden samples (Figure 11). In such cases, we increase the disocclusion efficiency of the occlusion camera by reducing the image area required to disocclude a given number of hidden samples.

We achieve this with asymmetrical splats. If the asymmetry factor is α, the distortion magnitude d_f varies linearly from r to 0, as the signed distance x to the edge increases from $-r$ to r/α. The expression for d_f is given by

$$d_f(x) = r - \frac{x+r}{1+\frac{1}{\alpha}}$$

Equation 1 Distortion magnitude variation for asymmetrical splats.

When the splats are symmetrical, α equals 1 and the expression for d_f becomes $(r\text{-}x)/2$, as derived earlier. The splat asymmetry is a powerful tool for increasing the disocclusion capability of the occlusion camera. Figure 12 shows a scene consisting of several rectangular occluders that float in front of a checkered background. The background is heavily occluded. When the side view is rendered from a regular

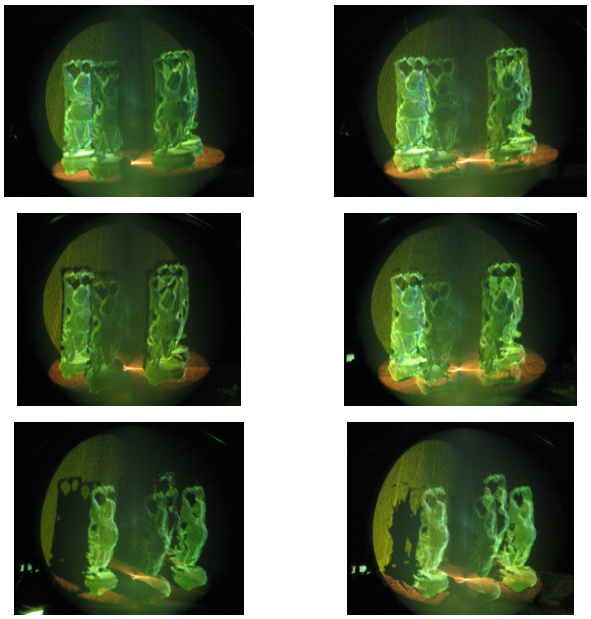


Figure 14 Photographs of the 3D display showing the Happy Buddha statues scene. The 3D image was rendered from a DI (*left column*) and from an OCRI (*right column*). The photographs were taken from the reference viewpoint (*row 1*), and from 4" above the reference viewpoint (*row 2*). Side view shows the "shadows" shrunk by the OCRI (*row 3*).

depth image, severe disocclusion errors occur. When using asymmetrical splats ($\alpha = 2$), virtually the entire background is captured. Asymmetrical splats decrease the sampling rate near depth discontinuities ($\alpha + 1$) times, since an $r + r/\alpha$ region is compressed into an r/α region. The decrease in resolution can be alleviated an increase the reference image resolution.

V. Occlusion Camera Reference Image construction

The scene is rendered with the occlusion camera $OC_0 = (PPHC_0, DMAP_0)$ to create the reference image $OCRI_0$. Each triangle mesh of the scene S is projected with OC_0 and then the projected mesh is rasterized in hardware. A triangle mesh is projected by projecting each of its vertices. A vertex V is projected with the following equation.

$$(u_u, v_u, z) = PPHC_0(V)$$
$$(d_u, d_v, z_n, z_f, d_f) = DMAP_0(\lfloor u_u \rfloor, \lfloor v_u \rfloor)$$
$$d(z) = \begin{cases} 0, z < z_n \\ \dfrac{1/z_n - 1/z}{1/z_n - 1/z_f} d_f, z_n \le z \le z_f \\ d_f, z > z_f \end{cases}$$
$$(u_d, v_d) = (u_u, v_u) + (d_u, d_v) d(z)$$

Equation 2 Projection with occlusion camera.

The occlusion camera is a non-pinhole camera which does not preserve lines and planes. To control the approximation error introduced by conventional rasterization, we subdivide each triangle until the screen space edge lengths of the resulting triangles are smaller than a user chosen threshold. In practice, we use a threshold of 1 pixel.

The subdivision stopping criterion directly impacts the OCRI construction time. For many scenes coarser subdivisions are acceptable. Consider a scene like the one in Figure 12, except that it has a single rectangular occluder, of width 10 pixels. If the maximum tolerable edge length is 20 pixels, it can happen that no background triangle vertex is distorted, and the OCRI is equivalent to a regular depth image. However, a threshold of 5 pixels will produce the same (good) results as a threshold of 1 pixel.

VI. Rendering using the OCRI

The OCRI provides a good approximation of the scene, tailored to the reference view. The OCRI is converted to a 3D triangle mesh, which is then used by the volumetric display driver to render the 3D image, in lieu of the original scene model. Each sample in the OCRI corresponds to a 3D point with color. To recover the 3D point from the OCRI sample, one needs to be able to unproject the sample back in 3D. The distorted coordinates (u_d, v_d, z) and the distortion five-tuple are not sufficient to recover the undistorted coordinates of the sample, since the distortion is not invertible.

For this we augment the OCRI with an additional two channels per pixel that store the distortion vector used to create the sample. The values of these channels are computed during OCRI construction by rendering the scene meshes a second time with the distortion vector components stored in the red and green channels of vertex color. The hardware interpolates these values during rasterization and stores the distortion vector for every pixel in the frame buffer.

Given (u_d, v_d, z) and the distortion vector (δ_u, δ_v), the undistorted coordinates (u_u, v_u) are computed as (u_d - δ_u, v_d – δ_v). The model space 3D point is obtained by unprojecting the pixel (u_u, v_u) to depth z with $PPHC_0$.

VII. Rotating screen volumetric 3D display

As stated earlier, all 3D displays transform the geometry and color scene description into a 3D image. Our method reduces the complexity of the scene by adapting the level of detail and by safely discarding surfaces that are not visible in any view of interest to the user. Therefore, our method is applicable to a variety of 3D displays.

Available to us is a volumetric display (Figure 13) that builds a 3D image one slice at the time, with a rotating screen [31]. The screen has a radius of 5", it is diffuse and semitransparent, and it rotates with an angular velocity of 720rpm. Since both faces of the screen carry an image, the refresh rate is 24Hz, which corresponds to 180° rotation. The display projects onto the screen the intersection between the scene and the plane of the screen 198 times for every complete rotation. The optical path is folded using 3 mirrors M_0-M_2. The mirrors and screen are enclosed in an inner glass sphere that rotates with the screen; the glass sphere is enclosed in a stationary outer glass sphere. The display is not perfectly balanced which causes it to wobble. We estimate the amplitude of the wobbling to be 0.5cm. Each slice has a

Scenes	DI		OCRI			Geometry	
	Tris ($x10^3$)	**Time** (s)	**Tris** ($x10^3$)	C_{time} (s)	***Time*** (s)	**Tris** ($x10^3$)	**Time** (s)
Bunny	612	12.0	612	2.73	11.8	321	8.02
Bunny QR	37.8	.766	37.8	.875	.75	321	7.81
Buddha statues	612	11.4	612	12.1	11.5	4,603	131
Thai statue	612	12.5	612	20.3	13.9	10,252	292

Table 1 Rendering performance measures for various scenes.

resolution of 768x768. The color resolution is 32bit RGBA but it is compressed to 3bit RGB. The reduced image brightness requires dimming the ambient lights when the display is in use (Figure 13).

The application runs on a host computer (IBM, Intel chipset, Windows XP operating system) connected to the display with an SCSI interface. The display manufacturer has provided a driver that supports OpenGL. The timing information reported in this paper was obtained with a display driver v1.5. The 3D image maps the model space unit sphere to the volume of the display.

The photographs shown throughout this paper were taken with a digital camera with the following settings: no ambient lights, aperture F2.8, exposure time 1/25s, and simulated film sensitivity ISO400. Our camera does not offer 1/24s as one of the possible exposure times, which would have allowed acquiring a complete 3D image. We used the slightly shorter exposure time since the wobbling produces excessive blurriness if the shutter remains open more than 180^o and the screen revisits a part of the 3D image. The slightly shorter exposure time misses (1/24-1/25)*(12*360^o) = 7.2^o of the 3D image. We took several snapshots for every position to place the missing 3D image sector in a convenient location (see black stripe that splits the vertical plane in Figure 5—*left* or the horizontal plane in Figure 6).

VIII. Results and Discussion

We have tested our approach on several 3D scenes, both with our volumetric display simulator and the actual volumetric display: the bunny (Figures 1-7), the vertical bars (Figure 12), the four Happy Buddha statues (Figure 14), the Unity, the auditorium, and the Thai statue (Figure 15) scenes.

OCRIs prove to be a robust solution to the problem of disocclusion errors, and can handle complex scenes. We measure the disocclusion errors present in a frame by rendering a ground truth image from geometry and counting how many ground truth image samples are not present in the frame. We rendered sequences of frames by moving the viewpoint on the edges of an 8" cube centered at the reference viewpoint. The disocclusion errors measured when using the OCRI were, on average, 4.5% of those measured when using a depth image as reference.

	DI	LDI	LF	ULF	RPS	OCRI
Construction time (s)	0.12	3.84	30.72	3.84	6.84	11.5
Memory size (MB)	2.6	3	332.8	41.6	76	5.2

Table 2 Construction performance comparison.

The OCRI provides efficient projection and is constructed with the help of graphics hardware. Table 1 reports the 3D image rendering times and the number of triangles for each of three scenes (bunny, Happy Buddha statues, and Thai statue), and for each of three scene representations (depth image, OCRI, and geometry). The OCRI approach has three main steps: the occlusion camera model is computed first, then the OCRI is constructed by rendering the scene with the occlusion camera, and then finally the 3D image is produced from the triangle mesh defined by the OCRI. The table reports the aggregate time for steps 1 and 2 as C_{time}, and the time for step 3 as *Time*. The resolution of the desired image and that of the reference image is 720x480. The depth image and the OCRI always generate the same number of triangles since the OCRI has a single layer where it stores the hidden samples at the cost of reducing the sampling rate for the visible surfaces.

In the case of the bunny scene, the depth image and the OCRI generate more triangles than present in the original model, with the consequence of a larger 3D image rendering time. For the bunny, creating a depth image or an OCRI at this resolution is wasteful—the new vertices do not bring any new information since they are computed by interpolation. Once a more suitable resolution is selected (180x120, see row *Bunny QR* in the table), the speedup is considerable. For the DI representation, we define the speedup as the ratio between the time needed to render the 3D image from the original geometric model and from the depth image. For the OCRI representation we compute the speedup by dividing by the sum of C_{time} and *Time*. Therefore the speedup is 7.81/0.766 = 10.2 for the DI and 7.81/(0.875+0.75) = 4.8 for the OCRI.

For the Happy Buddha statues scene, the speedup is 11.5 for the DI and 5.5 for the OCRI. For the 10 million triangles Thai statue, rendering the 3D image from the DI or the OCRI brings a speedup of 23 and 8.5, respectively. The advantage of the DI and of the OCRI increases with the complexity of the scene, since the DI and the OCRI generate the same number of triangles (e.g. 612,000) regardless of the complexity of the original scene model.

The DI approach is more efficient since it does not incur the cost of OCRI construction, but it suffers from disocclusion errors. We will work on reducing the OCRI construction time. Step one has a cost proportional to ED^2+WH, where E is the number of depth discontinuity pixels, D is the user chosen maximum distortion region radius (D = 30 in our experiments), and W and H give the width and height of the

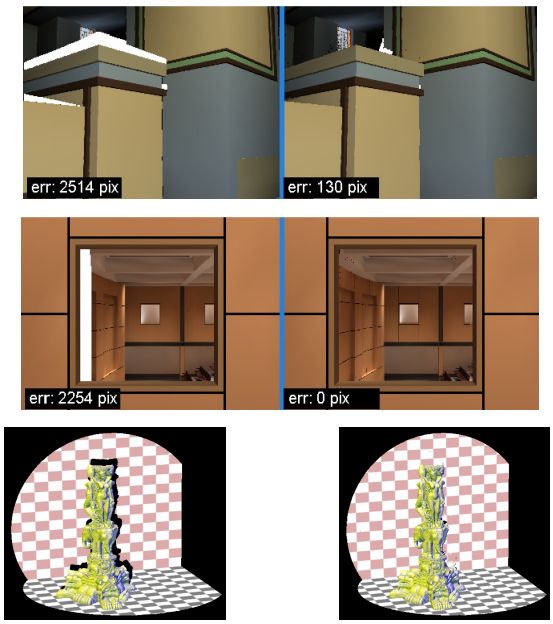


Figure 15 Frames rendered from DI (*left*) and OCRI (*right*) for the unity church (*top*), auditorium (*middle*) and Thai statue (*bottom*) scenes.

reference image. The occlusion camera construction takes uniformly about 1s for our scenes, consequently most of C_{time} goes to step two. Our current implementation projects the scene meshes in software (on the CPU of the host computer) using the distortion map, and then rasterizes the projected meshes on the graphics card. As future work we will move the entire second step on the GPU (graphics processing unit), by taking advantage of the vertex level programmability of recent GPUs. This will virtually eliminate the OCRI construction time C_{time} and will make the performance of the OCRI similar to that of the depth image. Note that the times of the third step of OCRI construction (third OCRI column in Table 1) are comparable to the DI times (second DI column in Table 1).

OCRIs are one of many possible 3D representations. Table 2 gives an approximate comparison between OCRIs and depth images (DIs), layered depth images (LDIs), light fields (LF), unstructured light fields (ULF), and ray-phase space (RPS) representation. The comparison is based on the four Happy Buddha statues scene, which our NVIDIA Quadro FX 3400 graphics card renders at ~8Hz, hence the 0.12s DI construction time. Constructing and LDI for such a scene requires first rendering and then merging approximately 4 x 4 = 16 construction depth images [37, 35, 4], at a time cost of 16 x 0.12 x 2 = 3.84s. The LF construction time is 16 x 16 x 0.12 = 30.72s, figure obtained with a rather modest back plane resolution (16x16). Constructing the ULF [8, 3] requires fewer images (32 for the table entry) since user interaction or a heuristic is used to identify the most important views.

The ray-phase space representation [40] is a 4D plenoptic representation which instead of using two planes in front of the desired viewpoints for parameterization, uses a 2D parameterized surface that surrounds the scene of interest, and then a 2D parameterization of the outgoing rays for each surface point. The approach is similar to surface light fields [44] and to models developed for general imaging systems [11]. In our case, a natural parameterization surface is the sphere described by the revolving screen, whose visible area is approximately 38% of the area of a sphere with a radius of 5 inches, or 120 square inches. For an average sampling rate of one point per square millimeter and 16 x 16 rays for each point, the total number of rays is 19 million. Generating these rays requires rendering the scene at least 57 times, for a construction time of 6.84s, which ignores the cost of rearranging the rays according to the ray-phase parameterization.

For the 720x480 resolution, the 8 bit R, G, B, A channels and the 32 bit floating point z channel amount to 2.6MB. The 16 floats needed to store the view are negligible. The LDI adds only a few non-redundant samples. The uncompressed LF requires considerable storage space. Compression could reduce the memory consumption 10 or 100 fold, with the corresponding compression and decompression time costs and loss of quality [16]. The ULF has a more manageable uncompressed size, but is less redundant and thus compresses less well. The 19 million color samples of the RPS representation translate to 76MB.

The OCRI requires twice the storage since the points are perturbed and the x and y coordinates need to be store explicitly (whereas in the DI or LDI, they are provided implicitly by the pixel coordinates). We have charged 8 additional bytes for per pixel floating point x and y, however a slimmer 2 byte fixed point representation would work equally well. Whereas DIs and OCRIs compress well using the coherence of the single layer, the variable depth of the multi-layered LDI pixels hinder compression. Note that the distortion map is only needed during construction.

The plenoptic representations are not supported by our 3D display. On a regular LCD, the scene can be rendered at refresh rate (60 Hz for our system) when using the DI, LDI, or OCRI. The LF and ULF representations have been shown to support frame rates as high as 20Hz. Quality wise, the OCRI produces images comparable to those rendered using the original geometry. DIs suffer from disocclusion errors. LDIs produce lower quality images since they lack connectivity and are rendered by splatting [37, 35, 4]. Estimating the size and shape of the splats cannot be done both efficiently and accurately. The splats are typically overestimated and modeled as rectangles or disks, which produces blockiness. Typical artifacts when rendering from plenoptic representations are coarseness (due to low spatial sampling resolution, as it is the case for the numbers chosen for this table), and compressions artifacts.

In conclusion, OCRIs, like DIs and LDIs, capture the scene well and are compact since they use the depth and the diffuse surface assumption to reuse color samples over a continuum of nearby views. OCRIs do away with disocclusion errors, the major disadvantage of depth images, while the The plenoptic representations have the advantage of not requiring geometry, and can be acquired with a tracked camera. The plenoptic

representations do provide limited support for view dependent effects, such as glossiness. Highly reflective surfaces are not supported since these entail the need of a very high spatial sampling resolution.

IX. Conclusion

We have described a novel occlusion camera that distorts the reference rays at depth discontinuities to reach behind occluders and to avoid disocclusion errors. We have demonstrated the effectiveness of the occlusion camera reference image for accelerating the rendering on a volumetric 3D display. The OCRI provides an efficient scene representation by adapting the level of detail to the reference view and by discarding samples that are not visible in any nearby views. A 3D image built from an OCRI supports disocclusion error free viewing for a fixed user. The OCRI stores most of the samples needed to form complete left and right eye images under normal head translations.

The OCRI brings a substantial speedup over rendering the 3D image from the complete geometric model. However, the frame rate is still far from interactive. Possible approaches for further increasing the 3D image rendering performance are simplification of the mesh produced by the OCRI, taking advantage of GPU versatility for efficiently converting triangles into the 3D image, and progressive refinement.

Volumetric displays cannot reproduce opaque surfaces, and the limitation will remain for the foreseeable future. Depth images and OCRIs remove hidden surfaces and improve the readability of 3D images that visualize surfaces. In some scientific visualization applications, the scene of interest contains opacity data. We will extend our approach to such data: a front volume becomes opaque if it is of sufficient thickness, case in which the data behind it can be safely eliminated, improving performance.

One of the great advantages of our display is its natural support for collaborative applications. Two or more users can simultaneously view the 3D image, each with the proper perspective, without the requirement of encumbering head gear. As presented, the OCRI approach works only for a single viewer. We will investigate creating occlusion cameras that provide all samples needed for two reference views.

Our solution for alleviating disocclusion errors is based on creating a custom non-pinhole camera with fast projection. This allows harnessing the impressive power of modern GPUs for solving a problem far from the classical computer graphics problem of providing perspective views of a 3D scene. We believe that the same methodology can be applied to solving other challenging problems in computer graphics and beyond.

Acknowledgments

We would like to thank Chunhui Mei and Elisha Sacks for their contributions to the development of the single-pole occlusion camera, on which this work builds. This work would not have been possible without the help of Christoph Hoffmann. This work was supported by the NSF through grant CNS-0417458, by Intel and Microsoft through equipment and software donations, and by the Computer Science Department of Purdue University. The bunny, Happy Buddha, and Thai statue models were obtained from the Stanford 3D Scanning Repository [39].

References

[1] Actuality Systems http://www.actuality-systems.com/
[2] ATI http://www.ati.com/
[3] C. Buhler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen. “Unstructured Lumigraph Rendering”. In Proc. of *SIGGRAPH 2001*.
[4] C. F. Chang, G. Bishop, and A. Lastra, “LDI Tree: A Hierarchical Representation for Image-Based Rendering,” in Proc. of *SIGGRAPH’99*.
[5] Dimension Technologies, http://www.dti3d.com/
[6] E. Downing et al, “A Three-Color, Solid-State, Three-Dimensional Display,” Science 273, 5279, August 1996.
[7] G. Favalora et al, “100 Million-voxel volumetric display,” in Proc. of *SPIE 16th Annual International Symposium on Aerospace/Defense Sensing, Simulation, and Controls*, 2002.
[8] T. Fujii, T. Kimoto, and M. Tanimoto, “A new flexible acquisition system of ray-space data for arbitrary objects”, IEEE Trans. Circuits Syst. Video Technol., vol. 10, pp. 218-224. Mar. 2000.
[9] A. Glassner, “An Introduction to Ray tracing,” Morgan Kaufman, 1989.
[10] S. Gortler, R. Grzeszczuk, R. Szeliski and M. Cohen, “The Lumigraph,” in Proc. of *SIGGRAPH 96*, 43-54.
[11] D. Grossberg and S. Nayar. A General Imaging Model and a Method for Finding its Parameters. In Proceedings of ICCV 2001.
[12] R. Gupta and R.I. Hartley, “Linear Pushbroom Cameras,” IEEE Trans. Pattern Analysis and Machine Intell., vol. 19, no. 9 963–975, 1997.
[13] A. Isaksen, L. McMillan, and S. Gortler, “Dynamically reparameterized light fields,” in Proc. of *SIGGRAPH 2000*.
[14] H. E. Ives, “A camera for making parallax panoramagrams,” Journal of Optical Society of America, 17, Dec. 1928, pp. 435-439.
[15] M. Halle, “Autostereoscopic displays in computer graphics,” in Proc. of *SIGGRAPH 97*, 31(2), May 1997, pp 58-62.
[16] M. Levoy and P. Hanrahan, “Light Field Rendering,” in Proc. of *SIGGRAPH 96*, 31-42, 1996.
[17] LightSpace Technologies. http://www.lightspacetech.com/
[18] M. Lucente, “Interactive three-dimensional holographic displays: seeing the future in depth,” in Proc. of *ACM SIGGRAPH 97*, 31(2), May 1997.
[19] L. Levkovich-Maslyuk et al., “Depth Image-Based Representation and Compression for Static and Animated 3-D Objects”, in IEEE Transactions on Circuits and Systems for Video Technology, Vol. 14, NO. 7, pp. 1032-1045.
[20] W. Mark, L. McMillan, and G. Bishop, “Post-Rendering 3D Warping,” in Proc. of 1997 Symposium on Interactive 3D Graphics, 1997.
[21] W. Matusik, H. Pfister, “3D TV: a scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes,” in Proc. of *SIGGRAPH 2004*.
[22] N. Max and K. Oshaki, “Rendering trees from precomputed z-buffer views,” in *Rendering Techniques ’95: Proceedings of the Eurographics Rendering Workshop 1995*, 45–54, Dublin, June 1995.
[23] D. McFarlane, “A true volumetric 3D display,” available at http://www.utdallas.edu/~dlm/A%20True%20Volumetric%20Three%20Dimensional%20Display.htm
[24] L. McMillan and G. Bishop, “Plenoptic modeling: an image-based rendering system,” in Proc. of *SIGGRAPH ‘95*, pp. 39-46.
[25] C. Mei, V. Popescu, and E. Sacks, “The Occlusion Camera,” in Proc. of *Eurographics 2005*, Computer Graphics Forum, vol. 24, issue 3, 2005.
[26] Microsoft DirectX, http://www.microsoft.com/windows/directx/
[27] NVIDIA Corporation http://www.nvidia.com/
[28] OpenGL http://www.opengl.org/
[29] T. Pajdla, “Geometry of Two-Slit Camera,” Research Report CTU–CMP–2002–02.
[30] K. Perlin, S. Paxin, J. Kollin, “An autostereoscopic display,” in Proc. of *SIGGRAPH 2000*, pp. 319-326.
[31] Perspecta Display, by Actuality Systems. http://www.actualitysystems.com/site/content/perspecta_display1-9.html
[32] V. Popescu and D. Aliaga, “The Depth Discontinuity Occlusion Camera,” to appear in *Proc. of ACM Symposium on Interactive 3D Graphics and Games*, 2006.

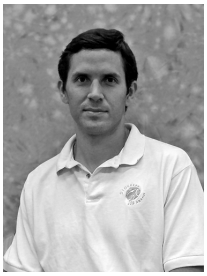
[33] V. Popescu and A. Lastra, "The Vacuum Buffer," in Proc. of *ACM Symposium on Interactive 3D Graphics*, Chapel Hill, 2001.
[34] V. Popescu et al, "The WarpEngine: An Architecture for the Post-Polygonal Age," in Proc. of *SIGGRAPH 2000*.
[35] V. Popescu, A. Lastra and M. Oliveira, "Efficient Warping for Architectural Walkthroughs Using Layered Depth Images," in Proc. of *IEEE Visualization 1998*.
[36] P. Rademacher and G. Bishop, "Multiple-center-of-Projection Images," in proc of SIGGRAPH '98, 199–206.
[37] J. Shade, et al, "Layered Depth Images," in Proc. of *SIGGRAPH 98*, 231- 242.
[38] A. Smolic and P. Kauff, "Interactive 3-D video representation and coding technologies", in Proceedings of the IEEE, vol. 93, issues 1, pp. 98-110, Jan 2006.
[39] The Stanford 3D Scanning Repository, http://graphics.stanford.edu/data/3Dscanrep/
[40] A. Stern and B. Javidi, "Ray phase space approach for 3D imaging and 3D optical data representation", IEEE/OSA journal of display technology, vol. 1(1), pp. 141-150, Sept. 2005
[41] A.C. Traub, "Stereoscopic Display Using Varifocal Mirror Oscillations," Applied Optics, Vol. 6, No. 6, June 1967, pp. 1085-1087.
[42] L. Westover, "Footprint evaluation for volume rendering," in Proc. of *SIGGRAPH 1990*, volume 24(4), pp. 367-376.
[43] T. Whitted, "An improved illumination model for shaded display," Communications of the ACM, v. 23, n.6, pp 343-349.
[44] D. N. Wood. et al. 2000. "Surface light fields for 3D photography". Proceedings, SIGGRAPH '00, ACM Press, pp. 287-296.
[45] J. Yu and L. McMillan, "General Linear Cameras", in Proc. of the *8th European Conf. on Computer Vision* (ECCV), 2004, Volume 2, 14-27.

Voicu Popescu received a B.S. degree in computer science from the Technical University of Cluj-Napoca, Romania in 1995, and a Ph.D. degree in computer science from the University of North Carolina at Chapel Hill, USA in 2001.

He is an assistant professor with the Computer Science Department, Purdue University. His research interests lie in the areas of computer graphics, computer vision, and visualization. His current projects include perceptual evaluation of rendered imagery and 3D displays, research and development of 3D scene acquisition systems, research of algorithms for fast and accurate rendering of view-dependent effects, and research and development of next generation distance learning systems.

Paul Rosen received a B.S. degree in computer science from Purdue University West Lafayette, Indiana in 2004.

He is a Graduate Research Assistant with the Computer Science Department, Purdue University. His research interests lie in the areas of computer graphics, 3D displays, image based rendering, and 3D scene acquisition and reconstruction.



Dan Aliaga received a B.S. degree in computer science from Brown University in 1991, and a Ph.D. degree in computer science from the University of North Carolina at Chapel Hill, USA in 1999.

He is an assistant professor with the Computer Science Department, Purdue University. His research interests lie in the areas of computer graphics, computer vision, and visualization. His research interests lie in the areas of computer graphics, computer vision, and scientific visualization.