Non-Pinhole Approximations for Interactive Rendering

Paul Rosen, Voicu Popescu, Kyle Hayward, and Chris Wyman, *Member*, *IEEE*

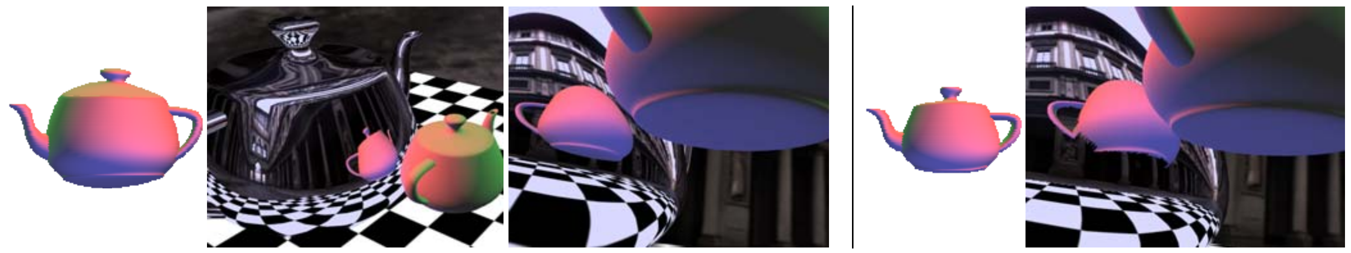
Figure 1. Non-pinhole depth image of teapot and reflections rendered with it (left), and conventional depth image and reflection rendered with it (right). The non-pinhole depth image captures the lid and bottom of the teapot and produces quality reflections.

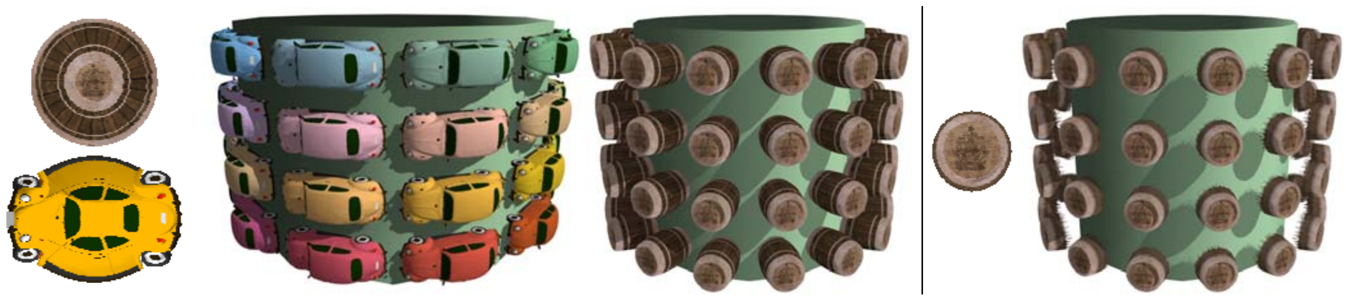
Figure 2. Non-Pinhole relief texture maps and frames rendered with them (left) and conventional relief texture mapping (right).

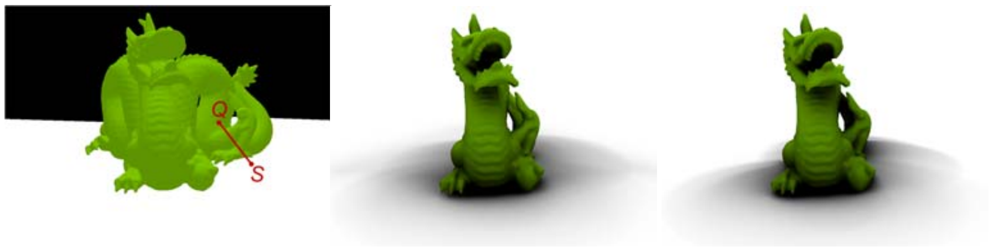

Figure 3. Non-pinhole z-buffer illustrated with color (left), ambient occlusion effect rendered using it (middle), and ambient occlusion effect rendered using output image z-buffer (right). The non-pinhole z-buffer captures hidden parts of the dragon for a more complete and stable ground shadow.

Abstract

Depth images have been used to approximate scene geometry in a variety of interactive 3-D graphics applications. In previous work, images were constructed using orthographic or perspective projection which limits approximation quality to what is visible along a single view direction or from a single viewpoint. This paper shows that images constructed with non-pinhole cameras improve approximation quality at little additional cost provided that the non-pinhole camera offers fast projection. For such a camera, the fundamental operation of ray depth-image intersection proceeds efficiently by searching along the one-dimensional projection of the ray onto the image. In the context of two-camera configurations, our work extends epipolar geometry constraints to non-pinholes. We demonstrate the advantages of non-pinhole depth images in the context of reflections, refractions, relief texture mapping, and ambient occlusion.

Keywords: non-pinhole camera, single-pole occlusion camera, graph camera, depth image, impostor, epipolar constraints, reflection, refraction, relief texture mapping, ambient occlusion, interactive 3-D graphics.

Index Terms: I.3.3. [Computer Graphics]—Three-Dimensional Graphics and Realism.

1 Introduction

In the quest for higher-quality and higher-performance rendering researchers have resorted to approximating scene geometry with more efficient representations. The three main desirable properties of such an alternative representation are high-fidelity approximation, efficient construction, and efficient rendering. The representation should capture the geometry it replaces sufficiently well such that the resulting images be virtually indistinguishable

- P. Rosen is with the Scientific Computing and Imaging Institute, The University of Utah, Salt Lake City, UT 84122. Email: prosen@sci.utah.edu.
- V. Popescu is with the Computer Science Department, Purdue University, West-Lafayette, IN 47907. Email: popescu@purdue.edu.
- K. Hayward is with Human Head Studios, Inc., Madison, Wisconsin 53704. Email: khayward@purdue.edu.
- C. Wyman is with the Computer Science Department, University of Iowa, Iowa City, IA 52242. Email: cwyman@cs.iowa.edu..

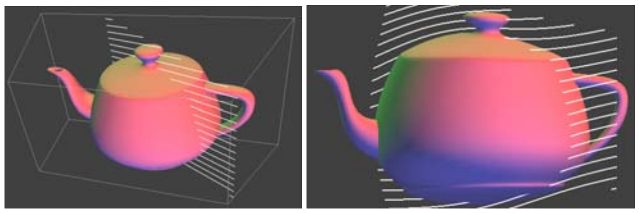
Figure 4. Visualization of rays in 3-D (left) and of their curved projection on the non-pinhole depth image (right) used in Figure 1.

from the images obtained when rendering with the original geometry. To support dynamic scenes, the representation has to be created on the fly, which requires fast construction.

Lastly, the alternative representation must deliver the desired performance boost to the application. We distinguish between applications where the representation can be rendered directly with the conventional feed-forward approach of projection followed by rasterization, such as for example when a distant tree is rendered using a billboard, and applications where the representation has to be rendered by intersection with one ray at a time, such as is needed for example in the case of reflections, refractions, relief texture mapping, and ambient occlusion. In this paper we focus on the second type of applications. For such applications a fast computation of the intersection between a ray and the alternative geometry representation is a central concern.

Images enhanced with per pixel depth have two of these desired properties. A depth image is constructed efficiently by rendering the geometry it replaces with the help of graphics hardware. Fast ray / depth image intersection is enabled by the fact that projection of a ray onto the depth image is a segment, which reduces the dimensionality of the intersection search space from two to one. Modern graphics hardware allows stepping along the ray projection, per pixel, at interactive rates. However, depth images are acquired from a single viewpoint—with a planar pinhole camera, or along a single view direction—with an orthographic camera, which limits their geometry modeling power. Such a depth image misses surfaces that become visible when the geometry approximation is rendered by the application, which lowers the quality of the result.

We propose constructing depth images using non-pinhole cameras. Such non-pinhole depth images offer a high-fidelity approximation of scene geometry while construction and rendering costs remain low. Once all rays need not pass through one point, the rays of a non-pinhole camera can be designed to sample all surfaces exposed by the application. To ensure construction and rendering efficiency, the non-pinhole camera is designed to provide a fast projection operation. This enables constructing the non-pinhole depth image in feed-forward fashion by projection followed by rasterization. The closed-form, unambiguous projection of the non-pinhole camera is leveraged a second time, during rendering, to compute the projection of the ray onto the non-pinhole image. As in the case of planar pinhole cameras, the ray / non-pinhole camera depth image intersection is found by walking on the one-dimensional projection of the ray.

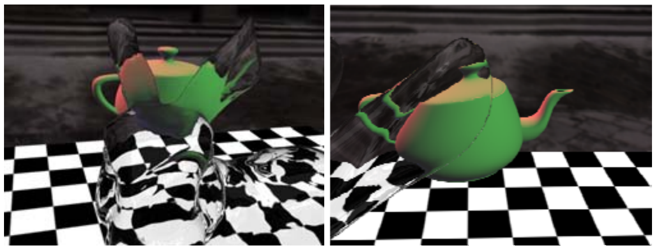
Figure 5. Refractions rendered with a non-pinhole depth image of the teapot similar to the one shown in Figure 1.

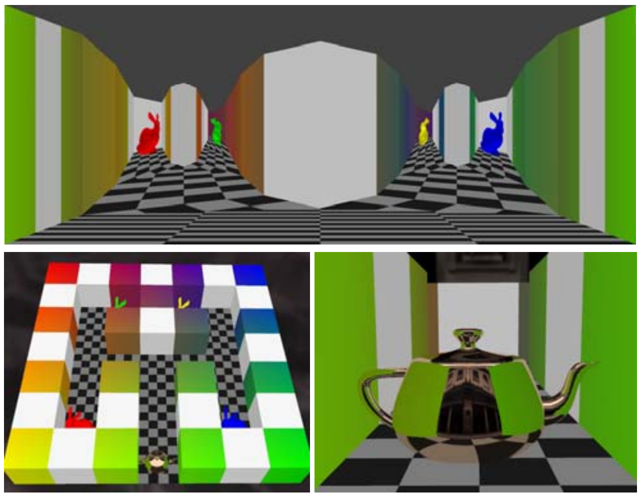
Figure 6. Graph camera depth image (top) capturing an entire 3-D maze (bottom left) and reflection rendered using it (bottom right).

Unlike for planar pinhole cameras, ray projection is not a straight line, though this does not raise intersection costs significantly.

The advantages of non-pinhole depth images are demonstrated in the contexts of reflection, refraction, relief texture mapping, and ambient occlusion (also see accompanying video). In Figure 1 the non-pinhole depth image captures all teapot surfaces exposed in the reflections, whereas a conventional depth image produces incomplete reflections. The intersection between a reflected ray and the non-pinhole depth image is found by searching along the curved projection of the ray (Figure 4). Refractions are supported similarly by intersecting the non-pinhole depth image with refracted rays (Figure 5). Non-pinhole cameras greatly enhance the modeling power of relief texture mapping (Figure 2). A single-layer non-pinhole relief texture captures the entire top and sides of a barrel or an entire car to produce quality frames with rich detail. By contrast a single-layer conventional relief texture misses a considerable part of the barrel. A multilayered relief texture is also impractical in this case since a prohibitively large number of layers are needed to sample the side of the barrel well.

A promising technique for rendering with ambient occlusion at interactive rates is to use the output image z-buffer to approximate the exposure of output image samples to the environment [BS08]. However, the amount of occlusion for visible samples also depends on samples that are not visible in the output image. This makes the ambient occlusion effect unstable, with dark regions appearing and disappearing as geometry subsets appear and disappear in the output image. We propose using a non-pinhole z-buffer which captures most samples needed to estimate the occlusion of output image samples, producing a more complete and more stable ambient occlusion effect (Figure 3).

Abandoning the pinhole constraint presents the opportunity to design the camera model such as to obtain the depth image best suited for the application and dataset at hand. In this work we construct depth images by adapting two recently introduced non-pinhole camera models: the *single-pole occlusion camera* (SPOC) [MPS05], and the *graph camera* [PRA09]. The SPOC reaches around an object's silhouette to gather samples not visible from the reference viewpoint but close to the silhouette. Such "barely" occluded samples are needed when the depth image is sampled from nearby viewpoints. The SPOC is well suited for approximating single objects, and it was used in the examples in Figures 1, 2, 4, and 5. The graph camera is a non-pinhole camera constructed starting from a planar pinhole camera through a series of frustum bending, splitting, and merging operations. The result is literally a graph of planar pinhole cameras. The graph camera circumvents occluders to sample an entire 3-D scene in a single-layer image. In Figure 6 the graph camera depth image models the entire reflected scene.

In summary, the contributions of our paper are as follows:

- We extend epipolar like constraints to non-pinholes. This fundamental contribution opens up the class of non-pinhole geometry approximations to interactive rendering applications.
- We provide a generic algorithm for intersecting a non-pinhole depth image with a ray, and we specialize the algorithm for two non-pinhole cameras, the SPOC and the graph camera.
- We demonstrate the advantages of non-pinhole depth images in the context of reflections, refractions, relief texture mapping, and ambient occlusion.

2 Prior work

We review prior research on image-based geometry approximation, on acceleration of rendering effects using depth images, and on non-pinhole cameras models.

2.1 Image-based geometry approximation

Maciel and Shirley [MS95] coined the term impostor which is now widely used to denote an image-based simplified representation of geometry for the purpose of rendering efficiency. The simplest impostor is a billboard, a quad texture mapped with the image of the original geometry, with transparent background pixels. Billboards are constructed efficiently, intersecting a billboard with a ray is trivial, and billboards provide good approximations of geometry seen orthogonally from a distance. When the billboard is close to the viewer the drastic approximation of geometry is unacceptable. Billboard clouds [DDS*03] use several quads to improve modeling quality. The quads and the assignment to original geometry are optimized for modeling fidelity. The number of quads is small and a billboard cloud can be intersected with a ray one quad at a time. However, the optimization makes construction of the billboard cloud a lengthy process that precludes dynamic scenes. Moreover, the approximation quality is still insufficient for close-up viewing.

Depth images [MB95] greatly improve the modeling power of billboards. Constructing a depth image is just as inexpensive as constructing a billboard, but the cost of intersection with a ray is not constant anymore, but rather linear in the depth image width. Searching for the intersection in the entire image is avoided by leveraging epipolar-like constraints: the intersection is known to belong to the image plane projection of the ray. Since the depth image is constructed with an orthographic or a perspective projection, the depth image only captures samples visible along the reference direction or from the reference viewpoint. When surfaces not captured by the depth image are exposed by the application, objectionable disocclusion artifacts occur.

The simplest method for alleviating disocclusion errors is the use of additional depth images [MMB97], which is expensive and only palliative. A breakthrough came with the introduction of layered representations such as the multi-layered z-buffer [MO95] and the layered depth image (LDI) [Sha98], which allow for more than one sample along a ray and control disocclusion errors effectively. However, expensive construction restricts layered representations to static scenes. Moreover, the lack of a connected representation makes ray intersection difficult, precluding rendering effects such as those addressed by this paper.

As graphics hardware has evolved, it became possible to intersect depth images with individual rays, and research focus shifted to rendering effects involving higher order rays.

2.2 Rendering effects accelerated using depth images

Reflection and refraction have been studied extensively in interactive rendering, yet no complete solution exists. We assign reflection and refraction rendering techniques to four groups: ray tracing [Whi80], image-based rendering [LH96, GGS*96, DYB98], projection [OR98], and reflected/refracted scene approximation. We only discuss group 4, since it is most relevant.

Environment mapping [BN76] approximates the reflected scene with cube map and it is currently the preferred approach for interactive applications due to its efficiency, robustness, and good results when scene geometry is far from the reflector/refractor. Environment mapping performs poorly close to the reflector/refractor. Improved results are obtained by approximating the scene with a sphere [Bjo04], but few environments are spherical so the fidelity is still quite limited. The scene approximation can be improved using depth images [SALP05, PMDS06]. Quality reflections are produced for simple objects or for select viewpoints, but the insufficient coverage is a limitation for non-trivial scenes or wide viewpoint translations.

Compared to reflection, refraction rays require additional work since most rays interact with the refractor at least twice—once entering and once leaving the object. Several techniques have been developed for computing the second refraction at interactive rates, including pre-computed distance fields [CW05], GPU ray tracing [RAH07], and image-space approximations [Wym05].

Another rendering effect that requires intersecting depth images with individual rays is relief texture mapping [POC05]. True geometric detail is added to a coarse model by texturing each triangle with a height map. A conventional relief texture samples surface detail orthographically, along the direction of the normal of the underlying coarse model, which limits the technique to height field surfaces. Even so, sampling degrades when the geometric detail becomes aligned with the normal of the underlying surface. The technique has been extended to non-height field surface detail by resorting to a relief texture with multiple layers, each sampled orthographically [PO06]. The extension works well when complex detail can be captured in a few layers, as it is the case for a chain link fence for example. A strength of the extended method is the ability to capture double-sided detail. However, capturing geometric detail perpendicular to the underlying surface remains challenging as a large number of layers is needed. Our work extends relief texture mapping in an orthogonal direction and multilayered non-pinhole relief textures could be developed to collect the advantages of both techniques.

Ambient occlusion techniques add realism to local illumination models by approximating the amount of light a surface point receives based on how much of the environment is hidden from the point by nearby geometry. The computational cost is high since a ray has to be cast from each point in all directions. First implementations precomputed ambient occlusion in model space off-line [ZIK98], which precludes dynamic scenes. Initial efforts of enrolling the GPU in accelerating ambient occlusion resorted to a large number (i.e. 128-1024) of spherical shadow maps of the scene [PG04]. Our non-pinhole z-buffer ambient occlusion method builds upon an image-space technique introduced by Bavoil and Sainz [BS08]. Their technique approximates the amount of ambient occlusion at an output pixel using the output image z-buffer. They noticed that in order to sample occlusion at a pixel for an entire half plane it is sufficient to traverse one z-buffer segment. The result is a fast ambient occlusion method that supports dynamic scenes. However, the technique computes ambient occlusion as if the geometry seen by the output image were the only geometry in the scene, which can cause missing and unstable ambient occlusion artifacts.

2.2 Non-pinhole cameras

The overwhelming majority of the images used in computer graphics, visualization, and computer vision are rendered or acquired with a pinhole camera: all rays pass through a common point, the pinhole. Whereas this leads to efficient computational and physical camera implementations which compute and acquire images that are similar to those captured by the human visual system, the pinhole requirement is quite restrictive. Non-pinholes are powerful cameras that can capture rays that originate at different points in space.

One context where non-pinholes have received attention is in computer vision where they were used to model complex lens and catadioptric systems, including the pushbroom camera [Gupta 1997], the two-slit camera [Paj02], and their generalization, the general linear camera [YM04b]. General linear cameras enable a powerful framework for designing multiperspective images, but rendering is done by ray tracing [YM04].

Another context is artistic rendering [AZM00], where non-pinholes were used to simulate deviations from conventional perspective adopted by artists for aesthetic reasons. In scene sampling, the context of highest relevance here, one prior non-pinhole is the multiple-center of projection camera [RB98] which samples the scene with a vertical slit along a user chosen path and thus avoids redundancy and offer sampling flexibility. However, construction requires rendering the scene for each position along the path. In cel animation multiperspective panoramas capture all 3-D scene samples seen along a camera path and allow simulating camera motion by sliding a frame over the panorama, but the view is confined to the path and the scene needs to be static [Woo97].

Occlusion cameras were designed to address disocclusion errors. Given a reference view and a 3-D scene, an occlusion camera builds a single-layer image that stores not only samples visible from the reference viewpoint, but also samples visible from nearby points. In addition to the single-pole occlusion camera (SPOC) discussed earlier, other occlusion cameras include the depth discontinuity occlusion camera (DDOC) [PA06] and the epipolar occlusion camera (EOC) [RP08]. The DDOC specifies the distortion through a map. The added flexibility comes at the cost of increased construction times. The EOC captures all samples visible as the viewpoint translates between two given points. The EOC effectively generalizes the view*point* of a planar pinhole camera to a view*segment*. However, the EOC only supports translation along a single direction.

3 NON-PINHOLE CAMERA DEPTH IMAGES

After removing the pinhole restriction, there is great flexibility in devising a camera model that best suits a given application and a dataset. Therefore we first discuss construction and ray intersection for non-pinhole depth images in general.

3.1 Construction

Given a non-pinhole camera with a fast projection operation that maps a 3-D point *(x, y, z)* to *(u, v, z)* where *(u, v)* are image coordinates and *z* is a measure of depth linear in image space, a non-pinhole depth image is constructed efficiently by projecting the vertices of the geometry it is called to approximate and by rasterizing the projected triangles conventionally. The unconventional projection occurs in a vertex program which implements the non-pinhole camera model. Since lines do not necessarily project to lines anymore and since rasterization parameters do not vary linearly (before the perspective divide) anymore, the triangles must be sufficiently small to provide an adequate approximation. Complex objects are frequently modeled with small triangles, so additional tessellation is usually not needed. Meshes with large triangles can be subdivided on-the-fly by taking advantage of primitive-level GPU programmability.

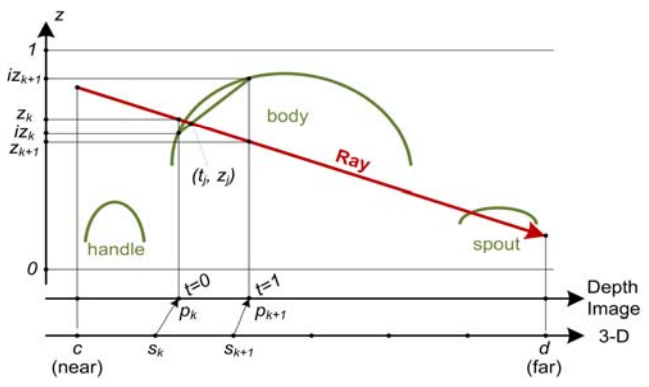

Figure 7. Ray / non-pinhole depth image intersection.

3.2 Ray intersection

Like a regular depth image, a non-pinhole depth image is defined by an image with color and depth per pixel, and a camera model which allows projection. The intersection of a ray *(a, b)* with a non-pinhole depth image *NPI* is computed as follows:

1. Clip the segment *(a, b)* with the bounding volume of *NPI* to obtain the segment *(c, d)*, see Figure 7.

2. Interpolate *(c, d)* in 3-D, from near to far to create *n* sub-segments. For each sub-segment (s_k, s_{k+1}):

    2.1. Project s_k and s_{k+1} to depth image at $p_k = (u_k, v_k, z_k)$ and $p_{k+1} = (u_{k+1}, v_{k+1}, z_{k+1})$.

    2.2. Lookup image depths iz_k, iz_{k+1} at (u_k, v_k), (u_{k+1}, v_{k+1}).

    2.3. Intersect in 2-D segments $[(0, z_k), (1, z_{k+1})]$ and $[(0, iz_k), (1, iz_{k+1})]$ to obtain intersection (t_j, z_j).

    2.4. If (t_j, z_j) is a valid intersection, return depth image color ic_j at lerp((u_k, v_k), (u_{k+1}, v_{k+1}), t_j), else continue with next sub-segment.

The ray is interpolated in 3-D since its projection is not a straight line, and one cannot simply rasterize the segment that connects the endpoint projections. Each intermediate point is projected with the non-pinhole camera to trace the curved projection correctly. Since the depth *z* stored by the depth image varies linearly in the image, the intersection can be computed efficiently in a 2-D space *(t, z)*, where *t* is the parameter locating the intersection along segment (p_k, p_{k+1}). These generic algorithms are specialized for the SPOC and the graph camera as follows.

4 SINGLE-POLE OCCLUSION CAMERA DEPTH IMAGES

The SPOC projection uses a conventional projection followed by a distortion that moves the projected sample away from a pole [MPS05]. The pole is the projection of the center of the object. The distortion magnitude increases with depth, so deeper samples move more, escaping the occluding front surface. For the SPOC depth image in Figures 1 and 2 the distortion pushes the object silhouettes back, revealing the lid and the bottom of the teapot, the entire side of the barrel, and the side and wheels of the car. Figure 8 shows that the SPOC depth image captures about half of the teapot, which is sufficient to intercept all reflected rays that would intersect the original teapot geometry.

SPOC construction and intersection closely follow the algorithms described in the previous section. The number of sub-segments *n* is chosen as the Euclidian distance between the projection of the endpoints of the clipped ray. This provides a good approximation of the actual number of pixels covered by the

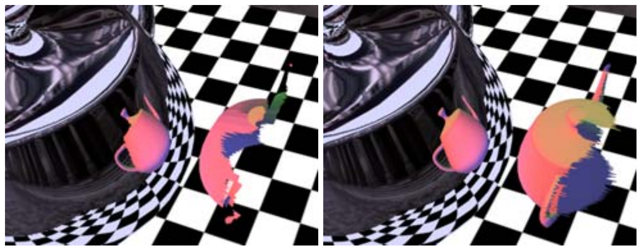
Figure 8. Visualization of samples stored by conventional (left) and SPOC (right) depth image.

curved projection of the ray. The curved projections of a set of coplanar rays are visualized in Figure 4.

5 Graph Camera Depth Images

The graph camera is constructed from a planar pinhole camera through a succession of bending, splitting, and merging operations [PRA09]. The result is a graph of planar pinhole camera frusta. The concept of camera ray is generalized to the set of points projecting at a given image location, which allows for rays that are not straight lines. The rays of the graph camera are piecewise linear. A ray changes direction as it crosses the shared face separating two frusta, but it remains continuous, which makes the graph camera image continuous. The graph camera constructed for the maze in Figure 6 is shown in Figure 9. The construction used a breadth first traversal starting from the entrance.

Projecting a point with the graph camera uses two steps. First, the frustum containing the given 3-D point is found. Then the point is projected directly to the output image with a 4-D matrix that concatenates the projections of all the cameras on the path from the current frustum to the root. The frustum containing the point can be found with an octree or another hierarchical space subdivision [PRA09]. Using the projection, graph camera depth image construction proceeds as in Section 3.1, except that a triangle has to be processed with each frustum it intersects.

5.1 Ray intersection

In order to intersect a graph camera depth image with a ray it is possible to follow the generic algorithm: interpolate the ray uniformly in 3-D space, and project each new point onto the graph camera image. However, since each frustum is a pinhole, the projection of a given ray is piecewise linear (Figure 10), which enables the following optimization. Given a ray *r*, for each graph camera frustum F_i:

1. Intersect ray *r* with F_i to produce 3-D sub-segment (s_i, e_i).
2. Project segment (s_i, e_i) to graph camera image segment (p_i, q_i).
3. Walk on (p_i, q_i) to find intersection.

The algorithm computes the linear pieces of the ray directly by intersecting the ray with all the frusta, resulting in a set of sub-segments (s_i, e_i). This is more efficient than the generic algorithm

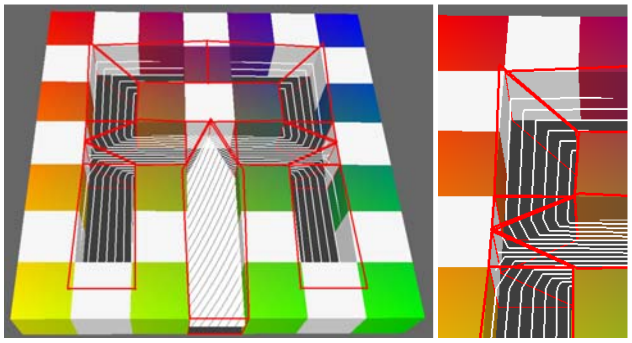
Figure 9. Graph camera model visualization. The frusta are shown in red and few rays are shown in white.

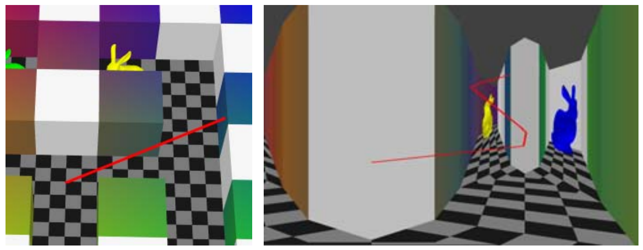
Figure 10. Visualization of 3-D ray (left) and its piecewise linear projection in a graph camera image (right).

which requires small 3-D steps just to model the breaking points of the piecewise linear with high fidelity. Each frustum is a planar pinhole camera, so each sub-segment projection remains a straight line segment (p_i, q_i) in the output graph camera image. The sub-segment is interpolated to search for the intersection step by step, similarly to the generic algorithm.

6 Applications to Interactive Rendering

Non-pinhole depth images accelerate reflection, refraction, relief texture mapping, and ambient occlusion as follows.

6.1 Reflection and refraction

To render a frame of a scene with specular reflections the first step is to update the depth images approximating reflected geometry that is dynamic. Then each reflector is rendered by computing a reflected ray per pixel, a step similar to environment mapping, and by intersecting the reflected ray with the depth images of reflected geometry.

Like in the case of environment mapped reflections, multiple reflections of the same object are obtained at no extra cost. Consider Figure 11, left, where the concave bunny reflects the teapot multiple times. The reflection is computed one ray at the time and the fact that two or more rays reflect the same 3-D point has no bearing on the cost of the method.

Fully dynamic scenes are supported since our method does not require any pre-computation involving the reflector and the non-pinhole depth images are computed efficiently. Higher-order reflections are supported by storing per pixel normals. The normal at a first intersection point is used to create a 2nd order reflected ray which intersects depth images again (Figure 11, right).

When deciding how to approximate reflected geometry the goal is to devise the simplest approximation that captures all samples visible in reflections. For example the black and white ground plane in the reflections in Figure 11 is perfectly captured with a billboard. The SPOC depth image of the teapot is captured along the direction that connects the centers of the bunny and teapot. The field of view of the SPOC is set to the smallest value that encompasses the teapot. The SPOC allows for some flexibility for tuning the disocclusion effect. In the case of the teapot more disocclusion means a better sampling of the lid and bottom and pushing back the silhouette of the body. However, a single SPOC cannot disocclude the entire teapot, as the handle and spout will start occluding the body. If the teapot were spinning or if there

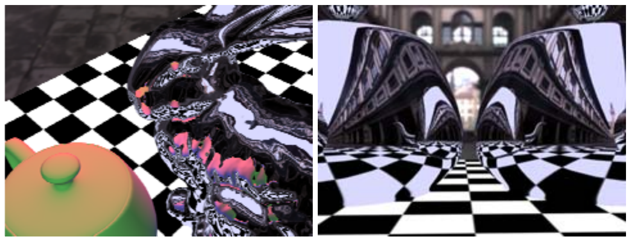
Figure 11. Multiple (left) and 2nd order (right) reflections.

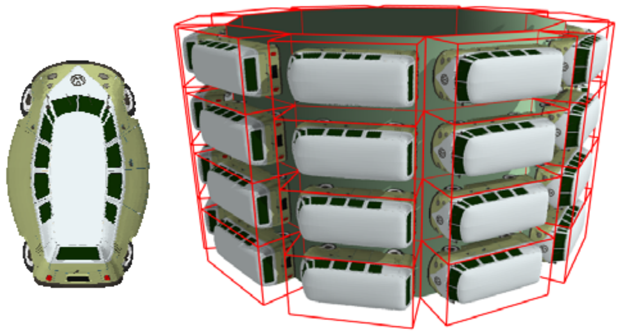
Figure 13. SPOC relief texture (left) and output image with relief bounding box visualization (right).

were multiple reflectors surrounding the teapot, the best solution would be to use two depth images capturing complementary halves of the teapot. SPOCs are suitable for capturing individual objects, and the graph camera for capturing an entire environment.

Regarding the resolution of the depth images, one should select the minimum resolution that captures the reflected geometry well. This way the best reflection is obtained regardless of the rate at which the depth image is sampled by reflected rays in any given frame. Consider a divergent pattern of rays, as obtained for example off a convex reflective surface or off a concave reflective surface beyond the convergence point. Such a pattern will minify the reflected object, which is handled straight-forwardly through mip-mapping. Like conventional depth images, non-pinhole depth images have the great advantage of reducing the problem of geometry minification to the much simpler problem of image minification. A convergent pattern of rays on the other hand magnifies the reflected object, and the reflection will become undersampled once the sampling rate exceeds the resolution of the depth image. Since this is also the resolution of the original geometry, the problem cannot be imputed to the depth image.

Refractions are rendered by intersecting the emerging ray with the depth images approximating geometry. The algorithm for computing emerging refracted rays is orthogonal to this work. We use an image-space approximation for computing the emerging refracted rays [Wym05]. The key idea behind this approximation is to use a first rendering pass to store depth and surface normals for back-facing surfaces, which are then used by a second pass to compute the ray emerging after a second refraction.

6.2 Relief texture mapping

Relief texture map rendering is triggered by rendering the primitives of the coarse underlying model. To obtain correct silhouettes we render the bounding box of each relief tile (Figure 13). For every pixel the eye ray is transformed to the coordinate system of the current relief tile and intersection proceeds as before. World space z is computed at the intersection for correct z-buffering with the rest of the scene and for casting and receiving correct shadows. Shadows could be computed by shooting a second ray from the intersection to the light source and intersecting it with the relief texture. We prefer to use a conventional shadow map such that the relief surface casts and receives shadows from other objects and from other relief tiles.

Non-pinhole relief textures capture complex objects in a single layer. Figure 12 shows that a conventional relief texture misses the wheels and severely under-samples the sides of the car. When tuning the non-pinhole camera parameters the only concern is to capture the relief adequately, independently of the underlying base geometric model. With an increased complexity of the geometry modeled with the relief texture comes the desire to modulate the appearance of individual instances of the relief texture. The different colored cars in Figure 2 were obtained with a single relief texture by simply modifying the color of the intersection if it corresponds to the car body, identified through its yellow color.

6.3 Ambient occlusion

The horizon-based screen-space algorithm [BS08] produces a plausible ambient occlusion effect at little cost, and is therefore of great appeal to interactive rendering applications. The ambient occlusion effect is measured solely based on the z-buffer of the output image. In other words, the geometry of the entire scene is approximated with the point-samples visible from the current viewpoint. This approximation is justified by the heuristic that for simple scenes the geometry occluding an output image pixel is likely to be visible from the current viewpoint and thus sampled by the output image z-buffer. However, this heuristic breaks down for more complicated geometry. Some output image pixels are occluded by geometry that is not visible from the current viewpoint, the occluding geometry is not represented in the output image z-buffer, and the algorithm fails to assign the appropriate ambient occlusion effect to these pixels. The output image z-buffer underestimates the geometry responsible for occluding the output image samples and consequently the algorithm underestimates the ambient occlusion effect.

Moreover, as the output view changes, occluding geometry can appear and disappear in the output image z-buffer which causes the ambient occlusion effect to appear and disappear. When only the view changes the ambient occlusion effect should be of course stable and the instability is a quite noticeable visual artifact, incorrectly suggesting that the lighting changes (see accompanying video).

Non-pinhole cameras present the opportunity of removing this fundamental limitation of the horizon-based screen-space algorithm. The main idea is to compute the ambient occlusion using a non-pinhole z-buffer that samples more of the geometry that occludes the samples in the output image. This leads to a better approximation of the ambient occlusion effect in each output image and also to more stability from image to image. In order to concretize this potential of non-pinhole z-buffers in the context of ambient occlusion two challenges need to be overcome:

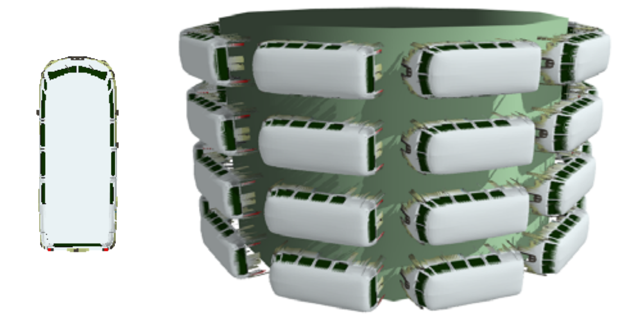
Figure 12. Conventional relief texture (left) & output image (right).

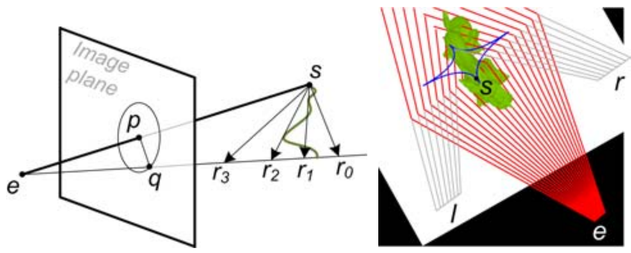

Figure 14. Screen-space ambient occlusion algorithm (left) and visualization of graph camera model (right) used in Figure 3.

specifying a non-pinhole camera that captures as much of the geometry needed to compute the ambient occlusion effect for the current view, and preserving the performance advantage of the original horizon-based ambient occlusion algorithm.

Whereas the effects discussed earlier such as specular reflection and relief texture mapping required tracing a single ray per output image sample, ambient occlusion requires probing visibility along a 2-D set of rays for each output image pixel. The visibility half-space at a pixel is sampled with half-planes and then each half plane is sampled with rays. The original horizon-based screen-space ambient occlusion algorithm is fast because it estimates occlusion in an entire half plane by traversing a single output image z-buffer segment. In Figure 14 left, *p* is an arbitrary output image pixel. All rays r_i project to *pq*, and occlusion on the *q* side of *ep* is estimated by simply traversing *pq*. Casting rays r_i is *not* needed. The dimensionality of the space of rays that need to be cast to probe visibility is reduced from 2 to 1.

This property needs to be maintained when porting the algorithm to non-pinhole z-buffers. The SPOC does not have this property. For any pixel other than the pole there is no set of planes spanning space with each plane projecting to a curve. The planes have a projection with non-zero area. Consequently, even though the SPOC might sample more of the geometry needed for a quality ambient occlusion effect, using it in this context is prohibitively slow.

We have designed a graph camera model that enhances the output image z-buffer with samples visible from two nearby viewpoints and also exhibits the desired property. The camera topology is a simple binary tree consisting of a root and two children. Figure 14, right, visualizes the rays of the graph camera used to render the non-pinhole z-buffer shown in Figure 3. The graph camera has 3 sub-frusta: the output image frustum *e* up to a vertical plane through the splitting point *s*, and the left and right frusta *l* and *r* beyond.

Recall that the graph camera projection is equivalent to a series of conventional projections. Once a plane is collapsed to a line by the leaf projection, the line is mapped to lines by subsequent projections. Consequently, just like for a conventional planar pinhole camera, a line segment in the graph camera image corresponds to a plane in 3-D space. Consider for example sample *S* in the graph camera z-buffer in Figure 3, and the line segment connecting it to sample *Q*. The plane defined by the viewpoint of the right frustum and 3-D points *S* and *Q* projects to graph camera image line *SQ*. Like before, visibility in the entire plane is examined by tracing the image line segment *SQ*.

The horizon-based screen-space algorithm enhanced with a graph-camera z-buffer proceeds as follows:

1. Render output image without ambient occlusion.
2. *Construct graph camera for output view.*
3. *Render z-buffer with graph camera.*
4. For each pixel in the output image, add ambient occlusion effect using graph-camera z-buffer.

Like for the conventional algorithm, the computation of the ambient occlusion effect is differed until after the first rendering pass, to only be performed for visible samples. Steps 2 and 3 are the new steps added to the conventional algorithm.

At step 2 the graph camera is constructed to achieve the desired disocclusion effect. The graph camera depends on the current view. The splitting plane where the root frustum ends and the left and the right frusta begin is perpendicular to the current view direction. One option is to keep the splitting point at the centroid of the object. For the example shown in this paper, the splitting point moves on the blue curve shown in Figure 14, right. The curve was designed off-line to move the splitting point smoothly behind the dragon as the dragon is seen sideways, in which case a conventional z-buffer suffices (also see accompanying video).

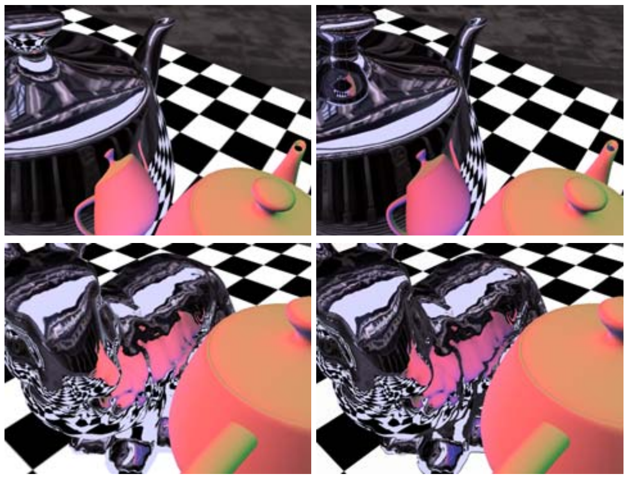
Figure 15. Comparison of our method (left) to ray tracing (right).

At step 3 the graph-camera z-buffer is rendered efficiently leveraging the closed-form and low-cost projection operation of the graph camera. At step 4, the graph camera z-buffer is used like a conventional z-buffer: the ambient occlusion effect at the current pixel is integrated along line segments that emanate radially from the pixel. Like a conventional camera, the graph camera allows recreating a 3-D point from a pixel and a depth value by unprojection.

7 Results and Discussion

7.1 Quality

Non-pinhole depth images enable quality reflections, refractions, relief texture mapping, and ambient occlusion as attested by the images in the paper and by the accompanying video. Figure 15 shows that our method achieves results comparable to ray tracing. The main limitations of our approach are as follows.

Absent self-reflection. Although our method could, in principle, support self-reflections by also intersecting the reflected rays with a depth image of the reflector, the additional intersection is probably a price interactive applications are not willing to pay.

Coarse silhouettes. An SPOC depth image does not sample the entire object it replaces. The sampled area ends with a jagged edge when the SPOC rays are tangential to the replaced geometry (Figure 8). When the jagged edge is exposed, the silhouette of the reflection becomes coarse. One possible solution is to smooth the edge as a pre-process, an approach that precludes dynamic scenes. Instead we alleviate the problem at run time by alpha blending the intersection sample with greater transparency when the SPOC ray becomes tangential to the sampled surface.

Undersampling. Like for all sample-based methods, the quality of the results obtained with non-pinhole depth images is contingent upon adequate sampling. The SPOC sampling rate is uniform and controllable. The graph camera sampling rate is not uniform: it is higher closer to the initial frustum and is lower for

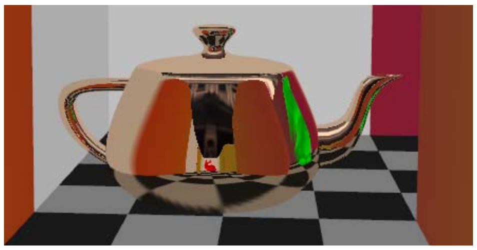
Figure 16. Undersampling artifacts on floor reflection.

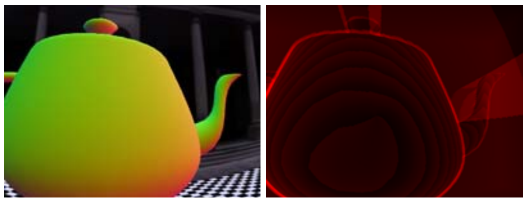
Figure 17. Diffuse teapot reflected in body of large teapot (left) and visualization of number of intersection steps per pixel (right).

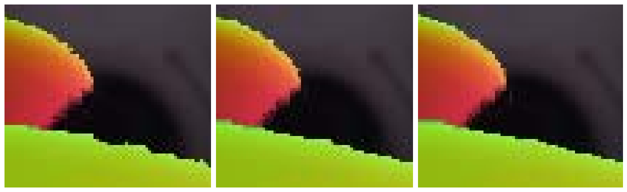
Figure 18. Silhouette detail with fine steps of 1, ½, and ¼ pixels.

the distant frusta. The graph camera depth image used here was constructed to capture the entrance at a higher resolution, where reflections are of highest quality (Figure 6). Deeper into the maze the resolution decreases leading to aliasing artifacts due to the large output image projection of depth image pixels, a problem similar to inadequate shadow map resolution. In Figure 16 the teapot is in the top left corner of the maze (see Figure 6), thus deepest in the graph camera depth image, where sampling insufficiency is noticeable. Note however that the case presented here is particularly challenging: a smooth highly-specular surface reflects a contrasting checker pattern. We use a graph camera depth image resolution of 1,920 x 1,175. A brute force solution is to increase the resolution further. Another possibility is to break up the maze into several parts each with its own smaller graph camera depth image. Lastly, hybrid sample-based/geometry-based techniques that incorporate "infinite frequency" edges into textures could also be used.

Missing samples. The most visible artifacts in non-pinhole relief texture mapping are caused by samples still missing from the relief texture due to residual occlusions. The rear bumper of the car shown in Figure 2 occludes some of the car body in the relief texture, which causes the shimmering "rubber band" surface seen in the video. One solution is to modify the car model to reduce the distance between the bumper and the body of the car by pushing the bumper in or by thickening the bumper. Another solution is to encode the bumper in a second relief texture layer.

7.2 Performance

The timing information reported here was collected on a 3.4GHz 2GB Intel Xeon PC with an NVIDIA 8800 Ultra 768MB card. We used NVIDIA's Cg 2.0 shading language with gp4 profiles. An important performance factor is the number of steps along the projection of the ray, which we analyzed for reflections.

We take coarse steps first and perform a fuzzy intersection of the coarse ray segment with the non-pinhole depth map. If the two endpoints project at unoccupied locations or if the coarse ray segment clearly does not intersect the impostor depth map, the coarse segment is trivially rejected. Coarse segments are refined by performing fine steps of 1, ½, or ¼ depth image pixels. Figure 17 illustrates the number of steps for a 512x512 SPOC depth image, a 6 pixel coarse step, and a ¼ pixel fine step. More steps are needed when the reflected ray narrowly misses the teapot, which causes the fuzzy test to return a false positive. The average number of steps is 48 per output pixel, including both coarse and fine steps. For fine steps of 1 and ½ pixels the average number of steps is 22 and 31, respectively. These numbers do not account for pixel processor idling due to SIMD processing constraints. Figure 18 shows reflection silhouette quality for various fine step sizes.

Res.	**640x480**			**800x600**			**1024x768**		
Fine step	**1**	**1/2**	**1/4**	**1**	**1/2**	**1/4**	**1**	**1/2**	**¼**
Avg FPS	26.5	20.7	15	20.9	16.5	11.6	15.2	11.8	8
Min FPS	18	14	8	14	10	4	10	8	6
Max FPS	54	46	36	48	15	34	40	34	28

Table 1. Frame rate [fps] variation with output resolution and fine step size [pixels] for scene in Figure 1.

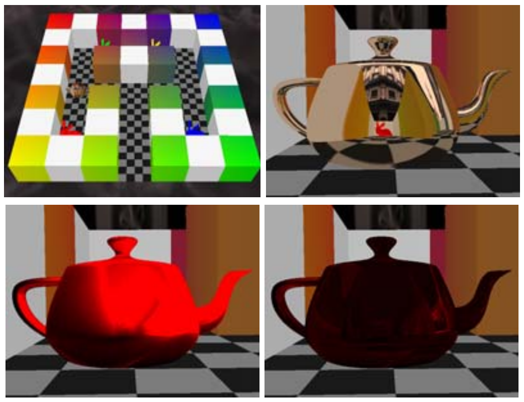
Figure 19. Teapot location and reflection (top), and number of steps visualization (bottom) w/o and w/ coarse stepping. Brighter red indicates a large number of steps being taken.

Performance depends on output image resolution and on the fine step size as shown in Table 1. Performance was measured on a typical path (see video) for the scene shown in Figure 1. Eight sample multi-sampling anti-aliasing (8x MSAA), a 512x512 SPOC depth image, and a coarse step of 6 pixels were used. For an output resolution of 640x480, with MSAAx8, the average frame rate for SPOC depth images of resolution 128x128, 256x256, 512x512, and 1024x1024, is 55.8, 36.6, 26.5, and 16.1 fps, respectively. For coarse steps of 3, 6, 9, and 12 texels, the average frame rates are 18.2, 31, 37.2, and 39.8 fps, respectively. The only feature thin enough to be affected by the coarser steps is the tip of the spout. For a sequence where the SPOC depth image is recomputed on the fly (see video), the average frame rates are 22 and 17.3 fps for no anti-aliasing and 16x MSAA, respectively.

Coarse stepping reduces the number of steps for the graph camera depth image as well, as can be seen in Figure 19, where the average number of steps decreases from over 155 to 27. The reflection of the main entrance where the graph camera impostor has highest resolution remains the hottest area on the teapot but only a few pixels have large step numbers. The graph camera impostor is constructed at over 100 fps. The average, min, and max frame rates for the path that follows the teapot through the maze (see video) are 45.5, 30, and 105 fps w/o antialiasing, and 26.8, 20, and 42 fps with 8x MSAA, respectively.

We will examine accelerating the ray/depth image intersection computation further by leveraging ray coherence. We envision a two pass approach that first renders the reflection at lower resolution and then up-samples by interpolation in coherent regions. The second pass actually

Figure 20. Short relief example.

computes intersections only at regions where the lower resolution results are not sufficient to reconstruct a quality intersection, such as at edge regions.

We construct non-pinhole relief texture maps with an SPOC and the discussion of performance of ray intersection provided above in the context of reflections still applies. For Figure 2 the overall performance, including shadow mapping, is 14 fps for the 40 cars and 18 fps for the 60 barrels example. Output resolution is 640x480 and relief texture resolution is 512x512. For 20, 10, and 1 car performance is 26, 51, and 219 fps, respectively. All examples shown use tall relief which implies long ray projections. For scenes with short relief performance is even higher, e.g. 46 fps for 160 cars half the size (Figure 20).

We investigated ambient occlusion performance for two quality occlusion sampling settings: regular (6 sampling directions and 6 steps per direction), and fine (32 and 20). The blur kernel width was 21 pixels and the output image resolution was 1024x1024 in both cases. Average performance for the dragon scene (Figure 3) is 35 and 16 fps for the regular and fine settings, respectively. Figure 3 and the video use the fine sampling setting—the regular setting produces a noisier ambient occlusion, issue orthogonal to the use of the non-pinhole z-buffer. Rendering the locally illuminated output image, the graph camera z-buffer, and adding the ambient occlusion effect takes 5.9, 16, and 6ms for the regular setting and 5.9, 16, and 40ms for the fine setting, respectively. For the conventional algorithm rendering the locally illuminated output image and adding the ambient occlusion effect takes 5.9 and 5.9ms for the regular setting and 5.9 and 32ms for the fine setting, respectively. Sampling occlusion in the graph camera z-buffer as opposed to the conventional z-buffer brings a 25% penalty, and most performance loss is brought by having to render the graph camera z-buffer of the 890K triangles scene. However, in some scenes it is unnecessary to update the graph camera z-buffer for every frame. The non-pinhole z-buffer does see more than what is visible in the output image which greatly extends its resiliency to output view changes. For example the z-buffer shown in Figure 3 can be reused if the dragon is only seen from the front, which increases performance to 21 fps for the high setting, approaching the 26 fps of the conventional algorithm. Another possible approach is to use a simplified version of the model when computing the non-pinhole z-buffer.

8 Conclusions and future work

We have shown that the desirable property of conventional depth images of efficient ray intersection holds for non-pinhole depth images, at the small additional cost of traversing a curved—as opposed to a straight—ray projection. We have leveraged this property to render specular reflections, refractions, relief texture mapping, and ambient occlusion. In prior work we have also used DDOCs to render soft shadows at interactive rates [MPW07]. The suitability of non-pinholes to so many rendering problems argues for the generality of the technique.

Compared to reflection rendering techniques that approximate the projection of reflected points such as explosion maps [OR98], our method produces multiple projections of the same object at no extra cost and handles complex reflectors. Compared to color-caching image-based techniques, our method supports dynamic scenes and has reduced memory requirements. Color-caching techniques excel at capturing the appearance of complex real-world materials that are glossy, but not specular. Compared to environment mapping, our method produces better results, at a higher per-pixel cost. Compared to ray tracing, our method more easily minifies and magnifies reflections by working in the color map at different levels of resolution, and achieves fast ray / geometry intersection. Ray tracing has a quality advantage since it does not approximate the reflected geometry. Compared to conventional relief texture mapping the non-pinhole relief maps bring greater modeling power at the cost of a more expensive ray/relief texture intersection.

In addition to the directions for future work already sketched, we will investigate integrating our non-pinhole rendering framework into popular digital 3-D content creation tools. Our work argues for the benefits and practicality of exploring camera models that abandon the pinhole constraint. Non-pinhole cameras can be designed to provide powerful yet inexpensive approximations for many applications in graphics and beyond.

References

[AZM00] Agrawala, M., Zorin, D., and Munzner, T. Artistic Multiprojection Rendering. In *Eurographics Workshop on Rendering Techniques*, pp. 125–136, 2000.

[BS08] Bavoil L. and Sainz M. Image-Space Horizon-Based Ambient Occlusion, SIGGRAPH 2008, Talk Program, 2008.

[Bjo04] Bjorke K. Image-based lighting. GPU Gems, Fernando R., (Ed.). NVidia, (2004), pp. 307–322.

[BN76] Blinn J.F., Newell M. E. Texture and Reflection in Computer Generated Images. CACM 19:10, pp. 542-547, 1976.

[CW05] Chan B. and Wang W. Geocube—GPU accelerated real-time rendering of transparency and translucency. The Visual Computer, vol 21, 2005, pp. 579-590.

[DYB98] Debevec P., Yu Y., Borshukov G. Efficient view-dependent image-based rendering with projective texture-mapping. In EG Workshop on Rendering, 1998, pp. 105–116.

[DDS*03] Decoret X. et al. Billboard Clouds for Extreme Model Simplification. ACM SIGGRAPH 2003, pp.689-696.

[GH97] Gupta, R., and Hartley, R. Linear Pushbroom Cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence 19*, 9, 963–975, 1997.

[MS95] Maciel P., and Shirley P. Visual Navigation of Large Environments Using Textured Clusters. I3D'95, pp. 95-102.

[MMB97] Mark W., Mcmillan L., and Bishop G. Post-Rendering 3D Warping. Symp. on Interactive 3D Graphics, pp. 7-16.

[MO95] Max N. and Ohsaki K. Rendering trees from precomputed z-buffer views. In Rendering Techniques '95: Proceedings of the EGRW, pp. 45–54.

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

[MPS05] Mei C., Popescu V., and Sacks E. The Occlusion Camera. Eurographics 2005, pp. 335-342.

[OR98] Ofek E. and Rappoport A. Interactive reflections on curved objects. In Proc. of SIGGRAPH '98, ACM Press, 333-342.

[Paj02] Pajdla, T. Geometry of Two-Slit Camera. Tech. Rep. CTU–CMP–2002–02, Czech Technical University, 2002.

[PG04] Pharr M. and Green S. Ambient Occlusion. In GPU Gems, edited by Randima Fernando, pp. 279–292, 2004.

[PO06] Policarpo F. and Oliveira M. Relief Mapping of Non-Height-Field Surface Details. In ACM I3D 2006, pp. 55-62.

[POC05] Policarpo F., Oliveira M., and Comba J. Real-Time Relief Mapping on Arbitrary Polygonal Surfaces. In ACM Symp. on Interactive 3-D Graphics and Games 2005, pp. 155-162.

[PRA09] Popescu V., Rosen P., and Adamo-Villani, N. The Graph Camera. ACM Trans. Graph. 28 (SIGGRAPH ASIA), 5, 2009.

[PMDS06] Popescu V., Mei C., Dauble J., and Sacks E. Reflected-Scene Impostors for Realistic Reflections at Interactive Rates. Computer Graphics Forum, (25):3 (EG 2006), pp. 313-322.

[PA06] Popescu, V. and D. Aliaga. The Depth Discontinuity Occlusion Camera. In ACM I3D 2006, pp. 139-143.

[QPW07] Qi, M. V. Popescu, and C. Wyman. The soft shadow occlusion camera. In Proceedings of Pacific Graphics 2007.

[RB98] RADEMACHER P. AND BISHOP G. Multiple-center-of-Projection Images. Proc. ACM SIGG. ’98, pp. 199–206.

[RAH07] ROGER D., ASSARSSON U., AND HOLZSCHUCH N. Whitted Ray-Tracing for Dynamic Scenes Using a Ray-Space Hierarchy on the GPU. In EGSR, 2007, pp. 99-110.

[RP08] ROSEN P. AND POPESCU V. The Epipolar Occlusion Camera, In ACM Symp. on Inter. 3-D Graphics 2008, pp. 115-122.

[SALP05] SZIRMAY-KALOS L. ET AL. Approximate Ray-Tracing on the GPU with Distance Impostors. Computer Graphics Forum, 24(3), 2005. pp. 171-176.

[Sha98] SHADE J. ET AL. Layered Depth Images. In Proceedings of SIGGRAPH 98, 231-242.

[Whi80] WHITTED T. An improved illumination model for shaded display. Comm. of the ACM (1980), 23, 6, pp. 343-349.

[Woo97] WOOD D. ET AL. Multiperspective Panoramas for Cel Animation. In Proc. of ACM SIGG. ’97, pp. 243-250.

[Wym05] WYMAN C. An Approximate Image-Space Approach for Interactive Refraction. ACM ToG, vol. 24, no 3, pp. 1050-1053.

[YM04] YU J. AND MCMILLAN L. A Framework for Multiperspective Rendering. In EGSR 2004.

[YM04b] YU, J., AND MCMILLAN, L. General Linear Cameras. In *ECCV ’04* vol. 2, pp. 14–27, 2004.

[ZIK98] ZHUKOV S., IONES A., AND KRONIN G. An Ambient Light Illumination Model. In EGRW '98, pp. 45–56.

Paul Rosen completed his Ph.D. in computer science at Purdue University in 2010. He is currently a Post Doctoral Research Associate in the Scientific Computing and Imaging Institute at the University of Utah. His research interests are primarily in camera model design and its applications in computer graphics and visualization, but also to computer vision and computer-human interaction. His interests also include multicore and parallel desktop computing, visualization of large-scale data sets, and uncertainty visualization.

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 associate professor with the Computer Science Department of Purdue University. His research interests lie in the areas of computer graphics, computer vision, and visualization. His current projects include camera model design, perceptual evaluation of rendered imagery and 3D displays, research and development of 3D scene acquisition systems, and research, development, and assessment of next generation distance learning systems.

Kyle Hayward graduated from Purdue University with a BS degree in Computer Science in 2009 and is a graphics programmer for Human Head Studios, Inc. His background is in real-time rendering, including facial and skeletal animation, natural effects rendering, volume rendering, and reflections and refractions using non-pinhole cameras. His current research interests are in real-time ray tracing on the GPU.

Chris Wyman received BS degrees in mathematics and computer science from the University of Minnesota in 1999 and a PhD in computer science from the University of Utah in 2004. He is currently an Associate Professor of Computer Science at the University of Iowa. His research interests focus on interactive global illumination, including both diffuse color bleeding and specular focusing of light, but also extend to other interactive and realistic rendering problems, visualization, and perceptual issues in rendering.