EUROGRAPHICS 2005 / M. Alexa and J. Marks
(Guest Editors)

Volume 24 (2005), Number 3

The Occlusion Camera

Chunhui Mei, Voicu Popescu, and Elisha Sacks

Purdue University

Abstract

We introduce the occlusion camera: a non-pinhole camera with 3D distorted rays. Some of the rays sample surfaces that are occluded in the reference view, while the rest sample visible surfaces. The extra samples alleviate disocclusion errors. The silhouette curves are pushed back, so nearly visible samples become visible. A single occlusion camera covers the entire silhouette of an object, whereas many depth images are required to achieve the same effect. Like regular depth images, occlusion-camera images have a single layer thus the number of samples they contain is bounded by the image resolution, and connectivity is defined implicitly. We construct and use occlusion-camera images in hardware. An occlusion-camera image does not guarantee that all disocclusion errors are avoided. Objects with complex geometry are rendered using the union of the samples stored by a planar pinhole camera and an occlusion camera depth image.

Categories and Subject Descriptors (according to ACM CCS): I.3.3. [Computer Graphics]—Three-Dimensional Graphics and Realism.

1. Introduction

Image-based rendering (IBR) creates novel views of a 3D scene by interpolating between reference color and geometry samples. One advantage is quality: IBR algorithms attempt to transfer the high-quality of the reference images to the desired image. Another advantage is efficiency: the desired image is rendered using reference samples, which avoids the full complexity of the scene.

In what are now more than 10 years of IBR great progress has been made. However, the fundamental challenges of IBR do not yet have complete solutions. One challenge is modeling. Gathering the reference color and geometry samples remains difficult. For this, IBR researchers have investigated problems traditionally associated with computer vision such as depth extraction and registration. Another challenge of IBR is inadequate sampling. The reference samples do not always suffice for a good reconstruction of the desired image. Particularly challenging are view dependent effects. One example is view dependent appearance due to phenomena such as reflection, refraction, and blending at depth discontinuities. Another example is view dependent geometry due to occlusions as the camera and/ scene objects move.

This paper addresses the problem of disocclusion errors, which is an artifact due to lacking samples for surfaces that are visible from the desired view, but not from the reference view. Disocclusion errors occur even for small translations of the view. The problem is challenging because disocclusion errors are scattered in the scene. Disocclusion errors occur wherever motion parallax occurs, which, ironically, is one of the most important clues for a user exploring a 3D scene.

Current methods for alleviating the problem of

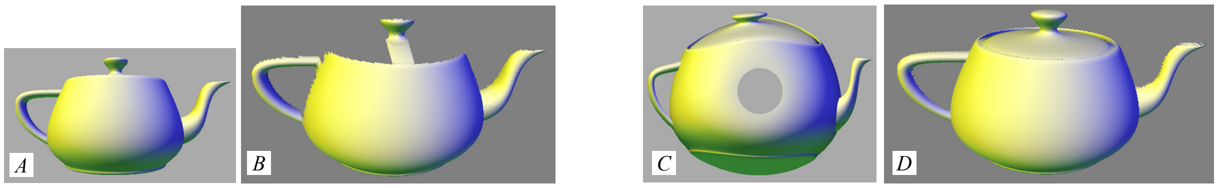

Figure 1: When a single depth image is used (*A*) disocclusion errors occur (*B*). The occlusion-camera reference image (*C*) stores samples for the lid, bottom and sides of the teapot, alleviating the problem of disocclusion errors (*D*).

disocclusion errors are based on using several reference views. The approach has diminishing returns: novel views come at the same price as the first view but contribute fewer and fewer new samples.

Our approach is based on a novel camera model, the *occlusion camera*, which is a non-pinhole generalization of the planar pinhole camera. A 3D radial distortion centered at an image plane point called a *pole* allows the occlusion camera to see around objects (*occluders*) along the ray defined by the pole. This way the occlusion camera also samples surfaces that are hidden from the reference view but are close to the silhouette of the occluder and are thus likely to become disoccluded in nearby views. Occlusion-camera images alleviate the disocclusion problem associated with regular depth images (Figure 1). Like depth images, they have a single layer so the rendering cost is bounded by their resolution and connectivity is defined implicitly by the regular grid of samples.

A planar pinhole camera acquires one sample along each ray. The 3D distortion trades *(u, v)* image resolution for resolution along the same ray. The occlusion camera image stores samples that are on the same ray in the corresponding planar pinhole camera image. In Figure 1 *C*, every lid sample L_i has a corresponding body sample B_i that lies on the same ray in *A*. The occlusion camera does not and should not attempt to store all the samples in the scene, since, like regular depth images, it is asked to provide view-dependent occlusion culling and level-of-detail. In Figure 1 *C* much of the back of the teapot body is missing. Those samples will not become visible until the view changes considerably from the reference view, when a new reference image should be used.

The occlusion camera does not guarantee that all disocclusion errors are avoided. For complex scenes, the set of samples captured by the occlusion camera is not necessarily a superset of the set of samples captured by the corresponding planar pinhole camera. For this we render such scenes with the union of the sets of samples captured by the two cameras. The planar pinhole and the occlusion camera images can be merged efficiently like any two depth images. The samples contributed by the occlusion camera image are samples that are almost visible in the reference view and are thus needed from nearby views. The combination produces good results for considerable translations away from the reference viewpoint.

The rays of the occlusion camera are non-concurrent, non-intersecting segments (Figure 2). There is exactly one ray through each 3D point in the field of view of the camera. These two facts imply that, like in the case of planar pinhole cameras, there is a one-to-one mapping between a scene *plane* and the image plane. This implies that an occlusion camera reference image can be constructed with the feed-forward graphics pipeline, by forward projecting vertices and then backward rasterizing triangles. Occlusion-camera reference images are constructed efficiently on the GPU.

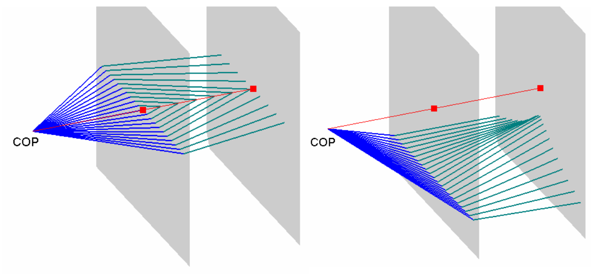

Figure 2: Visualization of an actual occlusion camera. The left/right image shows the rays for the central/bottom row of pixels. The 3D radial distortion centered at the pole is applied between two *z* planes. Each ray is defined by two segments: *COP* to near distortion plane (*blue*) and between distortion planes (*green*). The distortion brings the green segments closer to the pole ray (*red*). Rays that intersect the pole ray are clipped (*left*). The image plane can be placed anywhere between the *COP* and the near distortion plane.

The paper is organized as follows. We review prior work in Section 0. Section 0 describes the occlusion camera model. Section 0 gives the algorithm for constructing occlusion-camera reference images, and its GPU implementation. Section 0 describes using occlusion-camera reference images to render novel views of the scene. Results and discussion conclude the paper.

2. Prior work

McMillan and Bishop introduced 3D image warping, an IBR method that models and renders the scene with depth images [MB95]. The reference image is warped by projecting its depth-and-color samples onto the desired view. The desired image can be reconstructed by splatting, a technique that approximates the footprint of warped samples, or by connecting the warped samples in a triangle mesh and rendering the mesh. Using a single depth image causes disocclusion errors. Warping several depth images for each view greatly increases the cost of the method.

In post-rendering warping, Mark et al. investigate accelerating conventional rendering by warping two reference images [MMB97]. The reference views have to be very close to the desired view, which requires the system to update the reference images several times per second. Even so disocclusion errors occur.

Layered representations handle disocclusion errors by storing samples that are occluded from the reference view in additional layers at each pixel. Max proposes rendering trees using multi-layered z-buffers, which are built with a modified z-buffer algorithm [MO95]. Occluded samples are stored at deeper layers rather than being discarded. Keeping all samples along a ray generates high, uneven depth complexity, and many samples are never needed in nearby images. Limiting the number of samples along a ray requires a method for finding which samples are going to be visible in nearby views.

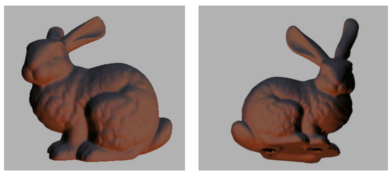

Figure 3: Normal (*left*) and reverse (*right*) planar pinhole camera image.

The Layered Depth Image (LDI) [SGH98, CBL99] solves the problem of choosing which samples to store by constructing the LDI from depth images from nearby views. The resulting LDI stores the union of the construction images, thus each of its samples is going to be needed at least when the desired view matches the construction view that contributed the sample. LDI construction requires rendering the scene multiple times. A disadvantage common to all layered representations is that sample connectivity cannot be easily computed and layered reference images are typically used by splatting.

None of the techniques reviewed so far guarantees the absence of disocclusion errors. If a surface was not sampled by any of the depth images used to construct the LDI, a disocclusion error occurs. The vacuum buffer [PL01] provides a conservative sample selection method for IBR. Given a desired view and a set of depth images, the technique finds the sub-volumes of the view frustum where disocclusion errors could occur. The method needs the desired view which implies running a complex z-buffer algorithm for every frame. Also verifying all view-frustum sub-volumes that could contain missed surfaces requires processing many depth images. Since most of the suspected sub-volumes turn out to be empty, conservatively checking for disocclusion errors is expensive.

The disocclusion error problem is the dual of the problem of occlusion culling, which has been much studied in conventional rendering. The reference depth image provides an over-aggressive occlusion culling solution for the desired image. Instead of attempting to undo some of the culling, we propose to use a camera model that prevents disocclusion errors in the first place.

Non-pinhole camera models have been studied in computer vision for 3D imaging applications. Examples include the pushbroom camera [GH97], which collects rays in parallel planes by sweeping a line, and the two-slit camera [Paj02], which captures all rays passing through two non-coplanar lines. The pushbroom and the two-slit cameras, as well as the pinhole and orthographic cameras, are subclasses of the general linear camera that collects linear combinations of three rays [YM04a]. We discuss using a general linear camera to address the disocclusion error problem in the next section.

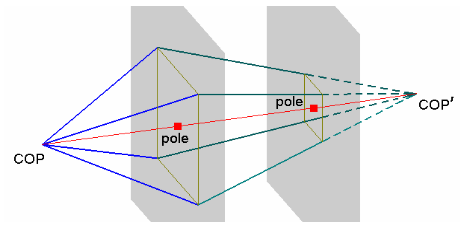

Figure 4: Reverse planar pinhole camera.

Computer graphics researchers have also explored non-pinhole cameras. The light field [LH96] and lumigraph [GGS*96] are a 2D array of planar pinhole cameras. The 4D ray database is queried during rendering, bypassing the need for geometry. The method does not have the problem of disocclusions and supports all view dependent effects. However the database grows to impractical sizes for large scenes and large viewing volumes. Light field approaches that separate sampling geometry from sampling view dependent appearance use conventional geometric models [WAA*00], and the problem of disocclusion errors become the problem of occlusion culling.

Multiple-center-of-projection cameras [RB98] sample the scene using a vertical slit that slides along a user chosen path. The interactive approach provides good sampling rate and coverage control. The resulting depth-and-color images are view independent, like a model, and fail to provide occlusion culling and level-of-detail adaptation. Because of the arbitrary path, 3D points cannot be projected directly onto the multiple-center-of-projection image and constructing such an image has to be done by ray tracing. Camera models developed for multiperspective rendering [WFH*97, YM04b] produce images that simulate 3D animation when viewed with a constrained camera motion but are not suitable for arbitrary views.

3. The occlusion camera model

To avoid disocclusion errors we set out to develop a camera model that has the following properties:

(1) in addition to visible samples, the camera should also collect samples close to the silhouette line of the occluder;

(2) each 3D point should project to a unique image plane location, with a closed form projection equation;

(3) there should be a one-to-one mapping between the plane of a scene triangle and the image plane.

Properties (2) and (3) ensure images can be rendered with the new camera using the feed forward graphics pipeline.

Reverse planar pinhole camera model

The first option we investigated was a simple camera model we called the *reverse planar pinhole camera* (Figure 4). The perspective foreshortening is reversed between the near and far planes. The camera is equivalent to a planar pinhole camera with an opposite view, which collects the *farthest* sample along each ray.

Figure 3 shows that the reverse planar pinhole camera has property (1) above. The image on the right half of the figure has samples that are beyond the silhouette of the

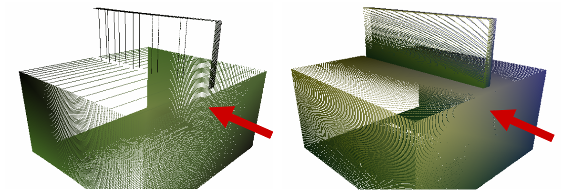
Figure 5: **R**everse planar pinhole camera (*left*) and 3D radially distorted camera image (*right*).

bunny as seen by a regular pinhole camera: the right side of the face, the top of the head, and the base are only visible in the right image. Such a reference image could be used to render novel nearby views of the bunny, and the originally hidden samples prevent disocclusion errors. Reverse planar pinhole camera images can be easily constructed even with the fixed graphics pipeline by using an opposite-view camera and flipping the z-test logic, thus the reverse planar pinhole camera also has properties (2 and 3).

The reverse planar pinhole camera has the advantage of great simplicity, but has an important disadvantage: the ability of sampling hidden surfaces decreases considerably for surfaces that are close to the view direction (red pole ray in Figure 4). In Figure 5 the scene consists of two boxes, one vertical and thin, and one horizontal and thick. The reference view direction is shown with the red arrow. The reverse planar pinhole camera reference image poorly samples the left and right faces of the thin box and the top face of the thick box. Moreover, the sampling rate decreases towards the front (see line frequency on the poorly sampled faces in Figure 5). The reverse planar pinhole camera is a special case of the general linear camera which captures rays that are a linear combination of 3 rays. Any linear camera has the same problem since, given a z plane, the amount of reverse foreshortening decreases to 0 towards the center of the image. We add a fourth desiderata for the occlusion camera:

(4) The distortion magnitude of a 3D point should be independent on the point's (x, y) coordinates).

Occlusion camera model

We have developed an occlusion camera model that has the desired four properties. The problem of the reverse planar pinhole camera is avoided (Figure 5, *right*). We started from a planar pinhole camera. In order to see around all sides of an occluder we have added a radial distortion centered at a pole chosen as the projection of the centroid of the occluder. The distortion pulls out samples according to their depth: farther samples are displaced by larger amounts. This way, two samples that are on the same ray in the undistorted image move to different distorted image locations. Hidden samples that are close to the silhouette in the undistorted image become visible.

In Figure 6, the 3D radial distortion moves the projection of scene point P from P_u to P_d. The distortion amount is given by a linear expression in $1/z$, where z is the camera-space z-coordinate of P. Larger z (smaller $1/z$) values produce larger distortion amounts. The distortion occurs in the image plane, on a direction away from the pole. A second scene point R that occludes P in the undistorted planar pinhole camera image is distorted less since it has a smaller z. Due to the different distortions, projections P_d and R_d do not coincide, and the sample P is part of the occlusion camera reference image.

The distortion of the occlusion camera is only apparently similar to the radial distortion characteristic to real-world lenses. The fundamental difference is that real-world lenses do not distort according to the depth of the sample; how far the ray has traveled to reach the lens is irrelevant.

The occlusion camera model is defined by a planar pinhole camera *PPHC* that gives the reference view and a six-tuple $(u_0, v_0, z_n, z_f, d_n, d_f)$ that specifies the 3D distortion. (u_0, v_0) give the image plane coordinates of the pole, (z_n, z_f) give the near and far z planes between which the distortion is applied, and (d_n, d_f) give the distortion magnitudes for points on the planes (z_n, z_f). The distortion magnitude varies linearly in $1/z$ between z_n and z_f.

The pole (u_0, v_0) is typically chosen as the *PPHC* projection of the centroid of the occluder, z_n is chosen epsilon closer than the closest point of the occluder, z_f is chosen to encompass the bounding box of the scene, and d_n is set to 0. d_f controls how much the silhouette of the occluder is receded. Using Figure 1 again, a larger d_f, would show more of the back of the teapot in C, and very large values would have *all* samples on the back of the teapot clear the front face. In Figure 8, d_f is sufficiently large to get all red checker background samples away from the cube. To provide room for the displaced samples, the original reference image resolution is extended by $2d_f$ in each direction to preserve the originally intended field of view. The occlusion camera images shown in this paper and in the video have a resolution of 720+200*2 (=1120) by 480+200*2 (=880).

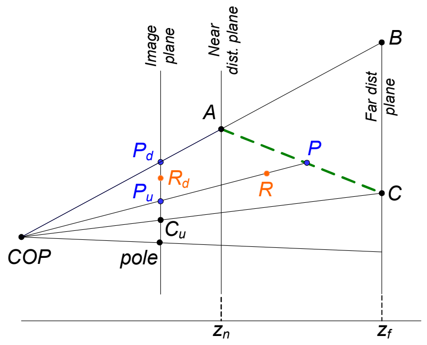

Figure 6 The 3D radial distortion is applied between near and far distortion planes. The projection of P is displaced from P_u to P_d. A closer sample R, seen along the same ray in the undistorted image, is displaced less, from P_u to R_d. A, P, and C all project at P_d in the distorted image. The planar pinhole camera ray (COP, P_d) has segment AB replaced with AC.

Projection equation

Given a 3D point $P(x, y, z)$, between z_n and z_f its occlusion camera $(PPHC, (u_0, v_0, z_n, z_f, d_n, d_f))$ image plane coordinates (u_d, v_d) are given by Equation 1.

$$(u_u, v_u, z) = PPHC(P)$$

$$d(z) = d_n + \frac{1/z_n - 1/z}{1/z_n - 1/z_f}(d_f - d_n)$$

$$(u_d, v_d) = (u_u, v_u) + \frac{(u_u - u_0, v_u - v_0)}{\|(u_u - u_0, v_u - v_0)\|}d(z)$$

Equation 1: Occlusion camera projection.

The undistorted coordinates (u_u, v_u) decide the direction but do not affect the distortion magnitude, which is completely defined by z. Points closer than z_n project at their undistorted *PPHC* projection. Because of our choice of z_f there are no points beyond z_f.

All 3D points in the view frustum of *PPHC* have exactly one projection onto the image plane, except for the points on the pole ray. For such points the distortion direction is undefined. In the limit, each of these points projects on a circle centered at the pole with a radius given by the distortion magnitude corresponding to the point's z (Figure 1 *A*, Figure 9).

Occlusion camera rays (unprojection)

Using the projection equation one can define the rays of the occlusion camera as the loci of scene points that project to the same occlusion camera image plane location. The points that project at an occlusion camera image plane location (u_d, v_d) are found by varying z from z_n to z_f. For a given z the distortion magnitude $d(z)$ is computed according to Equation 1 and then the 3D point P is computed according to the following equations.

$$(u_u, v_u) = (u_d, v_d) - \frac{(u_d - u_0, v_d - v_0)}{\|(u_d - u_0, v_d - v_0)\|}d(z)$$

$$P = Plane(z) \cap PPHC.Ray(u_u, v_u)$$

Equation 2: Occlusion camera ray.

The distortion direction is now computed using the known distorted coordinates. The distortion is applied in the opposite direction to find the undistorted coordinates (u_u, v_u). The point P is obtained as the point on *PPHC* ray (u_u, v_u) that is at distance z. For z's smaller than z_n, the point is found directly on the *PPHC* ray w/o distortion.

Since $1/z$ is linear in screen space, and since the distortion also varies linearly in $1/z$, the rays of the occlusion camera are straight between z_n and z_f. Using Figure 6 again and assuming $d_n = 0$, the occlusion camera ray at P_d consists of segments (COP, A) and (A, C). The distortion replaces (A, B) with (A, C). d_f equals the length of segment (C_u, P_d).

In Equation 2 the undistortion is not allowed to cross the pole. The undistortion amount cannot be larger than the distance from (u_d, v_d) to (u_0, v_0). This effectively clips the ray segments at the pole ray, as seen in Figure 2 left. Not all (u_d, v_d, z) triples correspond to a scene point.

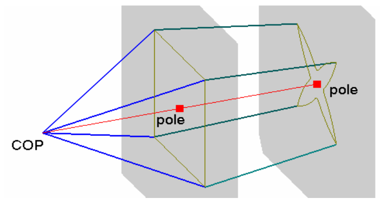

Figure 7: Visualization of an actual occlusion camera.

Figure 7 shows how the rays of the planar pinhole camera are broken at the near distortion plane. The yellow rectangle on the near distortion plane is gradually morphed into the star shape on the far distortion plane. A larger d_f would collapse the rectangle to a point.

The occlusion camera sees around an occluder along all image plane directions, thus has property (1) (see Figure 8). The occlusion camera is not a pinhole camera, since the lines of the ray segments are not concurrent. However, the occlusion camera does not suffer of projection ambiguity (Equation 1) and thus possesses property (2). This is an important difference when compared to non-pinhole cameras models used to describe the non-zero aperture of real-world cameras and its consequences on focus and depth of field. Property (4) follows from Equation 1. The occlusion camera has property (3) as explained next.

4. Occlusion camera image construction

A straight forward way of constructing an occlusion camera image is to render the scene triangles conventionally using the planar pinhole camera *PPHC* and then to distort every sample created before z-buffering. This forward mapping approach has to solve the problem of maintaining surface continuity. This problem has been studied extensively in IBR ([MB95, PZB*00, PEL*00, RL00]). A possible solution is splatting, a technique that replaces the point samples with surface elements that overlap, which prevents gaps. Another solution is to connect the samples using the connectivity implicitly defined in the undistorted image and to rasterize the distorted mesh conventionally.

4.1. Occlusion camera triangle rasterization

We avoid the difficulties of forward mapping by rasterizing

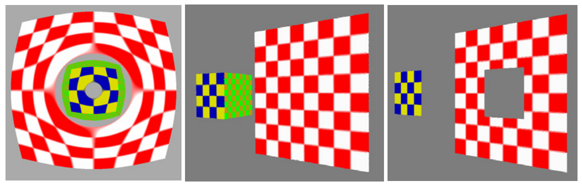

Figure 8: Occlusion camera captures 5 faces of the cube and complete background (*left*). Disocclusion errors are avoided (*middle*). *Right*: comparison to using depth image.

the triangles directly in the distorted domain with the following steps:

```
- Estimate triangle bounding box in distorted domain
- For each pixel $(u_d, v_d)$ in the bounding box
    - Compute undistorted $(u_u, v_u)$ and $z$
    - Zbuffer $(u_d, v_d, z)$
    - Evaluate edge equations using $(u_u, v_u)$
    - Compute color $c$
    - Set $(u_d, v_d, c)$
```

Since the edges of the distorted triangle are curved, we estimate the bounding box by distorting a few points on the perimeter. The undistorted coordinates (u_u, v_u) and z are computed simultaneously according to Equation 3.

$$\frac{1}{z}(u_u, v_u) = Au_u + Bv_u + C$$
$$d(u_u, v_u) = Du_u + Ev_u + F$$
$$(u_u, v_u) = (u_d, v_d) - \frac{(u_d - u_0, v_d - v_0)}{\|(u_d - u_0, v_d - v_0)\|} d(u_u, v_u)$$
$$(u_u, v_u) = (u_d, v_d) - \frac{(u_d - u_0, v_d - v_0)}{\|(u_d - u_0, v_d - v_0)\|} (Du_u + Ev_u + F)$$

Equation 3: Occlusion camera triangle rasterization.

For a given triangle t, $1/z$ is linear in undistorted screen space. The coefficients A, B, and C are computed by solving the linear system of 3 equations $1/z_{ti} = Au_{ti} + Bv_{ti} + C$, where i = 0, 1, and 2, and (u_{ti}, v_{ti}, z_{ti}) is the *PPHC* projection of vertex i of t. The $1/z$ expression is plugged into the distortion magnitude equation (see Equation 1) to compute D, E, and F. These coefficients define the linear variation of the distortion magnitude in undistorted screen space. The undistorted coordinates (u_u, v_u) are the distorted coordinates (u_d, v_d) minus the distortion vector. The direction of the distortion vector is known since (u_d, v_d) and (u_0, v_0) are known. The last of the four equations is a linear system of two equations with two unknowns u_u and v_u, which once known are used to recover z from the first equation. Once the triangle sample is known, rasterization proceeds as usual. Edge equations are evaluated in the undistorted domain where edges are straight.

Equation 3 provides a unique triangle-plane point for each (u_d, v_d) pair, except for the case in which the plane is aligned with one of the occlusion camera rays (silhouette triangle). Silhouette triangles collapse to a segment, just like silhouette triangles do for regular pinhole cameras. (A silhouette triangle in the undistorted image is a triangle whose plane passes through the center of projection of the planar pinhole camera.) The unique solution in the general case allows establishing a backward mapping from the distorted image plane to the triangle plane.

The most expensive part of the rasterization is the 2D vector normalization required to compute the distortion direction at each pixel. We have implemented the rasterization algorithm described on the GPU.

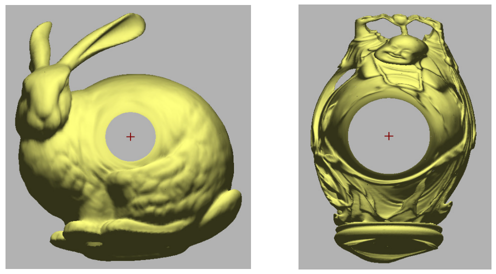

Figure 9: Occlusion camera images generated on the GPU at 11fps (bunny, 70Ktris) and 3fps/0.6fps (Happy Buddha, 293Ktris/1Mtris).

4.2. Hardware implementation

In order to implement a novel rasterization algorithm one needs to have programmability at triangle level. Existing GPUs feature programmability only at vertex and fragment level. Vertex programs cannot access vertex data for the other two vertices shared by a triangle. For this the data of all vertices is passed to the GPU for every vertex. Another difficulty comes from the fact that the edges of the projected triangle are curved. Our solution is to render each triangle by issuing a quad drawing command which rasterizes the bounding box of the curved-edge triangle.

The vertex program computes:
- the 2D vertices of the quad as the distorted-domain bounding box of a set of triangle perimeter points,
- the coefficients D, E, F (see Equation 3),
- the edge equations,
- and all the other linear expressions needed for regular rasterization (i.e. screen space or model space interpolation of texture coordinates, color, and normals).

Using the parameters passed by the vertex program, the fragment program computes the undistorted coordinates (u_u, v_u), performs the sidedness tests, shades and textures, and then returns color and z. Figure 9 shows examples of occlusion camera images rendered with our GPU programs.

5. Rendering with occlusion-camera images

Triangle mesh rendering

The occlusion camera image stores a single layer of depth-and-color samples. Once the occlusion camera image is constructed we read the color and z-buffer back and build a triangle mesh by unprojecting the samples to create 3D vertices (Equation 2). The mesh is rendered in hardware with per-vertex color and can be re-lighted.

We have also implemented a technique for rendering using the occlusion camera image that avoids reading back to main memory. The reference image is transferred to GPU memory and is processed using a vertex program that undistorts the depth and color samples forming triangles which are rasterized to render the desired view.

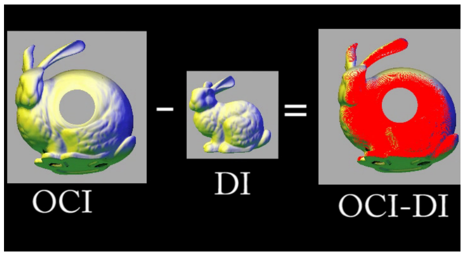

Figure 10: Set difference on occlusion camera images.

Merging

An occlusion camera images is less prone to disocclusion errors than a regular depth image since it also stores samples that are close to the silhouette as seen from the reference view. Such samples are visible in nearby views, filling in the gaps that would otherwise form, and extending the range of views for which the reference image is usable.

Occlusion camera images do not guarantee that all needed samples are present. It can happen that even some samples visible from the reference view are missing from the occlusion camera image. In the Buddha occlusion camera image shown in Figure 9 the feet of the statue are not visible, although they are part of a regular depth image. We avoid this problem by rendering a regular depth image *DI* together with the samples of the corresponding occlusion camera image *OCI* that are not part of *DI*.

We have implemented an occlusion-camera-image set-difference operator which also accepts depth images as operands since they are a special case of occlusion camera image. A sample in the first image is unprojected and the resulting 3D point is projected onto the image plane of the second image. If the *z*'s are similar the sample is marked as shared (Figure 10). Rendering *DI + (OCI – DI)* guarantees that no *DI* samples are lost while avoiding disocclusion errors (Figure 11, and Figure 12, and Figure 13).

6. Results, discussion, and future work

We have presented a technique for alleviating disocclusion errors. Occlusion-camera images share the advantages of depth images: they have a single layer, which bounds the complexity (even when used in conjunction with a depth image) and provides implicit connectivity, and are constructed and used in hardware. The distortion parameter allows handling occlusions on a continuous scale and provides an effective heuristic for deciding whether a surface point is likely to be visible in nearby views.

Our current GPU implementation constructs occlusion camera images from geometry at the average rate of 700Ktris per second (see Figure 9). The timing data reported in this paper was measured on a Pentium 4 2GHz 1GB system with an NVIDIA GeForce 6800 256MB AGP graphics card. The models used are courtesy of the Stanford 3D Scanning Repository [Sta05]. The occlusion camera image construction time is dominated by the time spent in the fragment program. We will investigate speeding up our implementation by deriving a tighter approximation of the bounding box of the distorted triangle.

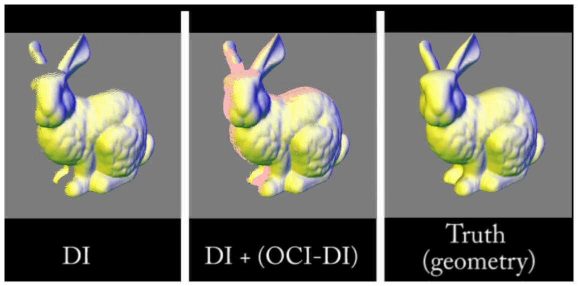

Figure 11: Samples contributed by the OCI are shown in pink in the middle image.

The 256MB of GPU memory currently allow processing 1Mtris at a time. The lack of programmability at triangle level prevents us from reusing vertices in the case of shared vertex meshes, and forces us to draw triangles individually. Moreover the vertex data needs to be replicated four times for each of the vertices of the quad needed to rasterize the curved-edge triangle. Removing this programmability limitation would provide a considerable reduction in memory needs. The Thai statue shown in Figure 13 has 10Mtris and its occlusion camera image was constructed on the CPU in 20 minutes. The original model is rendered by the fixed graphics pipeline at 2fps. The *OCI + (OCI-DI)* samples are rendered at refresh rate (40fps for our system).

The occlusion camera gathers samples all around the silhouette of the occluder without requiring rendering the scene multiple times, and without decreasing the sampling rate along the main view direction. Multiple depth image approaches, including LDIs, require rendering the scene several times. Typically an LDI is built from at least four reference images captured from viewpoints that box the reference viewpoint. An additional central image is needed to satisfy the sampling rate requirement.

In this paper we have considered the case of occluders. In order to address disocclusion errors occurring at the edges of a portal (e.g. a window in an architectural model), the maximum distortion should occur on the near plane and no distortion should occur on the far plane. The frame of the portal is enlarged in the occlusion camera reference image and more of the interior of the adjacent cell is sampled.

Path prediction can also be used to tune the occlusion camera model for improved disocclusion error avoidance. A predicted future view defines epipolar lines in the current view. All occlusions and disocclusions occur along epipolar lines. Placing the pole at the epipole generates occlusion camera images that alleviate most disocclusion errors.

Our current approach requires handling objects individually. In the single object case good results are obtained by placing the distortion pole at the projection of the center of the object. Handling complex inside-looking-out scenes by subdivision into objects can lead to a high depth complexity. We will investigate more complex

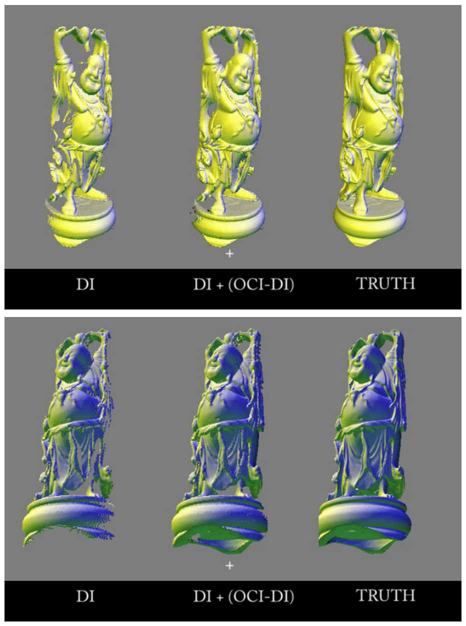

Figure 12: Happy Buddha model. The same occlusion camera reference image alleviates disocclusion errors on all sides of the statue.

occlusion camera models generated by 3D distortions controlled by more than one pole and also by line segments and curves in the image plane.

The ray pattern of the occlusion camera and the regular images produced indicate that the approach could probably be used to support several geometry processing tools such as view *independent* simplification, surface parameterization and 3D morphing.

7. Acknowledgments

This work was supported by the Purdue University Visualization Center, and by NSF grants SCI-0417458 and CCR-0306214. We thank the anonymous reviewers.

References

[CBL99] C-F Chang, G. Bishop, and A. Lastra. LDI Tree: A Hierarchical Representation for Image-Based Rendering, *Proc. SIGGRAPH'99*, (1999).

[GGS*96] S. Gortler, R. Grzeszczuk, R. Szeliski, M. Cohen. The Lumigraph. *Proc. of SIGGRAPH* 96, 43-54.

[GH97] R. Gupta, R. I. Hartley. Linear Pushbroom Cameras. *IEEE Trans. Pattern Analysis and Machine Intell.* vol. 19, no. 9 (1997) 963–975.

[LH96] M. Levoy, and P. Hanrahan. Light Field Rendering. *Proc. of SIGGRAPH* 96, 31-42 (1996).

[MB95] L. McMillan and G. Bishop. Plenoptic modeling: An image-based rendering system. *In Proc. SIGGRAPH* '95, pages 39-46, 1995.

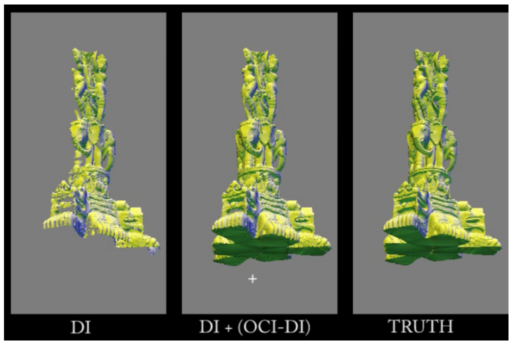

Figure 13: Thai statue model.

[MMB97] W. Mark, L. McMillan, G. Bishop. Post-Rendering 3D Warping. *Proceedings of 1997 Symposium on Interactive 3D Graphics (Providence, Rhode Island, April 27-30, 1997).*

[MO95] N. Max and K. Ohsaki. Rendering trees from precomputed z-buffer views. In *Rendering Techniques '95: Proceedings of the Eurographics Rendering Workshop 1995*, 45–54, Dublin, June 1995.

[Paj02] T. Pajdla. Geometry of Two-Slit Camera. *Research Report CTU–CMP–2002–02*, 2002.

[PEL*00] V. Popescu, J. Eyles, A. Lastra, et al. The WarpEngine: An architecture for the post-polygonal age. *Proc. ACM SIGGRAPH*, 2000.

[PL01] V. Popescu, A. Lastra. The Vacuum Buffer. *In Proceedings of ACM Symposium on Interactive 3D Graphics, Chapel Hill*, 2001.

[PZB*00] H. Pfister, M. Zwicker, J. V. Baar, M. Gross. Surfels: Surface Elements as Rendering Primitives. *Proc. of SIGGRAPH* 2000, 335-342 (2000).

[RB98] P. Rademacher, G. Bishop. Multiple-center-of-Projection Images. *Proc. ACM SIGGRAPH* '98 (1998)199–206.

[RL00] S. Rusinkiewicz, M. Levoy. QSplat: A Multiresolution Point Rendering System for Large Meshes. *Proc. SIGGRAPH* 2000.

[SGH98] J. Shade, S. Gortler, L. He, et al. Layered Depth Images, *In Proceedings of SIGGRAPH 98*, 231-242.

[Sta05] The Stanford 3D Scanning Repository, *http://graphics.stanford.edu/data/3Dscanrep/*

[WAA*00] D. N. Wood, D. I. Azuma, K. Aldinger, et al. Surface light fields for 3D photography. *Proceedings, SIGGRAPH '00, ACM Press*, pp. 287-296.

[WFH*97] D. N. Wood, A. Finkelstein, J. F. Hughes, et al. Multiperspective Panoramas for Cel Animation. *Proc. ACM SIGGRAPH* '97 (1997) 243-250.

[YM04a] J. Yu, and L. McMillan. General Linear Cameras *In 8th European Conference on Computer Vision (ECCV)*, 2004, Volume 2, 14-27.

[YM04b] J. Yu, and L. McMillan. A Framework for Multiperspective Rendering. *In Proceedings of Eurographics Symposium on Rendering (EGSR),* 2004.