2009

Non-Pinhole Imposters

Voicu Popescu
Purdue University, popescu@cs.purdue.edu

Kyle Hayward

Paul Rosen

Chris Wyman

Report Number:

09-006

Non-Pinhole Impostors

Voicu Popescu
Kyle Hayward
Paul Rosen
Chris Wyman

CSD TR #09-006
August 2009

Non-Pinhole Impostors

Voicu Popescu, Kyle Hayward, Paul Rosen, Chris Wyman

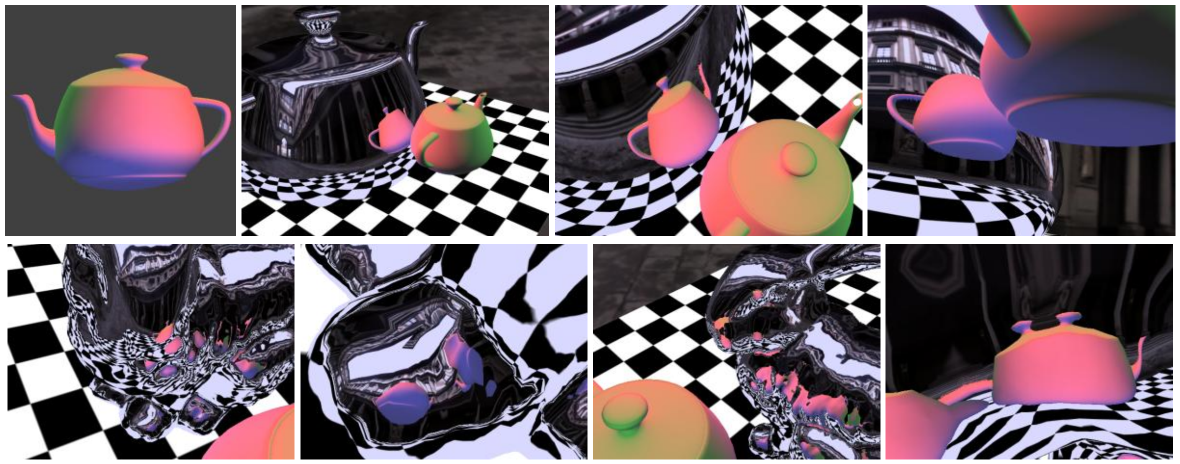
Figure 1: *Single-pole occlusion camera (SPOC) impostor of teapot (top left), and reflections rendered using it, 30fps.*

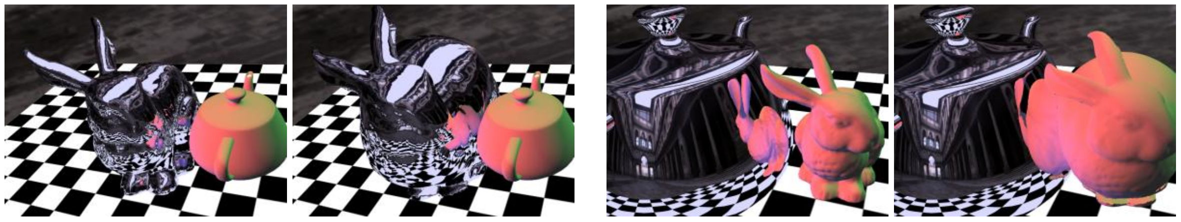
Figure 2: *Dynamic reflector (left) and dynamic diffuse object (right), 30fps.*

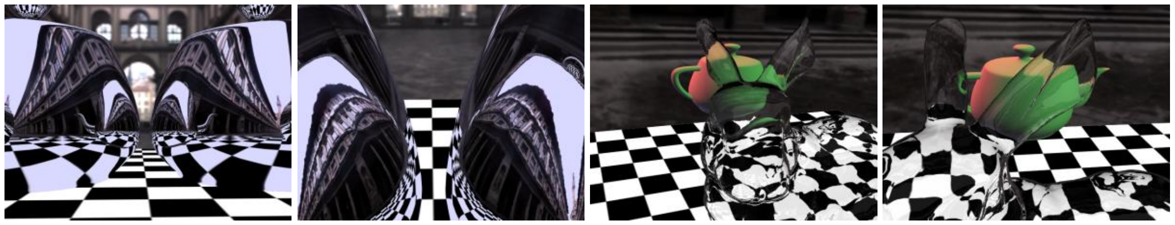
Figure 3: *Inter-reflecting teapots (left) and refractions (right), all rendered using SPOC impostors, 30fps.*

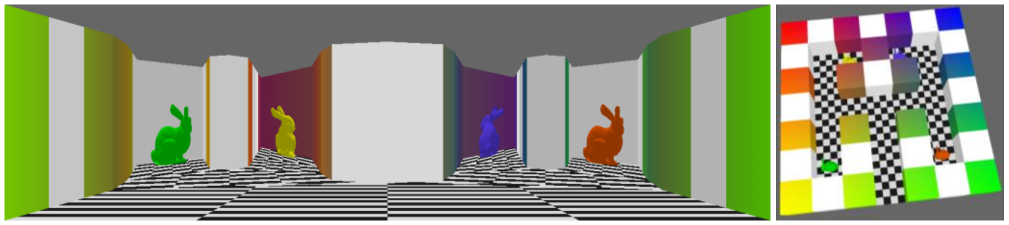
Figure 4: *Graph camera impostor (left) captures entire 3-D maze (right).*

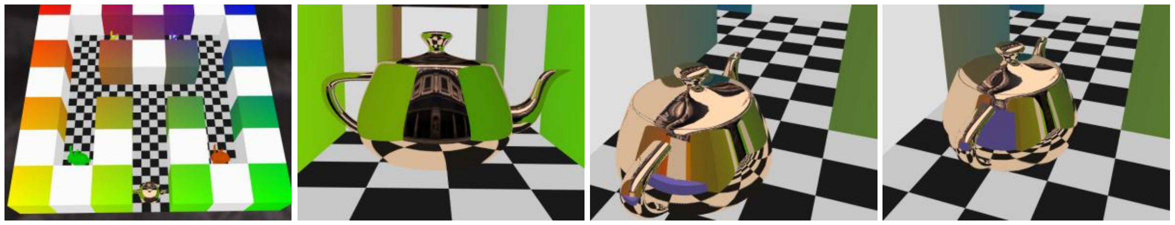
Figure 5: *Reflections rendered with graph camera impostor in a dynamic scene, 20fps.*

Abstract

Impostors are approximations of scene geometry with multiple applications in computer graphics. In previous work impostors are constructed with orthographic or perspective projections which limit the approximation quality to what is visible along a single view direction or from a single viewpoint. In this paper we show that impostors constructed with non-pinhole cameras improve the approximation quality at little additional cost if the non-pinhole camera provides fast projection. For such a camera, the fundamental operation of ray-impostor intersection proceeds efficiently by searching along the one-dimensional projection of the ray on the impostor image. In the context of two-camera configurations, our work extends epipolar geometry constraints, well known for pinholes, to non-pinholes. We demonstrate the advantages of non-pinhole impostors in the context of interactive reflection and refraction rendering.

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

1. Introduction

In the quest for higher-quality and higher-performance rendering, researchers have developed impostors, a general technique of substituting scene geometry with more efficient representations. The three main desirable properties of impostors are high-fidelity geometry approximation, efficient construction, and efficient rendering. An impostor should describe the geometry it replaces sufficiently well such that the output image rendered with the impostor is virtually indistinguishable from an output image rendered with the original geometry. To support fully dynamic scenes, impostors have to be created on the fly, which requires fast construction.

Lastly, impostors have to deliver the desired performance boost to the application that employs them. We distinguish between applications where the impostor is seen directly, as for example when the impostor replaces distant geometry for scene complexity management purposes, and applications where the impostor is seen indirectly, as for example in reflection and refraction rendering. Whereas in the first type of applications the impostor can be rendered directly, with the conventional feed-forward approach of projection followed by rasterization, rendering reflected or refracted impostors efficiently requires a fast ray-impostor intersection operation. The coherence of the desired view rays is perturbed by the reflector or refractor and no closed form projection exists that takes impostor 3-D points directly to the output image. The lack of projection operation precludes conventional rendering and requires that the reflector or refractor be rendered by intersecting the ray at each fragment with the scene impostors.

Several types of impostors have been developed, which we review in the next section. An important basic type of impostor is the *depth image*, which stores a depth value for each pixel. A depth image is constructed efficiently by rendering the geometry it replaces. Fast ray / depth image intersection is enabled by the fact that the ray projects on the depth image to a segment, which reduces the dimensionality of the intersection search space from two to one. Graphics hardware has reached a sufficient level of performance to allow 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 impostor is rendered by the application, which lowers the quality of the result (Figure 6).

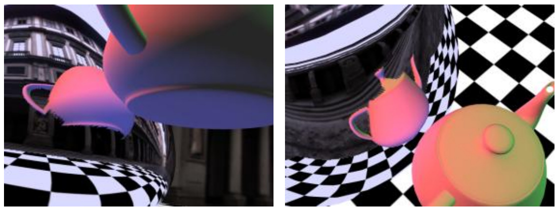
Figure 6: *Reflections rendered with a conventional planar pinhole camera depth image impostor, which does not capture the lid and bottom of the teapot.*

In this paper we propose to construct impostors using non-pinhole cameras. Such non-pinhole impostors offer a high-fidelity approximation of scene geometry while construction and rendering costs remain low. Once the restriction that all rays pass through one point is removed, the rays of non-pinhole camera can be designed such as to sample all surfaces that are exposed by the application during the use of the impostor. To ensure construction and rendering efficiency, the non-pinhole camera model is designed to provide a fast projection operation. This enables constructing the impostor in feed-forward fashion, with the help of graphics hardware, 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 on the non-pinhole image. Like in the case of planar pinhole camera impostors, the ray / non-pinhole camera impostor intersection is found by walking on the one-dimensional projection of the ray. Unlike in the case of planar pinhole camera impostors, the ray projection is not a straight line, which, however, does not raise the cost of intersection computation significantly.

We construct impostors with two recently introduced non-pinhole camera models: the single-pole occlusion camera (SPOC) [MPS05], and the graph camera [RPA08]. The SPOC has rays that reach around an object's silhouette to gather samples that are not visible from the reference viewpoint but that are close to the silhouette. Such "barely" occluded samples are needed to provide adequate reconstruction of the geometry when the impostor is sampled during the application by rays from nearby viewpoints. The graph camera is a non-pinhole camera constructed starting from a planar pinhole camera which undergoes a series of 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.

The SPOC is used for constructing impostors of single objects, whereas the graph camera is suitable for replacing an entire scene. The advantages of the two non-pinhole camera impostors are demonstrated in the context of interactive rendering of specular reflections and of refractions (Figures 1 through 5 and accompanying video). Figure 1 shows that an SPOC impostor captures the lid and the bottom of the teapot which were missing in Figure 6. Both types of non-pinhole camera impostors provide sufficient coverage of the geometry they replace to provide all samples seen by reflected or refracted rays; both types are constructed at interactive rates supporting fully dynamic scenes; finally, both types provide efficient ray / impostor intersection operations, which translate in interactive frame rates.

The remainder of this paper is organized as follows. Prior work is reviewed next. Section 3 discusses the construction and ray intersection operations for impostors constructed with a generic non-pinhole camera. Sections 4 and 5 describe the specialization to SPOC and graph camera impostors. Sections 6 and 7 present results, conclusions, and possible directions for future work.

2. Prior work

We review prior research on impostors, on non-pinhole camera models, and on reflection and refraction rendering.

2.1. Impostors

The term impostor was introduced by Maciel and Shirley [MS95] and is now widely adopted to denote an image-based simplified representation of geometry for the purpose of efficiency. The simplest impostor is a billboard, a quad texture mapped with the image of the original geometry, with transparent background pixels. Billboards are rendered efficiently, intersecting a billboard with a ray is trivial, and billboards provide good approximations of geometry seen orthogonally from a distance. When the impostor is close to the viewer or close to a reflector surface, 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 maximum modeling fidelity. The number of quads is sufficiently small to enable the intersection of a reflected or refracted ray with each quad. However, the optimization makes construction of the billboard cloud a lengthy process that precludes dynamic scenes. Moreover, the approximation quality is still not sufficient for close-up viewing. In the case of reflection for example, if a complex diffuse objects intersects the reflector surface, the intersection line will be poorly approximated by the billboard cloud.

Depth images [MB95] greatly improve over 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 belongs to the image plane projection of the ray. Since the depth image is constructed with a planar pinhole camera, the depth image only captures samples visible from the reference viewpoint. When surfaces not captured by the impostor become visible during the application, objectionable disocclusion error 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 applications such as reflections and refractions.

Another solution to the occlusions problem is relief texture-mapping [POC05], a hybrid geometry / depth image representation. True geometric detail is added to a coarse triangle mesh by texturing each triangle with a height (i.e. relief) map. Occlusions are avoided since the coarse mesh is view independent and since the geometric detail has one sample for each triangle point. The eye ray is projected onto the relief map and the intersection is computed along the projection, as it is for depth maps. A similar method is procedural or sample-based geometry generation through tessellation, leveraging the programmability at primitive level exposed by recent graphics hardware. Neither method can easily intersect a ray with an entire object—the ray needs to be intersected with the coarse triangle mesh first, which makes the methods ill-suited for applications such as reflections and refractions.

2.2. Non-pinhole cameras

Non-pinholes have been studied relatively little in computer graphics. The light field [LH96] and the lumigraph [GGS*96] can be seen as the color samples acquired by a 2D array of planar pinhole cameras. Their strengths lie in the acquisition of small-scale complex real-world scenes. Although possible in principle, using light fields as impostors is precluded by their large memory footprint and construction time. Multiple-center of projection cameras [RB98] sample the scene with a vertical slit along a user chosen path and thus avoid the redundancy of the light fields and offer good modeling power. However, construction requires rendering the scene for each position along the path, which is inefficient. Camera models developed for multiperspective rendering [Woo97, YM04] simulate camera motion through a 3-D scene but do not support viewing from novel views, nor dynamic scenes.

Occlusion cameras have been recently introduced 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]. Whereas the SPOC specifies the 3-D distortion of the reference view rays analytically, 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.

In our context of devising an impostor that represents scene geometry well from a wide range of viewpoints and that is efficient, the SPOC offers a good balance between modeling power and efficiency, and we have adopted it to construct object non-pinhole impostors. In order to construct environment impostors we chose the graph camera [RPA08], leveraging the malleability of its rays.

2.3. Reflection and refraction rendering

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 (e.g. light fields [LH96, GGS*96] and view dependent texture mapping [DYB98]), projection [OR98], and reflected/refracted scene approximation. We only discuss the latter, since most relevant to this work.

Environment mapping [BN76] is currently the approach preferred by applications due to its efficiency, robustness, and good results when the reflected/refracted scene is not close to 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 reflected/refracted scene approximation can be improved by resorting to depth image impostors [SALP05, PDSM06]. Quality reflections are produced for simple objects or for select viewpoints, but the insufficient coverage is an important limitation for non-trivial scenes or wide viewpoint translations (Figure 6).

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 techniques [RAH07], and image-space approximations [Wym05]. In order to illustrate non-pinhole impostors, we use an image-space approximation to compute the emerging refracted rays [Wym05], which are then intersected with the impostor. 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 emerging ray after a second refraction.

3. Non-pinhole camera impostors

Once the pinhole restriction is removed, there is great flexibility in devising a camera model that best suits a given application and a particular dataset. Therefore we first discuss the construction and ray intersection for non-pinhole impostors 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, gd) where (u, v) are image coordinates and gd is a measure of depth linear in image space, a non-pinhole impostor is constructed efficiently by projecting the vertices of the geometry it replaces and by rasterizing the projected triangle conventionally. The unconventional projection can be executed by a vertex program which essentially implements the non-pinhole camera model. Since lines do not project to lines and since rasterization parameters do not vary linearly (before the perspective divide) anymore, the triangles have to be sufficiently small to provide an adequate approximation. Complex objects are typically modeled with small triangles to provide a good approximation of their shape, so additional tessellation is usually not needed. Meshes of objects with large triangles can be subdivided on-the-fly by taking advantage of primitive-level GPU programmability.

3.2. Intersection

Like a regular depth image impostor, a non-pinhole impostor 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 impostor NPI is computed with the following steps:

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

2. Project (c, d) to $((u_c, v_c, gd_c), (u_d, v_d, gd_d))$.

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

    3.1. Project (s_k, s_{k+1}) to $((u_k,v_k), (u_{k+1},v_{k+1}))$

    3.2. Intersect $((u_k,v_k, GD(u_k , v_k)), (u_{k+1},v_{k+1}, GD(u_{k+1}, v_{k+1}))$ with $((u_c, v_c, gd_c), (u_d, v_d, gd_d))$, where $GD(u, v)$ is the depth stored by the impostor at image location (u, v). If an intersection is found, break, else continue.

The ray has to be interpolated in 3-D since its projection is not a straight line, and one cannot simply rasterize the segment that connects the projection of its two endpoints. Each intermediate point is projected with the non-pinhole camera of the impostor which traces the curved projection correctly. Since the depth gd stored by the impostor varies linearly in the image, the intersection can be computed efficiently in a 2D space (t, gd), where t is the parameter locating the intersection along segment $((u_k,v_k), (u_{k+1},v_{k+1}))$.

For applications such as reflections or refractions, the ray that has to be intersected with the impostor is computed for each reflector or refractor pixel, which requires sending the non-pinhole camera parameters to the pixel shader as well. The generic construction and ray intersection algorithms are specialized for SPOC and graph camera impostors as follows.

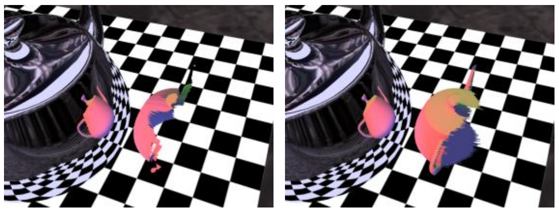

Figure 7: *Samples stored by a planar pinhole camera (left) and an SPOC (right) impostor. The SPOC impostor covers considerably more of the diffuse teapot.*

4. Single-pole occlusion camera impostors

The SPOC projection consists of a conventional planar pinhole camera projection followed by a distortion which 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 impostor in Figure 1 the distortion pushes the silhouette back, revealing the lid and the bottom. Figure 7 shows that the SPOC impostor 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 curved projection of the ray. The projection is visualized in Figure 8.

5. Graph camera impostors

The graph camera is constructed recursively starting from a planar pinhole camera through a succession of bending, splitting, and merging operations [RPA08]. 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 a parent from a child frustum, but it remains continuous. This makes the graph camera image continuous. The rays are disjoint, which makes that a point projects to a single image location, avoiding redundancy. The graph camera constructed for the maze in Figure 4 is shown in Figure 9. Here the construction followed a breadth first traversal of the maze graph starting from the entrance at the bottom of the maze.

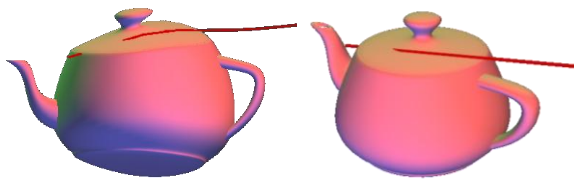

Figure 8: *Visualization of the curved SPOC projection of a ray and of its intersection with the impostor (left), and visualization of the ray intersecting the teapot (right).*

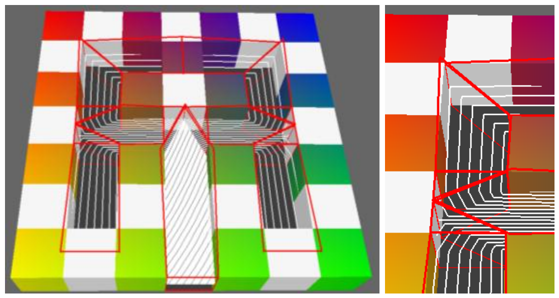

Figure 9: *Graph camera model visualization. The frusta are shown in red and a few rays are shown in white.*

Projecting a point with the graph camera implies two steps. The frustum containing the given 3-D point is found in a first step, followed by projection directly to the output image with a 4-D matrix that concatenates the projections of all the cameras on the path to the root. The frustum containing the point can be found with an octree or another hierarchical space subdivision [RPA08], but, for efficiency, we use a texture map of the floor of the maze that stores frustum ids.

With this projection operation the graph camera impostor construction proceeds according to the algorithm described in Section 3, with the only notable difference of clipping and rendering a triangle with each frustum it intersects.

We have developed two algorithms for intersecting a graph camera impostor with a ray. The difference is in how the ray is interpolated to model its non-linear projection. The first algorithm follows the generic algorithm closely: the ray is interpolated uniformly in 3-D space, and each new point is projected onto the graph camera image. This approach has the disadvantage that it does not know about the points where the ray intersects a frustum. The ray projection changes direction at these points and finding quality intersections requires using a fine interpolation step. Figure 10 shows how a ray is broken into pieces by graph camera projection.

The second algorithm models the piecewise linear projection of the ray well. The algorithm takes the

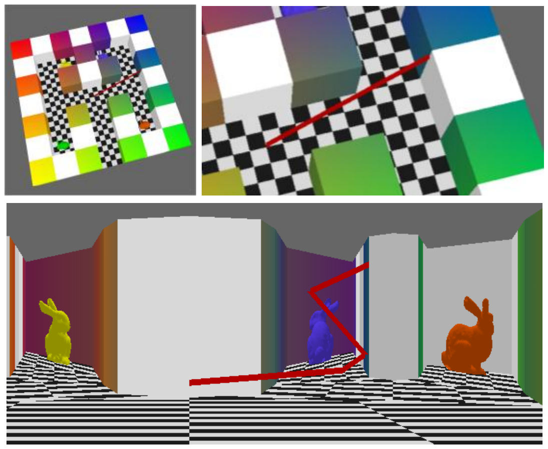

Figure 10: *Visualization of a ray intersecting the maze (top) and visualization of the piecewise linear graph camera projection of a ray and of its intersection with the impostor (bottom).*

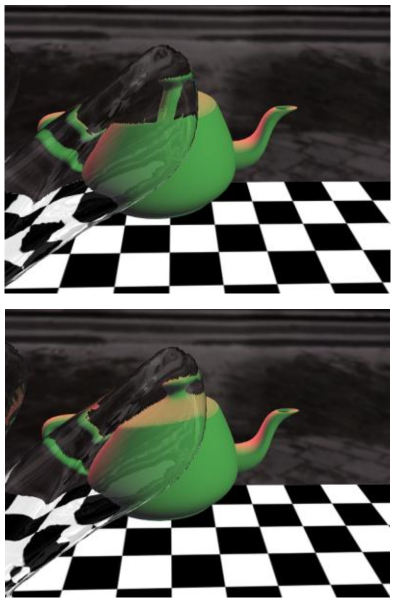

Figure 11: *Refraction rendered with regular depth image (top) and SPOC impostor (bottom).*

following steps for each graph camera frustum F_i:

1. Intersect ray r with F_i to produce sub-segment (s_i, e_i).

2. Project segment (s_i, e_i) to graph camera image segment (p_i, q_i).

3. Interpolate (p_i, q_i) to search for intersection with graph camera depth map.

The algorithm determines the intermediate points on the ray by intersecting it with all the frusta, resulting in a set of sub-segments (s_i, e_i). Each frustum is a planar pinhole camera, which implies that each sub-segment projects to 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.

The first algorithm has the advantage that it only works with the frusta intersected by the ray, whereas the second algorithm considers all frusta. For the graph camera used in this paper (Figure 9), which comprises 15 planar pinhole camera frusta, the second algorithm has superior performance.

6. Results

We have tested SPOC and graph camera impostors in the context of specular reflection and refraction rendering. Using the impostors is straight forward: once the reflected or refracted ray is computed in the pixel shader, the ray is intersected with the impostors. For the images rendered with an SPOC reflector the reflection of the grid is modeled with a billboard impostor, which captures it perfectly. The floor of the maze is part of the graph camera impostor.

For refractions, the superior modeling power of non-pinhole impostors is particularly evident over thin parts of the refractor where the refracted object, and any missing surface, can be clearly noticed (Figure 11***Figure 12***). Second order reflections are supported by storing normals

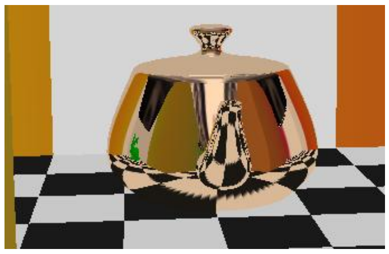

Figure 12: *The graph camera impostors samples distant parts of the maze at a lower resolution creating the aliasing artifacts for the floor.*

instead of color (Figure 3). Once the intersection is found, a second order ray is computed and the impostors are intersected again. Non-pinhole impostors enable reflection and refraction rendering with good quality and good performance.

6.1. Quality

Our method produces good results as attested by the images in the paper and by the accompanying video. In Figure 1 the complex bunny geometry exposes a considerable fraction of the teapot geometry, which is sampled by the SPOC impostor. The complex normals on the bunny lead to extreme reflection magnification and minification, which are handled well. The reflector and reflected objects can intersect, and the images show the expected reflection continuity (Figure 2). A graph camera captures a complex environment producing more accurate reflections than environment mapping.

Like all sample-based methods, the quality of the results obtained with non-pinhole impostors is contingent upon adequate sampling. The SPOC approximates only a single object so sampling rate is higher than for the graph camera. The graph camera sampling resolution is not uniform: it is higher closer to the initial frustum and is lower for the distant frusta. The graph camera impostor used here was constructed to capture the entrance at a higher resolution, where reflections are of highest quality (Figure 5). Deeper in the maze the resolution decreases leading to aliasing artifacts (Figure 12). Whenever the edge of the impostor is visible, the silhouette of the reflection is jagged.

6.2. Performance

The timing information reported in this paper was collected on a 3.4GHz 2GB Intel Xeon workstation with an NVIDIA 8800 Ultra 768MB card. We used NVIDIA's Cg 2.0 shading language with gp4 profiles. Performance depends on output image resolution as shown in Table 1.

	640x480	800x600	1024x768	1280x1024
Avg	52.4	42.78	36.1	23.9
Min	36	28	24	10
Max	70	56	58	44

Table 1: Frame rates along a typical path in the 2 teapot scene (Figure 1, top) with 8x multi-sampling antialiasing.

Performance also depends on the impostor resolution. Higher resolutions lengthen the projections of the rays and increase the number of steps taken along each ray to find the projection, as shown in table 2 (output image resolution is 640x480).

Impostor resolution [pix]	128	256	512	1024
Average frame rate [fps]	103.3	83.4	42.78	28.9
Maximum ray projection length [pix]	72	164	346	640

Table 2: Performance dependence on impostor resolution for 2 teapot scene (Figure 1, top), with 8x MSAA.

For the graph camera non-pinhole impostor scene (Figures 4 and 5), the minimum, maximum, and average performance along the path shown in the video is 20, 42, and 26.8 frames per second, with 8x multi-sampling antialiasing, with a 640x480 output resolution, and with a 1920x1175 impostor resolution. The graph camera impostor for the maze with 4 bunnies (66Ktris total) is constructed at over 100 frames per second, which enables updating the impostor in real time.

6.3. Discussion

Our method renders high-quality specular reflections on complex, dynamic reflectors, with complex, dynamic reflected objects. Compared to projection techniques such as explosion maps [OR98], our method has the advantage of producing multiple projections of the same object at no extra cost and of handling complex reflectors. Compared to image-based rendering techniques, our method has the advantage of supporting dynamic scenes and of reduced memory requirements. Image-based rendering 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 close to the reflector, 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 the reflected geometry is not approximated.

7. Conclusions and future work

The fundamental reason for the efficiency of the construction and rendering of these non-pinhole impostors, is the fact that the underlying non-pinhole camera model provides fast projection. This enables fast feed-forward construction of the non-pinhole color and depth maps, as well as a one dimensional search for the intersection of a ray with the impostor.

There are several promising directions for future work. One is developing a robust mip-mapping technique for non-pinhole camera images. Under-sampling should not lead to aliasing but rather to blurriness. Subsequent research could target porting to non-pinholes other solutions to the under-sampling problem such as geometry enhanced textures. Such an approach will also improve the quality of the silhouettes. Whereas this work has dealt exclusively with specular materials, more complex reflective materials are possible leveraging the known distance from the reflector surface to the reflected object.

Our work argues for the practicality and benefits of abandoning the pinhole constraint. Non-pinhole camera models can be designed to optimally serve the application and data set at hand through powerful yet inexpensive impostors.

8. Acknowledgments

<withheld for double-blind review>

References

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

[BN76] BLIN J.F., NEWELL M. E. Texture and Reflection in Computer Generated Images. *CACM 19:10*, 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*, 105–116.

[DDS*03] DECORET X., DURAND F., SILLION F., AND DORSEY J. Billboard Clouds for Extreme Model Simplification. *Proceedings of SIGGRAPH 2003*, pp 689-696.

[GGS*96] GORTLER S., GRZESZCZUK R., SZELISKI R., COHEN M. The Lumigraph. *In Proceedings of SIGGRAPH 96*, pp 43-54.

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

[MS96] MACIEL P., AND SHIRLEY P. Visual Navigation of Large Environments Using Textured Clusters, *Symposium on Interactive 3D Graphics* (1995) pp 95-102.

[MMB97] MARK W., MCMILLAN L., AND BISHOP G.. Post-Rendering 3D Warping. *Proceedings of Symposium on Interactive 3D Graphics*, 1997.

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

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

[MPS05] MEI C., POPESCU V., AND SACKS E. The Occlusion Camera, *Computer Graphics Forum*, volume 24, issue 3, Eurographics 2005, pp 335-342.

[OR98] OFEK E., RAPPOPORT A. Interactive reflections on curved objects. *In Proc. of SIGGRAPH '98*, ACM Press, 333-342.

[POC05] POLICARPO F., OLIVEIRA M., AND COMBA J. Real-Time Relief Mapping on Arbitrary Polygonal Surfaces.

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

[PA06] POPESCU, V. AND D. ALIAGA. Depth Discontinuity Occlusion Camera, In *Proc. of ACM Symp.I3D and Gaming*, 2006.

[RB98] RADEMACHER P, AND BISHOP, G. 1998: Multiple-center-of-Projection Images. Proc. ACM SIGGRAPH '98 (1998)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 Proceedings of Eurographics Symposium on Rendering, 2007, pp 99-110.

[RPA08] ROSEN P., POPESCU V., AND ADAMO-VILLANI N. The Graph Camera. *Purdue University Technical Report TR-08-005*.

[RP08] ROSEN P., POPESCU V. The Epipolar Occlusion Camera, In *Proceedings of ACM Symposium on Interactive 3-D Graphics and Gaming*, 2008.

[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.N. ET AL. Multiperspective Panoramas for Cel Animation. *In Proceedings of ACM SIGGRAPH* '97 (1997) pp 243-250.

[Wym05] Wyman C. An Approximate Image-Space Approach for Interactive Refraction. *In ACM Transactions on Graphics*, volume 24, number 3, pp 1050-1053.

[YM04] YU J. AND MCMILLAN L. A Framework for Multiperspective Rendering. *In Proceedings of Eurographics Symposium on Rendering (EGSR)*, 2004.