The Depth Discontinuity Occlusion Camera

Voicu Popescu*
Purdue University

Daniel Aliaga†
Purdue University

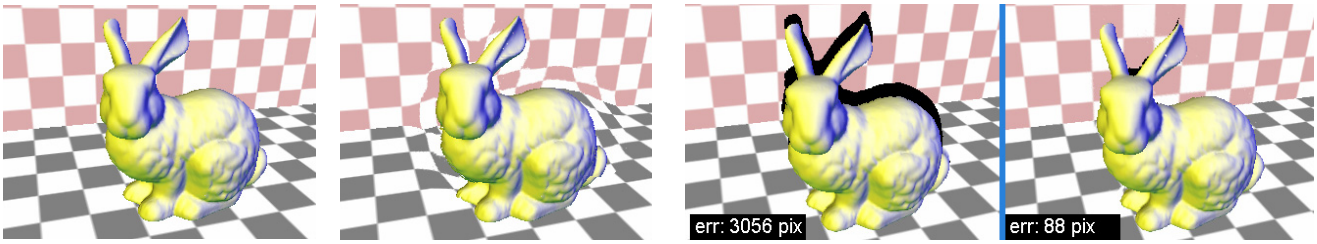

Figure 1 Depth image, DDOC reference image, and corresponding pair of frames. The DDOC reference image alleviates disocclusion errors, which are quantified as number of missing pixels.

Figure 2 Additional examples with the disocclusion errors highlighted in white.

Abstract

Rendering a scene using a single depth image suffers from disocclusion errors as the view translates away from the reference view. We present the depth discontinuity occlusion camera (DDOC), a non-pinhole camera that samples surfaces which are hidden in the reference view, but are likely to become visible in nearby views. The DDOC reference image alleviates disocclusion errors; since it has a single layer, it maintains the advantages of regular depth images such as bounded number of samples, efficient incremental processing, and implicit connectivity. The DDOC is defined by the reference view *and* the geometry it encompasses. The reference planar pinhole camera is 3D distorted at depth discontinuities. The distortion is fine-grain controlled with a distortion map, which allows handling complex scenes. The DDOC provides fast projection so the reference image is constructed efficiently with the feed-forward graphics pipeline.

CR Categories: I.3.3. [Computer Graphics]—Three-Dimensional Graphics and Realism.

Keywords: image-based rendering, disocclusion errors, occlusion culling.

*email: popescu@cs.purdue.edu
†email: aliaga@cs.purdue.edu

1. Introduction

Image-based rendering by 3D warping (IBRW) is well suited for interactive computer graphics applications because of compact scene representation and efficient rendering. The scene is modeled with images enhanced with per-pixel depth [McMillan 1995]. The depth and color samples of the reference depth images are re-projected (warped) to render the scene from novel views.

By definition, a depth image provides the best color, occlusion culling, and level of detail for the reference view. The level of detail remains appropriate for nearby views. However, even minimal viewpoint translations create two problems. First, the color data becomes obsolete for surfaces with view-dependent (e.g. specular) appearance. Second, the occlusion culling solution becomes obsolete: some surfaces that were not visible in the reference view are visible in the novel view, causing disocclusion errors.

This paper addresses the problem of disocclusion errors. Most previous solutions take the approach of rendering the desired image from several reference depth images. A first disadvantage of the approach is its high cost. Disocclusion errors are small groups of missing samples, scattered throughout the scene. No single additional depth image captures them all. Each additional depth image has the same high cost while contributing fewer and fewer new samples. A second important disadvantage is that the rendering cost, which was bounded in the case of a single depth image, becomes unpredictable, which is a severe limitation for applications that require a guaranteed minimum frame rate.

Instead of filling in disocclusion errors, we take the approach of preventing them. The reference image is called upon to provide samples for a continuum of nearby viewing locations. Therefore, the decision of including a sample in the reference image cannot

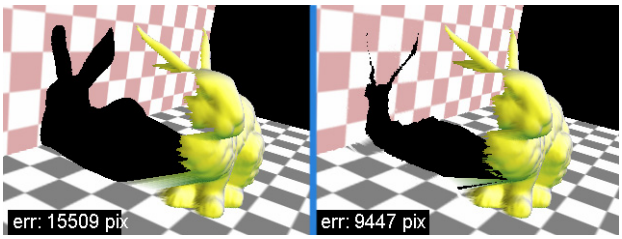

Figure 3 The DDOC reference image provides an occlusion culling solution (*right*) that is less aggressive than that of a regular depth image (*left*). The scene complexity is safely reduced without discarding samples needed in nearby views.

be simply based on whether the sample is visible in the reference view or not. A mechanism is needed for including samples that are hidden in the reference view but are likely to become visible in nearby views. A simple such mechanism is based on the observation that the samples needed are close to the discontinuities in the depth map of the reference view. We introduce the *depth discontinuity occlusion camera* (DDOC), a non-pinhole camera model obtained by 3D distorting the reference planar pinhole camera at depth discontinuities.

The DDOC reference image shown in Figure 1 (*second image*) stores additional samples of the floor and back wall all around the silhouette of the bunny. The DDOC reference image does not store all the samples of the scene, nor should it. Samples that remain hidden in nearby views are correctly discarded, which provides occlusion culling (Figure 3).

The DDOC reference image has a single layer thus it maintains the advantages of regular depth images such as bounded number of samples, efficient incremental processing, and implicit connectivity. The DDOC (camera) model consists of the reference planar pinhole camera augmented with a distortion map. The DDOC projects a vertex efficiently by first projecting with the pinhole onto the distortion map and then applying the distortion stored at the distortion map location. The availability of fast projection enables constructing the DDOC reference image efficiently with standard feed-forward graphics hardware.

Given a scene S and a reference view specified as a planar pinhole camera $PPHC_0$, the main steps of our algorithm are as follows:

1. Compute $DDOC_0$ from $PPHC_0$ and S
2. Construct ref. image $DDOCRI_0$ by rendering S with $DDOC_0$
3. For each desired view $PPHC$
    3.1. Render desired image by warping $DDOCRI_0$ to $PPHC$

The DDOC model depends not only on the reference view but also on the geometry it sees. The camera model is customized for every reference view. The resulting DDOC reference image provides a high-quality, bounded-cost approximation of the scene (see Figure 1, Figure 2, and accompanying video). The remainder of the paper is organized as follows. The next section discusses prior work. The following three sections describe each main step of our algorithm. Section 6 concludes with results and discussion.

2. Prior work

The first attempt to alleviate disocclusion errors in IBRW was to simply warp several depth images from viewpoints close to the desired viewpoint [McMillan 1995]. In post-rendering warping [Mark 1997] conventional rendering is accelerated by warping two reference images. Even when the two images have views close to the desired view, disocclusion errors still occur.

Another approach is to combine several depth images into a layered representation, in a preprocessing step. The layered representations generalize depth images to allow for more than one depth sample along a ray. Max [1995] renders trees with multi-layered z-buffers. The challenge is to decide how many samples to store along each ray. The goal is to store only and all the samples that will eventually be visible during rendering. The layered depth image (LDI) [Shade 1998] addresses this problem by combining several depth images from nearby view and storing the union of their depth-and-color samples. Popescu [1998] uses LDIs to accelerate rendering of architectural walkthroughs. Chang [1999] proposes a hierarchical scene representation based on LDIs. The disadvantages common to layered representations are high construction cost, unbounded number of samples, and lack of sample connectivity. Layered reference images are typically used by splatting, which lowers the quality of the desired image.

LDIs are not a conservative solution to the disocclusion error problem because it can happen that a surface visible in the desired view is not visible in any of the construction images. The vacuum buffer [Popescu 2001] proposes a conservative run-time solution based on warping additional depth images until all subvolumes of the desired view frustum have either revealed the surfaces they contain or have proven to be empty. The method is expensive because it requires executing a generalized z-buffer algorithm for every frame.

Visibility problems similar to disocclusion errors are encountered in techniques that reduce the geometry load using impostors [Maciel 1995]. Techniques have been developed for correctly depth-compositing the impostor with the surrounding geometry [Schaufler 1998], but such techniques rely on the planarity of the impostor and cannot be applied to depth image warping.

The problem of avoiding disocclusion errors is dual to the problem of occlusion culling, which has been studied extensively in conventional rendering. The novelty of our approach consists in reducing the burden of visibility computation to computing a camera model that handles samples close to depth discontinuities more leniently. Once the camera model is established, the power of hardware rasterization and z-buffering is harnessed to perform the actual visibility computation.

Non-pinhole cameras have been developed in computer vision to model complex lens and catadioptric systems. The general linear camera [Yu 2004], and its subclasses the pushbroom camera [Gupta 1997] and the two-slit camera [Pajda 2002] are not sufficiently powerful to address disocclusion errors for complex scenes.

Non-pinhole camera models have also been developed in computer graphics. The light field [Levoy 1996] and the lumigraph [Gortler 1996] are 2D arrays of planar pinhole cameras, which solve the problem of disocclusion errors for a sub-region of the scene, but are expensive and do not scale well. The multiple-center-of-projection camera (MCOP) [Rademacher 1998] samples the scene using a planar pinhole camera with a single column of pixels along a user chosen path. Disocclusion errors are avoided but the method requires user input. The light field and the multiple-center-of-projection camera are constructed by rendering the scene hundreds of times. The lengthy construction time is an important disadvantage when the goal is acceleration and precludes using the methods for dynamic scenes.

The single-pole occlusion camera

Our paper builds on the occlusion camera concept introduced by Mei [1995]. Their occlusion camera is defined by radially distorting the reference planar pinhole camera around a *pole*. The

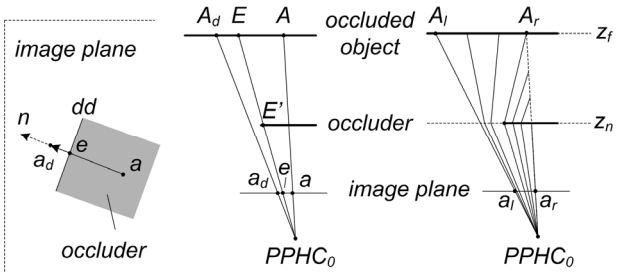

Figure 4 Illustration of distortion at depth discontinuity.

pole is defined as the projection of the centroid of the occluder onto the image plane. The single-pole occlusion camera is limited to a single simple occluder. In the case of more complex occluders, the coarsely specified distortion causes losing samples that are visible in the reference view. To recover the lost samples the occlusion camera reference image has to be complemented with a regular depth image. In the case of the DDOC, the automatically computed fine grain distortion produces reference images that successfully handle complex scenes.

3. Camera model computation

3. 1. Overview

Given a scene S and a reference view $PPHC_0$, the goal is to compute a camera model that sees around the occluders to capture samples that are barely hidden. We achieve this with a distortion that acts on samples close to depth discontinuities.

In Figure 4, the left drawing shows a fragment of the image plane of $PPHC_0$. The depth discontinuity dd separates the occluder (grey) from the occluded object. The $PPHC_0$ projection a of a point A is moved to a_d, along the normal n of dd.

The middle drawing shows the effect of the distortion in the plane defined by the center of projection of $PPHC_0$, a, and e. The image plane points a, e, and a_d are those seen in the left image. They are the projections of 3D points A, E and E', and A_d, respectively. The distortion magnitude depends on z: it is 0 up to the depth of the occluder z_n, and it is largest d_f at the depth of the occluded object z_f. Beyond z_f the distortion magnitude remains constant, d_f. This way, the occluded object samples move away from the occluder. The samples close to the depth discontinuity clear the edge and remain unoccluded, thus present, in the DDOC reference image. A projects where A_d would project in the absence of distortion.

The right drawing shows the same plane as the middle drawing, for the same scene, with and occluder that ends at E' (not labeled for clarity). The drawing shows the effect of the distortion on the rays of $PPHC_0$. The effect of the distortion is local, and the size of the neighborhood of the depth discontinuity affected by the distortion is specified by the user with a parameter D. Only the rays between A_l and A_r are affected. D is measured in pixels and equals half the distance between a_l and a_r. The points A, E, and A_d from the middle drawing are located between A_l and A_r.

No ray is affected before z_n. Between z_n and z_f, the rays are moved along n, towards the occluder (which causes the samples to move away from the occluder). After z_f, the rays continue straight, as the distortion remains constant d_f. It is interesting to note that the rays that sample the occluder are clipped by ray R. This simply means that points beyond ray R are not affected by the distortion caused by the edge of the occluder. The DDOC samples the entire viewing frustum of $PPHC_0$.

3. 2. Distortion map construction

The DDOC model is given by $PPHC_0$ and a distortion map. The distortion map has the same resolution as $PPHC_0$ and specifies the distortion for each $PPHC_0$ ray. A distortion sample is defined by the five-tuple (dir_u, dir_v, z_n, z_f, d_f), where dir is a 2D unit vector that specifies the image plane direction of the distortion, and (z_n, z_f) gives the z interval where the distortion increases from 0 to d_f. Like in the case of the single-pole occlusion camera, the distortion amount increases linearly with $1/z$, which makes that the perturbed rays are line segments, as shown in Figure 4, *right*. The parameter d_f specifies the amount of distortion. Larger distortion values make the DDOC reach farther back behind the occluder, and store more hidden samples.

Given the scene S and the reference view $PPHC_0$ the distortion map $DMAP$ is built as follows:

1. Compute z-buffer ZB by rendering S with $PPHC_0$
2. Compute 1-bit depth discontinuity map EB using ZB
3. For each non zero edge pixel e in EB
   Splat e in $DMAP$
4. For each non-zero edge pixel e in EB
   Adjust size of splat defined by e
5. For each pixel in $DMAP$
   Remove orphan distortion samples
6. For each pixel in $DMAP$
   Finalize distortion magnitude

Steps 1 and 2 compute the depth discontinuity map EB. The scene is first rendered with $PPHC_0$ to obtain the reference view z-buffer ZB. Depth discontinuity pixels are located where the second order finite depth difference is larger than a threshold [Popescu 2000].

Step 3 begins setting the $DMAP$ values in a first pass over the non-zero pixels in EB. For an edge pixel e, the distortion direction dir is set to be perpendicular to the local depth discontinuity direction, which is computed by least squares fitting a line to edge pixels in a neighborhood of e. The vector dir is set to point away from the occluder, towards the larger z values. z_n and z_f are set to the near and far z values that create the depth discontinuity at e. The edge pixel is splatted into the $DMAP$ as a circle centered at e with radius D. Recall that D is a user provided parameter that bounds the distortion magnitude. When a splat sample is written into the $DMAP$, dir, z_n, and z_f are copied from the edge pixel. The remaining scalar d_f is set later, at step 6.

During the construction phase, each $DMAP$ location stores 3 scalars in addition to the distortion sample five-tuple: the coordinates c_u and c_v of the center and the radius r of the splat that sets the distortion at the current pixel. When a splat covers a $DMAP$ location f that has already been set, the splat whose center is closest to f wins. The distance to the center of the splat that currently owns f is computed using the scalars c_u and c_v.

Step 4 takes a second pass over the depth discontinuity pixels, to detect and eliminate distortion conflicts. Two depth discontinuity pixels conflict if they affect the same $DMAP$ location and if their directions form an angle larger than a threshold, which we set to 90^o in our implementation. The area affected by the distortion of an edge pixel is given by the radius of the splat r. The radius of the splat is shrunk such that the splat circle passes just outside the closest $DMAP$ location with whom the splat conflicts.

Step 5 takes a first pass over the $DMAP$ locations to enforce the reduced splat radii. If a $DMAP$ location points to splat center that is farther than the radius of that splat, the distortion is reset (the distortion sample is deleted).

Step 6 is the second and final pass over the *DMAP* and finalizes the distortion samples at each location. The distortion magnitude d_f is set according to the relative position between *f* and the splat center. The distortion has to move samples from the hidden side of the edge to the visible side of the edge. Samples that were already on the visible side of the edge need to move away to make room for the newly visible samples. Otherwise samples visible in the reference view are overwritten. We achieve this effect with a distortion magnitude that linearly decreases from *r* to 0, as the signed distance from the edge increases from *–r* to *r* (Figure 5). This way, samples that have undistorted projection in the band [*-r,* 0] are distorted to the band [0, *r*/2]. The samples that would have projected in the interval [0, *r*] are condensed in the interval [*r/2*, *r*]. The distortion moves a sample from *j* to *e*, from *e* to *g*, and does not move a sample that is left of *h* or right of *j*.

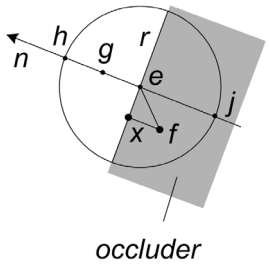

Figure 5 Distortion tuning according to distance from edge.

Let *x* be the signed distance from a distortion map location *f* to the edge. If *x* is between [*-r*, *r*], the distortion at *f* is given by

$$(2r-(x+r))r/2r = (r-x)/2$$

For *x* values of *–r, 0, r*, the distortion is *r, r/2*, and 0, respectively. The distortion works by compressing the visible [0, *r*] band to the band [*r/2, r*], and by shifting and compressing the hidden [*-r*, 0] band to the band [0, *r/2*]. Of course, this implies a loss of resolution, which can be avoided by increasing the resolution of the desired image.

Figure 6 shows a distortion map example. The radii of the splats at the ears of the bunny were reduced to accommodate the conflicting distortion requirements. The disocclusion capability of the DDOC is further increased if the splats are asymmetrical and a hidden band of width *r* is compressed to a visible band of width r/a_f, where a_f is a user chosen asymmetry factor. For example, when a_f is 2, the distortion maps [*-r*, 0] to [0, *r/4*], and [0, *r/4*] to [0, *r/2*]. This means that only *r/2* room is needed on the visible side of the edge, and comes at a price of reducing the resolution close to the edge by a factor of 4 compared to the original reference image (Figure 7).

4. Reference image construction

Once the distortion map is computed, the DDOC reference image is constructed by projecting the scene vertices with the DDOC and then by rasterizing the resulting triangle mesh in hardware.

$$(u_u, v_u, z) = PPHC_0(P)$$
$$(dir_u, dir_v, z_n, z_f, d_f) = DistMap(u_u, v_u)$$
$$d(z) = \begin{cases} 0, z < z_n \\ \dfrac{1/z_n - 1/z}{1/z_n - 1/z_f} d_f, z_n \le z \le z_f \\ d_f, z > z_f \end{cases}$$
$$(u_d, v_d) = (u_u, v_u) + (dir_u, dir_v) d(z)$$

Equation 1 Projection with DDOC.

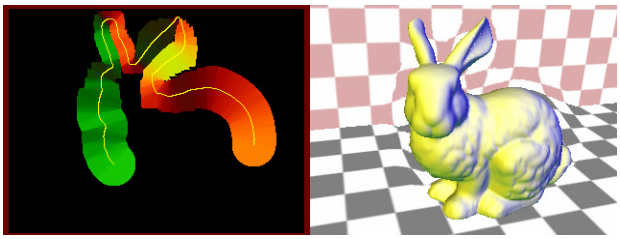

Figure 6 (*Left*) Distortion map visualization. The colors indicate the distortion direction. (*Right*) Resulting DDOC reference image.

A point *P* is first projected with the reference planar pinhole camera $PPHC_0$ (Equation 1) to its undistorted coordinates (u_u, v_u, z). The distortion is given by the distortion map location (u_u, v_u). The distortion magnitude is linearly interpolated in 1/z. The distorted coordinates are found by adding the distortion vector to the undistorted coordinates.

Because of the distortion, model space lines do not project to screen space lines. To control the error introduced by conventional rasterization of the projected triangles, we subdivide the triangles until their screen space edge lengths are smaller than a user chosen threshold. In practice we use a threshold of 1 pixel.

5. Reference image warping

Once the DDOC reference image is constructed, it is used in lieu of the scene to render the desired view. In order to efficiently warp the DDOC to the reference image we compute the triangle mesh inferred by the depth and color samples it stores. To recover the 3D position of a DDOC reference image sample one needs to be able to unproject the sample back into model space. The distorted coordinates (u_d, v_d, z) are not sufficient for unprojection, since the distortion is sample-based and cannot be inverted.

To enable unprojection we augment the DDOC reference image with two additional channels, one for each of the components of the distortion vector. The additional channels are computed by rasterizing the projected triangles a second time, with the distortion vector components of each vertex stored in the red and green channel of vertex color. The framebuffer is read back and provides for each DDOC sample the distortion vector used to

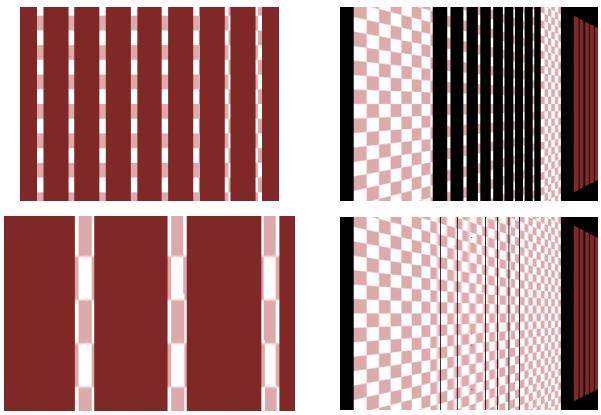

Figure 7 (*Top*) Reference depth image (*left*) used to render the scene from a novel view (*right*). (*Bottom*) DDOC reference image constructed with asymmetrical splats (magnified fragment, *left*) and novel view (*right*). Although the background is heavily occluded, the DDOC reference image drastically alleviates disocclusion errors.

Figure 8 Fragment of the depth image (*left*) and DDOC reference image (*right*) used to render the frames in Figure 2. The distortion introduced by the portal virtually enlarges its frame and more surfaces are sampled (see floor, seat rows).

create it. Once the distortion vector (d_u, d_v) of a sample is known, the undistorted sample (u_u, v_u, z) is recovered as (u_d-d_u, v_d-d_v, z). The model space 3D point is obtained by pushing back the center of pixel (u_u, v_u) to depth z.

Once the triangle mesh defined by the DDOC reference image is known, it is rendered for each novel desired view, effectively warping the reference image.

6. Results and discussion

The DDOC reference image trades (u, v) resolution for resolution along the same ray. When many hidden samples are stored along with the samples visible from the reference view, as is the case for asymmetrical splats, the resolution of the DDOC reference image needs to be increased to maintain the same sampling rate.

For our test scenes shown in Figures 1, 2, 7, and 8, the DDOC approach to alleviating disocclusion errors proved to be very effective. We quantify disocclusion errors in a frame by rendering a truth image from geometry and counting the number of truth image samples that are not present in the frame (Figure 1, 2, 3, and Figure 9 on color plate). We rendered sequences of frames on a cube centered at the reference view (see video). The disocclusion errors in the case of the DDOC reference image were, on average, between 7% and 13% of the disocclusion errors obtained when using a regular depth image as reference. The desired view has a horizontal field of view of 45° and we render 90°x90° reference images to avoid disocclusion errors at the reference image frame boundary. To support full desired view rotations, 6 such reference images are needed to cover all view directions.

The DDOC reference image construction is efficient compared to light fields, MCOPs or LDIs. The factors affecting performance are maximum splat size D, number of edge pixels, and subdivision level. The reference image construction times vary for our scenes between 1s and 30s. We will investigate constructing the DDOC reference image in hardware, by projecting with a vertex program. Once the DDOC reference image is built, the scene is rendered with it at refresh rate.

The DDOC is a promising new approach to visibility computation which we will further develop and apply to complete systems for interactive rendering of massive geometric models. An application of great interest to us is the reduction of the bandwidth requirement for 3D displays, which is presently a severe bottleneck for this exciting emerging technology.

We believe that the idea of creating a custom camera to solve a difficult computer graphics problem can be applied to other problems such as accurate interactive rendering of reflections and of soft shadows.

7. 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. We thank the reviewers for generously donating their time, and for their suggestions for improving this paper. 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 model was obtained from the Stanford 3D Scanning Repository [S3DSR].

References

CHANG, C. F., BISHOP G., AND LASTRA A.. *LDI Tree: A Hierarchical Representation for Image-Based Rendering*. In proc. of SIGGRAPH'99.

GORTLER S., GRZESZCZUK R., SZELISKI R. AND COHEN M.. *The Lumigraph*. In proc. of SIGGRAPH 96, 43-54.

GUPTA, R. AND HARTLEY, R.I. 1997: *Linear Pushbroom Cameras*. IEEE Trans. Pattern Analysis and Machine Intell. vol. 19, no. 9 (1997) 963–975.

LEVOY M., AND HANRAHAN P. *Light Field Rendering*. In proc. of SIGGRAPH 96, 31-42 (1996).

MEI, C., POPESCU V., AND SACKS E. *The Occlusion Camera*. In proc. of Eurographics 2005, Computer Graphics Forum, vol. 24, issue 3, sept 2005.

MACIEL P., AND SHIRLEY P. *Visual Navigation of Large Environments Using Textured Clusters*, Symposium on Interactive 3D Graphics pp 95-102, 1995

MARK W., MCMILLAN L., AND BISHOP G. *Post-Rendering 3D Warping*. In proc. of 1997 Symposium on Interactive 3D Graphics (Providence, Rhode Island, April 27-30, 1997).

MAX, N., AND OHSAKI, K. *Rendering trees from precomputed z-buffer views*. In Rendering Techniques '95: Proceedings of the Eurographics Rendering Workshop 1995, 45–54, Dublin, June 1995.

MCMILLAN, L., AND BISHOP, G. *Plenoptic modeling: An image-based rendering system*. In proc. SIGGRAPH '95, 39-46.

PAJDLA, T *Geometry of Two-Slit Camera*. Research Report CTU–CMP–2002–02.

PFISTER, H., ZWICKER M., BAAR, J. VAN, AND GROSS M.. *Surfels: Surface Elements as Rendering Primitives*. In proc. of SIGGRAPH 2000, 335-342.

POPESCU, V., AND LASTRA, A. *The Vacuum Buffer*. In proc. of ACM Symposium on Interactive 3D Graphics, 2001.

POPESCU V., et al. *The WarpEngine: An Architecture for the Post-Polygonal Age*. In proc. of SIGGRAPH 2000.

RADEMACHER P, AND BISHOP, G. *Multiple-center-of-Projection Images*. In proc of SIGGRAPH '98, 199–206.

SCHAUFLER G.. *Nailboards: A Rendering Primitive for Image Caching in Dynamic Scenes*. In proc. of Eurographics Rendering Workshop, pp. 151-162, 1997.

SHADE J. et al. *Layered Depth Images*, In proc. of SIGGRAPH 98, 231-242.

YU J., AND MCMILLAN L. *General Linear Cameras* In 8th European Conference on Computer Vision (ECCV), 2004, Volume 2, pp 14-27.