Graduate School ETD Form 9
(Revised 12/07)

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Paul Andrew Rosen

Entitled

Improved 3-D Scene Sampling by Camera Model Design

For the degree of Doctor of Philosophy

Is approved by the final examining committee:

Voicu Popescu
Chair

Christoph Hoffmann

Daniel Aliaga

Xavier Tricoche

To the best of my knowledge and as understood by the student in the *Research Integrity and Copyright Disclaimer (Graduate School Form 20)*, this thesis/dissertation adheres to the provisions of Purdue University's "Policy on Integrity in Research" and the use of copyrighted material.

Approved by Major Professor(s): Voicu Popescu

Christoph Hoffmann

Approved by: William J. Gorman 7/7/2010
Head of the Graduate Program Date

Graduate School Form 20
(Revised 1/10)

PURDUE UNIVERSITY
GRADUATE SCHOOL

Research Integrity and Copyright Disclaimer

Title of Thesis/Dissertation:

Improved 3-D Scene Sampling by Camera Model Design

For the degree of Doctor of Philosophy

I certify that in the preparation of this thesis, I have observed the provisions of *Purdue University Teaching, Research, and Outreach Policy on Research Misconduct (VIII.3.1)*, October 1, 2008.*

Further, I certify that this work is free of plagiarism and all materials appearing in this thesis/dissertation have been properly quoted and attributed.

I certify that all copyrighted material incorporated into this thesis/dissertation is in compliance with the United States' copyright law and that I have received written permission from the copyright owners for my use of their work, which is beyond the scope of the law. I agree to indemnify and save harmless Purdue University from any and all claims that may be asserted or that may arise from any copyright violation.

Paul A. Rosen

Printed Name and Signature of Candidate

July 2, 2010

Date (month/day/year)

*Located at http://www.purdue.edu/policies/pages/teach_res_outreach/viii_3_1.html

IMPROVED 3-D SCENE SAMPLING BY CAMERA MODEL DESIGN

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Paul Andrew Rosen

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

August 2010

Purdue University

West Lafayette, Indiana

ACKNOWLEDGMENTS

I owe tremendous gratitude to my advisors Voicu Popescu and Chris Hoffmann. Their guidance has been invaluable to me. Without their help, I am certain I would not have ended up as successful as I have been. I thank them both for all of the time and energy they have given to me. I would like to thank the other members of my committee as well, Daniel Aliaga and Xavier Tricoche. They too have been invaluable mentors to me.

I would also like to thank all of those folks with whom I have coauthored papers and collaborated on projects. In particular, I would like to thank Mete Sozen, Ayhan Irfanoglu, Ingo Brachmann, Zygmunt Pizlo, Rahul Sathe, Nicoletta Adamo-Villani, Jian Cui, Chris Wyman, and Kyle Hayward. I would additionally thank all of my friends and colleagues in the Computer Graphics and Visualization Lab and throughout the Computer Science Department.

Finally, I would like to thank my family for their love and support throughout this arduous process.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

LIST OF ABBREVIATIONS

3-D – Three dimensional

COP – Center-of-projection

DDOC – Depth discontinuity occlusion camera

DI – Depth image

EOC – Epipolar occlusion camera

GC – Graph camera

GPC – General pinhole camera

IBR – Image-based rendering

LDI – Layered depth image

LoD – Level of Detail

MPI – Multiperspective imaging

OCRI – Occlusion camera reference image

PPC – Planar pinhole camera

RPPC – Reverse planar pinhole camera

SPOC – Single pole occlusion camera

XAA – Extreme antialiasing

ABSTRACT

Rosen, Paul Andrew. Ph.D., Purdue University, August, 2010. Improved 3-D Scene Sampling by Camera Model Design. Major Professors: Voicu Popescu and Christoph Hoffmann.

Images are one of the most powerful means of communication available. They are pervasive throughout our lives and are the central focus of computer graphics, visualization, and computer vision. Most images, synthetic or photographic, are created using the planar pinhole camera model. This classic camera model has important advantages including its simplicity, enabling efficient hardware and software implementations, and its similarity to human vision, yielding images familiar to users. However, the planar pinhole camera model suffers from important limitations including sampling from a single viewpoint and requiring a uniform sampling rate along the image plane. These limitations result in problems with occlusions, when no direct line-of-sight exists to the viewpoint, and sampling rates which do not correlate well to the complexity of 3-D data.

This dissertation proposes a new paradigm of problem solving, dubbed Camera Model Design, which overcomes the limitations of the planar pinhole camera model to address many problems which still exist in computer graphics, visualization, and computer vision. The Camera Model Design paradigm stresses four important ideas. First, relax the constraints of the planar pinhole camera model allowing generalized camera rays which are no longer straight and no longer converge. This facilitates camera models that overcome occlusions and have variable sampling rates. Second, the choice of camera used for a

particular application need not be limited to the planar pinhole camera. Instead, a new camera should be designed to directly address the needs of the application. Third, camera models should no longer be static. Instead they should dynamically adapt to the 3-D data they are sampling. Finally, in order to support interactive exploration, a high level of computational efficiency should be maintained.

We introduce three new families of camera model, the occlusion cameras, the graph cameras, and the general pinhole cameras, all of which address one or more of the planar pinhole camera limitations. These new camera models are applied to a wide variety of applications which demonstrate the benefits of Camera Model Design for increasing computational efficiency, improving output image quality, and enhancing user performance in exploring 3-D datasets.

CHAPTER 1. INTRODUCTION

1.1. Motivation

Images are ubiquitous in computer graphics, visualization, and computer vision and pervasive throughout our daily lives. Images serve a number of important roles, the most obvious of which is for directly conveying information to a user through their eyes. The eye is arguably the most powerful sensory organ available to humans making images an important means by which we deliver information to people. As such, we are bombarded by images constantly throughout the day. These images come in many forms including visualization of complex scientific phenomena, photographs, and entertainment images from television, movies, and video games.

Aside from the use of images for direct consumption by a user, images are also used as a powerful primitive in computer graphics. In particular, they serve as an inexpensive and flexible intermediate representation of the color and geometry in a 3-D scene. These intermediate representations are used in support of many rendering algorithms. For example, texture, bump, and displacement maps are essentially images that enable inexpensive methods for significantly enhancing a scene's appearance without increasing geometric complexity. By assuming distant geometry, environment maps serve as an approximation of a scene which makes interactive rendering of reflections and refractions tractable albeit inaccurate. When enhanced with depth, images can also enable a wide variety of applications such as shadow mapping, accurate reflection and refraction, relief texture mapping, and ambient occlusion. The use of images for these

approaches enables efficient and interactive render of all of these techniques which might otherwise be impossible.

Tremendous strides have been made in computer graphics and visualization towards improving the visual quality of images. In computer graphics, the realism achieved by the latest approaches is phenomenal. Incredible strides have been made in rendering photorealistic images of synthetic 3-D worlds by improving geometric modeling techniques and by accurately modeling light transport. Visualization as a field has developed with the goal of extracting salient features from various complex data sets. The techniques used in both of these fields generally rely upon the use of the planar pinhole camera model as the method for reducing 3-D scene data into 2-D image data.

The planar pinhole camera model is widely used because of its ability to reduce complex 3-D scenes into images in an efficient, organized, and meaningful way. The planar pinhole camera accomplishes this by closely mimicking the construction of the human eye. This results in images familiar to users and enables efficient physical, hardware, and software implementations.

There are, however, several limitations placed upon the planar pinhole camera. These limitations come in the form of limited field-of-view, a single viewpoint requiring direct line-of-sight, and uniform sampling. These limitations significantly restrict the quantity of information which can be captured into a single image.

1.2. The Planar Pinhole Camera Model

Construction of the planar pinhole camera model is quite simple and its similarity to the human eye is critical to its popularity. Figure 1.1 shows a comparison between a simplified human eye model, a physical planar pinhole camera, and the mathematical planar pinhole camera.

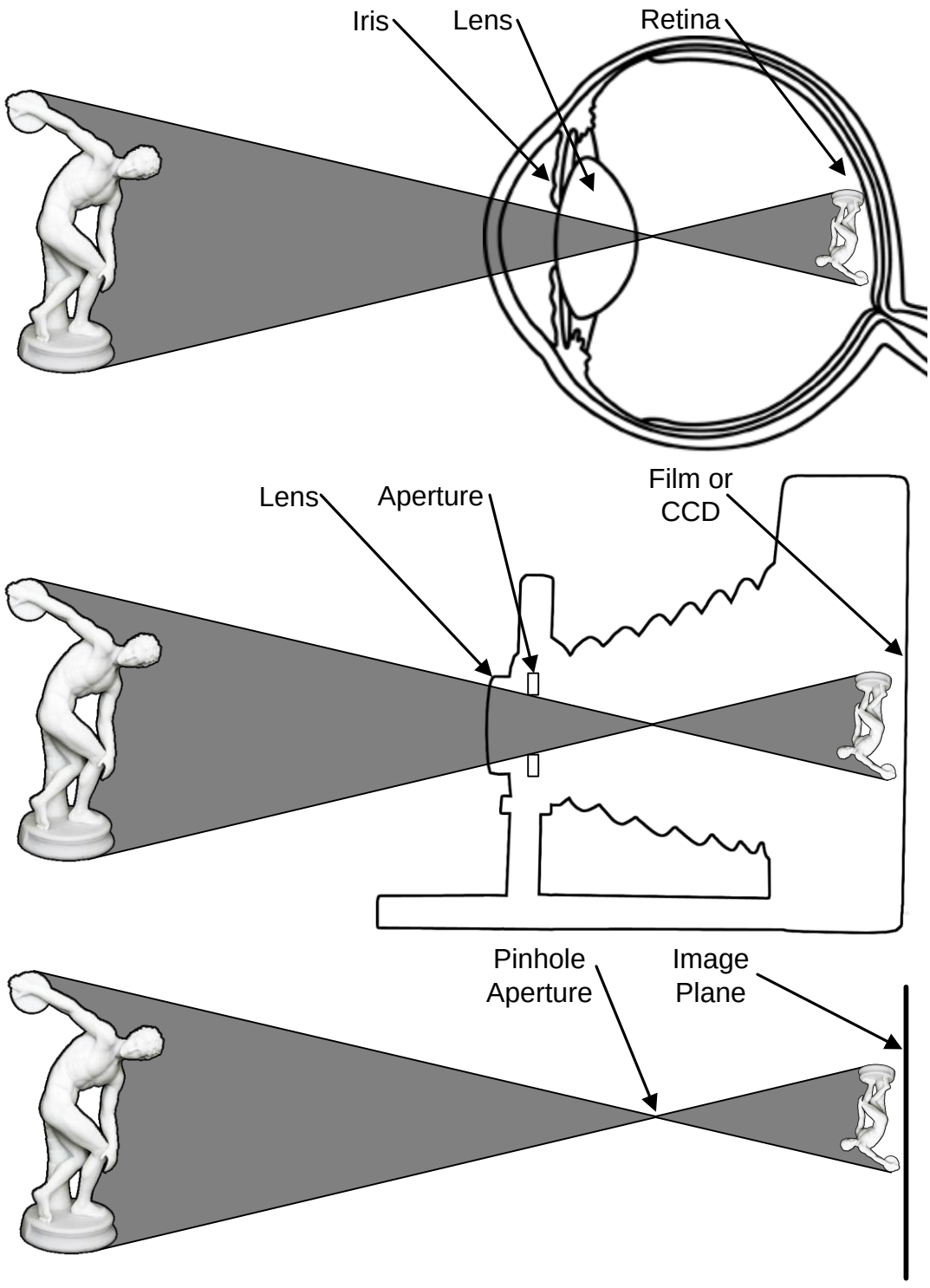


Figure 1.1. The planar pinhole camera model. A comparison of the construction of the human eye (*top*), a physical camera implementation (*middle*), and the planar pinhole camera model (*bottom*). Figure adapted from that of Short [144].

For the human eye and physical planar pinhole camera implementations there is a one-to-one match between the three key components used to construct both – the lens, aperture, and imaging surface. Light from the world enters the camera or eye first through the lens. The purpose of the lens is to focus the light such that objects at a certain distance from the lens are not blurry. This effect is known as depth-of-field and defines distance from the camera that will be in or out of focus. The aperture or the iris in the case of the human eye is used to control the amount of light which enters the camera and ultimately reaches the imaging surface. Finally, the imaging surface collects the light for the final image. For physical cameras, the imaging surface is generally flat and made up of film or a CCD (charge-coupled device). In the case of the human eye, the imaging surface is the retina which is rounded to the shape of the eye.

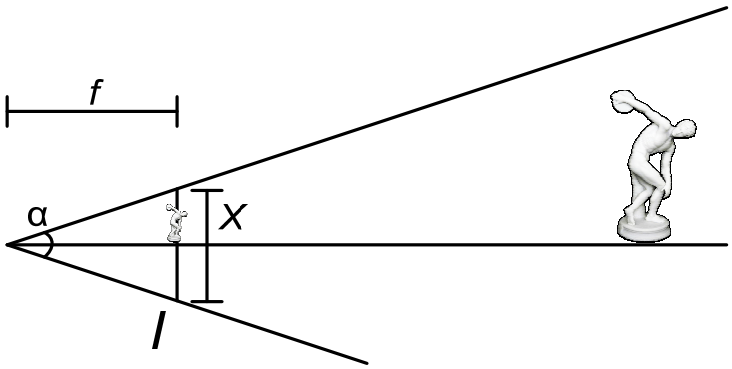


Figure 1.2. The construction of a 2-D perspective projection using the planar pinhole camera model.

The mathematical planar pinhole camera construction is similar to the human eye and physical planar pinhole camera constructions with a few key simplifications. Like with the human eye and physical camera, the planar pinhole camera model has an imaging surface for collecting the final image. In the planar pinhole camera model, this imaging surface is planar and referred to as the *image plane*. The image plane is subdivided into individual, usually square, elements referred to as *pixels* which contain color, depth, or other data. The key simplification of

the planar pinhole camera is in the use of a pinhole aperture, which is infinitely small, at the point where all of the camera rays converge. This point is generally referred to as the *center-of-projection* (COP). The pinhole aperture has the powerful property of infinite depth-of-field, further simplifying the camera by removing the need for any lenses.

The pinhole aperture simplification in the planar pinhole camera model is quite powerful because it enables inexpensive projection and rendering. The projection of a point in 3-D camera-space to 2-D image-space can be made a linear equation by use of the homogeneous coordinate system for an input point (x_i, y_i, z_i) and a camera with focal distance f as seen in Equation 1.1. Once vertices are projected into homogeneous coordinate space where lines in 3-D space remain lines in 2-D space, triangles can be quickly scan converted, or *rasterized*, into individual pixels. In homogeneous coordinate space, the color, texture, and other parameters vary linearly and can be interpolated in a perspective correct manner [111].

$$\begin{bmatrix} x/w \\ y/w \\ 1 \end{bmatrix} w = \begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = \frac{f}{z_i} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \qquad \text{Equation 1.1}$$

This compact projection and rasterization enables efficient hardware and software implementations that can render complex scenes at interactive rates. However, the planar pinhole camera model also suffers from three significant limitations that reduce the quantity of data it can encode.

(1) The field-of-view for the planar pinhole camera is limited, preventing panoramic views of a scene. Figure 1.2 shows an example planar pinhole camera construction. The size x of image plane I with focal distance f can be calculated using Equation 1.2. As the field-of-view α approaches 180°, the ratio x/f approaches ∞. This means that the image plane must be infinitely large or

the focal distance infinitely small to accommodate a field-of-view approaching 180°. This planar pinhole camera limitation has been addressed by others' work though innovations such as cube maps which give complete panoramas by combining six planar pinhole cameras each with a 90° field-of-view or spherical maps which allow the image plane to be spherical giving panoramic views. Further details of these approaches will be discussed in Chapter 2.

$$\frac{x}{f} = 2 \tan\left(\frac{\alpha}{2}\right) \qquad \text{Equation 1.2}$$

(2) By design, the rays of the planar pinhole camera are always straight, converge at a single point (the center-of-projection), and only collect a single sample. Relatively little has been done to remove all of these requirements, which, for example, limit shadow maps to hard shadows and lower scene approximation quality due to disocclusion errors.

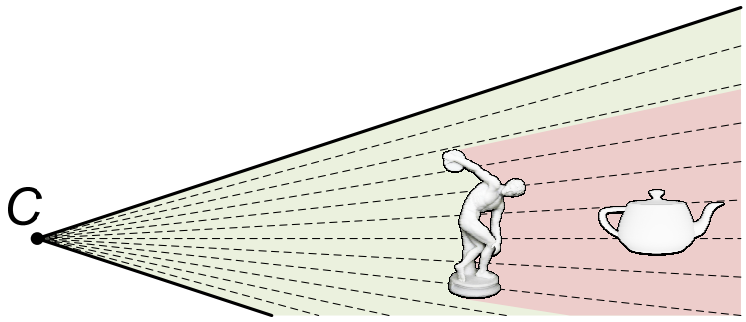


Figure 1.3. Illustration of occlusion limitation of the planar pinhole camera.

This means that occlusions within a scene might prevent important objects from being sampled. Figure 1.3 shows an illustration of the problems caused by occlusion within a scene. Here, all of the rays pass through the center-of-projection C and collect the first sample visible along each ray. The result is that rays capture samples of the disc thrower statue, while the teapot remains completely unsampled. Here, the region in green is sampled while the area in red remains occluded.

The occlusions created by the planar pinhole camera model are not just limited to inter-object occlusions. Complex object often self-occlude as well.

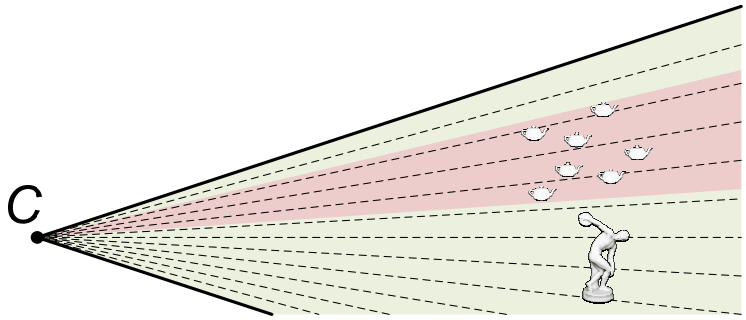


Figure 1.4. Illustration of sampling limitation of the planar pinhole camera.

(3) The pixels of the planar pinhole camera are always the same size and shape across the image plane. The camera rays associated with these pixels therefore sample in a spatially uniform manner[1]. This uniform sampling of the image plane precludes adapting the sampling rate according to the importance of individual regions or data subsets. There are many scenarios in which this uniform sampling is limiting. For example, when visualizing large data sets, one might want to locally zoom while maintaining a global context, a so called focus-plus-context visualization. Additionally, applications such as shadow mapping can benefit from non-uniform sampling where regions of high importance, such as edges, would receive additional samples.

In Figure 1.4 the rays of the planar pinhole camera sample the scene in a spatially uniform manner. In this example, the regions in red contain much

---

[1] For any given plane in space parallel to the image plane, the intersection of any neighboring rays of the planar pinhole camera and the plane are equidistant to one another. A distinction is required between this and the angular uniform sampling used in techniques such as cylindrical projections where the angle between neighboring rays is constant across the entire set.

higher frequency data than the regions in green. This means that either the rays in the red region are undersampling the data or the rays in the green region are wasted, oversampling low frequency data. For many applications, ideally fewer rays would be used in the green regions in favor of using more rays in the red region.

There are many applications where one or more of the planar pinhole camera limitations are a severe hindrance.

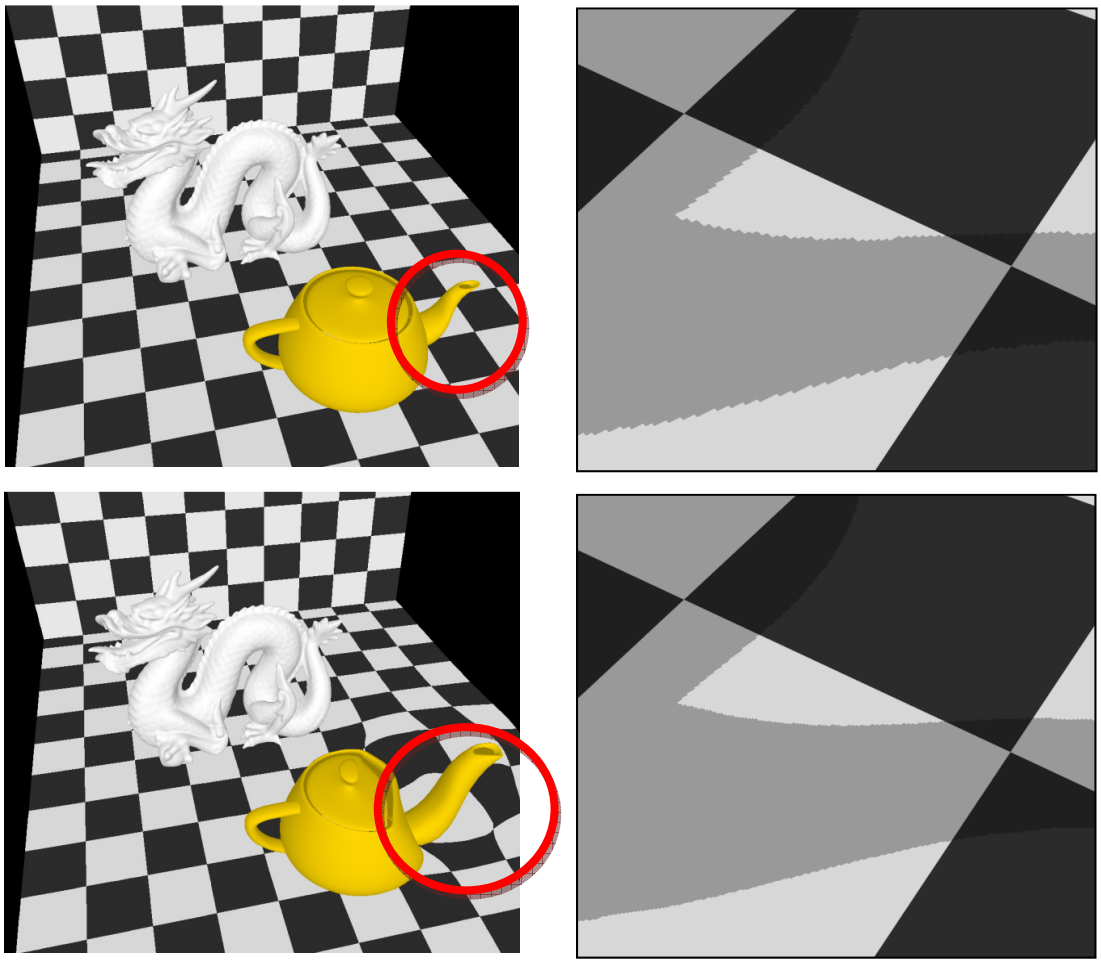


Figure 1.5. Shadow map aliasing example. Conventional shadow mapping (*top*) uses the map (*left*) to shadow a novel view (*right*) compared to a non-uniform shadow mapping (*bottom*) where less aliasing occurs on shadow borders.

The use of images as primitives in computer graphics applications, particularly in interactive graphics, for efficiently computing high-quality visual effects is wide spread. These intermediate images serve as high-quality geometry approximations which are generated primarily using the planar pinhole camera model. An application such as shadow mapping is a perfect example of where the limitations of the planar pinhole camera come into play. In shadow mapping, an image is rendered from the point of view of a light source. That rendered image can then be used to determine if a pixel is in or out of shadow within the desired view. This construction assumes that the light source is an infinitely small point in which case, shadowing is a binary decision, in or out of shadow. In reality, light sources have non-zero area resulting in penumbra regions. In addition, these shadow maps use uniform sampling. The quality of hard shadows is often measured by the quality of the boundary between regions in and out of shadow. Ideally, these regions would receive more samples compared to the other parts of the shadow map (Figure 1.5).

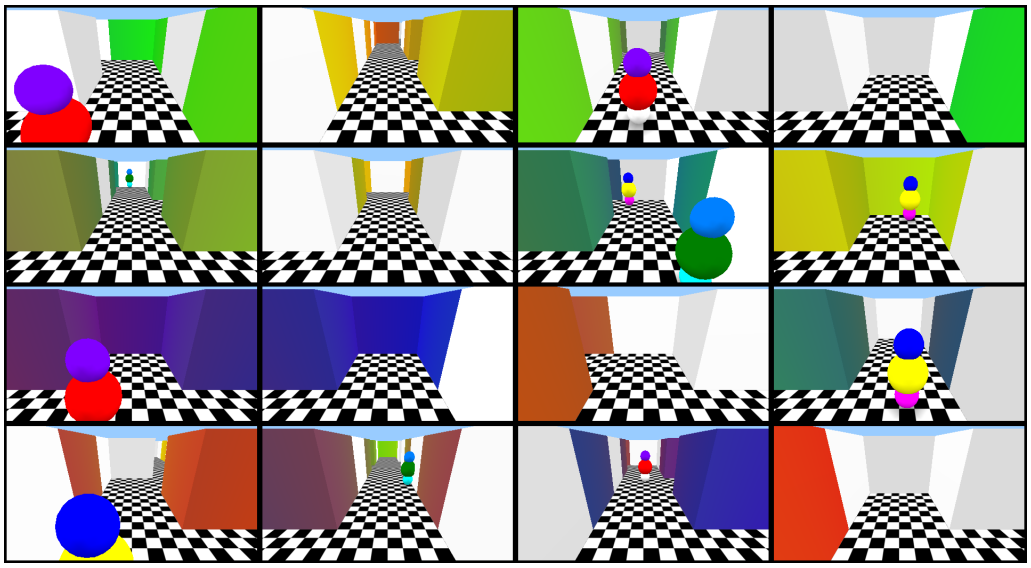


Figure 1.6. A comprehensive matrix of planar pinhole cameras. The comprehensive visualization of a 3-D space with a planar pinhole camera requires many redundant and discontinuous views of the scene.

Visualizing a large spatial data set can be challenging with a planar pinhole camera. It is rare for data sets to contain data that is uniformly interesting or important. In fact, many data sets contain structures which are highly important that are often occluded by less important structures. When trying to identify these interesting structures hidden behind occluding objects, a planar pinhole camera must be navigated around the occluding structures of the data set. This navigation can often be difficult and unintuitive. It is also the case that these data sets often have regions of interest that exist at widely different scales (i.e. globally interesting structures in addition to locally interesting structures at potentially many intermediate frequencies). The uniform sampling of the planar pinhole camera precludes the simultaneous monitoring of these multi-scale structures at adequate rates without using multiple planar pinhole cameras simultaneously.

A commonly addressed problem in computer vision involves the monitoring of complex 3-D spaces with multiple cameras for the purpose of security and situational awareness. The conventional approach to monitoring these spaces involves using tens of cameras and surrounding a user by a large collection of video monitors. The user is then expected to watch those tens of video feeds simultaneously with the goal, for example, of identifying suspicious individuals and keeping track of their activities. As can be noted in Figure 1.6, this scenario presents three significant problems. In order to adequately cover a space, many cameras will need to be used and there is the possibility of significant areas of overlap between views. Also, tracking an individual as they move through the world can be challenging as they switch between different camera views. Finally, requiring a user to track multiple monitors is inefficient since they are only truly able to watch a single monitor at any given moment.

These are just a small sampling of the scenarios and problem which the planar pinhole camera presents serious limitations. In order to address the limitations of the planar pinhole camera, we introduce Camera Model Design, a new paradigm

of problem solving which relaxes the constraints of planar pinhole camera model to address problems in computer graphics, visualization, and computer vision.

1.3. Camera Model Design

The use and acceptance of the planar pinhole camera model for projection and rasterization is so pervasive that its limitations are often ignored or accepted as a fact of life. I believe that any apprehension about removing the planar pinhole camera constraints is simply unjustified.

I advocate a new problem solving paradigm for computer graphics, visualization, and computer vision called *Camera Model Design*. The Camera Model Design paradigm advocates no longer limiting ourselves to the planar pinhole camera model. Instead, we design camera models that best suit a given application and optimize the camera models dynamically according to the data currently being sampled. The ultimate goal is to maximize the quantity of visual information contained within an image, at the same time maintaining a high level of computational efficiency.

My thesis therefore is that:

> *In order to create images that better sample 3-D scenes, I propose abandoning the constraints of the conventional planar pinhole camera model by no longer requiring that rays be straight, converge, or sample space uniformly. Camera models can then be designed for specific applications and optimized dynamically for each 3-D scene or dataset so as to achieve adequate sampling. At the same time, camera models should also be designed to preserve image computation efficiency in order to support interactive rendering of dynamic scenes.*

There are four principles that Camera Model Design advocates for on a general basis. Compliance with all four principles is by no means required or possible in all situations.

(1) The shape and structure of camera rays in the planar pinhole camera is one of its most significant restricting factors. Therefore, removing key planar pinhole camera constraints is the first principle of camera model design. This includes no longer requiring that camera rays[2] converge, be straight, or sample space uniformly. The result is that camera models become flexible entities that can take up any of a virtually unlimited number of configurations to overcome occlusions and sampling limitations.

(2) Currently, the choice of camera models available to applications is quite limited, using variations of either orthographic projections or the planar pinhole camera. The reality is that not all applications need the same type of information to be gathered into an image and these rigid camera models do not account for the potential needs of individual specific applications. While the only adjustable parameter of the planar pinhole camera remains the field-of-view, Camera Model Design instead advocates developing unique cameras models whose properties address the needs of individual applications.

(3) Once a camera model has been designed to satisfy the needs of a particular application, the camera model does not need to remain rigid throughout its use in the application. Instead, the camera should be designed in a data-centric manner, dynamically adapting according to changes in viewpoint, changes in the

---

[2] The term camera rays will be used loosely as the camera rays in most of our approaches are no longer mathematical rays. Instead a camera ray is defined as the locus of 3-D points which project to the same output image plane location.

data captured, or changes in other user selected parameters (such as regions or objects of interest).

(4) Any collection of camera rays can be raytraced and combined into an image. The idea of raytracing implies efficient ray calculation and efficient ray-primitive intersection. Despite many advances in real-time raytracing, the conventional feed-forward graphics pipeline remains the approach of choice for interactive rendering. In order to target large data sets and interactive applications, camera models should be designed to maximize computational efficiency. Doing this requires rendering as much within the standard feed-forward graphics pipeline as possible. This requires designing camera models that have efficient point projection functions which produce images that are mostly coherent.

By adhering to the four principles presented, Camera Model Design represents a flexible framework for generating images with multiple viewpoints and variable sampling rates. The images generated have a single layer, they are mostly continuous and non-redundant, and they can be computed efficiently with the help of graphics hardware.

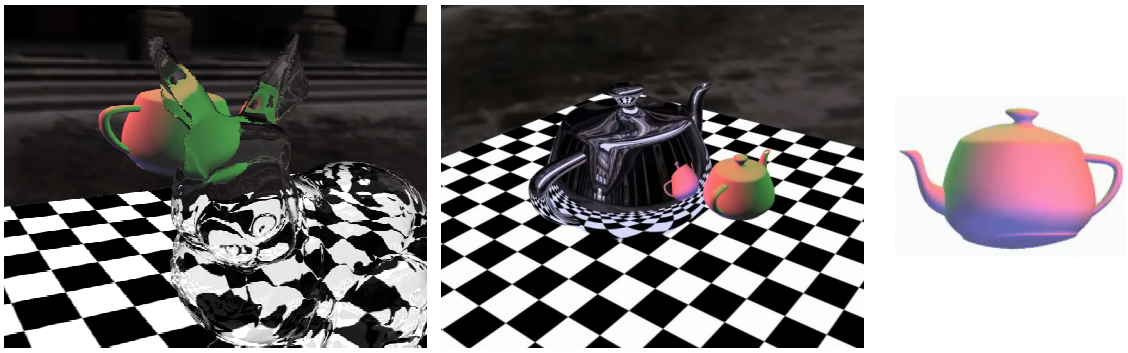


Figure 1.7. Reflection and refraction example. Accurate refractions (*left*) and reflections (*middle*) calculated using a single pole occlusion camera image (*right*).

1.4. Overview of Results

Thus far, three classes of camera have been developed based on the Camera Model Design paradigm. Each class of camera is designed specifically to address one or more of the limitations of the planar pinhole camera. As such they can also be used to address many of the scenarios presented in Section 1.2.

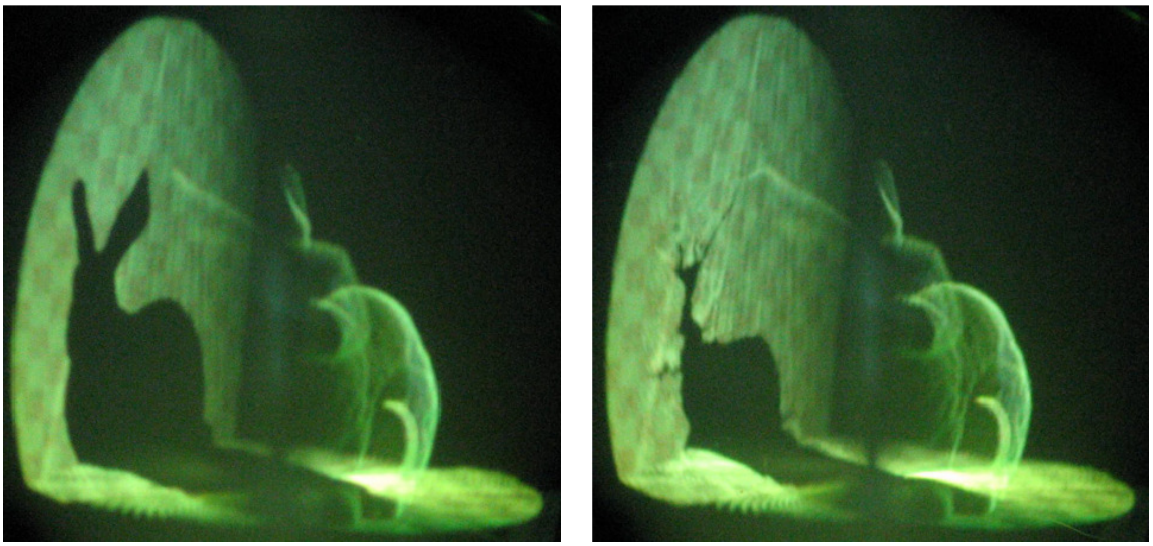


Figure 1.8. Occlusion camera application to volumetric display acceleration. Planar pinhole camera (*left*) and depth discontinuity occlusion camera (*right*) images used to replace geometry in three-dimensional display rendering.

1.4.1. The Occlusion Camera Family

Occlusion cameras have been designed to reduce occlusion locally around foreground objects. There are three camera models which belong to this class: the single pole occlusion camera, the depth discontinuity occlusion camera, and the epipolar occlusion camera. All of these cameras work in similar ways by sending rays around the silhouette of objects in order to capture samples which are barely hidden from the current viewpoint. They differ in the method by which they accomplish this goal. The single pole occlusion camera simply bends the

camera rays radially around a user defined pole through the scene. The depth discontinuity occlusion camera detects and bends rays around the silhouettes of objects. Neither the single pole occlusion camera nor the depth discontinuity occlusion camera has any mechanism for determining the valid set of reconstruction viewpoints a priori. The epipolar occlusion camera detects object silhouettes along epipolar lines, bending the rays around them and guaranteeing a set of viewpoints which will have valid reconstructions.

Figure 1.9. Occlusion camera application to image compression. *Top:* Occlusion camera image (*left*) used to compress a single image (*right*) from the sea of images. The occlusion camera image and a planar pinhole camera image (*not shown*) are warped to the output viewpoint location (*middle*) and a residual is calculated (*bottom*).

There are many applications for the occlusion cameras. For example, by replacing scene geometry with image-based approximations, high-quality rendering effects such as reflections and refractions can be calculated very efficiently (Figure 1.7). Using occlusion cameras to produce the image-based

approximations instead of the planar pinhole camera, results in higher visual quality at virtually no additional computational cost.

Figure 1.10. Graph camera continuity illustration. The graph camera (*top*) produces has an abrupt change of perspective while the curved ray camera (*bottom*) minimizes the local distortion such that it is almost not visible.

Rendering geometry on a three-dimensional display is a computationally expensive process. Variable scene complexity results in a wide variance of scene rendering time. Images can serve as a high-quality level-of-detail approximation of the original scene geometry. By using images to replace the original geometry, the render time of the device becomes constant based upon the resolution of the geometry image. Using occlusion camera images as opposed to planar pinhole camera images provides better scene coverage at identical costs (Figure 1.8).

A final application of occlusion cameras is in image compression of a large collection (a “sea”) of planar pinhole camera images. A sea of images normally

contains hundreds or even thousands of images, with a high degree of redundancy. Here, a small collection of representative images, either planar pinhole camera or occlusion camera images can be gathered. The representative image data can be transformed, or warped, from its original viewpoint to the viewpoints of many images from the complete sea of images (Figure 1.9). Once warped, residuals can be calculated and stored allowing error free reconstructions. The occlusion camera provides a better method of encoding the inter-image redundancies reducing storage requirements for the data set.

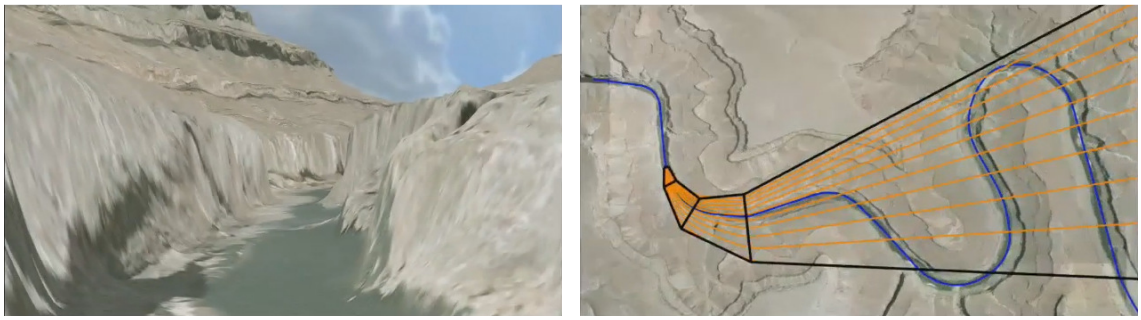


Figure 1.11. A curved ray camera used to linearize a path. A curved ray camera (*left*) allows forward and backward navigation along a user defined path (*right*).

1.4.2. The Graph Camera Family

While the occlusion camera family was designed to alleviate local occlusions, the graph camera family was designed to alleviate occlusions globally within the 3-D scene, facilitating comprehensive visualization of the entire scene. The graph camera functions by taking a planar pinhole camera frustum and repeatedly applying bending, splitting, or merging operations to the frustum. The result is a graph of planar pinhole cameras. As seen in Figure 1.10, the original graph camera produces images with piecewise linear rays resulting in C^0 continuity across changing perspectives. The abrupt changes in perspective produce distortions for objects that cross from one perspective to the next. The curved

ray camera was developed to minimize these distortions by using rays which are conic sections for transition regions. This results in rays which are C^1 continuous while still permitting efficient projection and rasterization.

One of the more important applications of the graph camera is enhanced navigation through 3-D scenes by allowing users to preview regions of the scene not visible from the current viewpoint. The graph camera introduces the idea of a continuum of navigation methods. The most basic form allows for unrestricted navigation by simply enhancing a planar pinhole camera with portals that preview upcoming occluded space. More restrictive navigation only allows the user to move forward or backward along a predefined path through the space (Figure 1.11). Finally, a comprehensive graph camera can be created which allows a user to monitor an entire space in a single image requiring no navigation at all.

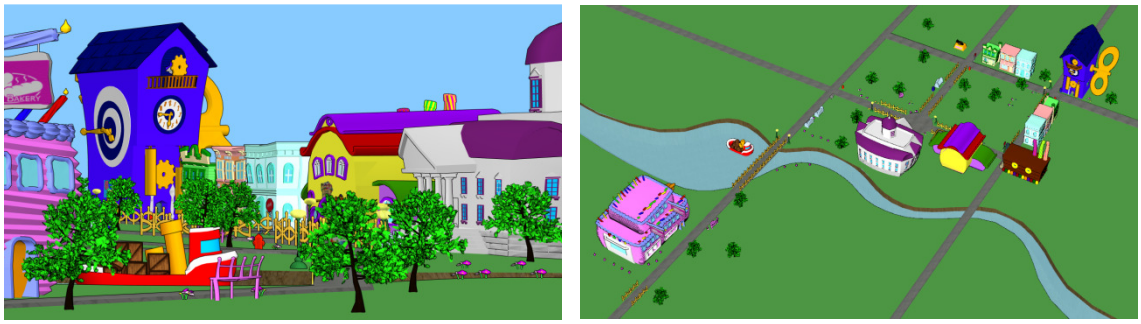


Figure 1.12. Graph camera application to scene summarization. The graph camera image (*left*) summarizes the 3-D scene (*right*).

Another application of the graph camera involves summarizing a 3-D scene into a single image. Artists and animators often wish to create posters that highlight important features of a 3-D scene they have created. The standard approach to do this either requires matting and blending multiple planar pinhole camera images of the scene or remodeling the scene in such a meaningful way that a single planar pinhole camera image summarizes it. Both of these processes can take many hours to complete. The graph camera provides multiple methods to

interactively generate and render posters reducing the construction time to only a few minutes (Figure 1.12).

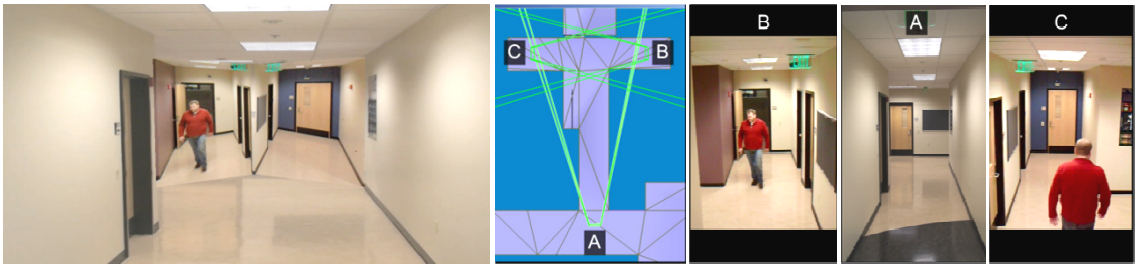


Figure 1.13. A real-world graph camera. The graph camera (*left*) can be used to integrate the 3 views (*right*) of the t-shaped hallway.

A final application of the graph camera is for enhanced surveillance of real-world spaces. Traditional surveillance involves simultaneously monitoring tens of planar pinhole camera video feeds which can be redundant and discontinuous. The redundancy or overlap of multiple views can make it difficult to perform tasks like counting the number of objects visible. The discontinuity of views can make it difficult to track an object as it transitions between various views. Using the graph camera, those planar pinhole camera video feeds can be combined into a single graph camera video feed (Figure 1.13). The graph camera image removes most of not all discontinuities and redundancies from the input images.

1.4.3. The General Pinhole Camera Family

The occlusion camera and graph camera families where developed specifically to address the single viewpoint limitation of the planar pinhole camera. The general pinhole camera family was developed to instead address the uniform sampling limitation. This is done by sampling the image plane of a planar pinhole camera in a non-uniform manner capturing regions of high interest with higher sampling rates than the rest of the image. The general pinhole camera is built upon the

planar pinhole camera allowing it to work on many types of data including geometry, volume data, and image data.

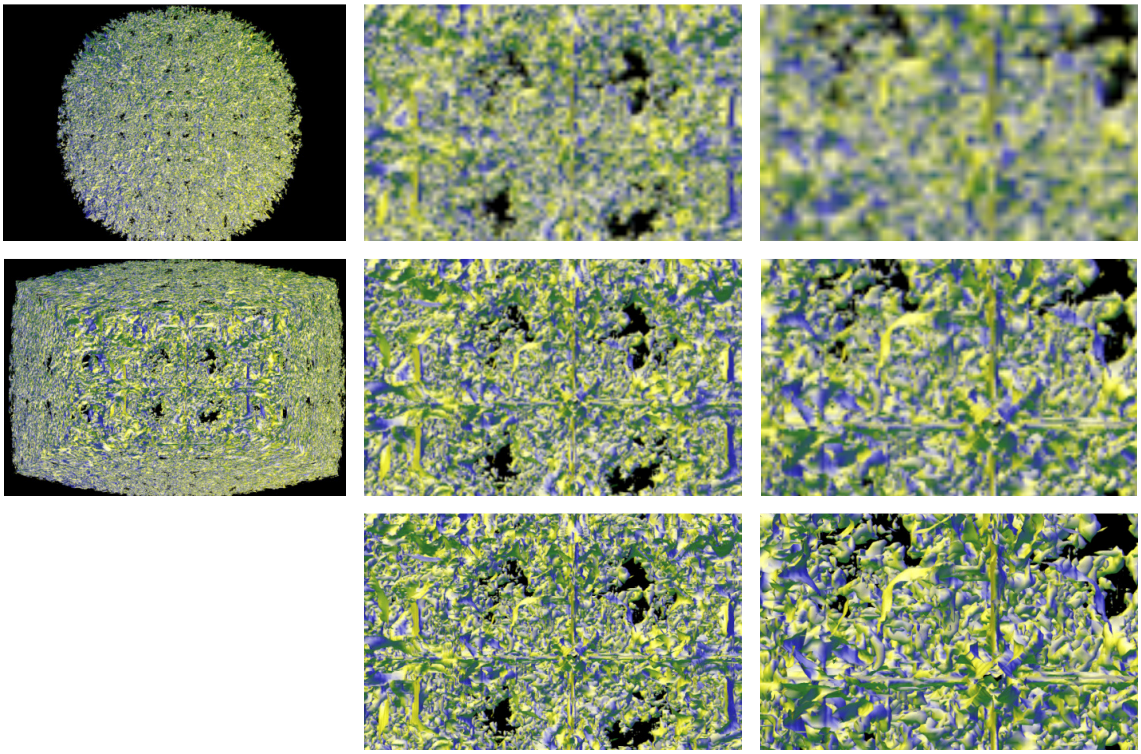


Figure 1.14. Remote visualization application of the general pinhole camera. *Top:* a planar pinhole camera image (*left*) used for remote visualization and two resampled output images (*middle* and *right*). *Middle*: a general pinhole camera image and corresponding frames. *Bottom*: ground truth output frames rendered from original scene geometry.

One powerful application of the general pinhole camera is in remote visualization (Figure 1.14). Here, the server will render a general pinhole camera image at a variable sampling rate. The server will transfer it to the client. The client will then use the general pinhole camera image to reconstruct a planar pinhole camera image of the scene. Since the general pinhole camera encodes more data than is needed for a single view, the user can zoom and rotate the view, producing high-quality output, without requesting additional data from the server. When

enhanced with depth, the general pinhole camera image can also support viewpoint translation at the client through 3-D warping.

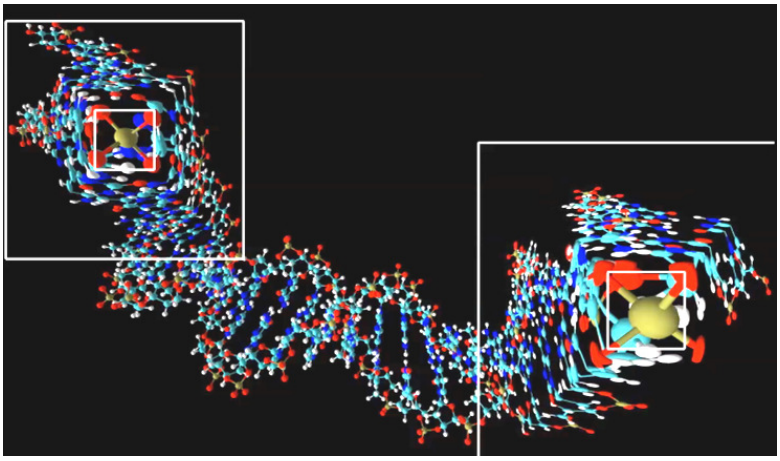


Figure 1.15. General pinhole camera used for a focus-plus-context visualization application. The general pinhole camera is used for focus-plus-context visualization of two separate regions with 3x and 6x magnification respectively.

A common problem in visualization of large-scale data sets stems from the fact that users often want to zoom in on a particular piece of data while still maintaining a sense of the global context, a so called focus-plus-context visualization (Figure 1.15). The general pinhole camera naturally supports such visualization allowing the user to select important regions, zoom in on them, and still monitor the global context as well.

There are instances of data sets which contain a few small discrete regions of very high-frequency data (Figure 1.16). Rendering these regions using conventional hardware antialiasing can result in shimmering effects. The general pinhole camera can be used to perform extreme antialiasing on these discrete regions. This is done by rendering the high-frequency data in high resolution off-screen buffers all but eliminating the shimmering effect.

Figure 1.16. General pinhole camera extreme antialiasing application. The general pinhole camera is used for 361x extreme antialiasing (*left*) compared to conventional 16x antialiasing (*right*). Images magnified 5 times are shown for illustrative purposes.

1.5. Organization

The remained of this dissertation is organized as follows. Chapter 2 will discuss work related to Camera Model Design. Chapters 3, 4, and 5 will introduce the 3 families of camera models designed thus far and their applications. These include: the occlusion camera family, the graph camera family, and the general pinhole camera family. Chapter 7 discusses the challenges of rendering camera models with non-linear projection. Chapter 8 takes a look at the potential perceptual benefits of Camera Model Design. Finally, Chapter 9 will conclude the work and discuss future directions.

CHAPTER 2. PREVIOUS WORK

There exists no previous work which is directly parallel to that of Camera Model Design. There are, however, a large number of topics worthy of discussion that are related to improving the sampling of a 3-D scene within images for the purpose of addressing occlusions or sampling in a non-uniform manner. These approaches can be broken down into three primary groups: planar pinhole camera based approaches; non-pinhole camera based approaches; and model modification based approaches.

2.1. Planar Pinhole Camera Approaches

2.1.1. Depth Enhanced Images

Many early image-based attempts at trying to more effectively capture 3-D scenes focused primarily on using planar pinhole camera images enhanced with per pixel depth [98, 100, 102]. When attempting to render a scene from a novel viewpoint, one of these image-based representations located nearby the novel viewpoint is warped to the location of the novel viewpoint to synthesize a new view. In most cases, one depth image is insufficient for addressing the disocclusion errors that exist within complex 3-D scenes.

To address the disocclusion errors, additional nearby depth images can be warped to the same novel viewpoint and combined with the original synthesized view. Brute force techniques [98] simply warp n nearby images and hope that all disocclusion errors are removed. Other techniques, such as the vacuum buffer

[129], identify subvolumes of the view frustum which are potentially missing samples, allowing for conservative avoidance of disocclusion errors. Such an approach still requires that one iterate over many depth images until all subvolumes of the desired view frustum are filled.

Unfortunately, even with a scene densely sampled by reference images, removing all disocclusion errors might require warping an unbounded number of reference images to the novel viewpoint. Most of those images contain large quantities of data which are redundant with neighboring depth images resulting in wasteful storage and computation.

To address the problems of redundancy, new scene representations were developed which capture more than one sample per ray. Multi-layered z-buffers [100], depth peeling [42, 163], layered depth images (LDIs) [142], and LDI trees [25] are all based on the planar pinhole camera model, but relax the constraint that each ray only captures one sample, by instead capturing multiple layers of the scene from a single viewpoint. While these approaches do reduce redundancies, all require significant additional costs in construction, storage, and rendering. This additional cost can be quite extensive based upon the number of layers used, often precluding the use of dynamic scenes. Determining the number of layers needed to conservatively sample a scene is also a difficult problem to solve in a reasonable amount of time. Instead, a fixed number of layers is generally used which does nothing to guarantee scene coverage and may lead to inefficiencies in regions which require fewer layers.

There have also been a number of approaches which relax the uniform sampling requirement of the planar pinhole camera in order to improve the quality of applications such as shadow mapping. These methods rely on multiple rendering passes [45], offline rendering [4], or the use of irregular data structures [70]. These approaches all produce high-quality or even perfect hard shadows,

but none of them represents an ideal solution to the problem given the currently available hardware.

2.1.2. Depth-Free Representations

The methods mentioned thus far assume a depth value is associated with each pixel of an image. Another class of representation methods model 3-D scenes by sampling a dense set of rays in space, removing the need for any depth to be stored.

The plenoptic function is a depth-free representation used for modeling [1, 102] and rendering [7] visual data for 7 degrees of visual freedom (3-D position, 2-D angle, 1-D time, and 1-D wavelength). A planar pinhole camera at a fixed position is a simple plenoptic capture device which encodes 2 degrees of angular freedom. Recording video, a planar pinhole camera also captures time making it a 3-D plenoptic capture device. The lightfield [63, 84] is a depth-free representation that encodes a 4-D (5-D when video is used) plenoptic function. Lightfields model a 3-D scene by combining data from a dense array of planar pinhole cameras. This technique models only a set of rays which sample the scene, never capturing any actual geometry. Similarly, the lumigraph [19, 53] is a quasi depth-free 4-D plenoptic function which uses a geometric proxy on top of a lightfield. A true 7-D plenoptic capture device will contain every ray potentially needed for reconstructing any novel viewpoint in space. These techniques do a good job capturing complex 3-D scenes, but their data is large, unstructured, and redundant.

2.2. Non-Pinhole Camera Approaches

2.2.1. Panoramas

There are several non-pinhole camera models which capture into a single image more than would be seen from any single viewpoint. Panoramas [26] are single-layer images which address the field-of-view limitation of the planar pinhole camera model by capturing samples in all possible directions from a single viewpoint. When enhanced with depth [10], a small amount of translation from the center-of-projection is possible. Mosaic techniques [152] relax the single viewpoint requirement of panoramas, but still require that the viewpoints be nearly coincident.

A number of multiple viewpoint panorama methods also exist.

Multiple center-of-projection images [133] sample scenes with a vertical slit or pushbroom camera [56] that slides along a user selected path around a scene. The resulting images provide a continuous transition between a wide variety of different viewpoints. The rendering of multiple center-of-projection images can be quite expensive. The path which defines the image must be finely discretized and the scene must be rendered once for each column of the image corresponding to the various viewpoints. A similar approach has been used to create multiperspective panoramas used in cel animation [170]. Like the multiple center-of-projection images, these cel panoramas are rendered offline using a sliding frame across a finely discretized path.

Street panoramas [2, 137] were created for photographing urban landscapes by sliding a vertical pushbroom camera down a street. The resulting multiple perspective images provide a continuous view for a single street. In order to make construction automatic, street panoramas make an important assumption about the geometry in the scene always being located at the building façades.

This creates distortions with cars on the street or when the panorama comes to an alley or cross street. By estimating the geometry [33] of the scene and using a crossed-silt projection [178] in place of a pushbroom camera, the perspective can be dynamically varied compensating for these distortions [138]. Nevertheless, for all street panorama implementations, the resulting images require sampling from many viewpoints which makes construction slow and precludes dynamic scenes. Additionally, the output of a street panorama is a very wide format image which maps poorly to modern video displays.

2.2.2. General Linear Camera

Another system for removing the planar pinhole camera constraints is the general linear camera [126, 172, 173] which is constructed by interpolating intermediate rays between three non-concurrent rays in space. Combining multiple general linear cameras together to produce a continuous image is not a trivial problem. Blending the rays of neighboring general linear cameras together [174] produces a continuous ray space which leads to continuous output images. The resulting camera model does not provide a fast projection operation requiring raytracing to be performed in order to produce images.

2.2.3. Artistic Rendering

For centuries artists have been manipulating perspective, in some cases unknowingly, in order to further their artistic expression. Some rendering systems have also attempted to reproduce these same effects. One such system [3] composites planar pinhole camera images of objects from various perspectives together into a single image. This approach sometime has difficulty with visibility ordering, does not scale well with scene complexity, and does not support multiple perspectives per object.

Artistic collages of real-world scenes have been achieved by the use of flexible camera arrays [107]. These arrays of planar pinhole camera images are roughly aligned resulting in a multiperspective image with intentionally visible seams.

Work has also been done in generating multiple perspective images by resampling video cubes of real-world scenes. A video cube is constructed by gathering a stack of images from a video camera moving along a continuous path. Arbitrary cuts can be made through the video cube to generate multiperspective videos or images [141]. Video cubes have been used to create abstract stylized video effects such as impressionist and cubist effects [72].

2.2.4. Computer Vision

Some computer vision research has focused on removing the limitations of the planar pinhole camera model. Most early efforts focused on producing omnidirectional cameras. For example, the disparity embedded with a catadioptric non-pinhole camera has been exploited for extracting 3-D geometry from a single image [77]. A number of non-pinhole cameras models have been studied in order to model complex lens and catadioptric systems. These systems include pushbroom cameras [56], two-slit or crossed-slit cameras [114, 178], and their generalization, the general linear camera [173], but these camera models only partially relax the single viewpoint constraint, not sufficiently overcoming problems such as occlusion.

2.2.5. Occlusion Cameras

Occlusion cameras attempt to capture samples visible from a reference viewpoint in addition to those samples barely hidden from the reference viewpoint. The goal of these cameras is to alleviate disocclusion errors that occur as the desired view translates away from the reference viewpoint. The single pole occlusion

camera [103] does this by performing a radial distortion of the camera rays around a user specified pole through the scene. The depth discontinuity occlusion camera [127] automatically detects the silhouettes of objects and distorts the rays of the camera to disoccluded objects hidden just out of view behind those silhouettes. Both of these occlusion cameras are tightly coupled to the thesis of this work and will be discussed in greater detail in Chapter 3.

2.3. Model Modification Approaches

A final class of method that can be used to overcome the limitations of the planar pinhole camera is one which continues to use the planar pinhole camera, but modifies the scene to enhance what is visible from the desired viewpoint. These approaches almost exclusively address the problem of occlusion. For a more detailed state-of-the-art description, the reader is referred to a comprehensive taxonomy of over 50 occlusion management techniques [39].

2.3.1. Transparency and Cutaway

Both transparency and cutaway techniques have been developed as natural approaches for removing occlusion from scenes. These methods have the advantage of not distorting the shape of data in any way, maintaining spatial relationships between object subsets.

Transparency [34, 44, 62] has the advantage of still showing a complete model, but simply reducing the opacity of less important features, making them less visible. The disadvantage of these techniques is that they only work well in models with a few layers of transparency. As the number of layers increases, either each individual layer contributes less influence to the output image or the transparent layers combine to become opaque.

To address the limitations of transparency, the cutaway approaches [20, 35, 44, 86] have been developed. Cutaway works by simply removing outer layers of the dataset, revealing occluded subsets. Cutaway approaches can remove many layers of occluding surface, but as more layers are cutaway important data subsets might be lost. The complete removal of occluding layers in this approach prevents simultaneous monitoring of multiple layers of the dataset. Alternatively, cutaway data can be displayed in a separate view. This alternative approach is limited in the number of layers it can support.

2.3.2. Dataset Distortion

An alternative approach for handling occlusions in datasets is to distort the dataset such that data subsets of interest become visible to the user. Earlier 2-D work including Generalized Fisheye Views [48], the Hyperbolic Browser [79], and EdgeLens [168] used graph and hierarchical methods for visualizing data. For 3-D datasets, two general approaches have been developed to date, deformation and explosion.

The deformation techniques strive to disoccluded data subsets of interest while preserving original connectivity and minimizing dataset distortions. Deformation approaches have been used in a wide variety of applications including providing occlusion-free street-level animation of driving routes [154], constructing panoramic tourist and ski maps [31] which maximize visibility and minimize distortion, comprehensively visualizing short travel routes [32] which distorts space around the corners of a path, and in focus-plus-context visualizations of bird's eye views of urban environments [132]. All of these techniques strive to modify the spatial relationships between various data subsets as little as possible leading to results which are similar to the original dataset. Nevertheless, the distortions that they introduce can lead to confusion, in particular when spatial

relationships such as distance and direction are important. All of the techniques completely fail when the data of interest is contained within the occluding object.

Explosion techniques rely on datasets that are hierarchically subdivided by either user interaction [18, 37] or automatically [85]. Using the hierarchical information, subsets of data can be moved away from one another in order to reveal elements of the scene that would normally be occluded. The explosion technique is well suited for datasets where objects are separable into meaningful subsets and where some data subsets are completely occluded by others. Explosion techniques try to maintain configurations which closely match the original dataset configuration, but they still require significant distortions of the data subsets to reveal occluded objects. This disturbs the spatial relationships between objects, relying upon the user's ability to mentally connect the pieces.

2.4. Camera Model Design in Nature

The idea of application specific cameras is not a purely human invention. Nature itself has a number of interesting examples of Camera Model Design. A number of snake species including some boa constrictors and rattlesnakes have eyes design to see in the infrared spectrum, far beyond what is visible to humans, allowing them to see minute differences in temperature [29]. Many birds of prey also have specialized visual systems. Eagles and falcons have eyes with multiple levels of magnification. The magnification is at its highest near the center of the eye allowing them to focus at great distance [29, 80]. Owl eyes have an extremely large aperture (iris) allowing them to have extraordinary night vision [147]. Many inserts such as bees, flies, and dragon flies have wide angle multifaceted lenses on their eyes. These types of lenses allow for panoramic views of the world, and their eye sensors are designed to detect motion allowing them to quickly avoid predators [29]. This is by no means an exhaustive list of animals that benefit from Camera Model Design in nature but simply a few

interesting examples. In general, nature has designed predators with forward looking eyes for hunting and prey with eyes set to the side for more panoramic views to avoid predators.

CHAPTER 3. THE OCCLUSION CAMERA FAMILY

A depth image, defined as an image with per-pixel depth, is an important computer graphics primitive. One use of depth images is in the context of image-based rendering where they replace geometry in order to reduce rendering cost for 3-D warping [102] or for high-quality interactive rendering effects (e.g. impostors [95, 127], relief texture mapping [112, 125], or ambient occlusion [14]). Depth images are also used to determine visibility from the light's point of view in shadow computation. In the context of compression of rendered imagery, depth image key frames are automatically mapped to intermediate frames by 3-D warping, reducing the amount of data residual images need to store [5]. In the context of 3-D displays, depth images can be used as an intermediate representation that accelerates the rendering of the 3-D image and reduces the bandwidth to the 3-D display.

An important challenge with depth images is posed by disocclusion errors. A depth image is typically rendered with a planar pinhole camera and only stores samples visible from the camera's viewpoint. Even minimal viewpoint translations expose regions that were not visible from the original viewpoint, which causes disocclusion errors. In image-based rendering this leads to highly objectionable "holes" in the output image or reduced fidelity in rendering effects. In shadow mapping, the single viewpoint constraint limits the approach to hard shadows. In compression, the disocclusion errors lead to high residuals, lowering compression performance (i.e. compression ratio). In the context of a 3-D display, disocclusion errors limit image quality when the display is seen from additional viewpoints (i.e. second-eye viewpoint in a single-user scenario, and additional user viewpoints in a multi-user scenario).

The classic approach for alleviating the problem of disocclusion errors is to use additional depth images, rendered from nearby viewpoints, in the hope of filling in disocclusion errors. However, such an approach is inefficient. First, disocclusion errors occur throughout the volume of the scene, and one cannot find a small set of additional depth images that provide all missing samples. Second, the additional depth images have considerable overlap, which causes costly redundancy. Third, the output image is rendered from a variable number of depth images, so the advantage of bounded rendering cost is lost.

Disocclusion errors occur because the depth image is constructed with a planar pinhole camera which samples the scene from only a single viewpoint, yet the depth image is asked to support additional viewpoints. This is only possible when the scene is trivial and the additional viewpoints do not require additional samples.

To alleviate the problem of disocclusion errors we have innovated at the camera model level. Instead of constructing the depth image with a planar pinhole camera, the depth image is constructed with a non-pinhole called an occlusion camera. In addition to the samples seen from a reference viewpoint, an occlusion camera also gathers samples that are likely to become visible from nearby viewpoints. Moreover, although occlusion cameras are non-pinholes, they do provide closed-form projection and therefore occlusion camera images can be constructed efficiently in feed-forward fashion, by projecting and rasterizing scene triangles.

The first such camera model is the single pole occlusion camera [103]. The single pole occlusion camera performs a 3-D distortion of the rays of a planar pinhole camera around a pole through the scene. This distortion of the rays effectively enables the camera to see around the silhouette of objects to gather samples previously hidden from the reference viewpoint. Like traditional depth images, single pole occlusion camera images are single-layer, non-redundant,

and their connectivity is implicit. The single pole occlusion camera does not guarantee all samples will be captured, but taking the union of a planar pinhole camera image with a single pole occlusion camera image provides much better scene coverage for a wide variety of viewpoints as compared to a planar pinhole camera image alone.

The single pole occlusion camera does a good job alleviating disocclusion errors, but it lacks scene awareness. This results in cases of inefficient image allocation (i.e. wasted pixels) and poor scene coverage for complex occluders. The depth discontinuity occlusion camera [127] was developed to be more aware of the 3-D scene. It detects the silhouettes of objects by finding depth discontinuities and distorts the camera rays around those silhouettes to expose samples hidden from the reference viewpoint. Like the single pole occlusion camera, the depth discontinuity occlusion camera produces single-layer and non-redundant images with implicit connectivity. This allows them to be efficiently substituted for planar pinhole camera depth images in many applications.

Both of the previously mentioned occlusion cameras are successful at capturing samples not visible from the current reference viewpoint. One weakness of both methods however is that they cannot tell you which set of desired viewpoints will have adequate samples for reconstruction. The third occlusion camera model, the epipolar occlusion camera, is a camera model designed to capture most of the samples seen by a planar pinhole camera translating between two viewpoints. The camera works by searching for depth discontinuities along epipolar lines which are created by linear camera motion. Whenever a depth discontinuity is detected, space is added in the image for potentially hidden samples. The resulting view-segment produces a single layer 2-D image that samples almost all of the visible geometry without any redundancy.

The epipolar occlusion camera overcomes one major disadvantage of earlier occlusion cameras. The set of viewpoints with valid reconstructions was not

known a priori for either the single pole occlusion camera or the depth discontinuity occlusion camera. The epipolar occlusion camera overcomes this limitation by gathering all of the samples for a 3-D scene visible from along a segment of viewpoints.

As part of my work, I have designed applications for both the single pole occlusion camera and the depth discontinuity occlusion camera. Therefore, for completeness, I describe the single pole occlusion camera and depth discontinuity occlusion camera in detail. All images and any text are used with the authors' permission.

3.1. Single Pole Occlusion Camera [103]

To address the problem of occlusion in scenes, the single pole occlusion camera (SPOC) was developed. The SPOC is a non-pinhole camera which is a generalization of the planar pinhole camera (PPC) designed for disoccluding around a single object within a scene. The SPOC performs a 3-D radial distortion, defined by a pole, on geometry within the scene to sample surfaces hidden from the reference viewpoint. These newly exposed surfaces are ones which lie slightly hidden from the reference view behind the silhouette of occluders and are therefore likely to become exposed as the desired view translates away from the reference viewpoint.

A PPC gathers exactly one sample per ray as does the SPOC. The 3-D distortion of the SPOC effectively trades image resolution (*x*-*y* domain) for increased resolution along the rays of the planar pinhole camera (*z* domain). In Figure 3.1, the lid of the teapot is not sampled using the PPC. In the SPOC, the lid of the teapot receives many samples at the cost of lower resolution sampling for the body of the teapot.

The SPOC does not guarantee that all disocclusion errors can be avoided. In addition, the SPOC does not capture the entire scene, nor should it because this would be wasteful oversampling. In Figure 3.1, for example, the front, top, bottom, and sides of the teapot are well sampled, but back of the teapot remains unsampled. In complex scenes, the samples captured by the SPOC might not contain all the samples corresponding to the planar pinhole camera, resulting in disocclusion errors from the reference viewpoint. To address this limitation, an SPOC can be efficiently merged with the reference PPC image. To do this, only those samples not visible from the reference PPC are added to the SPOC image (Figure 3.2).

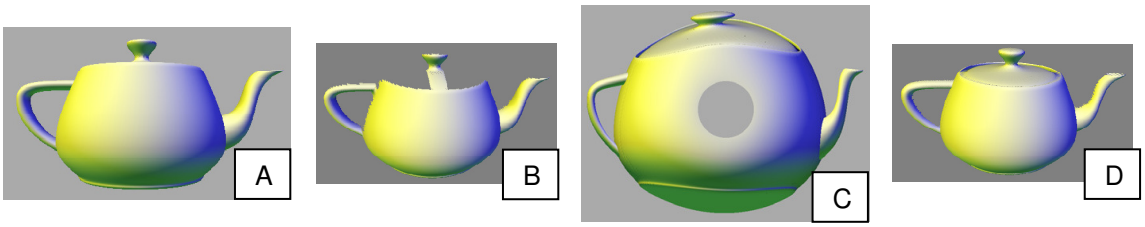


Figure 3.1. Samples captured with a planar pinhole camera compared to a single pole occlusion camera. With a PPC depth image (*A*), disocclusion errors can occur as the view translates away from the reference view (*B*). Using the SPOC in place of the PPC (*C*), a wide variety of viewpoints is supported without disocclusion errors appearing (*D*).

The SPOC rays are non-intersecting segments which cover the entire camera field-of-view. There is exactly one ray through each 3-D point in the scene, with the exception of those lying exactly on the pole. The SPOC supports closed form projection and there is a one-to-one mapping between scene triangles and image plane triangles. The implication being that SPOC images can be rendered on the GPU using the feed-forward graphics pipeline.

3.1.1. Camera Model

The SPOC was designed with a couple of goals in mind. The first was to capture the samples visible from the reference view. In addition, it should capture samples barely hidden from the reference view at the object silhouettes, thereby allowing disocclusion error free translation away from the reference viewpoint. The camera model should also be designed to render quickly by fitting as tightly as possible into the standard feed-forward graphics pipeline. Doing this requires closed-form projection for 3-D points at a minimum. Closed-form projection implies that the projection of every 3-D point will have a unique image plane location that can be found in a finite number of operations. Additionally, fitting into the feed-forward graphics pipeline requires the image be coherent in that there should be a one-to-one mapping between a triangle in the scene and on the image plane.

PPC *PPC + (SPOC – PPC)* *Truth*

Figure 3.2. Single pole occlusion camera samples merged with planar pinhole camera samples. The samples contributed by the SPOC (*middle*), marked in pink, are added to the samples from the PPC (*left*) to approach the quality of the ground truth (*right*).

The reverse planar pinhole camera model (RPPC) has many of these properties (Figure 3.3, left). The RPPC reverses the perspective foreshortening effect resulting in a view frustum which starts larger and shrinks as it moves away from

the camera. This is similar to collecting the farthest visible sample along each ray of a PPC.

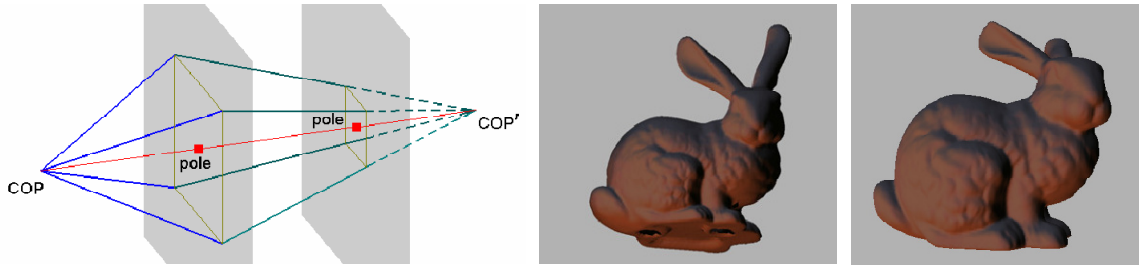


Figure 3.3. The reverse planar pinhole camera model. The RPPC model (*left*) used to capture the bunny dataset (*center*) is compared to using a conventional PPC (*right*).

Figure 3.3 compares the results of capturing the bunny dataset with both a conventional PPC and well as the RPPC. The RPPC captures many samples which are barely hidden from the conventional PPC behind the silhouette of the object. Due to the simplicity of the camera model, the scene can be easily rendered using fixed-function hardware by simply rendering the scene using the opposite view PPC and flipping the z-buffer test.

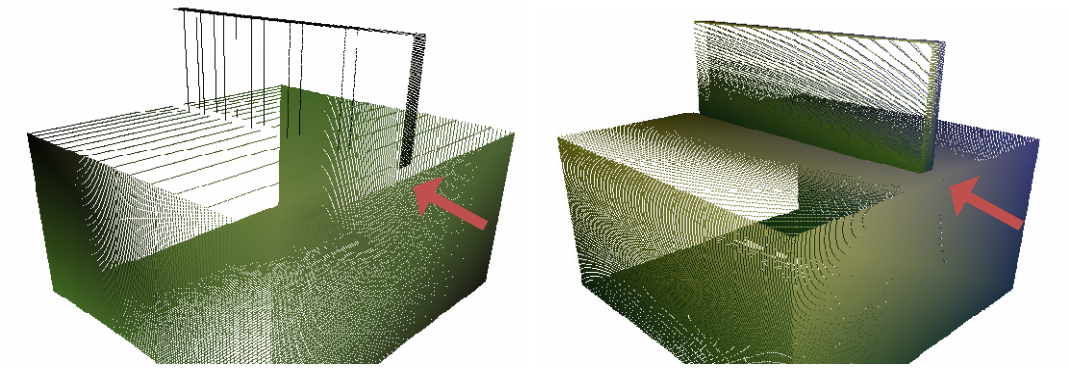


Figure 3.4. Limitation of the reverse planar pinhole camera model illustrated. An RPPC (*left*) captures some surfaces poorly. Applying a radial distortion (*right*) captures those surfaces better.

The RPPC captures many surfaces not visible from the reference viewpoint except for those surfaces whose normals are near perpendicular to rays capturing them. Figure 3.4 left shows an example of this problem. A simple scene is constructed of two boxes. The scene is viewed from the direction of the red arrow. The surface normals of the top of the large box and sides of the small box are nearly perpendicular to the rays in these regions resulting in insufficient sampling. This inspired the introduction of a radial distortion around a pole (Figure 3.4, right).

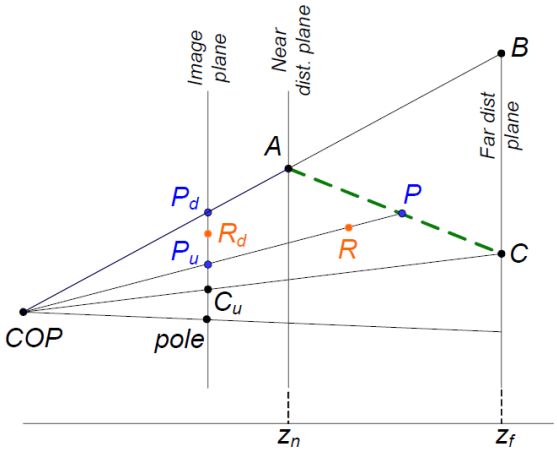


Figure 3.5. Diagram of the 3-D radial distortion used in the single pole occlusion camera model.

The SPOC was therefore designed to have many of the properties of the RPPC, in addition to a radial distortion designed to capture samples from around all sides of an occluder. The pole is a point chosen on the image plane, and thus a 3-D line. The best choice in general is for the pole to go through the centroid of the occluder object, but this is not a requirement. The distortion then pushes samples away from the pole according to their depth in the scene. 3-D points which are closer to the center-of-projection are pushed less, while points farther from the center-of-projection are pushed further away. This distortion means that two points which lie on the same PPC ray will lie on different rays with the SPOC, thereby exposing regions which are barely hidden from the reference viewpoint.

Figure 3.5 shows an example of the 3-D radial distortion working. Here the point P is projected to the image plane location P_u where it would normally be occluded by the projection of point R. Instead, the projection of R is displaced to the location R_d and P_u is displaced by a larger amount (since it is farther from the center-of-projection) to the location P_d. In this configuration, R no longer occludes P in the output image. Now the ray which effectively samples P no longer travels from *COP* to B as is the case with the conventional PPC. Instead the ray travels from *COP* to A and then from A to C. The amount of distortion which the projections of R and P receive is a linear expression in $1/z$, where z is the camera space z-coordinate of the point.

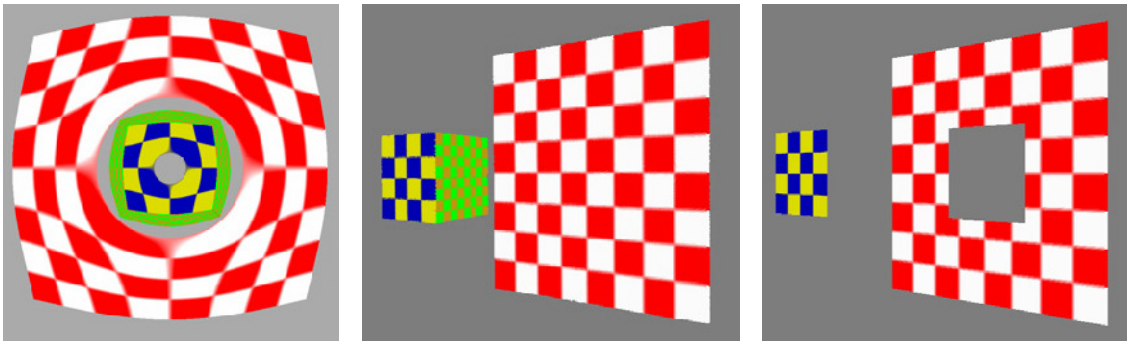


Figure 3.6. Example of the single pole occlusion camera disocclusion capabilities. The SPOC image (*left*) captures 5 of 6 faces for the cube and the entire background avoiding any disocclusion errors (*middle*) as opposed to using a PPC image (*right*).

The SPOC model is defined by a PPC and a six-tuple (u_0, v_0, z_n, z_f, d_n, d_f) that describes the distortion. The image plane coordinates (u_0, v_0) define the coordinates of the pole. The coordinates of the pole are typically chosen to be the centroid of the PPC projection of the occluder. The values of (z_n, z_f) define the range of the scene in which the distortion will be applied. The value of z_n will typically be chosen to be some value smaller than the closest point of the occluder to the camera. The value of z_f will typically be large enough to encompass the entire scene. Finally, the values of (d_n, d_f) define the distortion

magnitudes for points on the planes represented by (z_n, z_f). The distortion value d_n is typically set to zero while the distortion value d_f defines how much of the occluded object becomes visible. The larger d_f becomes, the farther samples are pulled out from behind the occluding object. Looking at Figure 3.1, the large d_f exposes a significant portion of the top and bottom of the teapot and eventually a larger value would expose the back of it as well. Similarly in Figure 3.6, d_f is sufficiently large to capture the entire background hidden behind the occluding cube.

3.1.2. Projection

For a 3-D point P(x, y, z) which lies between z_n and z_f the image plane coordinates of the point when projected with an SPOC (*PPC*, (u_0, v_0, z_n, z_f, d_n, d_f)) are given by Equation 3.1. By examining the projection equation a little closer, we can see that the magnitude of the distortion is calculated by using only $1/z$ while the projected coordinates (u_u, v_u) only affect the direction of the distortion.

$$(\mathrm{u_u}, \mathrm{v_u}, \mathrm{z}) = \mathrm{PPC(P)}$$

$$\mathrm{d(z)} = \mathrm{d_n} + \frac{1/\mathrm{z_n} - 1/\mathrm{z}}{1/\mathrm{z_n} - 1/\mathrm{z_f}}(\mathrm{d_f} - \mathrm{d_n}) \qquad \text{Equation 3.1}$$

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

In the case of a point whose z value is less than z_n, the planar pinhole camera projection is applied. By definition, no points should lie beyond z_f, but if they do, the original SPOC projection equation can be applied or the distortion can be clamped at d_f.

The pole represents a singularity in the projection. All of the 3-D points that land within the view frustum of the PPC have exactly one projection point onto the image plane, except for points lying on the pole. By definition, these points have no distortion direction. Each of these points in fact project to a circle centered at the pole whose radius is the magnitude of distortion at the point's z value. This singularity can be hidden by pushing the camera's z_n beyond some geometry lying on the pole, effectively masking the hole.

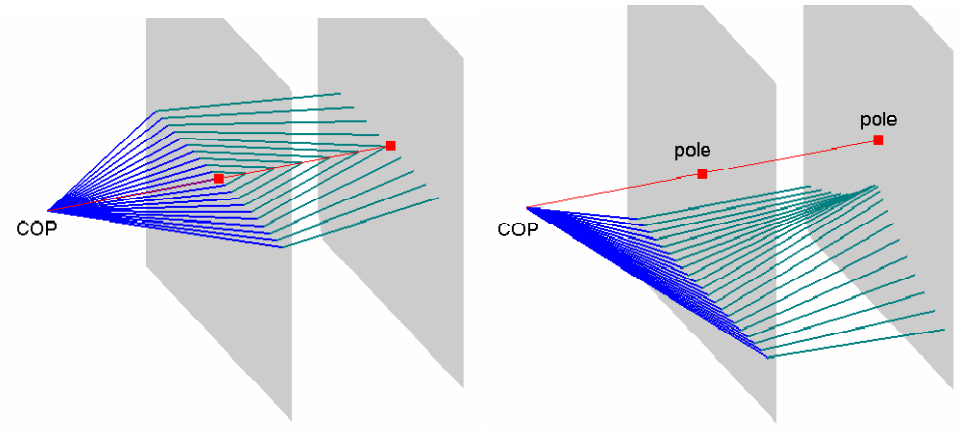


Figure 3.7. A visualization of the single pole occlusion camera rays. The pole is denoted in red. The rays of the camera remain undistorted (*blue*) up to the near plane and are pulled toward the pole beyond it (*green*).

3.1.3. Rays and Unprojection

In mathematical terms, a ray is a point and a direction, which accurately describes the shape of the PPC rays. The definition of camera rays in terms of the SPOC needs to be relaxed. For the SPOC, a camera ray is defined as the locus of 3-D points which project to the same output image location.

Given an SPOC output image plane location (u_d, v_d), the set of 3-D points which represent that ray can be found by varying z from z_n to z_f. The 3-D point P for a

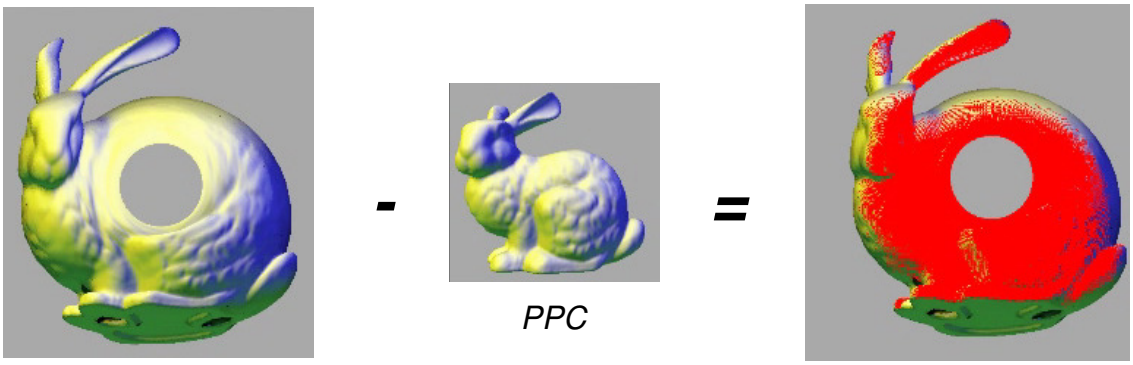
given *z* value can be computed by calculating the distortion magnitude *d(z)* from Equation 3.1 and using that value to calculate *P* using Equation 3.2.

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

Equation 3.2

$$P = \text{Plane}(z) \cap \text{PPC.Ray}(u_u, v_u)$$

The direction of the distortion can be computed by using the known distorted coordinates (u_d, v_d) position relative to the coordinates of the pole (u_0, v_0). To calculate the undistorted coordinates (u_u, v_u), the distorted point is pulled toward the pole with a magnitude defined by d(*z*) for the known *z* value. Points with *z* values less than z_n receive a zero distortion magnitude making their (u_d, v_d) and (u_u, v_u) coordinates identical. The output 3-D point P is obtained by calculating the point which lies on the PPC ray (u_u, v_u) at the distance of *z*.



SPOC *PPC* *SPOC-PPC*

Figure 3.8. Result of an image-set-difference on a single pole occlusion camera image and a planar pinhole camera image.

The rays of the SPOC are effectively piecewise linear rays consisting of two segments. Referring again to Figure 3.5, the ray which effectively samples point *P* is comprised of the segments (*COP*, *A*) and (*A*, *C*). This relatively simple

structure is possible because $1/z$ is linear in screen space and the distortion is linear with respect to $1/z$, making the rays straight within the distortion region.

Not all values of (u_d, v_d, z) are valid for unprojection to produce a 3-D point. The undistorted coordinates (u_u, v_u) are not allowed to cross the pole. This requires the distortion magnitude $d(z)$ be less than the distance from (u_d, v_d) to the pole (u_0, v_0). This also means that the rays of the SPOC cannot cross the pole. As seen in Figure 3.7, they simply terminate when reaching the pole.

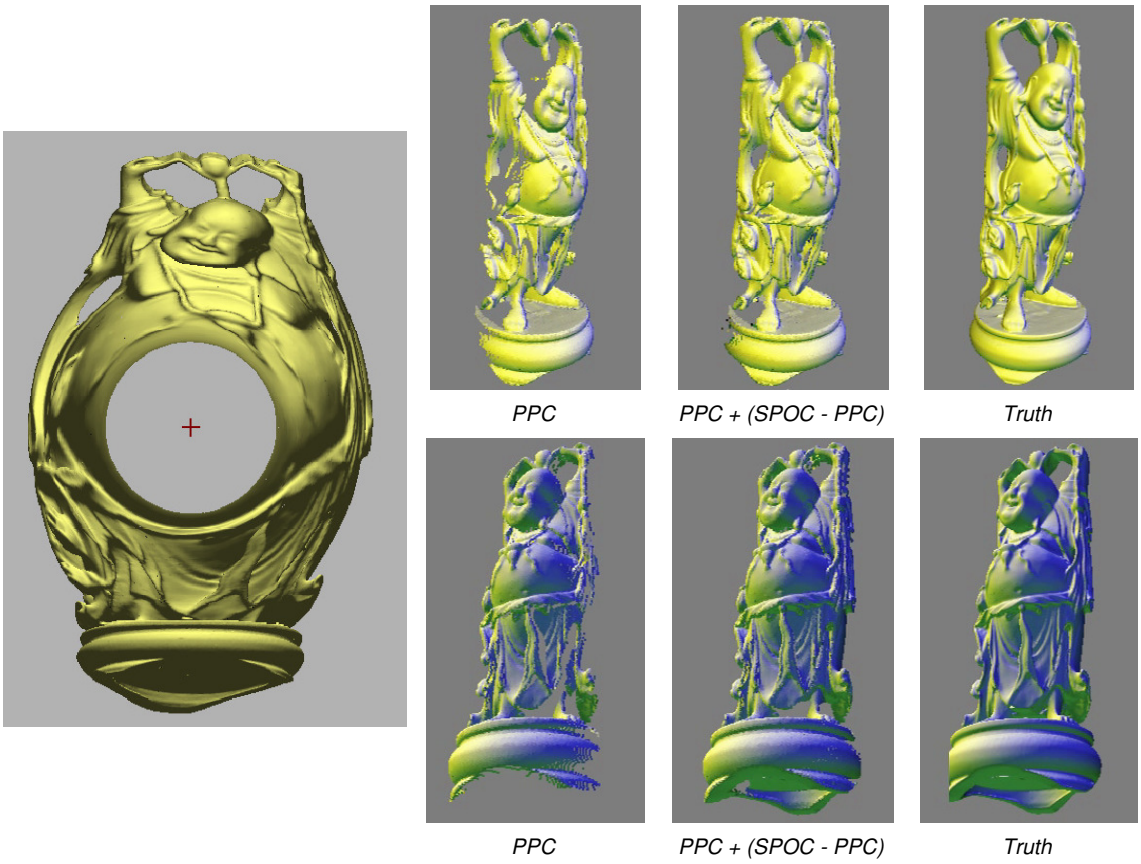
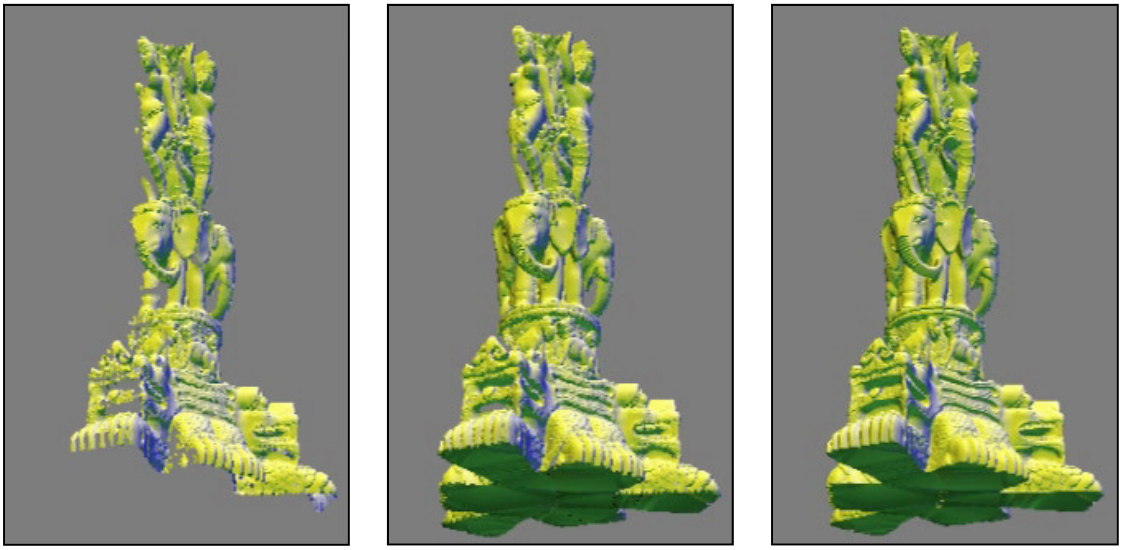


Figure 3.9. Single pole occlusion camera image of the Happy Buddha model. The SPOC reference image alleviates distortions from a variety of viewpoints.

3.1.4. Point-Set Merging

PPC images are very prone to disocclusion errors when the view translates even a small distance away from the reference viewpoint. SPOC images are far less prone to these disocclusion errors since they store samples barely hidden from the reference view at the silhouettes of objects. These samples fill the gaps which would normally appear when using a PPC image, thereby extending the usefulness of the reference image far beyond just the reference viewpoint.



PPC *PPC + (SPOC - PPC)* *Truth*

Figure 3.10. A single pole occlusion camera reconstruction of the Thai statue model.

Even so, the SPOC images make no guarantee that the samples they have will be the complete set required to reconstruct a view. In fact, due to the distortion, samples visible in the planar pinhole camera reference image can become occluded in the SPOC image. In Figure 3.9, the feet of the statue are occluded in the SPOC image despite being visible from the PPC's perspective. This problem is mitigated by rendering a PPC image along with an SPOC image that only collects samples not visible in the PPC image.

This is accomplished by using an SPOC image-set-difference operator. Given two input images, a sample from the first image is unprojected to its 3-D position and the resulting 3-D point is projected onto the image plane of the second image. If the resulting *z* value is similar to the *z* value stored in the second image, the sample is a duplicate and marked as such. This operator works on two SPOC images, but can accept PPC image as well, since they are a special case of an SPOC image with no distortion (Figure 3.8). Rendering the samples from the PPC and the SPOC image-set-difference (SPOC – PPC) pair results in no loss of samples from the PPC view while avoiding most disocclusion errors (Figure 3.9 and Figure 3.10).

3.2. The Depth Discontinuity Occlusion Camera [127]

The SPOC suffers from significant limitations due to its coarsely specified distortion. The SPOC is essentially limited to a single occluder or non-complex set of occluders. When used on multiple objects or complex occluders important samples can be lost. To recover from those errors, the SPOC images are used in conjunction with PPC images.

To address these challenges in a more robust manner, a new more scene-aware occlusion camera was developed, the depth discontinuity occlusion camera (DDOC). While the SPOC used one global deformation of camera rays to avoid occlusions, the DDOC automatically computes fine-grain local camera ray distortions to produce images which more successfully handle complex scenes and multiple occluders.

In Figure 3.11, the DDOC stores additional samples of the floor and back wall not visible from the PPC. The DDOC does not store all of the samples for the scene, but does store enough to generate a much better novel viewpoint reconstruction of the scene when compared to the PPC.

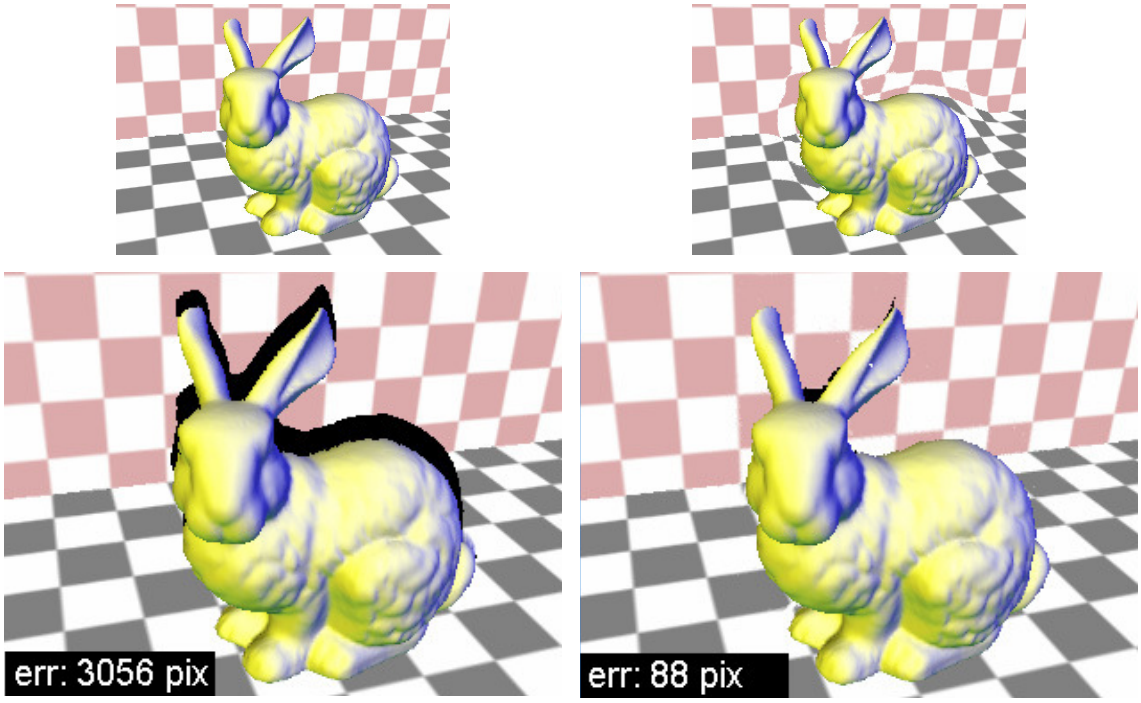


Figure 3.11. A comparison between a planar pinhole camera image and depth discontinuity occlusion camera image. The PPC (*left*) and DDOC (*right*) reference images are shown along with corresponding novel viewpoint reconstructions. The DDOC does a better job alleviating disocclusion errors which are measure as the number of missing pixels.

The DDOC model consists of a PPC augmented with a distortion map. Efficient projection is achieved by first projecting a 3-D point with the PPC, then applying the distortion stored at the distortion map location. The images produced by the DDOC have the same advantages as those of a PPC image such as a bounded number of samples and implicit connectivity.

3.2.1. Camera Model

Like the SPOC, the DDOC strives to alleviate occlusions by capturing samples which are barely hidden behind the silhouette of occluders. The DDOC accomplishes this in a more scene-aware manner. Given a scene and a

reference viewpoint, the DDOC accomplished the goal by applying a distortion that acts only on samples near object silhouettes.

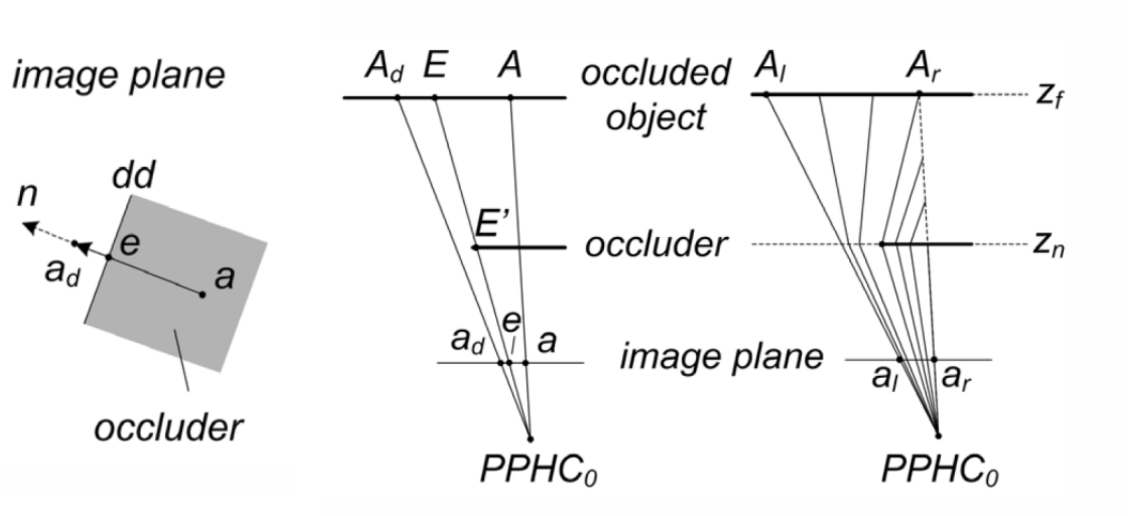


Figure 3.12. Illustration of the distortion used in the depth discontinuity occlusion camera.

Given a reference PPC image, the silhouette of objects can be detected by finding depth discontinuities within the image. These are identified by finding pixels whose depth values significantly vary from those of their neighbors. The left image of Figure 3.12 shows a small portion of a reference image plane. The depth discontinuity *dd* lies between the occluder and the background on the image plane. When projecting a point with the DDOC, the sample point *a*, which lies behind the occluder in 3-D space, will be moved in the direction of the normal *n* of *dd* to the location a_d.

The middle panel of Figure 3.12 shows a cross section of the scene. Here, the depth discontinuity *dd* is represented by the 3-D points *E* and *E'* and the image plane location *e*. The 3-D point *A* is projected with a conventional PPC to the image plane location *a*. Here, the distortion magnitude depends upon the depth of the point within the scene. The magnitude is zero up to z_n, the depth of the occluder *E'*. The distortion is its largest d_f at z_f, the depth of the occluded object

E, and remains constant beyond that point. In this way, objects which are close to the occluder move less in favor of objects farther away, since distant objects are more likely to become visible as the desired view translates.

The right panel of Figure 3.12 shows the same cross section with the effect of the previously described distortion on the rays of the camera. The distortion effect is only performed locally on rays that lie between A_l and A_r and thus only points which project between a_l and a_r using the PPC. Beyond that, no other points are affected by the distortion. The locations of a_l and a_r are determined by the size of the distortion neighborhood which is a user specified parameter *D*. The points a_l and a_r are each chosen to be *D* pixels away from *e*. The points *A*, *E*, and A_d from the middle panel all lie between A_l and A_r.

The rays of the camera are initially the rays of the PPC. They remain unaffected until they reach z_n. At that point, the rays move along the depth discontinuity normal *n* towards the occluder, effectively pushing the samples away from the occluder. At the end of the distortion region z_f, the rays continue with a constant amount of distortion d_f until they reach the end of the view frustum.

3.2.2. Distortion Map Construction

The DDOC model is defined by a reference planar pinhole camera ($PPHC_0$) and a distortion map. The distortion map is a five-tuple which specifies the distortion for each $PPHC_0$ ray and has the same resolution as $PPHC_0$. Each distortion sample is a five-tuple containing (dir_u, dir_v, z_n, z_f, d_f). The pair (dir_u, dir_v) is a 2-D vector which describes the image plane direction for the distortion (*n* from the Figure 3.12). The scalars (z_n, z_f) define the interval of distortion along the z-axis. The distortion magnitude varies between zero at z_n and d_f at z_f. Like the SPOC, the distortion varies linearly with $1/z$, making the distorted portion of the DDOC rays straight segments.

Construction proceeds as follows. Given the 3-D scene *S* and a reference view $PPHC_0$, the distortion map *DMAP* is built in a six step process.

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

Steps 1 and 2 compute the depth discontinuity map *EB*. This is done by first rendering the scene with the planar pinhole camera $PPHC_0$ obtaining its depth buffer *ZB*. Depth discontinuities are then detected on *ZB* at locations where the second order finite depth difference is larger than a threshold [128].

Step 3 begins the process of setting the *DMAP* values. For every edge pixel *e* in *EB*, the five-tuple for that cell must be determined. The distortion direction *dir* is set to be perpendicular to the local depth discontinuity direction by computing a least squares fitting line to the edge pixel in the neighborhood of *e*. The orientation is set to point away from the occluder towards the larger *z* value. The values of z_n and z_f are set to be the near and far z values which formed the depth discontinuity in the first place. The last scalar d_f is calculated in step 6. Once the five-tuple for the edge pixel has been determined, that edge pixel is splatted onto the *DMAP* using a circle centered at *e* with a radius of *D*, the distortion magnitude.

During the construction phase, each *DMAP* pixel stores three temporary scalars in addition to the distortion five-tuple. Those values are c_u and c_v, the center of the splat and r, the radius of the splat. During the next processing step, this data is used to remove conflicts when a *DMAP* location has already been set. In the case that two splats conflict, the splat whose center is closest to the current sample is used. The distance to the splats can be computed using the current *DMAP* location and the splat center (c_u, c_v).

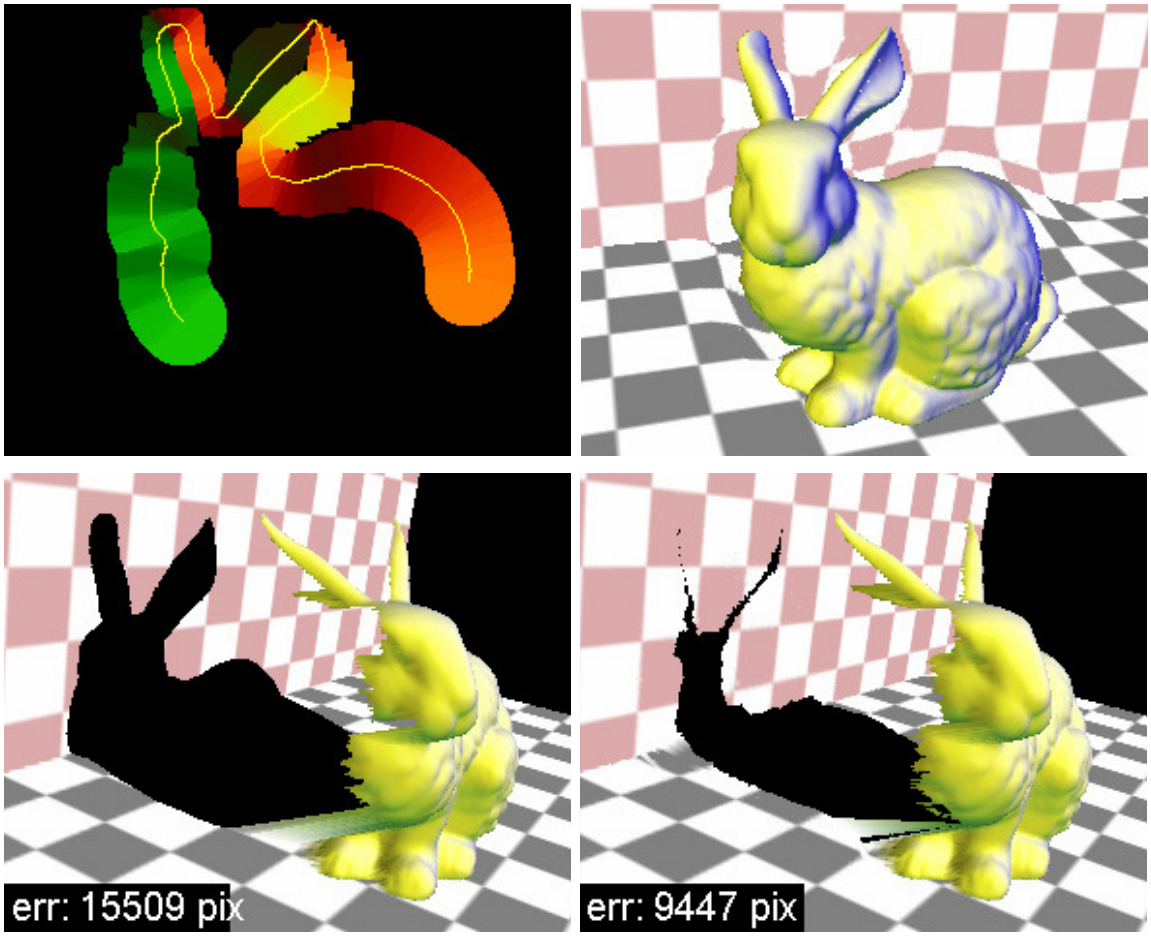


Figure 3.13. Depth discontinuity occlusion camera visualization of the bunny. *Top:* A visualization of the distortion map (*left*) used to generate the DDOC image (*right*). *Bottom:* A novel viewpoint reconstruction illustrates how the DDOC image (*right*) improves sampling of the scene, shrinking the shadow of the bunny as compared to a PPC image (*left*).

Step 4 attempts to detect and eliminate any distortion conflicts that may have arisen as part of step 3. In particular, if any *DMAP* location is affected by two or more depth discontinuities they might be conflicted. The normal directions of the two depth discontinuities are compared and if they are above a threshold, 90° in the authors' implementation, they are considered to be conflicted. In that case, the splat radius of the two conflicted depth discontinuities are shrunk such that the splat circles no longer overlap thereby eliminating the conflict.

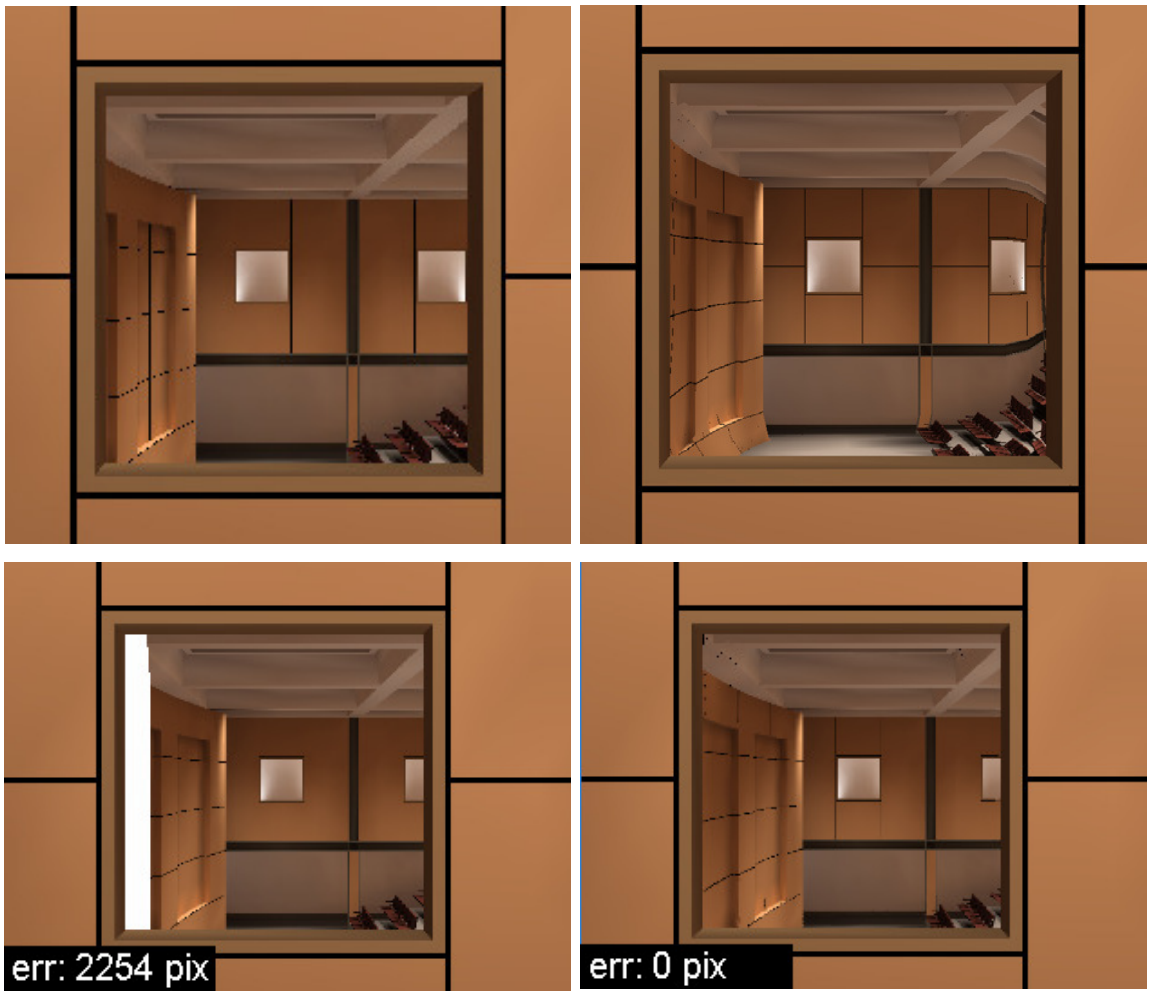


Figure 3.14. Depth discontinuity occlusion camera example of the auditorium. The PPC reference image (*top*, *left*) does not have enough samples for a novel viewpoint reconstruction (*bottom*) while the DDOC reference image (*right*) does.

Step 5 takes a pass over each cell of *DMAP* eliminating samples invalidated by step 4. Distortion samples whose distance to their splat center is larger than the radius of that splat are deleted.

Step 6 finalizes the distortion samples at each location in *DMAP*. In particular, the distortion magnitude needs to be set. It is obvious that the samples on the hidden side of the depth discontinuity need to move out from behind the occluder. It is also necessary to move samples on the visible side of the edge to make space for the hidden samples.

To achieve this effect, the distortion magnitude d_f is varied linearly from r to zero as the signed distance from the edge increases to $-r$ to r. In this way, samples, that are already visible, exist in the band $[0,r]$ and are compressed to the interval $[r/2,r]$ to make space for hidden samples. The hidden samples in the band $[-r,0]$ are compressed to the interval $[0,r/2]$ making them visible in the output image.

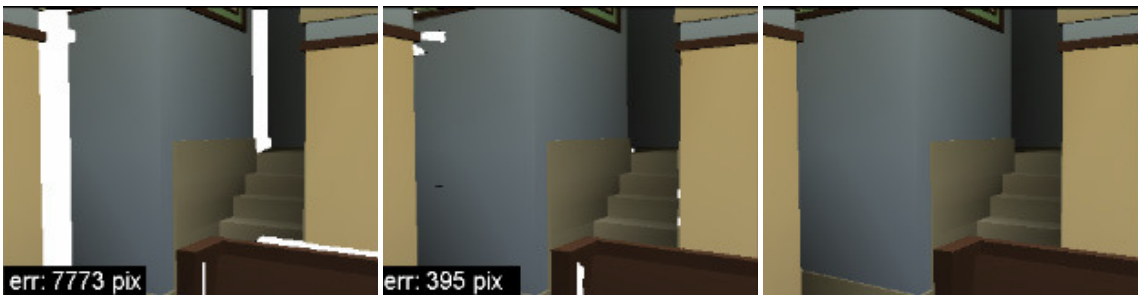


Figure 3.15. A depth discontinuity occlusion camera example of the Unity. A three-way comparison between reconstructions that use a PPC reference image (*left*), DDOC reference image (*middle*), and the original scene geometry (*right*).

Let x be the signed distance between a distortion map location and the edge. The distortion value d_f can be found using a bias and scale operation $d_f = (r - x)/2$. This operation implies a loss of resolution in the distortion region in exchange for additional sampling along the rays of the camera (Figure 3.15). This tradeoff can be mitigated by increasing the output image resolution.

Figure 3.13 shows an example distortion map, DDOC, and a novel viewpoint reconstruction. The radii of the splats are at their maximum around the body of the bunny, but needed to be compressed around the ears of the bunny. When viewed from the side, it is apparent that the DDOC does a better job sampling the scene than a PPC does from the same viewpoint.

3.2.3. Projection and Unprojection

Projecting a 3-D point using the DDOC is similar to that of the SPOC. Using Equation 3.3, a 3-D point *P* is first projected with the planar pinhole camera $PPHC_0$ to the undistorted image plane coordinates (u_u, v_u, z). The direction of distortion (dir_u, dir_v) applied to point can be retrieved from the distortion map along with z_n, z_f, and d_f. The magnitude of distortion can then be linearly interpolated using $1/z$ from zero at z_n to d_f at z_f. The output coordinates (u_d, v_d) are then found by adding the distortion vector to the undistorted coordinates.

$$(\mathrm{u_u}, \mathrm{v_u}, \mathrm{z}) = \mathrm{PPHC_0(P)}$$

$$(\mathrm{dir_u}, \mathrm{dir_v}, \mathrm{z_n}, \mathrm{z_f}, \mathrm{d_f}) = \mathrm{DMAP(u_u, v_u)}$$

$$\mathrm{d(z)} = \begin{cases} 0, \mathrm{z} < \mathrm{z_n} \\ \dfrac{1/\mathrm{z_n} - 1/\mathrm{z}}{1/\mathrm{z_n} - 1/\mathrm{z_f}}\mathrm{d_f}, \mathrm{z_n} \le \mathrm{z} \le \mathrm{z_f} \\ \mathrm{d_f}, \mathrm{z} > \mathrm{z_f} \end{cases} \qquad \text{Equation 3.3}$$

$$(\mathrm{u_d}, \mathrm{v_d}) = (\mathrm{u_u}, \mathrm{v_u}) + (\mathrm{dir_u}, \mathrm{dir_v})\mathrm{d(z)}$$

Unlike the SPOC, the distortion of the DDOC is sample-based and therefore noninvertible. This makes the distorted coordinates (u_d, v_d, z) insufficient for unprojecting the point.

To enable unprojection, the DDOC reference image is augmented with two additional channels of data containing the distortion vector (d_u, d_v). Once the distortion vector is known, the value of (u_d-d_u, v_d-d_v, z) provides the undistorted image plane coordinates of the sample (u_u, v_u, z). The 3-D point can then be recovered by unprojecting the undistorted sample using the planar pinhole camera $PPHC_0$.

3.3. The Epipolar Occlusion Camera

Occlusion cameras offer an elegant solution to the problem of disocclusion errors. Like a regular depth image, an occlusion camera image has a single layer, it does not store redundant samples, and it is rendered in hardware. Unlike regular depth images, an occlusion camera image also stores samples needed in nearby views.

The SPOC is constructed by using a 3-D radial distortion on a PPC around a pole defined by the occluder's center. The SPOC has the merit of introducing the occlusion camera concept but the simple camera model only works for a few relatively simple objects.

The DDOC disoccludes samples by distorting them along the direction perpendicular to the occluder edge, and away from the occluder. The distortion is specified per pixel with a distortion map which provides the flexibility needed to handle complex scenes. However, the DDOC makes room for the disoccluded samples by reducing the resolution on a band parallel to the occluder edge. The visible samples do not use the entire band anymore which makes room for the disoccluded samples. When there is enough room between occluders this approach only implies halving the resolution close to occluder edges, when occluders are close together, the DDOC's disocclusion capability is greatly reduced. In Figure 3.19 the picket fence has a distance between pickets of 1 pixel and there is no room for disoccluding the back wall samples hidden by the

pickets, which are needed for intermediate viewpoints. In this case, a DDOC would not perform better than the PPC image shown.

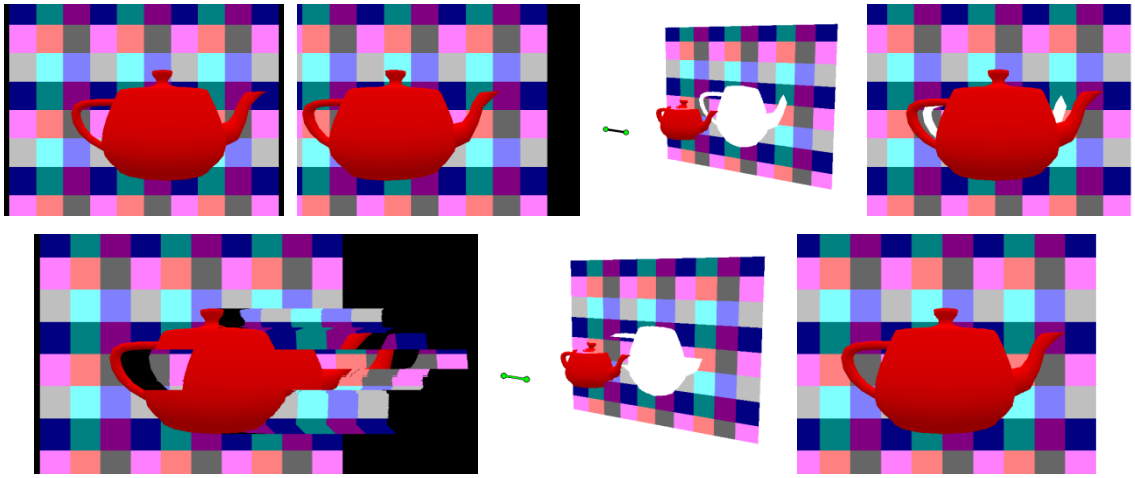


Figure 3.16. An epipolar occlusion camera example of the teapot. *Top*: Even when two PPC depth images are used (*left*), one for each end of a segment of viewpoints (*middle*), important disocclusion errors occur for an intermediate viewpoint (*right*). *Bottom*: A single EOC image (*left*) captures all samples visible from the viewpoint segment (*middle*), and avoids disocclusion errors (*right*).

An additional limitation of the SPOC and DDOC considers the set of viewpoints for which the cameras gather sufficient samples. The cameras are designed to support viewpoint translations in all directions, however, the application intended range of translations can be overruled by the constructor which has to arbitrate between competing occluders.

To overcome these disadvantages we introduce the epipolar occlusion camera (EOC). Given a 3-D scene, a planar pinhole camera *PPHC,* and a segment of viewpoints *LR*, a non-pinhole EOC is constructed that gathers samples that *PPHC* would see when translating between *L* and *R*. While a PPC gathers the samples visible from a *point*, an EOC gathers the samples visible from a *segment*. The EOC model leverages epipolar constraints to avoid disocclusion errors on individual epipolar lines. The EOC image has a single layer, yet

occluder spacing is not a problem. The EOC provides efficient projection so its image is constructed directly by rendering the scene using the EOC, with hardware support, and *not* by combining several PPC images.

The EOC concept is illustrated in Figure 3.16. Both PPC images miss back wall samples that are visible from intermediate viewpoints. For example, some of the samples occluded by the teapot handle in the left image are then occluded by the teapot body in the right image. These samples are visible in the intermediate view and their absence causes the white gap around the handle in the top right image. The EOC is constructed from the planar pinhole camera *PPHC* of the left image, the segment of viewpoints *LR*, and the scene. The goal is for the EOC to gather the samples visible by *PPHC* as it translates on the viewpoint segment. Due to epipolar geometry constraints, the disocclusion events that occur can be conveniently described as the sum of independent disocclusion events on individual epipolar lines. To take advantage of this fact, EOC rows are defined according to the epipolar lines induced by *LR* on *PPHC*. For each row, the EOC gathers additional samples as needed, according to the occluders crossed by the epipolar line of the row. In the case shown, *LR* is parallel to the *PPHC* rows, thus the epipolar lines are the *PPHC* image rows, and consequently the EOC has the same rows as *PPHC*. The EOC image has more samples than the *PPHC* image on rows where more disocclusion events occur (i.e. rows that cross the handle, the body, and the spout), and the same number of samples on rows without disocclusion events (i.e. the top and bottom rows). Figure 3.17 shows the EOC working on a complex scene. Figure 3.18 shows an EOC built to support forward translation.

3.3.1. Camera Model

The EOC model is constructed from a 3-D scene *S*, a planar pinhole camera *PPHC*, and a segment of viewpoints *LR*. The rays of the EOC are line segments

which do not pass through a common point. The EOC model is encoded with *PPHC*, *LR*, a projection map *PM*, and a ray map *RM*. *RM* stores the ray segments of the EOC and *PM* enables fast projection.

3.3.1.1. Construction Algorithm

The EOC is constructed with the following steps.

EOC (*S, PPHC, LR*) {

1. Render *S* with *PPHC* from *L* to create depth buffer ZB_0
2. For every epipolar line e_v defined by *LR* in ZB_0
    a. For every pixel *u* on e_v
        - If *u* is not a depth discontinuity
            - Update *RM* with ray (*PPHC.Ray*($e_v(u)$))
        - If *u* is a depth discontinuity
            - Compute additional set of rays A_i from (v, ZB_0, u, e_v)
            - Update *RM* with rays A_i
            - Update *PM* using first and last rays A_1, and A_n

}

The first step creates a planar pinhole camera depth buffer ZB_0 of the scene, as seen from the left endpoint of the viewpoint of segments. The rows of the projection and ray maps correspond to the epipolar lines induced in ZB_0 by the translation *LR*. The second step walks on epipolar lines and sets the two maps one row at the time. While no depth discontinuity is found, the *PPHC* ray is simply added to *RM*. For a depth discontinuity, *PM* and *RM* are updated to introduce a set of additional rays.

Figure 3.20 left shows the rays of a row of an EOC constructed for a scene with a single foreground object (*red*) occluding a background object (*green*). The row has the rays of *PPHC* located at *L* except for the additional set of rays (*blue*)

which are created to handle the near to far depth discontinuity encountered when stepping off the foreground object, from P_3 to P_4. The corresponding EOC image row stores all background object samples needed as *PPHC* translates from left to right. EOC and Truth show the rows of a reconstruction from an intermediate viewpoint obtained from the samples of the EOC and from geometry, respectively.

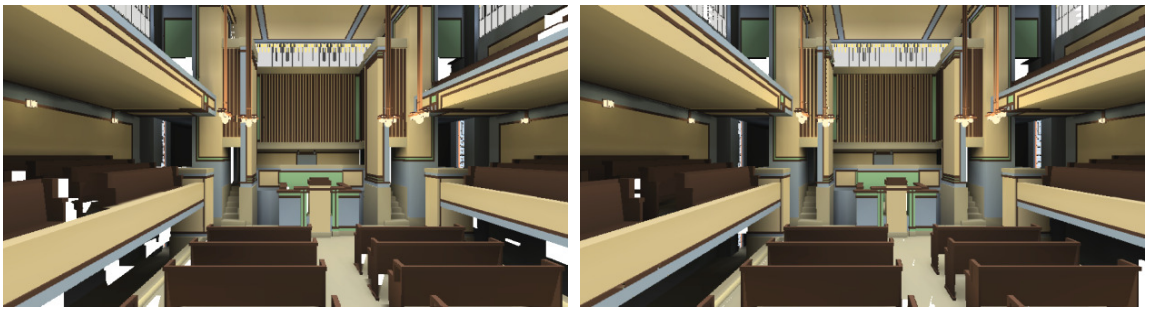


Figure 3.17. Epipolar occlusion camera image of the Unity church. The output images have a viewpoint on the segment *LR* reconstructed once from two depth images with viewpoints *L* and *R* (*left*) and once from an EOC image with segment of viewpoints *LR* (*right*).

The set of additional rays is computed as follows. Let z_f be the depth at P_4. P_1 is computed as the intersection of RP_3 with the plane $z=z_f$. P_0 is the *PPHC(L)* projection of P_1 onto the image plane. For clarity, the image plane is chosen here to be coincident with the hither plane. P_2 is the image plane projection of P_3. The number of additional rays is the length of segment P_0P_2 in pixels. The additional rays are defined as pairs of rays (r_L, r_R), where r_L/r_R passes through *L/R.* The far point of r_R is the same as the near point of r_L, and they are defined by intersecting *R* rays with the $z=z_f$ plane. The additional rays shorten the original *L* rays between P_0 and P_2, but only if the intersection with the blue ray is not visible from *L*. In other words, the blue rays are not allowed to reemerge from underneath the occluder—their function is to sample the occluded regions. Figure 3.20 right shows a case where the occluder is narrow and the viewpoint segment is long enough to reveal all hidden samples. Some r_R blue rays are

clipped by P_0P_1, do not reach z_f, and do not have a r_L pair. In the case shown here, these rays do not find any samples, and they correctly leave a black gap in the EOC. Similar black gaps can be observed in Figure 3.16 and Figure 3.19 to the right of the teapot handle and the pickets, respectively.

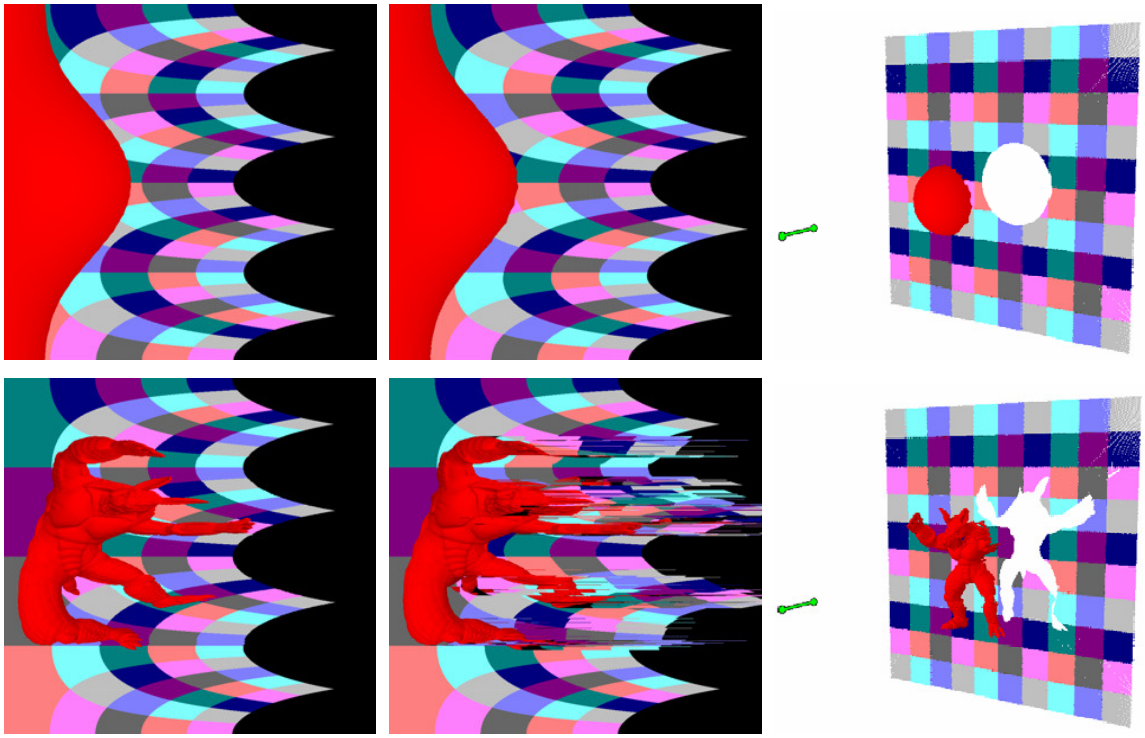


Figure 3.18. Epipolar occlusion camera examples with radial epipolar lines. EOC images constructed for a short (*left*) and a long (*middle*) viewpoint segment aligned with the *PPHC* view direction, and the samples gathered by the EOC shown from a third view (*right*). As before, the EOC disoccludes on rows, now defined by radial epipolar lines.

The ray map *RM* is updated with the set of additional rays A_i by simply appending the rays A_i to the rays that are already in the current row. Each *RM* location has room for a (r_L, r_R) pair of rays. Appending the additional rays effectively extends the row of the EOC image to accommodate hidden samples. Sufficient *RM* locations are used to provide the samples at full resolution, as dictated by *PPHC*.

The function of *PM* is to enable projecting 3-D points in the EOC image where rays have been inserted. *PM* has the same resolution as *PPHC*. A *PM* location stores two offsets (o_0, o_1) and two depth values (z_0, z_1). These four values model the EOC rays visible at a *PPHC* pixel. For example, using Figure 3.20 left again, at the location u_{P0} corresponding to P_0, *PM* stores 0, u_{P2}-u_{P0}, z_f, and z_f, for o_0, o_1, z_0, and z_1, respectively. For P_2, *PM* stores 0, u_{P2}-u_{P0}, z_n, and z_f. For an intermediate ray (not shown), z_1 would be a value between z_f and z_n and the other 3 values remain the same. In order to project a 3-D point P with the EOC, P is first projected with *PPHC* to find its corresponding *PM* location (o_0, o_1, z_0, z_1) along epipolar line e_v at pixel u (row v and column u). If the depth z_P of P is less than z_0, then the EOC image projection of P is ($u+o_0$, v). If z_P is greater than z_1, P projects to ($u+o_1$, v). If z_P is in between z_0 and z_1, P projects to the interval [($u+o_0$, v), ($u+o_1$, v)].

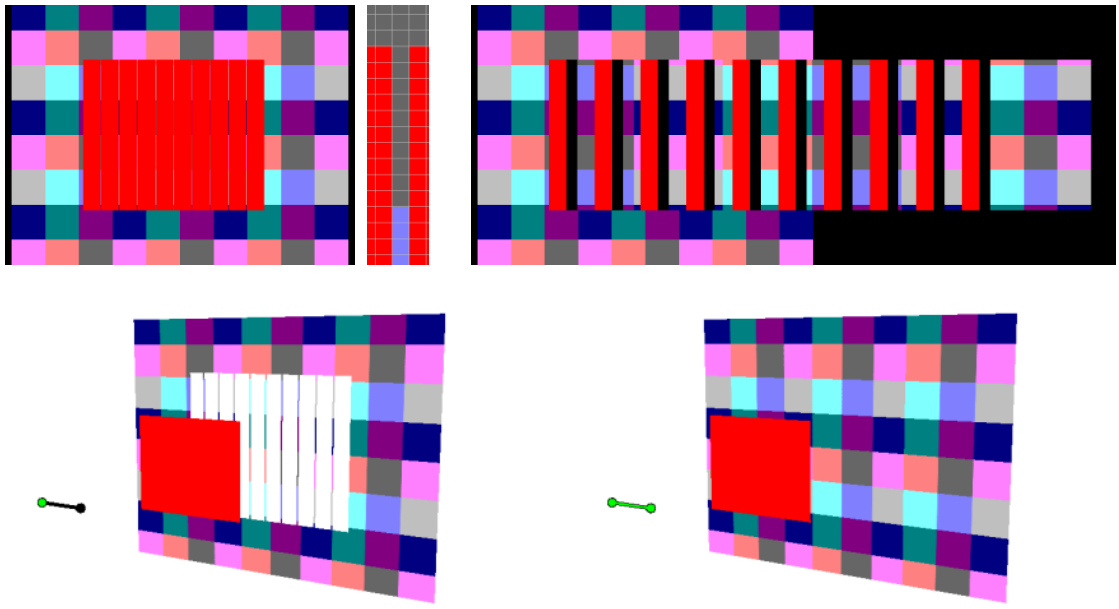


Figure 3.19. Epipolar occlusion camera used to alleviate disocclusions within a one pixel gap. *Top*: Regular depth image and a magnified fragment (*left*), and an EOC image (*right*). *Bottom*: The depth image misses a considerable part of the back wall (*left*) while the EOC (*right*) does not, preventing disocclusion errors.

Once an additional set of rays is inserted into *RM*, *PM* is updated with the first and last of the additional rays. In Figure 3.20 left, the *PM* locations that have to be updated are those corresponding to the segment P_0P_2. For a given location, a candidate z_0 value is computed by intersecting the location *L* ray with ray A_1. The candidate value replaces z_0 if closer than z_0. The candidate z_1 value is given by z_f and it replaces z_1 if farther. Offset o_1 is updated to $o_0 + u_{P2}-u_{P0}$ if this candidate value is larger. Offset o_0 does not change.

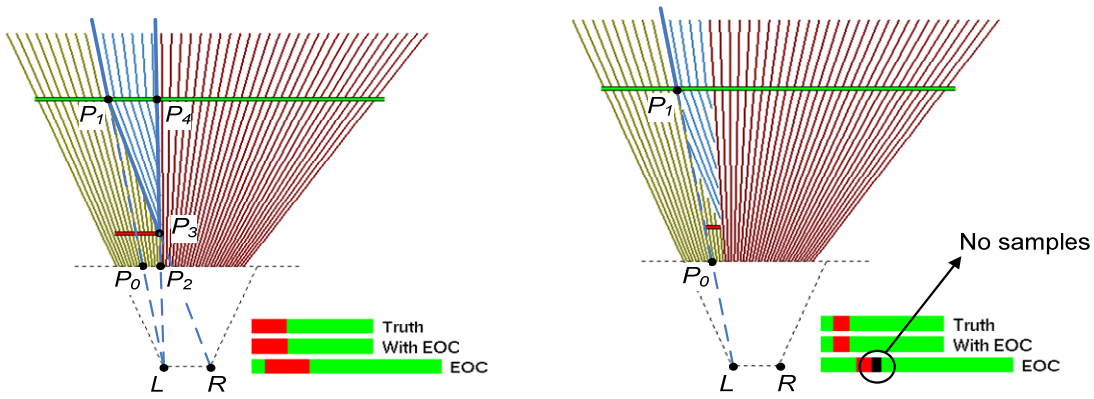


Figure 3.20. Visualization of the epipolar occlusion camera rays for one epipolar line. Visualization of an EOC row for a wide (*left*) and narrow (*right*) occluder.

Figure 3.22 gives another example of an EOC image. The bunny is not visible from the left or right viewpoints. The EOC opens the doors to sample the bunny which is needed as the viewpoint translates from left to right. The EOC image avoids disocclusion errors which are considerable when the two regular depth images are used, since they do not sample the bunny. Like in the case in Figure 3.19, the small gap between the two red occluders precludes any significant disocclusion when a DDOC is used. Figure 3.18 shows a case when the epipolar line (i.e. the intersection between the *PPHC* image plane and the segment of viewpoints *LR*) is inside the *PPHC* image, which causes radial epipolar lines.

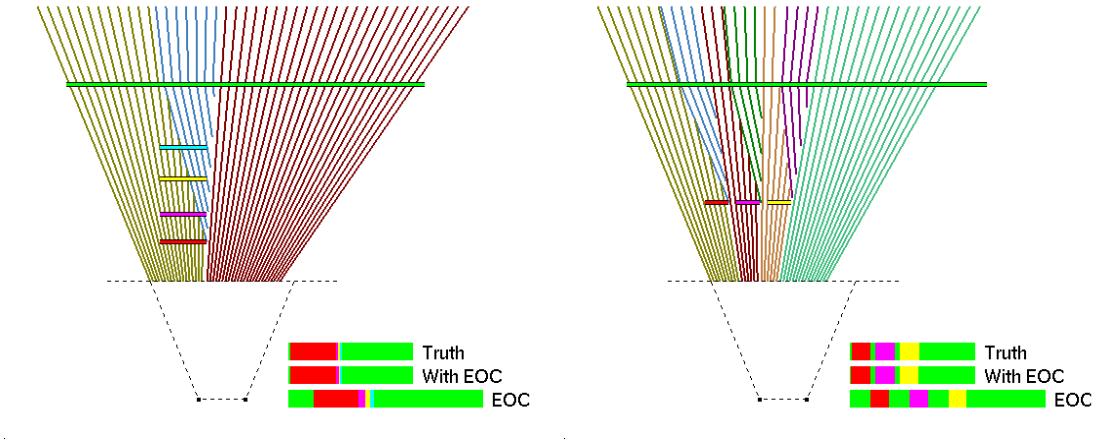


Figure 3.21. Visualization of extreme cases of occlusion alleviated by the epipolar occlusion camera. Visualization of an EOC row for a stack of occluders hidden by the closest one (*left*), and side by side occluders separated by small gaps (*right*).

3.3.1.2. Performance

The EOC rays sample the space fully and without any redundancy resulting in good disocclusion capability. The EOC is not conservative, since it addresses only depth discontinuities that appear in *L*. However, the rays inserted handle additional "unforeseen" disocclusion errors well. For example, in the left image in Figure 3.21, the blue, yellow, and pink occluders are hidden by the red occluder and are not visible from *L,* so they are not addressed explicitly. Yet the disocclusions they cause are handled well by the additional rays introduced to handle the depth discontinuity caused by the red occluder. The right case corresponds to the picket fence shown in Figure 3.19.

The EOC rendering algorithm is implemented on the GPU as described in Section 3.4.2. A vertex shader implements the projection of the vertices of a triangle with PPHC. A geometry shader subdivides the triangle into spans and

projects the spans with the projection map. A pixel shader intersects the span with the one or two ray segments of each pixel of the projected span. Table 3.1 shows the algorithm performance for a variety of scenes. Performance depends on the number and size of triangles which determines the number of spans and on the number and magnitude of depth discontinuities. That impacts the tightness of the EOC projection and implicitly the amount of overdraw (i.e. empty ray span intersections).

Table 3.1. Epipolar occlusion camera performance.

Scene	Figure	Triangles	Render Time (ms)
Teapot	Figure 3.16	1K	19
Unity	Figure 3.17	110K	1,793
Picket Fence	Figure 3.19	1K	16
Sphere	Figure 3.18, top	1K	17
Armadillo	Figure 3.18, bottom	346K	1,130
Double Door	Figure 3.22	69K	68
		16K	25
		4K	20
		1K	15

The EOC makes room on epipolar lines for hidden samples that are needed as the viewpoint translates. We define the width of the EOC image (also the width of the ray map) as the maximum width of any of its rows. The width *w* of a row verifies the following inequality $w < w_0 + Ns$, where w_0 is the width of *PPHC*, *N* is the number of depth discontinuities on the row, and *s* is the maximum shift caused by one depth discontinuity. The value of *s* can be computed as $s = LR \cdot (zF - zN) / zF$, where *LR* is the length of the viewpoint segment, and z_F

(z_N) is the farthest (closest) z_f (z_n) over all depth discontinuities. In practice, the width of an EOC image is typically less than $2w_0$.

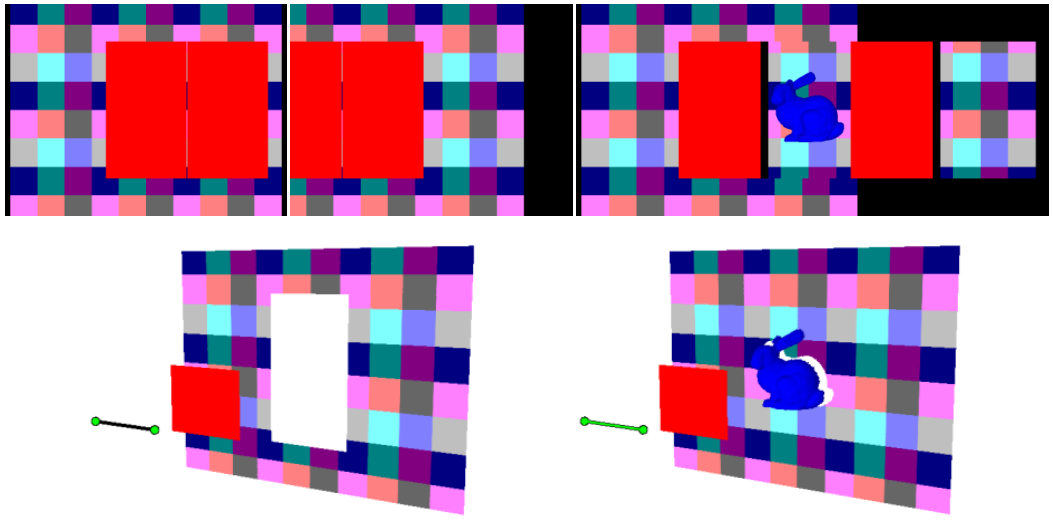


Figure 3.22. Epipolar occlusion camera used to alleviate disocclusion events within a small gap. *Top*: PPC images (*left and middle*) at the endpoints of the view segment of the EOC (*right*). *Bottom*: Visualization of the samples gathered by the two depth images (*left*) and the EOC (*right*).

A projection map location stores 2 short integers and 2 floats for a total of 12 bytes. A ray map location stores 2 short integers and 4 floats for a total of 20 bytes. Consequently the memory size of an EOC constructed for a *PPHC* with resolution w_0 x h_0 is $12 \cdot w_0 h_0 + 20 \cdot 2 \cdot w_0 h_0 = 52 \cdot w_0 h_0$ bytes.

3.3.2. Unprojection

An EOC image sample is converted to a 3-D point in one of two ways according to the data it stores. If each image sample (*u, v, z*) is enhanced with a single offset value, the sample can be undistorted and a 3-D point computed by unprojection with *PPHC*. If instead the EOC image stores depth discontinuities, the EOC model is recreated and the 3-D point is computed by unprojection along the corresponding ray stored by the ray map.

3.4. Rendering Occlusion Camera Images

3.4.1. The Single Pole Occlusion Camera and Depth Discontinuity Occlusion Camera

There are a wide variety of rendering methods for both the SPOC and DDOC. The choice of technique depends upon the desired levels of quality, accuracy, and performance. A detailed description and analysis of the available techniques is presented in Chapter 6.

In the case of the SPOC, the simple distortion model allowed easily establishing a conservative bounding box for projected triangles. Thus, a hybrid raycasting approach was chosen as the original method for producing images. In subsequent work however, triangle subdivision was used instead due to the large performance gains which are traded for only minimal loss of accuracy.

For the DDOC, having a sample-based distortion made calculating a compact and conservative bounding box for projected triangles difficult. To address this limitation triangle subdivision was chosen from the beginning as the rendering method of choice. To minimize the error, triangles where subdivided until they were approximately 1 output pixel in size, making the result almost indistinguishable from a raycasting result.

3.4.2. The Epipolar Occlusion Camera

The epipolar decomposition of the EOC makes triangles no longer contiguous on the image. The lack of row to row continuity precludes the use of any triangle-based rendering methods, making rendering an EOC a more challenging problem than rendering an SPOC or DDOC. Point-based rendering could be used, assuming enough scene points exist. Instead, a modified version of the

hybrid raycasting approach was used in rendering EOC images. Since a rendered triangle is no longer necessarily contiguous on an EOC image, the resulting bounding box for that triangle can be quite large. Instead, using the epipolar decomposition allows rasterization on segments with tight bounding boxes in the output domain.

Given the 3-D scene *S* and the *EOC*(*PPHC, LR, PM, RM)*, the epipolar occlusion camera image is computed by processing each triangle *t* in *S* with the following algorithm.

1. Project *t* with *PPHC* to *t'*
2. For every epipolar span e_0e_1 of *t'*
    a. Project e_0 e_1 with *PM* into *RM* to $e_0'e_1'$
    b. Rasterize $e_0'e_1'$ by intersecting e_0e_1 with rays $RM(e_0' \ldots e_1')$

The triangle is split into spans according to the epipolar lines induced by *LR* into *PPHC*. A span e_0e_1 is projected with the EOC to find the *RM* segment $e_0'e_1'$ that stores the EOC rays that could intersect e_0e_1. A segment P_0P_1 is projected with an EOC by first projecting its endpoints P_0 and P_1 as described in the previous section. Let $[P_{00}, P_{01}]$ and $[P_{10}, P_{11}]$ be the projections of P_0 and P_1 (a point projects with an EOC to a point or segment, so P_{00} and P_{01}, and P_{10} and P_{11} could be identical). The projection of P_0P_1 is the bounding segment $[P_{00}, P_{11}]$. For every *RM* location e_j between e_0' and e_1' the span is intersected with the rays at e_j. If an intersection is found, the sample is written in the EOC image at the location corresponding to e_j.

3.5. Occlusion Camera Applications

Like PPC images, occlusion camera images have a single layer, are non-redundant, and can be rendered efficiently by projection followed by rasterization with hardware support. Unlike a PPC image, the occlusion camera images have

samples for clusters of viewpoints and not just for a single viewpoint. There are several applications, inherited from PPC depth images, in which the occlusion cameras can outperform the PPC.

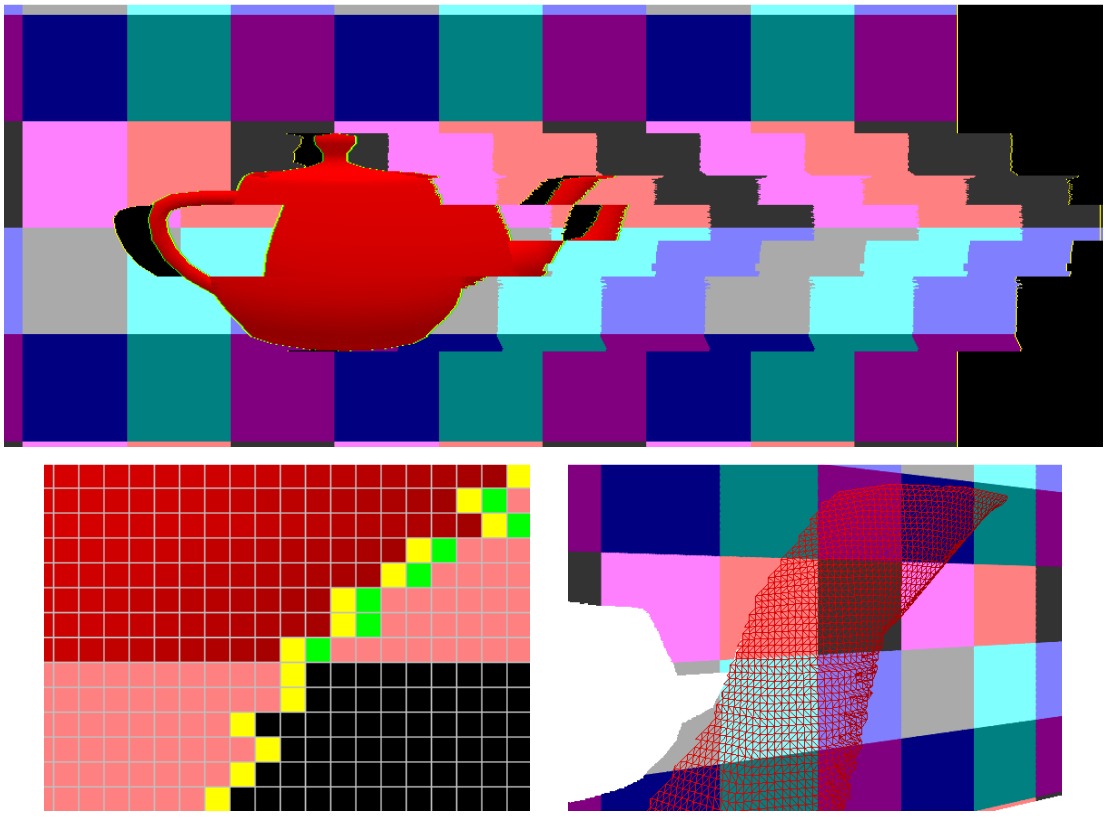


Figure 3.23. An epipolar occlusion camera image used as a geometry replacement. The EOC image (*top*) has the spout magnified with a pixel grid and span start/end points highlighted with green/yellow (*bottom, left*). A wireframe visualization of the reconstructed mesh is shown as well (*bottom, right*).

3.5.1. Geometry Replacement

Depth images are a convenient solution to the challenging problems of level-of-detail adaptation and occlusion culling, as work in image-based rendering by 3-D warping [102], impostors [95, 127], and geometric detail texturing [112] have shown. However, the occlusion culling provided by a PPC depth image is too

aggressive and it quickly becomes obsolete as the viewpoint translates, which leads to the problem of disocclusion errors. For all of these applications, substituting PPC depth images with occlusion camera depth images brings the advantage of avoiding disocclusion errors. The occlusion camera computes occlusion culling that remains valid as the viewpoint translates away from the reference viewpoint.

Once an occlusion camera image is converted into 3-D points, point-based rendering techniques (e.g. surfels [119], qsplat [139]) can be used to produce high-quality output images at interactive rates.

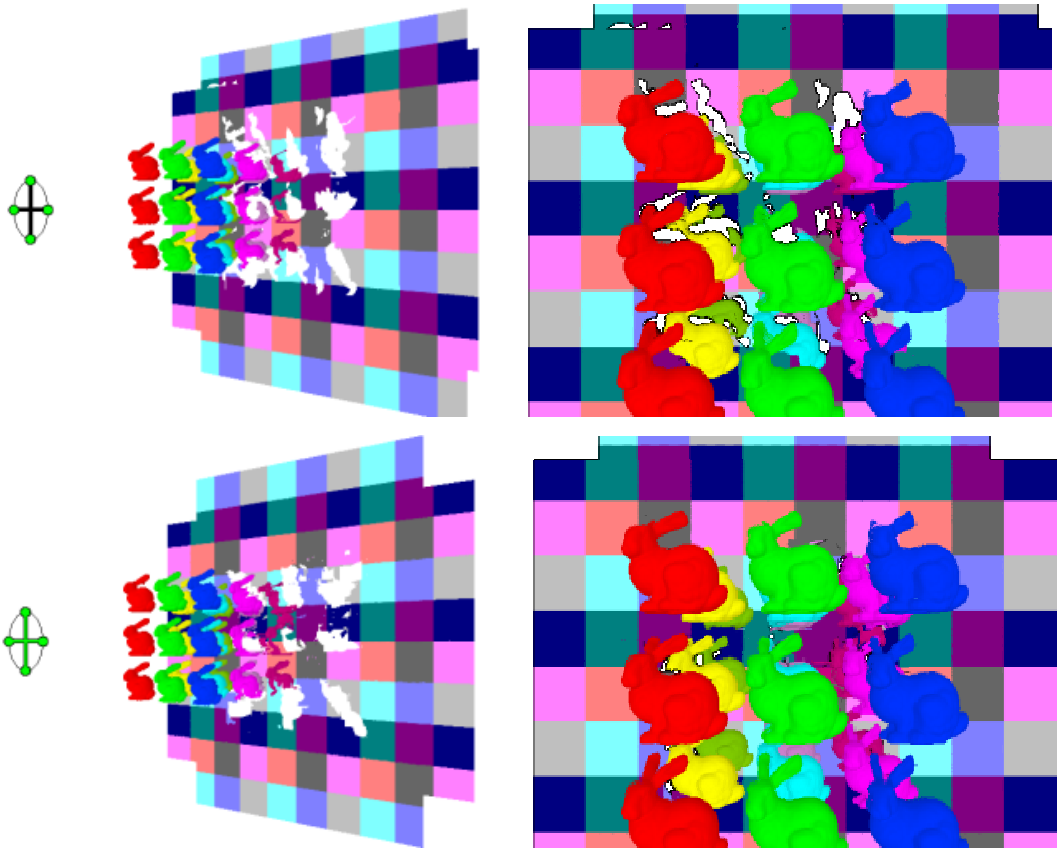


Figure 3.24. Two crossing epipolar occlusion camera images used for geometry replacement. *Top*: Four depth images rendered from the corners of the cross (*left*) leave important disocclusion errors (*right*). *Bottom:* Two EOCs do not sample the entire back wall (*left*) but most disocclusion errors are avoided (*right*).

Like PPC depth images, SPOC and DDOC depth images both have implicit connectivity making triangle mesh rendering trivial. Even though the rows of the EOC are misaligned, sufficient coherence remains to enable a straight forward triangulation by connecting corresponding spans on adjacent rows. Figure 3.23 shows how the triangulation of corresponding spans realigns the spout samples to form a mesh.

In order to increase the range of translations supported, configurations of multiple occlusion cameras can be used to provide even better scene coverage. A combination of two EOC images, one with a vertical and one with a horizontal viewpoint segment, is particularly effective (Figure 3.24).

3.5.2. Geometry Approximations for High-Quality Rendering Effects

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 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 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, like those needed in the case of reflections, refractions, and relief texture mapping. 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 PPC, 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 occlusion cameras. Such occlusion camera depth images offer a high-fidelity approximation of scene geometry while construction and rendering costs remain low. Once the constraints of the PPC are relaxed, the rays of an occlusion camera can be used instead to sample all surfaces exposed by the application. To ensure construction and rendering efficiency, the occlusion camera is designed to provide a fast projection operation. This enables constructing the occlusion camera depth image in feed-forward fashion by projection followed by rasterization. The closed-form, unambiguous projection of the occlusion camera is leveraged a second time, during rendering, to compute the projection of the ray onto the occlusion camera image. As in the case of PPCs, the ray / occlusion camera depth image intersection is found by walking on the one-dimensional projection of the ray. Unlike for PPCs, ray projection is not a straight line, though this does not raise intersection costs significantly.

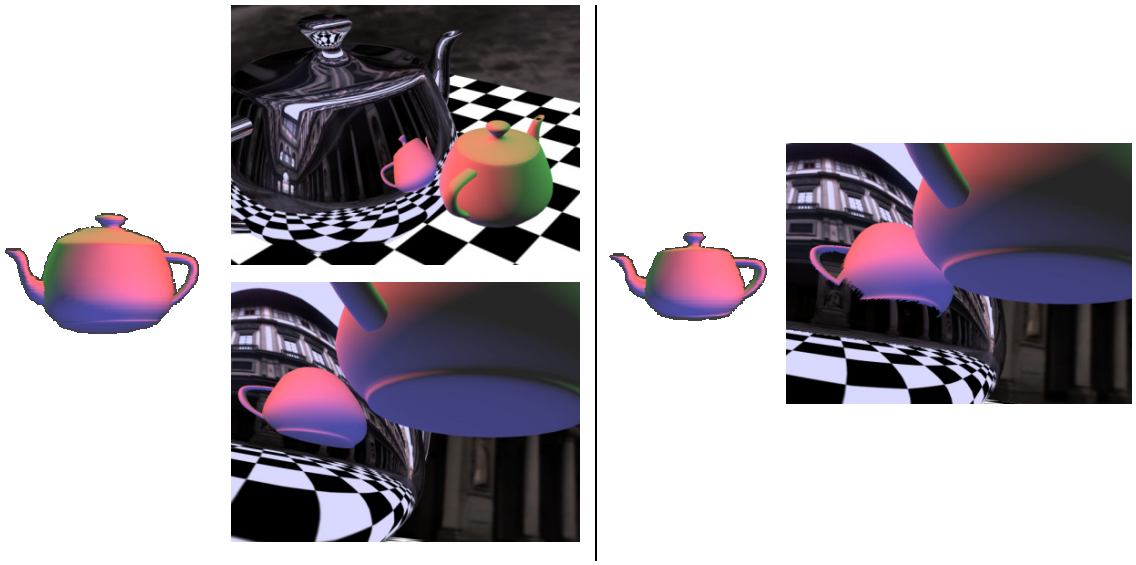


Figure 3.25. Reflection rendered with an occlusion camera geometric approximation. An occlusion camera depth image of the teapot used to render reflections (*left*) is compared to a PPC depth image and the reflections rendered with it (*right*). The occlusion camera depth image captures the lid and bottom of the teapot and produces quality reflections.

The advantages of occlusion camera depth images are demonstrated in the contexts of reflection, refraction, and relief texture mapping. In Figure 3.25 the occlusion camera 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 occlusion camera depth image is found by searching along the curved projection of the ray (Figure 3.28). Refractions are supported similarly by intersecting the occlusion camera depth image with refracted rays (Figure 3.27). Occlusion cameras greatly enhance the modeling power of relief texture mapping (Figure 3.26). A single-layer occlusion camera 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.

Abandoning the pinhole constraint presents the opportunity to design the camera model to obtain the depth image that is best suited for the application and dataset at hand. For these applications we construct depth images by using the SPOC.

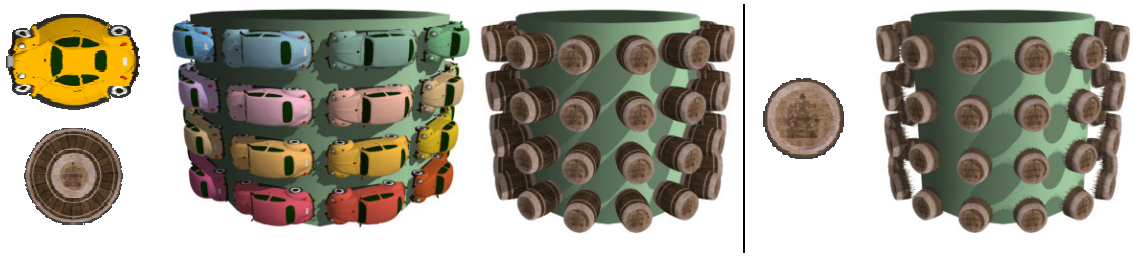


Figure 3.26. Occlusion camera image used for relief texture mapping. Occlusion camera relief texture maps and frames rendered with them (*left*) and conventional relief texture mapping (*right*).

3.5.2.1. Prior Work

3.5.2.1.1. Image-Based Geometry Approximation

Maciel and Shirley [95] 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 [30] 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.

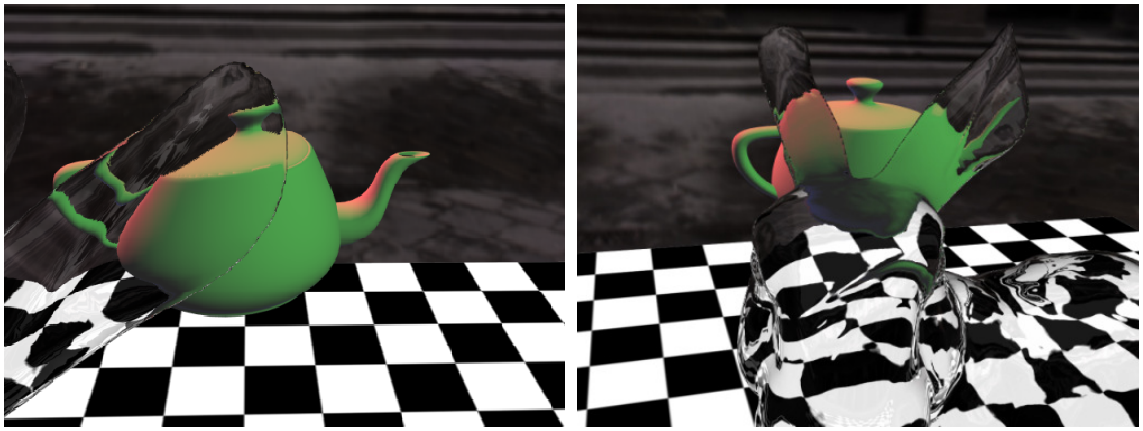


Figure 3.27. Refractions calculated using an occlusion camera image. Refractions rendered with an occlusion camera depth image of the teapot similar to the one shown in Figure 3.25.

Depth images [102] 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.

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

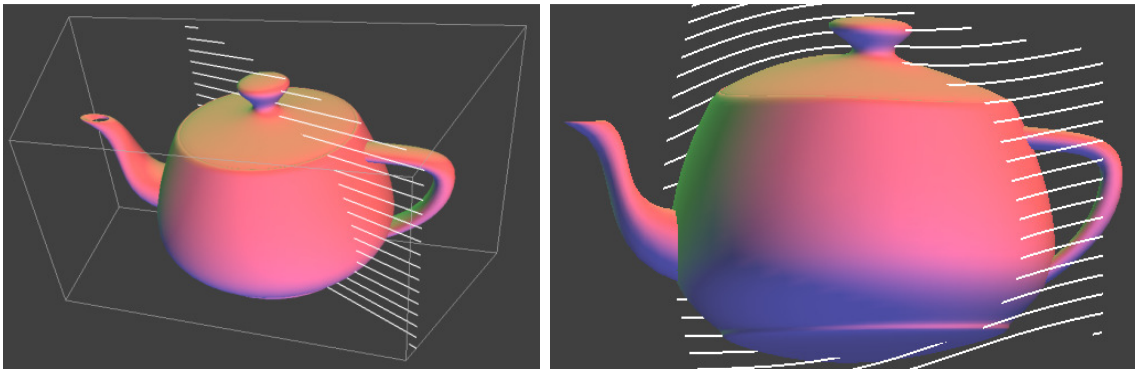


Figure 3.28. Visualization of the single pole occlusion camera rays. The rays in 3-D (*left*) and of their curved projection on the SPOC depth image (*right*) used in Figure 3.25.

3.5.2.1.2. 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 [165], image-based rendering [28, 53, 84], projection [110], and reflected/refracted scene approximation. We only discuss group 4, since it is most relevant.

Environment mapping [17] 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 [16], but few environments are spherical so the fidelity is still quite limited. The scene approximation can be improved using depth images [127, 153]. 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 rays interact with the refractor at least twice—once entering and once leaving the object. Several techniques have been developed for computing the second refractor interaction at interactive rates, including pre-computed distance fields [24], GPU raytracing [136], and image-space approximations [171].

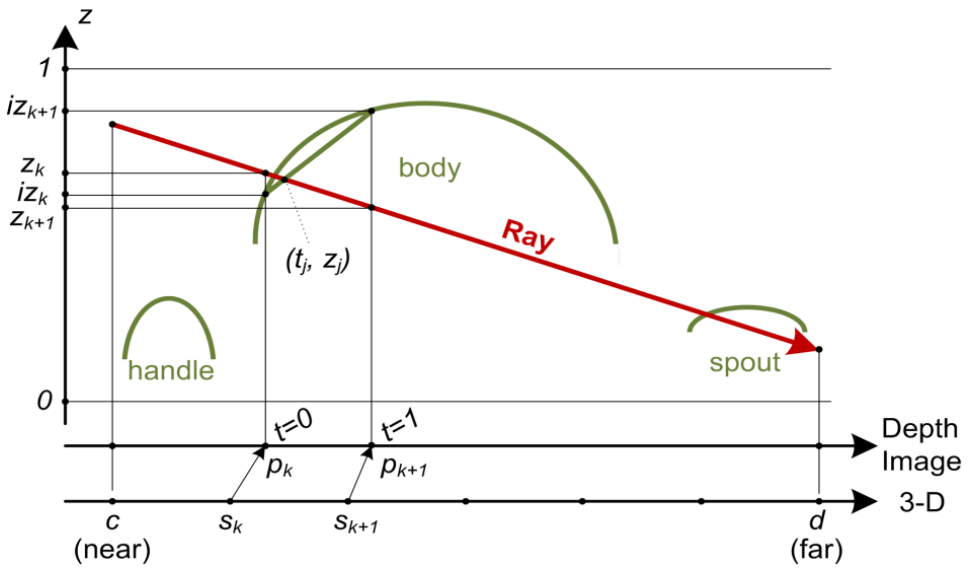


Figure 3.29. Ray / occlusion camera depth image intersection.

Another rendering effect that requires intersecting depth images with individual rays is relief texture mapping [125]. 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. 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 [124]. The extension works well when complex detail can be captured in a few layers, as is the case for a chain link fence for example. The 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 occlusion camera relief textures could be developed to collect the advantages of both techniques.

3.5.2.2. Ray Intersection

Like a regular depth image, an occlusion camera 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 an occlusion camera depth image *OCC* is computed as follows:

1. Clip the segment (a, b) with the bounding volume of *OCC* to obtain the segment (c, d), see Figure 3.29.
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}):
    a. Project s_k and s_{k+1} to depth image at $p_k = (u_k, v_k, z_k)$ and $p_{k+1} = (u_{k+1}, v_{k+1}, z_{k+1})$.
    b. Lookup image depths iz_k, iz_{k+1} at (u_k, v_k), (u_{k+1}, v_{k+1}).
    c. 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).
    d. 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 occlusion 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}). For these applications, this generic algorithm is used on the SPOC.

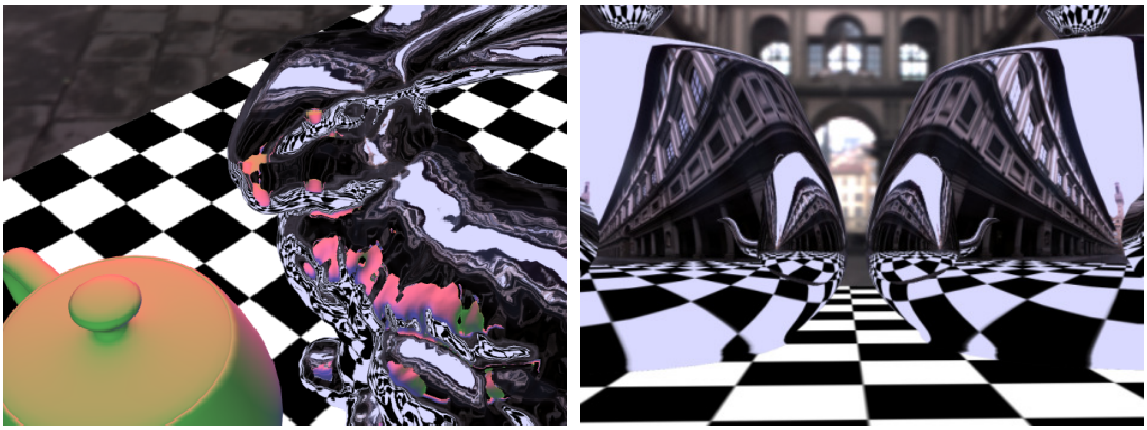


Figure 3.30. Multiple and second order reflections. Our algorithm implicitly supports multiple reflections (*left*) and can easily be modified to support second order reflections (*right*).

3.5.2.3. Applications to Interactive Rendering

Occlusion camera depth images accelerate reflection, refraction, and relief texture mapping as follows.

3.5.2.3.1. Reflection and Refraction

To render a frame of a scene with specular reflections the first step is to update the depth images approximating the dynamically reflected geometry. 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. Multiple reflections of the same object are obtained at no extra cost (Figure 3.30, left). Fully dynamic scenes are supported since our method does not require any pre-computation involving the reflector and the occlusion camera depth images are computed efficiently. Higher-order reflections are supported by storing per pixel normals. The normal at the first intersection point is used to create a second order reflected ray which intersects the depth images again (Figure 3.30, right).

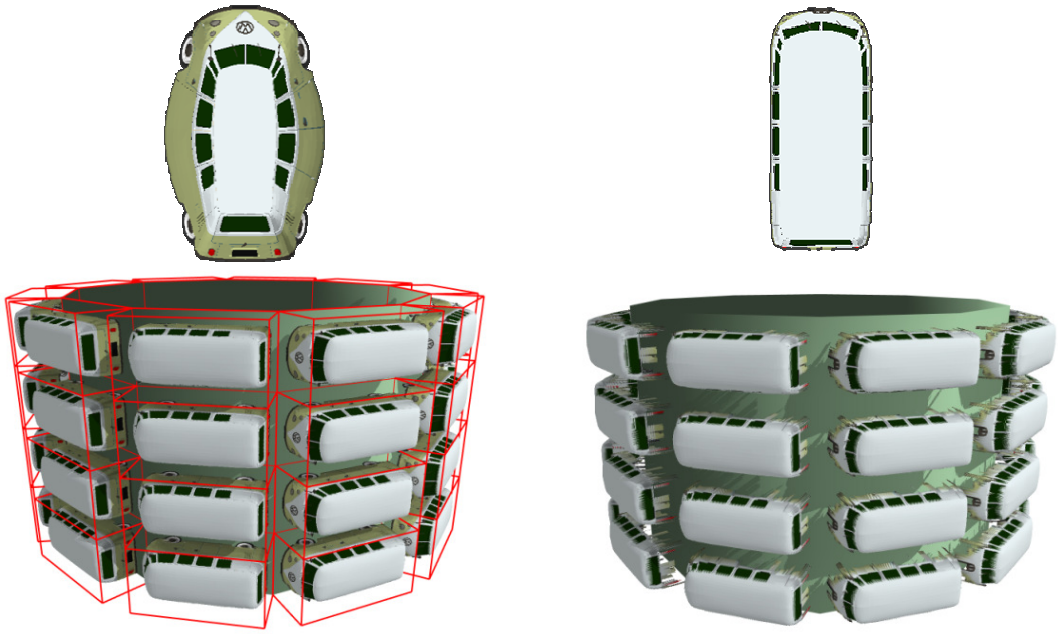


Figure 3.31. Occlusion camera image used for relief texture mapping is compared to a planar pinhole camera image. SPOC relief texture (*top, left*) and output image with relief bounding box visualization (*bottom, left*) compared to a conventional relief texture (*right*).

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 3.30 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. If the teapot were spinning or if there were multiple reflectors surrounding the teapot, the best solution would be to use two depth images capturing complementary halves of the teapot.

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 [171]. 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.

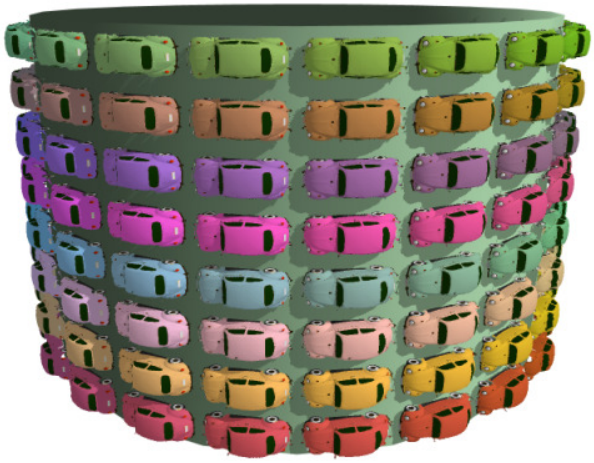


Figure 3.32. Short relief example.

3.5.2.3.2. 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 3.31, left). 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.

Occlusion camera relief textures capture complex objects in a single layer. Figure 3.31, right shows that a conventional relief texture misses the wheels and severely under-samples the sides of the car.

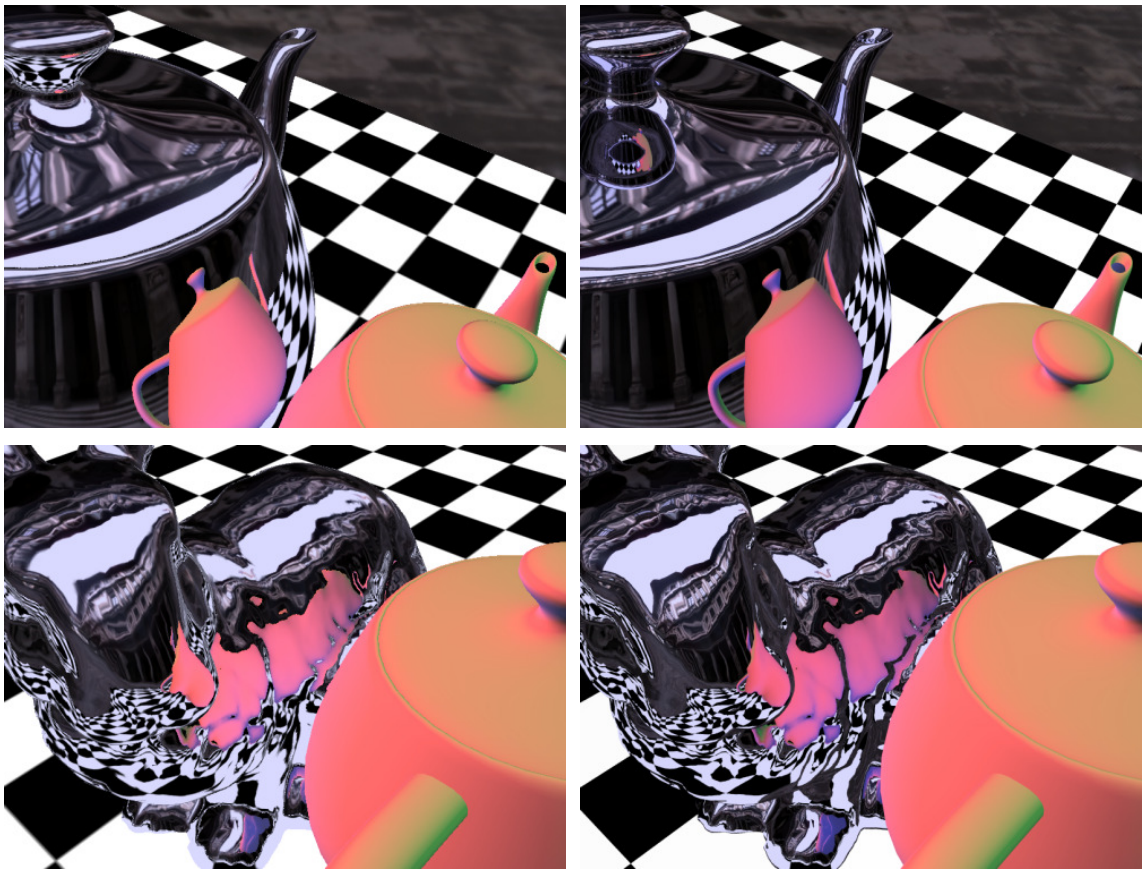


Figure 3.33. Comparison of the quality of reflections between raytracing and the occlusion camera. Our method (*left*) compares favorably to raytracing (*right*).

3.5.2.4. Results and Discussion

3.5.2.4.1. Quality

Occlusion camera depth images enable quality reflections, refractions, and relief texture mapping. Figure 3.33 shows that the method achieves results comparable to ray tracing. The main limitations of the approach are as follows.

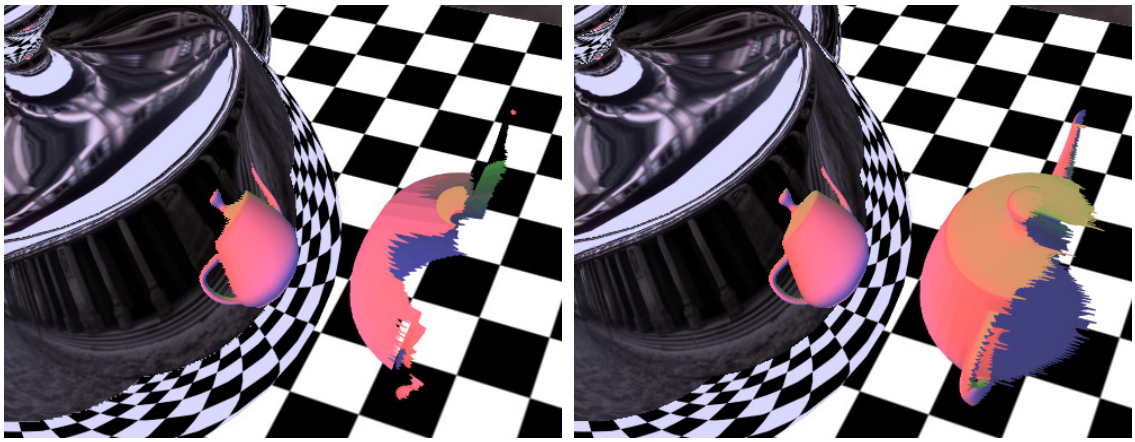


Figure 3.34. Visualization of the samples stored in a planar pinhole camera image to those of an occlusion camera image. The SPOC depth image (*right*) store far more samples than the PPC (*left*) resulting in a higher-quality approximation.

Absent self-reflection

Although the 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 3.34). 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.

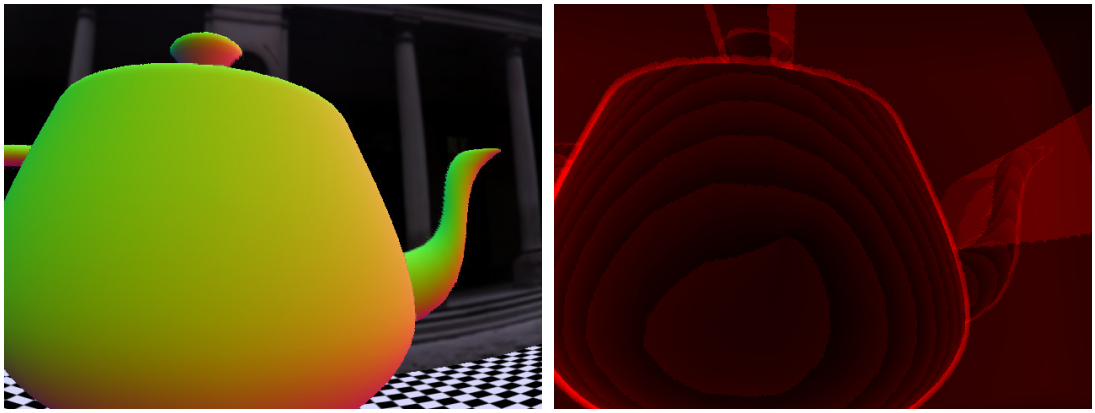


Figure 3.35. Visualization of the workload for calculating a reflection. Diffuse teapot reflected in body of large teapot (*left*) and visualization of number of intersection steps per pixel (*right*).

Undersampling

Like all sample-based methods, the quality of the results obtained with occlusion camera depth images is contingent upon adequate sampling. The SPOC sampling rate is uniform and controllable.

Missing samples: The most visible artifacts in occlusion camera 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 3.31 occludes some of the car body in the relief texture, which causes the shimmering “rubber band” surface. 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.

3.5.2.4.2. Performance

The timing information reported here was collected on a 3.4 GHz Intel Xeon PC with 2 GB of RAM and an nVidia 8800 Ultra card. An important performance

factor is the number of steps taken 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 occlusion camera 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 3.35 illustrates the number of steps for a 512^2 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 3.36 shows reflection silhouette quality for various fine step sizes.

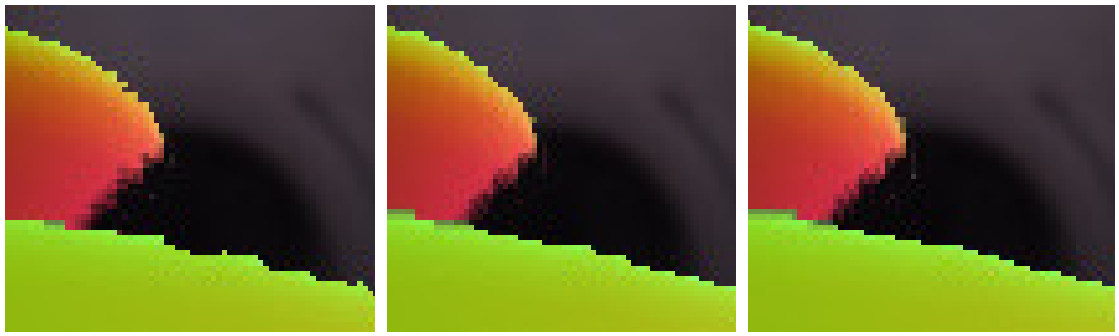


Figure 3.36. Silhouette detail with fine steps of 1, ½, and ¼ pixels.

Performance depends on output image resolution and on the fine step size as shown in Table 3.2. Performance was measured on a typical path for the scene shown in Figure 3.25. Eight sample multi-sampling anti-aliasing (8x MSAA), a 512^2 SPOC depth image, and a coarse step of 6 pixels were used. For an output resolution of 640x480, with 8x MSAA, the average frame rate for SPOC depth

images of resolution 128^2, 256^2, 512^2, and $1{,}024^2$, is 55.8, 36.6, 26.5, and 16.1fps, 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, the average frame rates are 22 and 17.3 fps for no anti-aliasing and 16x MSAA, respectively.

Table 3.2. Frame rate variation with output resolution and fine step size. This data is for the scene in Figure 3.25.

Resolution	**640x480**			**800x600**			**1,024x768**		
Fine	1	1/2	1/4	1	1/2	1/4	1	1/2	¼
Average	26.5	20.7	15	20.9	16.5	11.6	15.2	11.8	8
Minimum	18	14	8	14	10	4	10	8	6
Maximum	54	46	36	48	15	34	40	34	28

We construct occlusion camera 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 3.26 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 512^2. For 20, 10, and 1 car performance is 26, 51, and 219 fps, respectively. All of the examples shown used tall relief which implies long ray projections. For scenes with short relief performance is even higher. For example, for 160 cars half the size (Figure 3.32) performance is 46 fps.

3.5.3. The Soft Shadow Occlusion Camera [105]

The most common approach for generating shadows in interactive applications is shadow mapping. Shadow mapping has the advantage of being a very efficient

method for generating shadows, but it has some serious disadvantages as well. The first problem is that the uniform sampling of a shadow map can cause aliasing effects on the output image. More importantly, from the occlusion camera perspective, a shadow map returns a binary in-or-out of shadow for any given 3-D point. This limits shadows to only hard shadows.

The hard shadow assumption means that all light sources are infinitely small points. In reality, light sources have some non-zero area to them. This non-zero area produces a shadow transition region, known as the penumbra, from fully in to fully out of shadow. This limitation is caused by the use of the PPC which only captures a single viewpoint for the shadow map.

To address this limitation, Mo et al [105] adapted the DDOC to the application of soft shadows. They did so by replacing the PPC in the shadow map with a DDOC. Since the DDOC captures samples visible from a locus of viewpoints, it would follow that the DDOC used in a shadow map would represent a light source with a non-zero area.

3.5.4. Image Warping for Compressing and Spatially Organizing a Dense Collection of Images

Computer graphics applications such as telepresence, virtual reality, and interactive walkthroughs require a 3-D model of real-world environments. Students can “visit” famous historical sites, such as museums, temples, castles, battlefields, and entire history rich cities; archeologists can capture excavation sites as they evolve over time; soldiers and fire fighters can train in safe simulated environments; real estate agents can show potential buyers the interiors of homes for sale via the Internet; and, people all over the world can enjoy virtual travel and multi-player 3-D games. Thus, a growing desire exists for methods which can efficiently capture important and visually stunning environments.

In recent years, image-based rendering (IBR) approaches have addressed this problem [1, 7, 19, 53, 84, 100, 102, 145]. They do so by using the plenoptic function directly resampling images to generate new views, thus avoiding the need for a detailed geometric model.

IBR techniques require a large collection of images that must be efficiently stored and accessed. In particular, for interactive walkthroughs image access cannot be limited to coincide with the capture paths, but instead requires access along arbitrary viewpoint paths. Furthermore, disk-to-memory bandwidth limitations require algorithms that reduce both the size of the images on disk and the amount of data that must be transferred to main memory as a virtual observer navigates through a captured 3-D environment.

Traditional compression techniques do not provide both access flexibility and full exploitation of data redundancy. Image compression, such as JPEG [160], JPEG2000 [65], and 2-D wavelets [158] exploit intra-image redundancy to reduce individual image sizes but do not take advantage of inter-image redundancy. Video compression achieves a significant improvement in overall compression by encoding sparse reference images and using motion-estimation algorithms to determine the coherence for interpolation between reference images (e.g., MPEG [82]). However, motion-estimation is expensive and is intended for pre-determined linear sequences of images, making it ill-suited for image access along arbitrary viewpoint paths.

We propose a spatial image hierarchy combined with a model-based compression algorithm that provides quick access to images along arbitrary viewpoint paths during interactive walkthroughs and enables efficient compression of high-resolution images whose centers of projection (COPs) irregularly sample a plane (Figure 3.37). Specifically, our method arranges reference images and residual images into a binary tree. A captured image is extracted via a sequence of image warping operations. Each operation warps a

reference image to the viewpoint of a residual image and the two are added together. We eliminate the need for costly motion estimation by using a simple user-supplied geometric proxy of the environment. The proxy, which may consist of only a dozen polygons approximating the environment, helps compensate for the translation between the two viewpoints and provides an inexpensive but accurate mapping from the reference to the residual image. We further reduce the size of the residual images by employing occlusion camera reference images (OCRIs). These images store samples visible from the reference viewpoint as well as samples likely to become visible from nearby viewpoints. Warping OCRIs instead of regular images produces more compact residuals when disocclusions occur and thus may achieve improved overall compression.

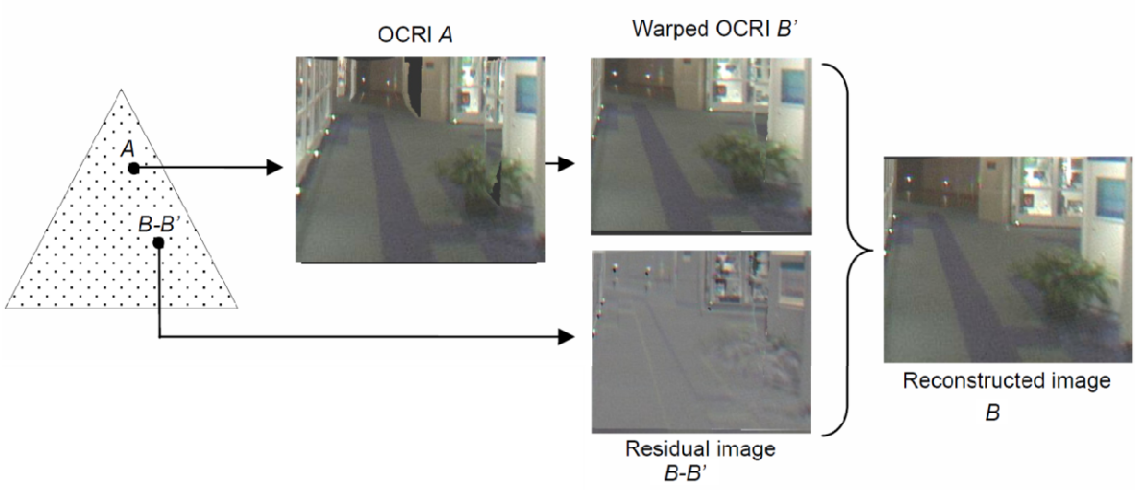


Figure 3.37. Diagram of the system used to compress images. The system uses a spatial image hierarchy combined with a compression scheme that uses image warping and coherence to provide quick efficient image access for IBR applications.

Residual images primarily account for surfaces that are visible from the viewpoint of the residual but not from the reference viewpoint. In the case of non-trivial scenes, even small viewpoint translations disocclude many surfaces that are not sampled in the reference image. Furthermore, the disocclusions are not grouped

in one contiguous region of missing samples, but are rather scattered throughout the scene. Therefore, when regular reference images are used, residual images have to compensate for a substantial number of samples and are difficult to compress.

An OCRI is built using a virtual non-pinhole camera model that “sees” around occluders to gather samples that are hidden but close to the edge of their occluder and therefore are likely to become visible even for small viewpoint translations. For synthetic imagery, a virtual camera model is implemented using graphics hardware. For real-world imagery, the proxy is used in place of the synthetic model to support creating a virtual camera model. In both cases, OCRIs, like regular depth images, have a single layer and therefore share the advantages of bounded number of samples, implicit connectivity, and efficient incremental processing. When disocclusions are present, these additional samples reduce the number of missing samples for which the residual has to compensate, producing residual images with less energy.

In experiments with environments of 2,000 to 10,000 omnidirectional images, $1{,}024^2$ pixels each, our method gives nearly a factor of a 100-to-1 compression without significant loss in quality. This work extends the previous work [5] by adding the support of a model-based compression algorithm based on OCRIs.

3.5.4.1. Previous Work

Several image-based rendering systems use schemes for organizing and compressing images. For example, the Lightfield [84] and the Lumigraph [53] form flat hierarchies of images captured in front of an object of interest. The original Lightfield paper exploits image redundancy by using vector quantization [50] followed by Lempel-Ziv [177] encoding. For interactive rendering, the entire dataset is first decoded using Lempel-Ziv followed by random image

decompression using the vector quantization codebooks. Vector quantization achieves only about 6:1 to 24:1 compression for their datasets.

Subsequently, more elaborate compression algorithms have been proposed for Lightfields. Magnor and Girod [97] describe a DCT-based coder for regular Lightfields using datasets of 1,024 images. The images are arranged into a quad-tree that encodes images as either I- or P-frames. Images of 256^2 pixels are decoded and reconstructed interactively. Optionally, disparity information can be estimated and used to further compress images, although interactive performance is lost. For display, all I-frames and residuals are decoded and stored in memory. This approach achieves 100:1 to 1,000:1 compression depending on desired quality and scene characteristics.

Peter and Strasser [118] describe a wavelet-based compression algorithm for the same Lightfield datasets. A three-layer cache system achieves interactive rates (15-20 fps) for 256^2 pixel images – upon too many cache misses or without the cache, performance is reduced to 3 fps. This system demonstrates up to 100:1 compression.

Gotz et al. [54] describe a novel image representation for cylindrical projection images. Their representation exploits spatial coherence and rearranges the columns of pixels. Columns that correspond to the same world-space viewing direction are grouped and stored either as index columns or as residual columns. Each column is coded using a one-dimensional wavelet and the coefficients are quantized and stored using a Huffman code. Overall they achieve compression performance similar to JPEG (15-20x) but, unlike JPEG, are able to incrementally code new columns and efficiently decode individual columns.

Wilson et al. [167] create a system for interactive walkthroughs of synthetic models that used spatially encoded video to represent image-based impostors [6]. They decompose the model into a regular grid of rectangular viewing cells.

For viewpoints inside a cell, they represent the geometry outside the cell by mapping pre-computed images of the far field onto the interior walls of the cell. Their scheme exploits the redundancy between the images of neighboring cells. Using MPEG nomenclature, cells are classified as either I-cells or B-cells. An image-based impostor of a B-cell references two images from the nearest two I-cells. Using depth information obtained from the synthetic model, they compute motion vectors and code the images using MPEG. To access a coded image at runtime in constant time, they store the bit stream offsets to each image in a globally accessible array. The average compression ratio for their dataset of 22,000 images of 512^2 pixels is 48:1.

In contrast to previous work, our goal is a method to compress a large number of high-resolution images of a real-world environment irregularly sampled over a plane in a manner that it yields high compression performance, is highly scalable, uses single layer images, and permits fast image access along arbitrary viewpoint paths. Our OCRIs reduce disocclusions and are automatically created using a fine-grain distortion method. Reference and residual images can be processed, stored, and decoded using graphics hardware already present on most computers. None of the systems described above support all these features.

Our image hierarchy and compression scheme uses the dense datasets captured with the Sea of Images system [8]. This work replaces the difficult computer vision problems of 3-D reconstruction and surface reflectance modeling with the easier problems of motorized cart navigation, dense sampling, and working-set management.

3.5.4.2. Image Hierarchy

Our algorithm builds a tree that exploits image coherence. This section describes how we build the image hierarchy using a binary tree, how reference

and residual images are created and encoded in the tree, and how the original images are extracted and decoded from the image hierarchy. Figure 3.38 provides a summary of the entire process.

Binary tree construction
• Create a node for each image. • Calculate Delaunay triangulation of the centers-of-projection of the images. • Build the tree using a priority queue and edge collapse operations.
Image encoding
• Use image warping to exploit inter-image redundancy. • Use an optimization process to further reduce image energy.
Image extraction and decoding
• Extract images in either logarithmic or constant time, at the expense of compression performance.

Figure 3.38. Summary of the process for building an image hierarchy.

3.5.4.2.1. Binary Tree Construction

We incrementally build the tree by collapsing the edges of a Delaunay triangulation of the centers-of-projection (COPs) of all the original images. First, a node is created for each image. Second, the edges of the Delaunay triangulation are placed into a priority queue. Third, the next highest priority edge is collapsed until no more edges remain.

There are several possible metrics for edge priority. To maximize compression, edges should be processed in an order that maximizes inter-image coherence.

One metric is the energy of the image difference between the two images of the edge. Image energy is estimated by calculating the total absolute pixel luminosity of the residual image or by measuring the size of the intra-coded compressed residual image. However, both these measures are computationally expensive.

A simpler, but effective, approximation is to use the Euclidean distance between the images (i.e., the edge length) as the metric of image similarity. This is based on the assumptions that nearby images have smaller image differences than those farther apart, and therefore should be processed first. This metric is very fast to evaluate, requiring only a distance calculation, and our experiments prove it gives almost as good a compression performance as the more costly image energy metric.

Using each edge and its associated node pair in priority order, we perform a half-edge collapse operation to convert the node pair into children of a new common parent node. This operation produces three types of tree nodes: I-node, P-node, and N-node. The new reference image parent node (I-node) is placed at the same spatial location as one of its children and contains the image formerly stored with that child. The child node at that location (N-node) simply becomes a pointer to the parent. The other child node (P-node) becomes a residual image, the difference between the original child and the I-node image warped the viewpoint of the child node. Finally, the nodes are locally re-triangulated and the queue is updated with the new edges, while the edges that no longer exist are removed from the queue. After all edges have been processed, the resulting forest of trees joins to form a single tree for the entire original image set (Figure 3.39).

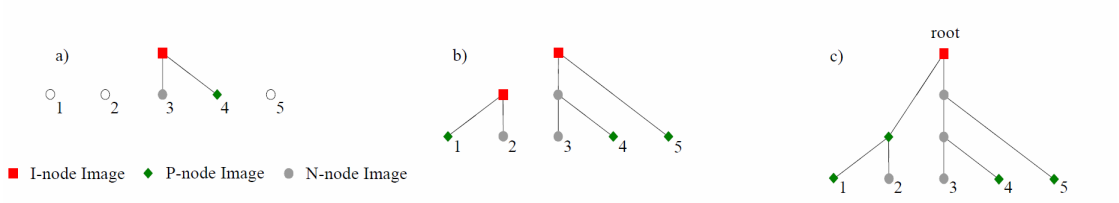


Figure 3.39. The tree building process begins from a set of five images. In this didactic example, the images are arranged in a line. From (*a*) to (*c*), image pairs (edges) are collapsed, producing I-nodes, P-nodes, and N-nodes.

3.5.4.2.2. Creating Reference and Residual Images

The tree-building process determines a set of reference and residual images that we must create. Given such single-layer images, our algorithm can use any image coding method, such as DCT-based compression (e.g., JPEG) or Wavelet-based compression for the images at each node in the binary tree. For JPEG, we use the quality parameter to set the amount of quantization, and thus determine the amount of loss. We select a lower quality value for residual images, since they contain less information and thus contribute less to overall quality.

There are several ways to reduce the energy in the residuals beyond using straightforward image differencing. We want to identify pixel correspondences and use this information to lower the energy in the residuals. Loosely, we categorize existing approaches into those that infer pixel correspondences using pixel search algorithms and those that require 3-D information of the scene. Motion estimation, typically used in video compression, uses search algorithms to find for a block of pixels in an image *A* that have a similar block of pixels in an image *B*. However, such search algorithms are very expensive computationally.

Proxy-Warped Reference Images

We reduce residual image energy using approximate 3-D scene information and warping the reference image to the COP of the residual image prior to image

differencing. This approach uses a geometric proxy to project a common set of 3-D point features onto both image planes, establishing a feature-based correspondence. Then, the features in one image are triangulated and a projective mapping warps the pixels of each triangle to their corresponding position in the other image. The warp operation may be performed by software or, as in our implementation, by exploiting texturing hardware that is commonly available on graphics cards.

The algorithm generates 3-D features by subdividing the polygons of the proxy and using the vertices as features. In order to obtain an approximately even distribution of features in image space, the algorithm adaptively subdivides proxy polygons according to the size of their screen-space projection. Polygons near the COP are subdivided more than polygons farther away. Pre-computing and storing features for every image pair would increase the amount of data to store, so we choose to generate features on the fly.

It is worth mentioning that a ray casting approach which creates set of 2-D features in a first image and finds the corresponding locations in a second image might obtain a better feature distribution, but it also introduces several disadvantages. In particular, features are not necessarily created along edges and corners of the proxy (i.e., places that often correspond to sharp visual change) and ray casting cannot easily be adapted to hardware.

Occlusion Camera Reference Images

To further reduce the size of the residuals, we need to have samples available for the surfaces that may become visible after the image warp. In general, the energy in the residual images comes from three main sources. One is due to the view dependent appearance of scene surfaces. For example, a highly specular surface will look different in images with different COPs, and although the mapping induced by the proxy correctly finds the same surface point in both images, it does not help reducing the color difference. Another source of residual

difference is the approximate nature of the proxy; the small geometric features ignored by the proxy do create small inconsistencies between the warped reference image and the image to be compressed. A third and large source is due to disocclusion errors. The image to be compressed samples some surfaces that are not visible from the reference viewpoint. Simply using the reference image to recreate a second image produces disocclusion errors. Compensating for disocclusion errors in the residual image does not lend itself to good compression. Thus, we reduce the size of the residual images by alleviating disocclusion errors. Instead of regular pinhole camera reference images we use occlusion camera reference images (OCRIs), which store additional samples to prevent disocclusion errors. The OCRI has a single layer. The visible samples are “squeezed” together in the neighborhood of depth discontinuities to make room for barely hidden samples. The OCRI does not store samples for all surfaces; samples that are clearly hidden and are unlikely to become visible in nearby views are discarded.

The depth discontinuity occlusion camera is used for this implementation. Its construction was described in Section 3.2. The only difference here from the previous description is that the scene is the proxy model rendered with projective texture mapping for color.

Warping Optimization

For each image differencing operation, we optionally adjust the registration of the proxy with the images so as to minimize the energy of the residual image. This optimization attempts to reduce errors introduced by camera pose estimation as well as attempts to compensate for the approximate nature of the proxy. The optimization process uses the COP A of a first image IA and the COP B of a second image IB to initialize the vector $V=B-A$, containing the translation and rotation offsets from image IA to IB. We minimize the energy of the residual image between image IA warped to position $A+V$ and image IB. A

multidimensional minimization method (e.g., Powell's method [131]) is used to find an optimal vector *V*.

3.5.4.2.3. Extracting and Decoding Original Images

In an IBR application we need to extract an arbitrary image sequence from the original set of *M* images. We find the corresponding leaf node, and trace the links up to the I-node ancestor. Then starting with the I-node, we reverse the path following the P-node or N-node branches, adding the P-node images to the I-node until we reach the leaf-node. This procedure is applied recursively until the tree is traversed down to the node of the desired image.

There are several variants to the tree building process that yield different original image extraction methods. We present three such methods here and explore their tradeoffs in the results section.

Root-based Extraction

A first extraction method always refers to the root node and is the most straightforward one. It requires at most an $O(log\, M)$ sequence of image additions. For this method, the root of the tree is the only I-node and all other nodes are either P-nodes or N-nodes. Since the root I-node must be accessed, the number of image additions required to extract an arbitrary image is equal to the height of the tree. Unfortunately, the accumulation of a long sequence of incremental image residuals may result in a large reconstruction error. This problem can be mediated by creating residual images top-down instead of bottom-up, ensuring that only captured images are used to calculate image residuals. In order to create residual images top-down, we must first create a tree skeleton which determines the types of the nodes and the tree connectivity. Such a tree skeleton is easily constructed if we use a metric for edge collapse priority that depends on the Euclidean distance between the images and not on image energy.

Indirect I-node-based Extraction

A second extraction method reduces decompression time, at the expense of storage, by decreasing the maximum number of image additions. This option forces I-nodes to be distributed throughout the tree, akin to forced intra-coding in MPEG. This distribution may be proportional to the residual image energy (adaptive) or set by a pre-defined constant. For example, if during an edge collapse the residual image energy is greater than a pre-defined threshold or the distance between a newly created I-node and the farthest P-node descendant is greater than a pre-defined constant, we change the newly created P-node to an I-node.

Direct I-node-based Extraction

A third image extraction method further reduces the number of image additions to exactly one in all cases (i.e. constant time) by defining the residual image of a P-node directly relative to the closest I-node up the tree. To build such a tree, we either have knowledge of which nodes will be I-nodes or make residuals after creating the tree skeleton.

3.5.4.3. Implementation

We have implemented our algorithms on a Pentium IV 3.0 GHz system equipped with a modern graphics card. We use JPEG compression both for the I- and P-nodes, adjusting the quality setting to achieve the desired quantization. We use an OpenGL programming interface and the graphics subsystem to achieve image differencing and image addition at interactive rates. Both operations are performed using multiple-pass texture mapping operations.

Proxy-based reference image warping, as described previously, is implemented using OpenGL. To determine the set of features for reconstructing an image using an I-node and P-node image pair, we recursively subdivide proxy polygons on the main CPU for the COP of either of the images. Using a low resolution

frame buffer, we render the proxy polygons into the z-buffer and then render the proxy vertices into the color buffer encoding unique vertex IDs in the color channels. Next, we read back the frame buffer and determine which vertices are visible. We compute the location of the same features in the other image. Next, we load the two images into texture memory. We render the triangular mesh for the I-node image using its feature positions in image space as texture coordinates, but render the mesh using the vertex coordinates of the P-node mesh to create the warped image. Finally, the reconstructed image is obtained by adding the signed P-node residual image to the warped I-node image.

Like in the case of a regular reference image, each pixel of an OCRI defines a 3-D point with color and the regular 2D array of samples implicitly defines a triangle mesh. The OCRI is efficiently warped by rendering the OCRI mesh (or proxy) from the novel view in hardware.

The complexity of the image warp operation is proportional to the size of the images. In order to reduce the complexity, we can optionally reduce the size of the warped images. This reduction in image resolution improves runtime performance but shifts more energy into the residual image, resulting in diminished compression performance.

Table 3.3. Summary of datasets: number of images, raw size, and average center-of-projection spacing.

Dataset	**Number of Images**	**Raw**	**Avg. COP Spacing**
Museum	9832	30.9 GB	2.2 inches
Office	3475	10.9 GB	0.7 inches
Library	1947	6.1 GB	1.6 inches

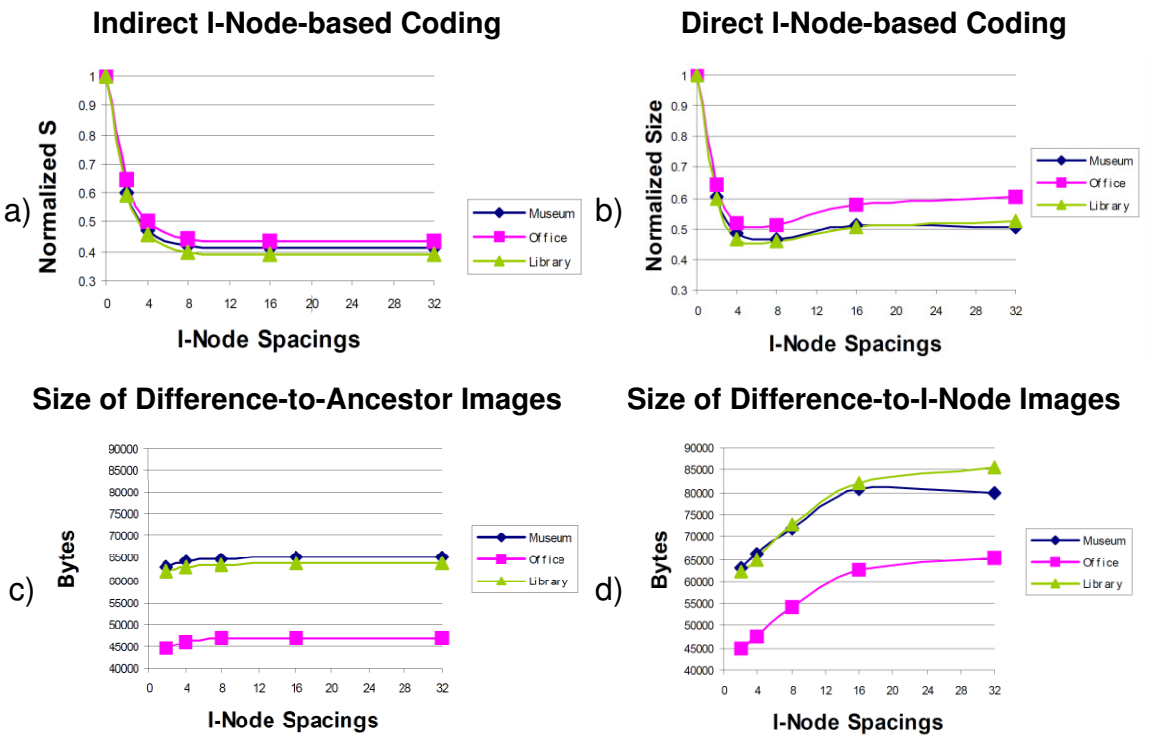


Figure 3.40. The compression results for several I-node spacings. An I-node spacing of 0 corresponds to only intra-coded images. An I-node spacing of 32 corresponds to a single I-node for the entire dataset. In order to show all dataset sizes in the same graph, normalized size values are used. These values, used in (*a, b*), are computed by dividing the compressed dataset sizes by the maximum compressed size of the dataset in each graph. (*a, c*): Results using residual images computed relative to immediate ancestors (indirect I-node-based coding). (*b, d*): Results using residual images computed from the closest I-node up the tree (direct I-node-based coding).

3.5.4.4. Results

We have applied our algorithms to three datasets (Table 3.3). For each dataset, we captured images at $1{,}024^2$ pixels from an eye-height plane using an omnidirectional camera [106] mounted on a motorized cart driven through the environment via remote control. The first dataset, Museum, consists of 9,832 images covering 900 square feet. The second dataset, Office, consists of 3,475

images covering 30 square feet. Finally, the third dataset, Library, contains 1,947 images covering 120 square feet. To create the proxy for each environment, we obtain a coarse floor plan of each environment and extrude the walls to construct a 3-D model of about a dozen polygons each. Floor and ceiling surfaces are approximated with two large planes. On average, we are able to extract and reconstruct from proxy-warped images $1{,}024^2$ pixel-resolution images at a rate of 15 to 20 images per second.

3.5.4.4.1. Hierarchy Alternatives

The hierarchy, as described, supports either residual images that are relative to their immediate ancestor in the tree (e.g. indirect I-node-based coding) or residual images that are directly relative to the closest I-node up the tree (e.g. direct I-node-based coding). The former yields better compression but more expensive image extraction. The latter provides constant image extraction cost but less compression performance. The tradeoff also depends on the spacing between I-nodes in the tree.

To better understand the aforementioned tradeoff, Figure 3.40 (a-d) show the compression performance for various maximum I-node spacings (i.e., maximum levels of the tree between I-nodes) using proxy-warped reference images and both residual-image strategies. Since none of the compression trees has more than 32 levels, the examples at an I-node spacing of 32 are those having a single I-node for the entire dataset. The examples at I-node spacing of 0 represent trees where every node is an I-node (this configuration is effectively the same as compressing each image as an independent JPEG image). The average preprocessing time for datasets, such as those in Figure 3.40, is approximately 1 second per image. This includes building the tree, computing residual images, and intra-coding images.

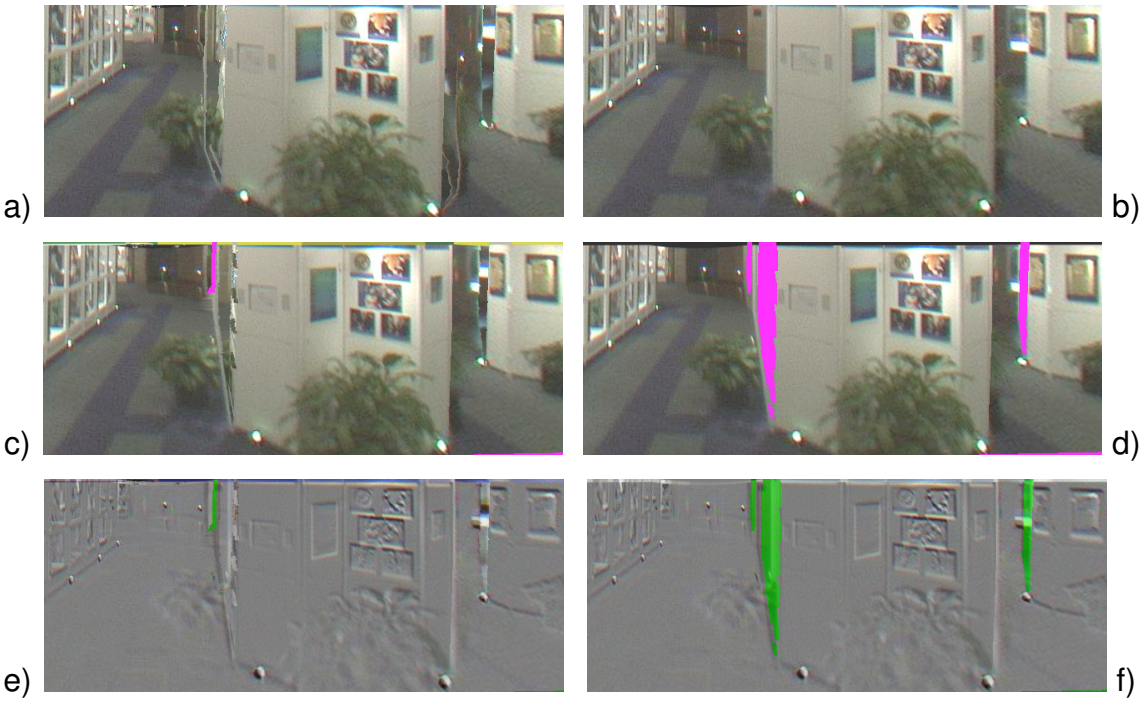


Figure 3.41. An example occlusion camera reference image usage. *Top*: Planar re-projection of an OCRI for the Museum environment (*left*) and the desired image to reconstruct (*right*). *Middle*: OCRI (*left*) and standard reference image warped to the desired image viewpoint with the missing samples colored purple. *Bottom*: Residual for the warped OCRI (*left*) and standard reference image (*left*) with missing samples are highlighted in green. Notice the large quantity of missing samples that the standard reference image residual has to store.

Figure 3.40, top, shows the compressed dataset sizes against I-node spacings. Other compression parameters are kept constant so the variance observed is solely due to changing I-node spacings. Fewer I-nodes generally yields better compression at the expense of either quality or image extraction performance.

However, residual images directly relative to their I-node tend to vary in size proportional to their distance from the I-node (Figure 3.40d). They also tend to maintain similar image quality although at potentially greater cost in terms of space. As evidenced in Figure 3.40b, the best tradeoff between residual image size and number of I-nodes occurs between I-node spacings of 4 and 8.

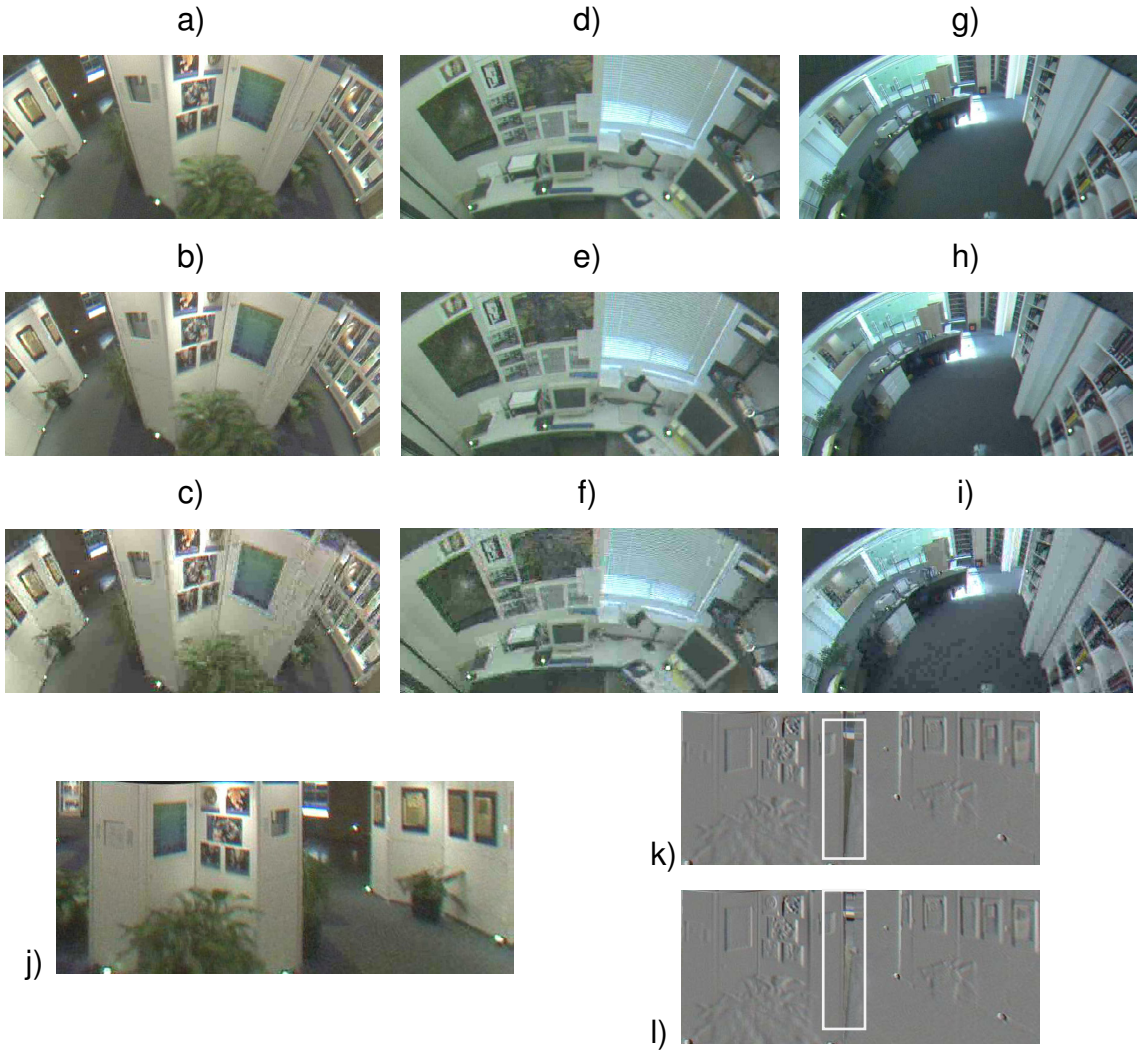


Figure 3.42. Examples of images compressed by several ratios. Images (*a-c*) are of the Museum environment. Images (*d-f*) are of the office environment. Images (*g-i*) are of the library environment. The original omnidirectional images are shown in (*a, d, g*). For our datasets, we able to compress to 83:1 (*b*), 100:1 (*e*), and 69:1 (*h*) without significant loss in quality. Further compression slowly exhibits artifacts such as those visible at 121:1 (*c*), 149:1 (*f*) or 120:1 (*i*). Image (*j*) was reconstructed using OCRIs and the resulting more compact difference images. Image (*k*) highlights the area of the difference image for proxy-warping which needs to store the samples of disoccluded surfaces in image (*k*). Image (*l*) shows the OCRI difference image lacking those samples.

In summary, I-node spacings between 4 and 8 yield a reasonable tradeoff between reconstruction time, reconstruction quality, and compression performance. Using residual images directly relative to their I-node produces a more consistent image quality although at the expense of compression. Thus, in rest of the results section we use an I-node spacing of 4 and direct I-node-based coding.

3.5.4.4.2. Compression Analysis

Table 3.4 breaks down how each part of the algorithm contributes to the total compression. As can be observed in the table, approximately one third of the total compression comes from each of intra-coding (35%), proxy-based image warping (35%), and image-warping optimization (23%). Another source of compression (approximately 7%) is adaptive quality settings for the residual images based on image energy. When using intra-coding images without adaptive settings, all images are coded at fixed quality settings. In particular, we fix I-node images to be at quality setting 75 (default for JPEG) and residual images to be at quality setting 65.

To adaptively choose JPEG quality settings, we calculate the image energy range for the residuals and use interpolation to obtain the quality setting (this is similar to rate-control in video coding). The energy range is computed as the mean energy plus/minus its standard deviation. Through experimentation, we found that compressing low-energy residuals 30% more aggressively yields good results.

The error function for the image warping optimization requires computing residual images and their average energy value. Since the error function is called many times per optimization and one optimization is performed per residual, this step is computationally expensive. To reduce the preprocessing time, we use 256^2 resolution residuals during the optimization. The translation and rotation offsets

are similar to those obtained using full resolution images but at reduced cost. On average, optimization using proxy-based warping adds 3 seconds to the per-image preprocessing time. Using the results of the optimization does not, however, affect the runtime decoding performance.

Table 3.4. Compression contributions as various elements of the algorithm are enabled. Results in this table use residual images relative to I-Nodes and I-Node spacings of 4. The table should be read top-down. For each environment, we show compressed dataset sizes in megabytes and cumulative contributions as percentages and ratios. The percentage/ratio indicates how much of the total compression has been achieved thus far.

	Museum		**Office**		**Library**		**Average**	
Operation	Contrib.	MB	Contrib.	MB	Contrib.	MB	**Contrib.**	**Ratio**
Raw Data	0%	30929	0%	10931	0%	6128	**0%**	**1:1**
Intra-coding	35%	1559.7	35%	376.4	34%	317.9	**35%**	**23:1**
Proxy image-warping	72%	754.8	67%	195.3	72%	148.8	**70%**	**46:1**
Adaptive settings	77%	709.6	77%	169.2	78%	137.8	**77%**	**51:1**
Optimization	100%	538	100%	129.7	100%	106.6	**100%**	**66:1**

3.5.4.4.3. OCRI Benefit

Using occlusion camera reference images yields us improved compression performance over warping standard reference images whenever disocclusion errors are present. For each I-node of the tree, we create an OCRI by computing the distortion map using the proxy and obtain color samples from a neighborhood of images surrounding the OCRI. It takes approximate an additional 3 to 5

seconds to create each OCRI. Figure 3.41a shows a planar re-projection of a subset of an OCRI for the Museum environment. (Our current OCRI implementation handles only planar projections, so we tile the omnidirectional image into planar re-projections and then optionally re-assemble them). Figure 3.41b contains the desired destination image. Figure 3.41c shows how the OCRI is warped to the viewpoint of the desired image and is able to fill-in disocclusions. Figure 3.41d contains an image from the same viewpoint but warped using regular reference images. Figure 3.41e shows the residual image to be added to the OCRI in order to complete the image in Figure 3.41b, while Figure 3.41f is the residual image when warping a standard reference image. The reduced number of samples and energy in the OCRI residuals allows us to obtain improved compression performance in this example.

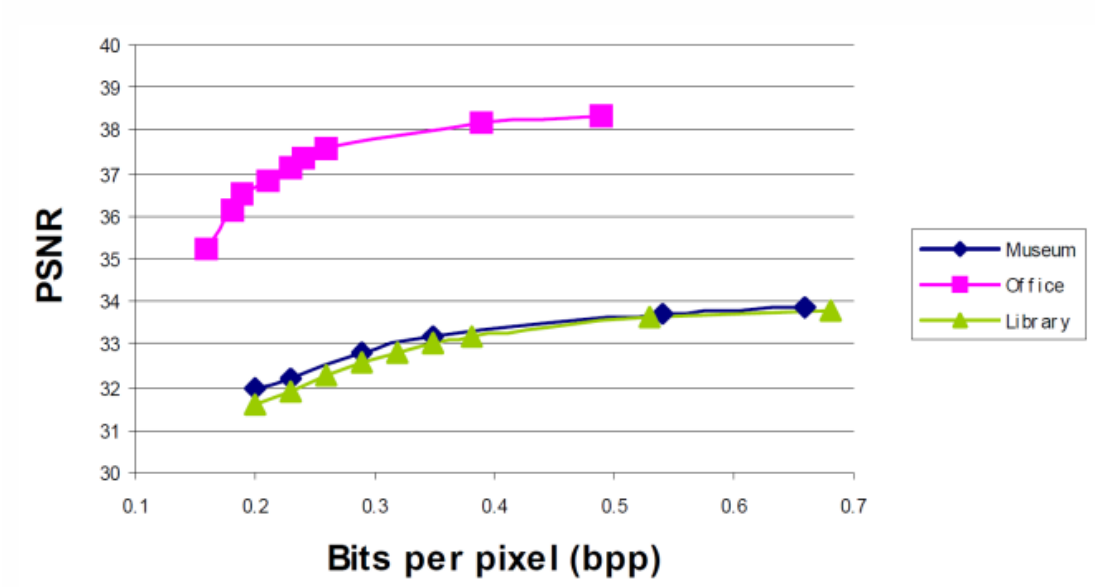


Figure 3.43. A graph showing the tradeoff between compression and quality. The tradeoff between compression (in terms of bits-per-pixel) and quality (in terms of peak-signal-to-noise-ratio) using proxy-based warping is shown. Original images have 24 bpp; thus, 0.68 bpp is equivalent to 35:1 compression and 0.16 bpp is equivalent to 149:1 compression.

3.5.4.4.4. Coding Performance

Using the best parameter settings, as previously described, we vary the effective compression rate (bits-per-pixel, or bpp) and report in Figure 3.43 the peak-signal-to-noise-ratio for each of our datasets when using proxy-based warping. The higher peak-signal-to-noise-ratio values for the office environment are due to the higher image density in that environment (see Table 3.3). This increase in image density creates lower energy residuals.

For our datasets, we show example reconstructed images for several compression ratios, ranging from 35:1 to 149:1, in Figure 3.42. On average, our algorithm can compress images without significant artifacts by a factor of 84-to-1 (average compression ratio of Figure 3.42b, Figure 3.42e, and Figure 3.42h).

For the museum environment, our largest dataset, we used OCRIs to yield improved compression. The compression performance, by definition, will be the same or better than proxy-based warping depending on the presence of disocclusions in the image sequence. For our simple proxy model, disocclusions mostly occur near the kiosk located in the middle of the museum. For a sequence of planar re-projections containing views of this subset of the model (similar to Figure 3.41), OCRIs are able to reduce disocclusions and image energy. Using a sequence of 2,000 images facing the kiosk, compressed at low to medium compression ratios (35:1 to 83:1), we observed individual OCRI difference images that were up to 3.3% smaller and up to 4.4% more compact at higher compression ratios (121:1). Figure 3.42 (j-l) show an image reconstructed using OCRIs and a comparison of difference images. The overall average improvement we saw in images containing disocclusions was only about one percent. In our particular environment, the missing samples were often of a single color (e.g., white colored surfaces) and thus compress well in the difference image – this offsets the maximum gains to be obtained from OCRIs in this case.

a) b) c)

Figure 3.44. An example of our method compared to MPEG-2. (*a*): A portion of an original captured omnidirectional image within the Museum environment. (*b*): The same frame but reconstructed from MPEG-2 coded images such that the total compression equals 85:1. (*c*): The same frame reconstructed using the compression algorithm of this paper with standard reference images. Reconstruction uses difference-from-I-nodes, I-node spacing of 5, proxy-based image warping, and image warping optimization, yielding 85:1 total compression. Notice the improved quality of our reconstruction as compared to the artifacts visible in the MPEG-2 frame.

In Figure 3.44, we compare our compression algorithm to a standard MPEG-2 encoding of a linearization of the entire image database. First, we select an approximate target compression ratio for our algorithm (e.g., 85:1) and then select a bpp setting for an MPEG-2 encoder that yields approximately the same overall compression. Even though we are only using proxy-based warping in this example, on average, our algorithm results in better quality images because we are able to capitalize on the 2-D nature of the inter-image redundancy, as opposed to the linear (1-D) inter-image redundancy in MPEG-2. Furthermore, the proxy model, image warping, and optimization process contribute to our superior quality.

3.5.5. Three-Dimensional Display Rendering Acceleration

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

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

We describe a method to accelerate rendering on volumetric 3-D displays, based on adapting the scene level-of-detail before the 3-D image is computed, and reducing the number of 3-D image samples that are computed and transferred. For example, if the 3-D scene represents Manhattan, a view that maps the entire island to the volume of the 3-D display can be safely computed from a coarser

representation than a view that only shows Times Square. Moreover, for a single user that is seated or stands in one place, many of the background buildings are completely occluded and do not become visible for normal gaze changes and head motions. The hidden buildings can be ignored when computing the 3-D image.

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

A research path in computer graphics known as image-based rendering (IBR), models the scene by rendering from pre-computed or pre-acquired reference images. In one variant, the scene is modeled with planar pinhole camera *depth images* (DIs), which are images enhanced with per-pixel depth [102]. The depth information allows warping the reference samples to any novel desired view. A DI provides a good level-of-detail solution, which holds for nearby views.

Unfortunately, the occlusion culling solution of the reference image cannot be applied to nearby views. Even small translations of the viewpoint produce disocclusion errors, which are artifacts due to lack of samples for surfaces that become visible but were not sampled by the reference DI. In our context, representing the scene with a DI computed from the left eye's viewpoint produces disocclusion errors in the image seen by the right eye.

To address this problem we replace the planar pinhole camera with an occlusion camera that samples not only surfaces visible in the reference view, but also surfaces that are likely to become visible in nearby views. The resulting occlusion camera reference image (OCRI) stores samples that are hidden in the reference view but are needed to alleviate disocclusion errors when the view translates. We represent the scene with an OCRI computed for the user's reference view, which is the average of the left and right eye views in the normal head position. Like a regular DI, the OCRI is a single layer representation with the advantages of bounded number of samples, implicit connectivity, and efficient incremental processing. Another advantage shared with regular DIs is that OCRIs adapt the scene's level-of-detail to the reference view. Unlike a regular DI however, the OCRI has all samples needed for a continuum of views centered at the reference view. Interpupillary distance and normal head motion do not produce disocclusion errors.

Figure 3.45 through Figure 3.51 illustrate our approach. Figure 3.45 through Figure 3.48 show images computed with our volumetric 3-D display simulator, and Figure 3.49 through Figure 3.51 show actual photographs of our volumetric 3-D display. Both simulated and real 3-D displays produce spherical images with a diameter of 10". Figure 3.45 and Figure 3.47 show a depth image (DI) and an OCRI constructed from the same viewpoint. Figure 3.46 shows the DI and OCRI from a viewpoint 4" left of the reference viewpoint. The severe disocclusion errors that occur for the DI are alleviated by the OCRI. Figure 3.48 shows the DI and OCRI from a side view. The OCRI does not sample all surfaces in the scene, nor should it. The OCRI provides occlusion culling by safely discarding the samples that are not needed in nearby views. The OCRI shrinks the "shadow" of the bunny. Figure 3.49 shows reference view photographs of the 3-D images rendered from the DI, OCRI, and geometric model. Figure 3.50 and Figure 3.51 correspond to Figure 3.46 and Figure 3.48.

SIMULATOR IMAGES

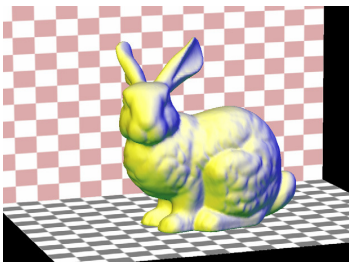


Figure 3.45. Depth image.

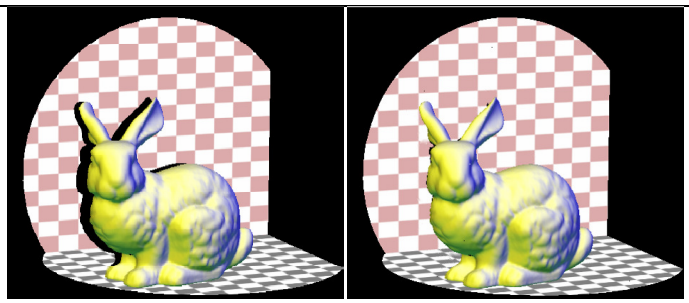


Figure 3.46. Images rendered from depth and occlusion camera reference images with the viewpoint four inches left of reference viewpoint.

Figure 3.47. Occlusion camera reference image.

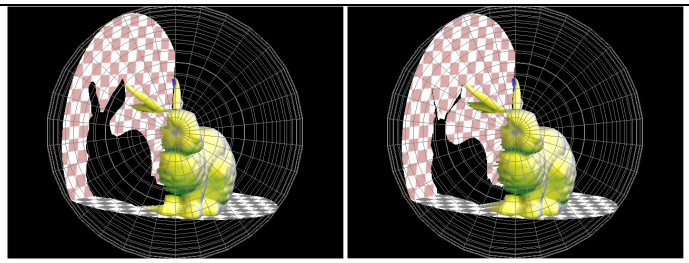


Figure 3.48. Images rendered from depth and occlusion camera reference images. Wireframe shows spherical display volume.

3.5.5.1. Prior Work – Three-Dimensional Displays

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

PHOTOGRAPHS OF 3-D DISPLAY

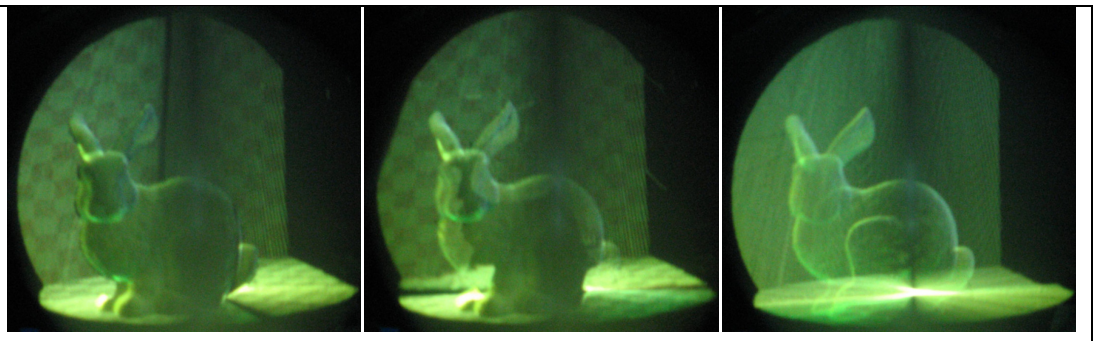


Figure 3.49. Photographs of images rendered from the reference viewpoint in the 3-D display. 3-D images rendered from DI (*left*), OCRI (*middle*), and original geometric model (*right*).

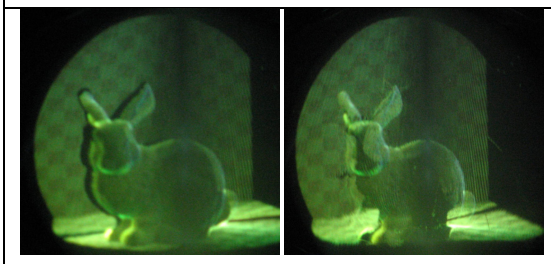


Figure 3.50. Photographs of display four inches left of the reference view.

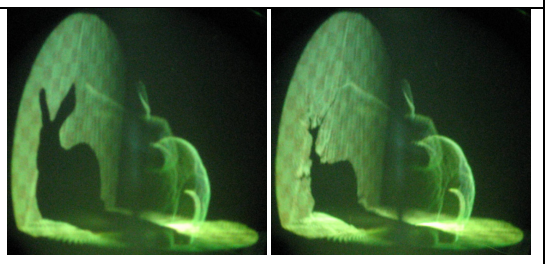


Figure 3.51. Photographs of the 3-D display from a side view.

Autostereoscopic displays [59] produce a 3-D image without the need of special eyewear. Parallax autostereoscopic displays provide different images for the left and right eyes using slits [66, 115] or lenslets [36, 63, 99]. The disadvantages are reduced resolution and range of supported viewpoints.

Volumetric displays produce a truly three dimensional image. One approach is to fill space, for example with a stack of transparent LCDs [87]. The approach has the disadvantage of limited z resolution. Another approach is to use a varifocal mirror whose oscillations are synchronized with a 2-D display it reflects [156]; the difficulty with such a display is building the varifocal mirror.

Another type of volumetric display technology is based on sweeping the display volume. 2-D slices of the scene are displayed in rapid succession and the eye

integrates them into a 3-D image [43, 116]. The greatest challenge is the mechanical scanning, which is noisy, imprecise, and fragile.

Several emerging technologies show potential for producing 3-D images. Electroholography [91] produces an interference pattern (holographic fringe) which is then illuminated to produce a 3-D image by diffraction (modulation of holographic fringe). The approach is hampered by the enormous amount of data

To the best of our knowledge, the only volumetric displays available commercially are those produced by Actuality Systems [116] and LightSpace Technologies [87]. All volumetric displays convert a 3-D scene description into a 3-D image. Our method produces a simplified description of the scene which is then used to compute the 3-D image. Therefore, in principle, the method can be applied to other volumetric display technologies. We demonstrate the effectiveness of our method on the Perspecta volumetric display [116], which we characterize in detail later.

3.5.5.2. Algorithm Overview

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

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

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

3.5.5.3. Occlusion Camera

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

1. *Disocclusion:* Some rays of the camera sample surfaces that are not visible in the reference view, but are likely to become visible in nearby views.
2. *Single layer:* The camera acquires a 2-D image; at each pixel, the image stores the depth and color of the closest surface sample along the ray at that pixel.
3. *Unambiguous projection*: A 3-D point projects to at most a single image location (no two rays intersect).
4. *Efficient projection*: The projection of a 3-D point is computed in a constant number of steps.

The first property ensures that the OCRI is less prone to disocclusion errors than a regular depth image. Because of the second property, the OCRI has a bounded number of samples.

The last two properties ensure that the OCRI can be constructed efficiently with the feed-forward graphics pipeline.

3.5.5.4. Occlusion Camera Reference Image Construction

We demonstrated the application of the occlusion camera to 3-D display acceleration using the depth discontinuity occlusion camera (DDOC) and epipolar occlusion camera (EOC). Rendering of the DDOC proceeds as described in section 3.2, and rendering the EOC proceeds as described in section 3.3.

3.5.5.5. Rendering Using the OCRI

The OCRI provides a good approximation of the scene, tailored to the reference view. The OCRI is converted to a 3-D triangle mesh, which is then used by the volumetric display driver to render the 3-D image, in lieu of the original scene model. Each sample in the OCRI corresponds to a 3-D point with color. To recover the 3-D point from the OCRI sample, one needs to be able to unproject the sample back in 3-D.

3.5.5.6. Rotating Screen Volumetric 3-D Display

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

Available to us is a volumetric display (Figure 3.52) that builds a 3-D image one slice at the time, with a rotating screen [116]. The screen has a radius of 5", it is diffuse and semitransparent, and it rotates with an angular velocity of 720 rpm. Both faces of the screen carry an image resulting in a refresh rate is 24 Hz, which corresponds to a 180° rotation. The display projects onto the screen the intersection between the scene and the plane of the screen 198 times for every complete rotation. The optical path is folded using 3 mirrors M_0-M_2. The mirrors and screen are enclosed in an inner glass sphere that rotates with the screen; the glass sphere is enclosed in a stationary outer glass sphere. The display is not perfectly balanced which causes it to wobble. We estimate the amplitude of the wobbling to be 0.5 cm. Each slice has a resolution of 768x768. The color resolution is 32-bit RGBA but it is compressed to 3-bit RGB. The reduced image brightness requires dimming the ambient lights when the display is in use (Figure 3.52).

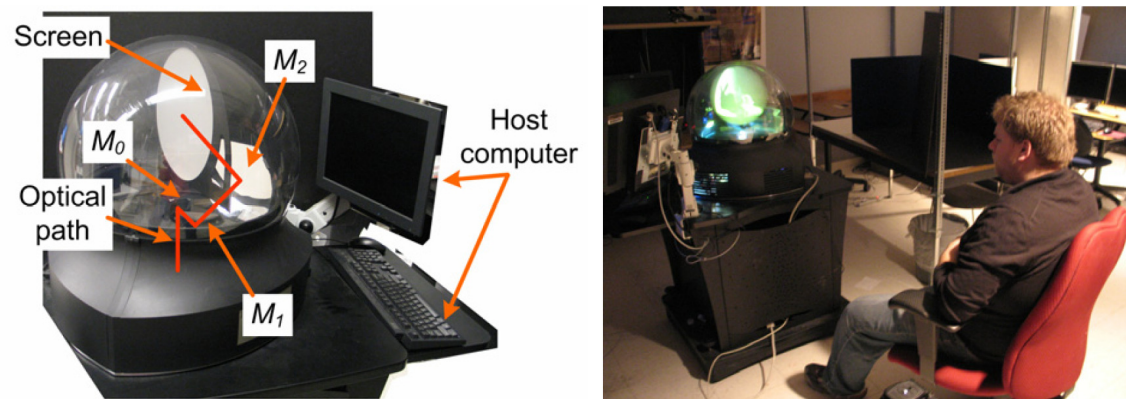


Figure 3.52. Photograph of the volumetric 3-D display. The 3-D display used to validate the OCRI approach (*left*) is typically viewed at a distance of 50" (*right*).

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

The photographs shown throughout were taken with a digital camera with the following settings: no ambient lights, aperture F2.8, exposure time 1/25s, and simulated film sensitivity ISO400. Our camera does not offer 1/24s as one of the possible exposure times, which would have allowed acquiring a complete 3-D image. We used the slightly shorter exposure time since the wobbling produces excessive blurriness if the shutter remains open more than 180° and the screen revisits a part of the 3-D image. The slightly shorter exposure time misses (1/24-1/25)*(12*360°) = 7.2° of the 3-D image. We took several snapshots for every position to place the missing 3-D image sector in a convenient location (see black stripe that splits the vertical plane in Figure 3.49—*left* or the horizontal plane in Figure 3.50).

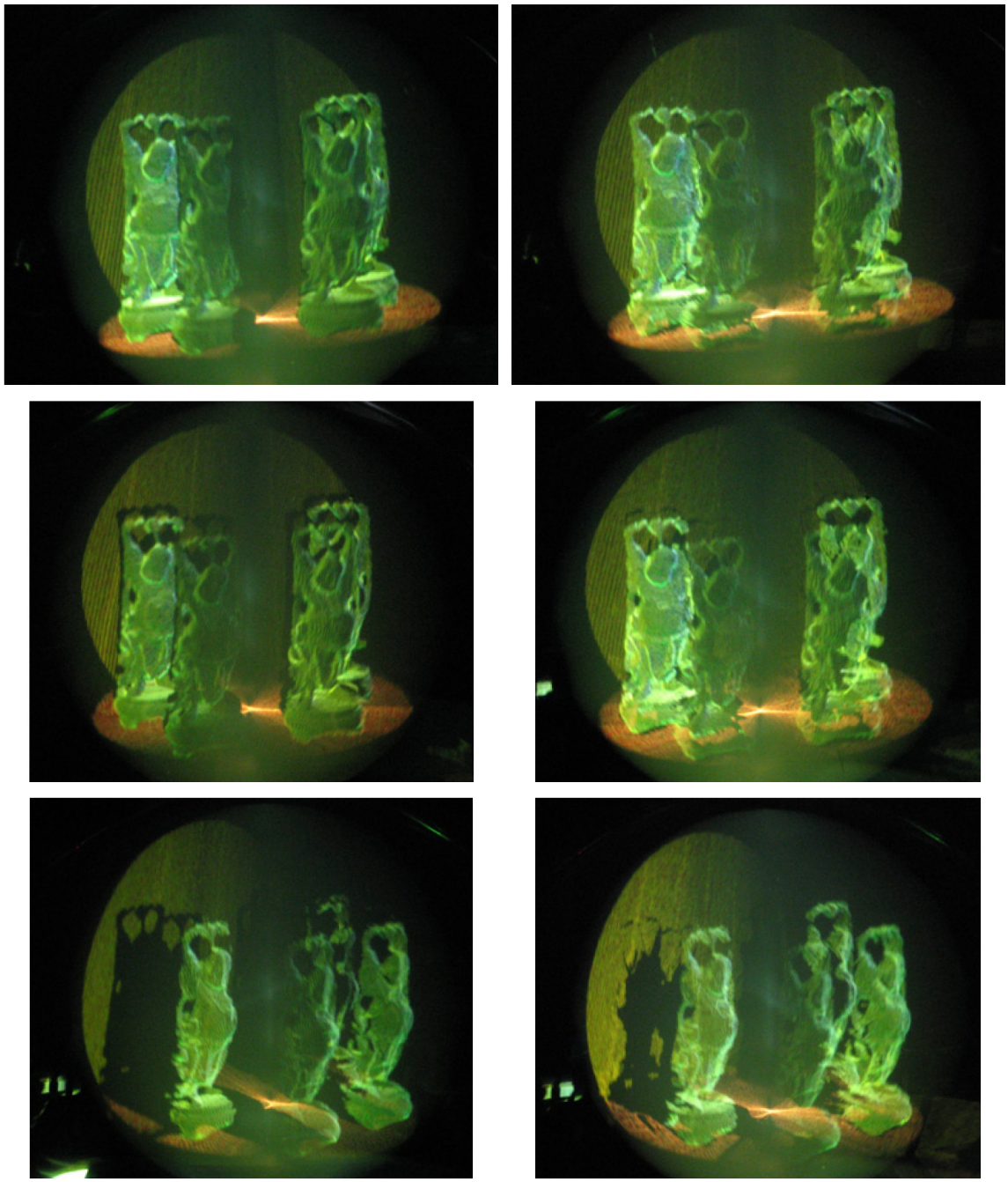


Figure 3.53. Photographs of the 3-D display of the Happy Buddha statues. The 3-D image was rendered from a DI (*left*) and from an OCRI (*right*). The photographs were taken from the reference viewpoint (*row 1*), and from 4" above the reference viewpoint (*row 2*). Side view shows the "shadows" shrunk by the OCRI (*row 3*).

3.5.5.7. Results

We have tested our approach with the DDOC on several 3-D scenes, both with our volumetric display simulator and the actual volumetric display: the bunny (Figure 3.45 through Figure 3.51), the four Happy Buddha statues (Figure 3.53), and the Thai statue (Figure 3.54) scenes. We have also tested our approach using the EOC on the teapot scene (Figure 3.55).

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

The OCRI provides efficient projection and is constructed with the help of graphics hardware. Table 3.5 reports the 3-D image rendering times and the number of triangles for each of three scenes (bunny, Happy Buddha statues, and Thai statue) and for each of three scene representations (depth image, OCRI, and geometry). The OCRI approach has three main steps: the occlusion camera model is computed first, then the OCRI is constructed by rendering the scene with the occlusion camera, and then finally the 3-D image is produced from the triangle mesh defined by the OCRI. The table reports the aggregate time for steps 1 and 2 as C_{time}, and the time for step 3 as *Time*. The resolution of the desired image and that of the reference image is 720x480. The depth image and the OCRI always generate the same number of triangles since the OCRI has a single layer where it stores the hidden samples at the cost of reducing the sampling rate for the visible surfaces.

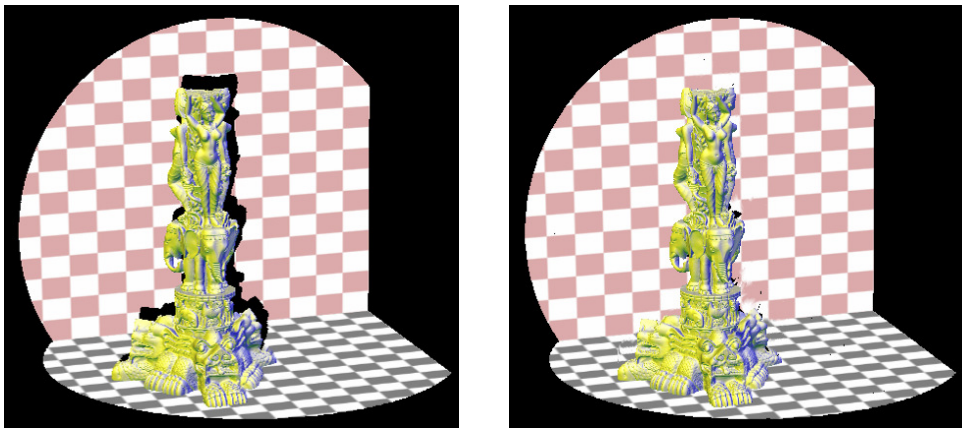


Figure 3.54. Simulator images of the Thai statue scene. Frames rendered from a DI (*left*) and an OCRI (*right*).

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

For the Happy Buddha statues scene, the speedup is 11.5 for the DI and 5.5 for the OCRI. For the 10 million triangles Thai statue, rendering the 3-D image from the DI or the OCRI brings a speedup of 23 and 8.5, respectively. The advantage of the DI and OCRI increases with the complexity of the scene, since the DI and the OCRI generate the same number of triangles (e.g. 612,000) regardless of the complexity of the original scene model. The DI approach is more efficient since it does not incur the cost of OCRI construction, but it suffers from greater disocclusion errors.

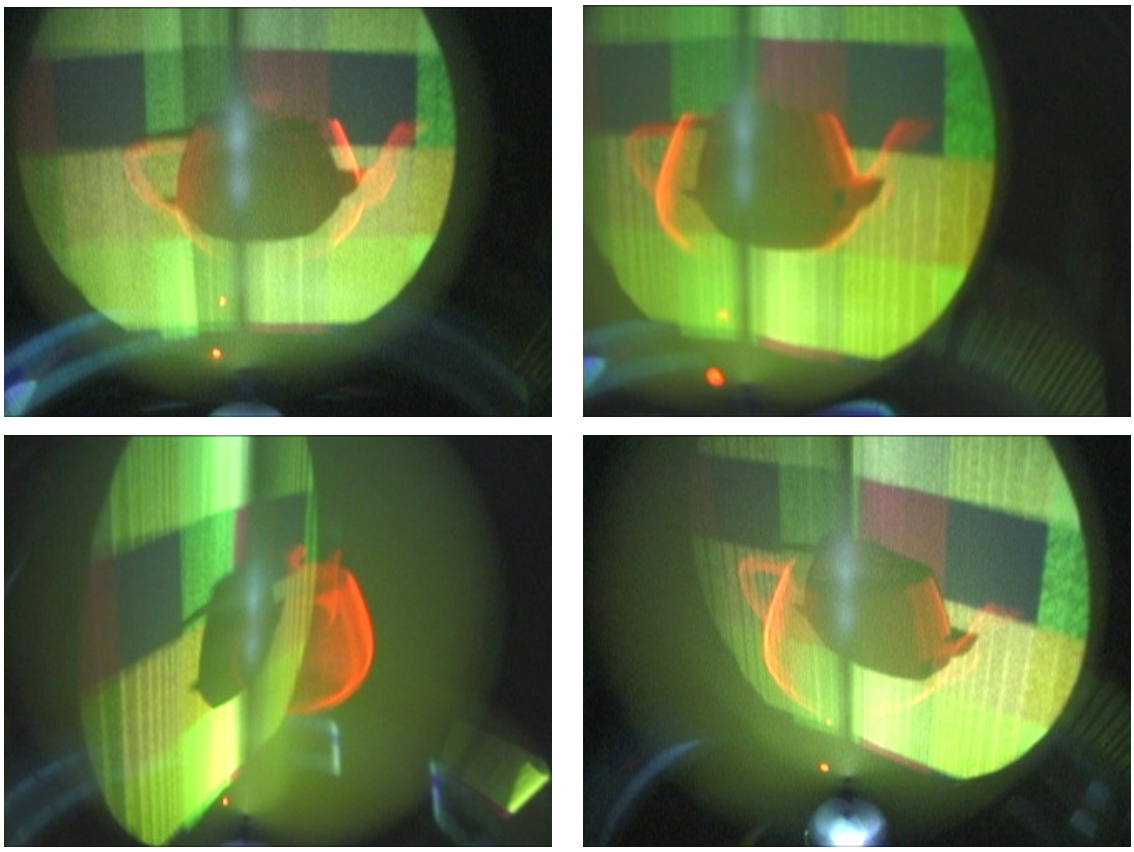


Figure 3.55. Photographs of the 3-D display showing the samples captured by an epipolar occlusion camera image.

OCRIs are one of many possible 3-D representations. Table 3.6 gives an approximate comparison between OCRIs and depth images (DIs), layered depth images (LDIs), light fields (LF), unstructured light fields (ULF), and ray-phase space (RPS) representation. The comparison is based on the four Happy Buddha statues scene, which our NVIDIA Quadro FX 3400 graphics card renders at ~8 Hz, hence the 0.12 s DI construction time. Constructing and LDI for such a scene requires first rendering and then merging approximately 4x4 = 16 construction depth images [25, 130, 142], at a time cost of 16x0.12x2 = 3.84 s. The LF construction time is 16x16x0.12 = 30.72 s, obtained with a rather modest back plane resolution (16x16). Constructing the ULF [19, 47] requires fewer images (32 for the table entry) since user interaction or a heuristic is used to identify the most important views.

Table 3.5. Rendering performance measured for various scenes.

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

The ray-phase space representation [151] is a 4-D plenoptic representation which instead of using two planes in front of the desired viewpoints for parameterization, uses a 2-D parameterized surface that surrounds the scene of interest, and then a 2-D parameterization of the outgoing rays for each surface point. The approach is similar to surface light fields [169] and models developed for general imaging systems [55]. In our case, a natural parameterization surface is the sphere described by the revolving screen, whose visible area is approximately 38% of the area of a sphere with a radius of 5 inches, or 120 square inches. For an average sampling rate of one point per square millimeter and 16x16 rays for each point, the total number of rays is 19 million. Generating these rays requires rendering the scene at least 57 times, for a construction time of 6.84 s, which ignores the cost of rearranging the rays according to the ray-phase parameterization.

For the 720x480 resolution, the 8-bit R, G, B, A channels and the 32-bit floating point z channel amount to 2.6 MB. The 16 floats needed to store the view are negligible. The LDI adds only a few non-redundant samples. The uncompressed LF requires considerable storage space. Compression could

reduce the memory consumption 10 or 100 fold, with the corresponding compression and decompression time costs and loss of quality [84]. The ULF has a more manageable uncompressed size, but is less redundant and thus compresses less well. The 19 million color samples of the RPS representation translate to 76 MB.

Table 3.6. A comparison of the construction performance.

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

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

The plenoptic representations are not supported by our 3-D display. On a regular LCD, the scene can be rendered at refresh rate (60 Hz for our system) when using the DI, LDI, or OCRI. The LF and ULF representations have been shown to support frame rates as high as 20Hz. Quality wise, the OCRI produces images comparable to those rendered using the original geometry. DIs suffer from disocclusion errors. LDIs produce lower quality images since they lack connectivity and are rendered by splatting [25, 130, 142]. Estimating the size and shape of the splats cannot be done both efficiently and accurately. The splats are typically overestimated and modeled as rectangles or disks, which produces blockiness. Typical artifacts when rendering from plenoptic

representations are coarseness (due to low spatial sampling resolution, as it is the case for the numbers chosen for this table), and compressions artifacts.

In conclusion, OCRIs, like DIs and LDIs, capture the scene well and are compact since they use the depth and the diffuse surface assumption to reuse color samples over a continuum of nearby views. OCRIs do away with disocclusion errors, the major disadvantage of depth images. The plenoptic representations have the advantage of not requiring geometry and can be acquired with a tracked camera. The plenoptic representations do provide limited support for view dependent effects, such as glossiness. Highly reflective surfaces are not supported since these entail the need of a very high spatial sampling resolution.

CHAPTER 4. THE GRAPH CAMERA FAMILY

The planar pinhole camera (PPC) can only sample data to which there is direct line-of-sight from the pinhole, due to the single viewpoint limitation. In the context of complex 3-D datasets, occlusions hide regions of interest and reduce the visualization payload of PPC images. The problem of occlusions has been addressed in visualization using a variety of approaches. One approach is to render the occluding layers transparently, or to cut a hole into the occluding layers to reveal the hidden data subset. Such transparency and cutaway techniques work well when the number of occluding layers is small and when a summary representation of these layers is acceptable. A second approach is to distort the 3-D dataset such that the alignment between the viewpoint, the occluder, and the data subset of interest is broken. The approach has the advantage of a clear and complete visualization, but specifying a dataset distortion that achieves the desired disocclusion effect while minimizing the visualization distortion is challenging.

Another approach is to simply rely on the user to navigate the camera around occluders interactively in order to establish a direct line-of-sight to data subsets of potential interest. Such sequential visualization can be inefficient. When the disoccluded data subset turns out to be of no interest, the camera path has to be retraced which is wasteful and can confuse the user. The single viewpoint limitation has also been addressed by using several PPC images simultaneously. However, the approach suffers from visualization discontinuity between individual images. The user cannot monitor all images in parallel and has to spend considerable cognitive effort to adapt sequentially to each one of the many visualization contexts.

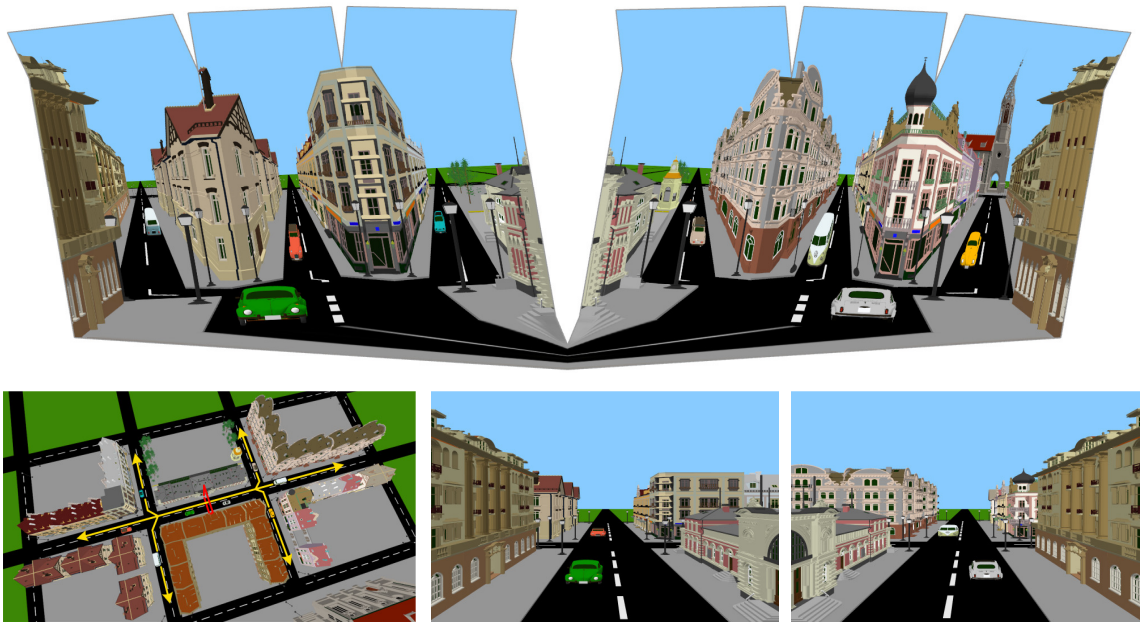


Figure 4.1. Enhanced virtual 3-D scene exploration. The graph camera image (*top*) samples longitudinally the current street segment as well as the 3 segments beyond the first intersections (*bottom, left*). The 4 side streets are occluded in conventional images (*bottom, right*).

Multiperspective visualization is a promising approach based on integrating data sampled from multiple viewpoints into a single image. The multiple viewpoints are integrated tightly which alleviates the visualization discontinuity problem of multiple individual images. Like in the case of the dataset distortion approach, multiperspective visualization amounts to a warp of global spatial relationships between data subsets. However, multiperspective visualization allows specifying the desired disocclusion effect directly in the image, as opposed to indirectly, through a dataset distortion. Finally, multiperspective visualization does not preclude but rather enhances interactive exploration of datasets. The multiperspective image provides a preview of data subsets to come which improves interactive visualization efficiency.

To further address the single viewpoint limitation, we introduce the graph camera family, a family of non-pinhole cameras that samples simultaneously multiple regions of interest in a 3-D scene. The graph camera integrates several PPC images into a single image with good continuity and little redundancy. The graph camera is a graph of PPC frusta constructed from a regular PPC through a sequence of frustum bending, splitting, and merging operations. Despite the camera model complexity, a fast 3-D point projection operation allows rendering at interactive rates.

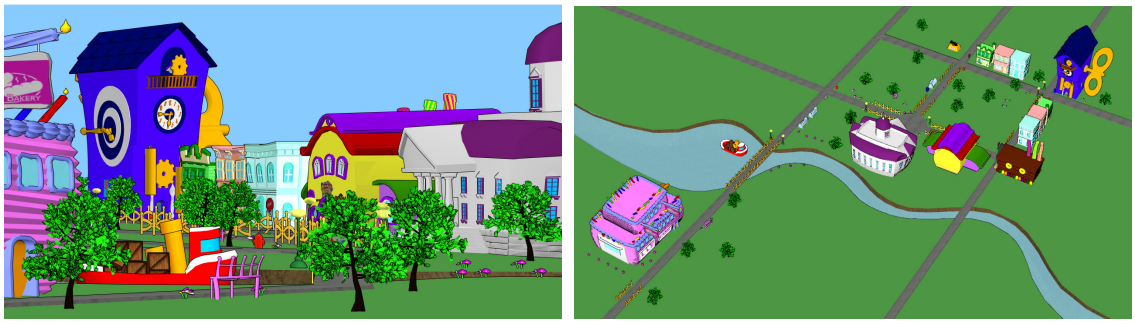


Figure 4.2. A 3-D scene summarization example. The graph camera image that summarizes a cartoon town scene (*left*) and PPC image for comparison (*right*).

Multiperspective visualization has challenges of its own. The multiperspective image is computed using a non-pinhole camera model that does not project 3-D lines to image plane lines, so one challenge is achieving the desired disocclusion effect while minimizing visualization distortions. The non-pinhole camera model is considerably more complex than the PPC model, so a second challenge is to achieve adequate rendering performance to support interactive visualization and dynamic datasets. The elimination of the single viewpoint constraint of the PPC model results in a higher dimensional camera model design space. Whereas for the PPC model the only intrinsic parameter of significant relevance in shaping the visualization is the focal length, optimizing multiperspective visualization requires tuning a larger number of parameters. Consequently a third challenge is to specify the camera model that best visualizes a given dataset from a given location.

The flexibility of the graph camera model makes it useful in many contexts. The most direct application is the enhancement of navigation in virtual 3-D scenes. Instead of being limited to a single viewpoint, the user benefits from an image that integrates multiple viewpoints, which allows viewing multiple scene regions in parallel, without having to establish direct line-of-sight to each scene region sequentially. The enhanced navigation enabled by the graph camera promises to reduce the time needed to find static targets and to greatly increase the likelihood that dynamic targets—moving or transient—are found. In Figure 4.1 the user position is shown with the red frame (bottom, left). The graph camera image lets the user see up to as well as beyond the first street intersections.

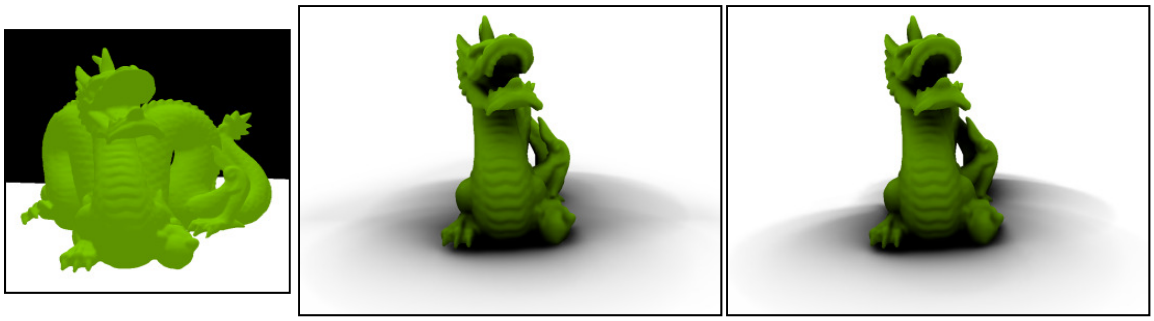


Figure 4.3. An ambient occlusion example. A graph camera depth buffer illustrated with color (*left*) is used to render the ambient occlusion effect (*middle*) and compared to using a PPC depth buffer (*right*). The graph camera captures hidden parts of the dragon for a more complete and stable shadow.

Another graph camera application is in the context of 3-D scene summarization, where the goal is to inventory the representative parts of a scene in a visually eloquent composition. A graph camera can be quickly laid out such as to sample any desired set of scene parts, producing a quality summarization image at a fraction of the time costs associated with previous techniques. In Figure 4.2 the graph camera was constructed to sample most building façades. Sampling the same buildings with a PPC leads to poor image space utilization.

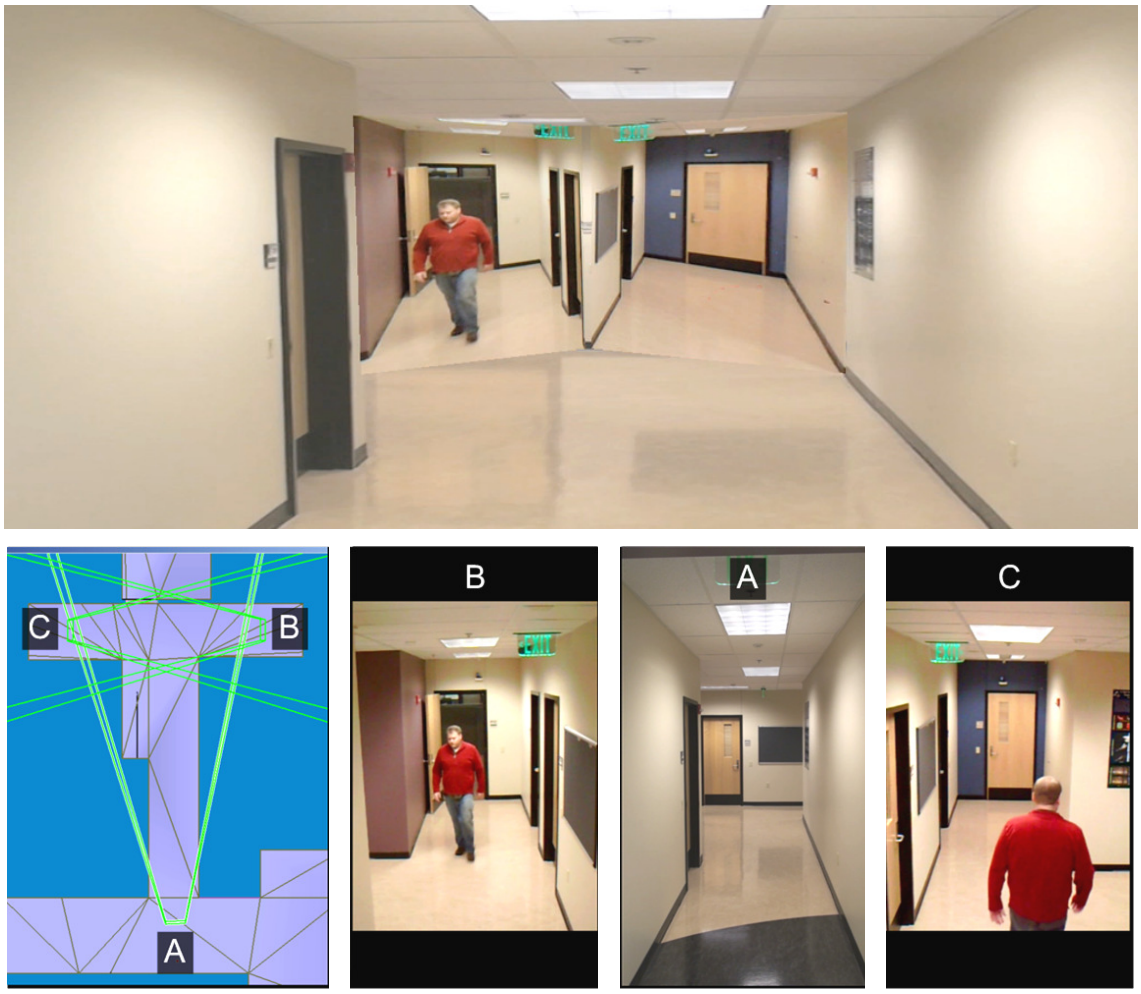


Figure 4.4. A real-world graph camera. The graph camera can be used for single-image comprehensive visualization of real-world scenes. The graph camera image (*top*) seamlessly integrates 3 video feeds (*bottom*) and shows all 3 branches of the t-shaped corridor intersection.

A third application of the graph camera is for use as a geometric approximation in high-quality rendering effects. Since the graph camera captures more than is visible from a single viewpoint, it produces a scene approximation that is more accurate and valid for a greater set of views that a PPC approximation. We demonstrate this in the context of ambient occlusion and environment approximations for reflections. Figure 4.3 shows an example of the ambient occlusion application. Using the graph camera depth buffer, the results are more

accurate in a number of regions (the floor left of the dragon and the back leg and tail to the right of the dragon).

Finally, the graph camera can also be used for visualizing real-world scenes. A physical implementation is obtained by assigning a video camera to each of the PPCs in the graph camera. The result is a seamless integration of the video feeds, which enables monitoring complex real-world spaces with a single image. Unlike individual video feeds, the graph camera image is non-redundant and mostly continuous. In Figure 4.4 monitoring the hallway is facilitated by the graph camera image which bypasses the need to monitor individual video feeds sequentially. Moreover the moving subject is easier to follow in the graph camera image which alleviates the jumps between individual video feeds.

4.1. The Graph Camera

We define 3 basic construction operations on the frustum of a PPC (Figure 4.5). Given a PPC with center-of-projection (COP) C, a plane p, and a point C', the bending operation changes the viewpoint to C' beyond p. The splitting operation introduces two viewpoints C_l and C_r beyond planes p_l and p_r. Splitting is equivalent to two bending operations that act on subsets of the rays of the initial PPC. The merging operation takes two PPCs with COPs C_l and C_r and reduces the two viewpoints to one (C_m) beyond a given plane p. For all operations the resulting rays are defined by two connected segments.

By definition, a graph camera is a directed graph with PPCs at its nodes (Figure 4.6). The graph is constructed starting from a root PPC through a sequence of bending, splitting, and merging operations. For each operation, graph edges connect the input PPC(s) to the output PPC(s). The graph camera rays originate at the COP of the root PPC and are defined by a chain of connected segments. The graph camera image is collected at the root PPC before any frustum operation (see image plane in Figure 4.6).

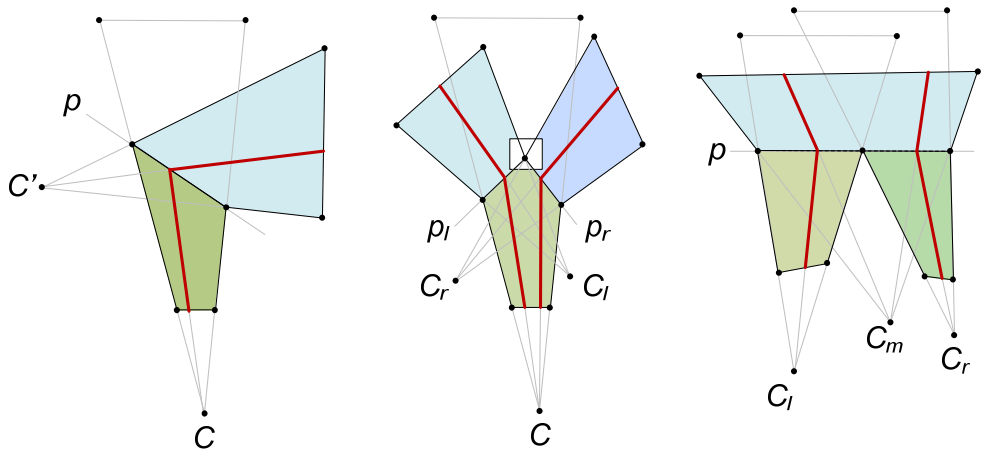


Figure 4.5. Basic graph camera construction operations. Bending (*left*), splitting (*middle*), and merging (*right*) are shown with sample rays in red.

Since each ray is C^0 continuous, the graph camera image is C^0 continuous, except for where splitting lines are visible. For example, in Figure 4.5 the line at the intersection of planes p_l and p_r is hidden by geometry (white rectangle) which avoids a discontinuity in the output image. The graph camera image is non-redundant as long as the frusta are disjoint. Unlike most camera models, the graph camera is defined based on the actual 3-D scene it is visualizing. Its rays are designed to circumvent occluders to reach deep into the scene.

Once the camera model is defined, graph camera images can be rendered using ray tracing, by intersecting the piecewise linear rays with scene geometry. However, faster feed-forward rendering is possible due to the availability of a fast projection operation. Given a graph camera with root frustum PPC_0 and a 3-D point P inside a frustum PPC_i, one can directly compute the image plane projection P' of P using a 4-D matrix $M_{0i} = M_0M_1...M_i$, where M_k (k = 0..i) are the 4-D PPC matrices for the frusta on the path from PPC_0 to PPC_i. For example point P in Figure 4.6 is projected with matrix $M_0M_1M_3$. A PPC matrix is the product between the usual projection and the view matrices of the PPC. The matrix M_{0i} is computed for each frustum PPC_i at graph camera construction. Further derivation of the projection function can be found in Appendix A.

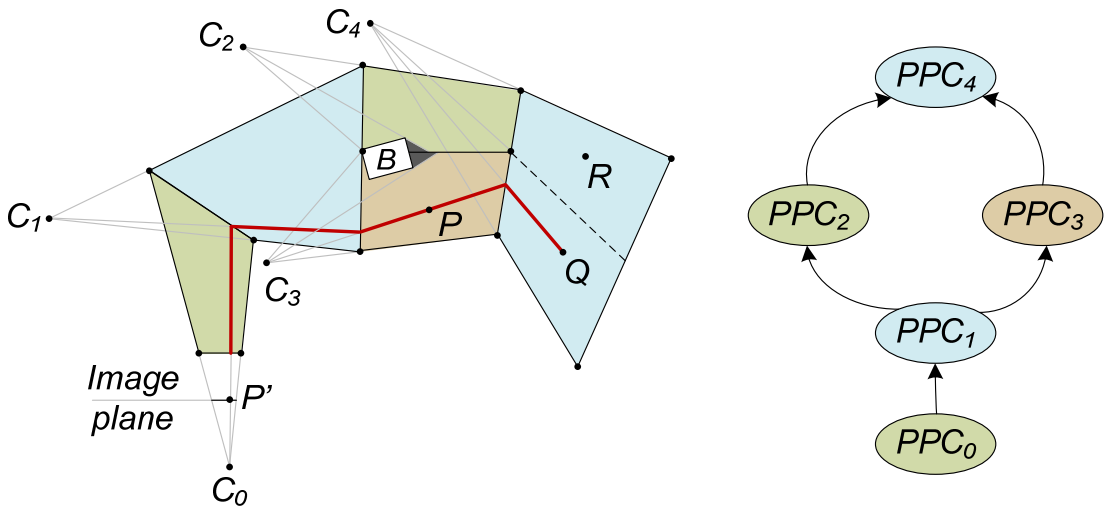


Figure 4.6. A graph camera with five frusta. First, PPC_0 is bent. PPC_1 is split into PPC_2 and PPC_3 then merged into PPC_4 to shrink the occlusion shadow of the rectangular object B. P and Q are projected at P'.

Special care needs to be taken when deriving the projection matrix for frusta downstream from a merging operation. For example in Figure 4.6 point Q is projected with matrix $M_0M_1M_3M_4$ and point R with matrix $M_0M_1M_2M_4$. Although all rays inside frustum PPC_4 converge to C_4, the frustum is implemented with two sub-frusta, each with its own projection matrix. Projection is defined using a tree and not a graph, whose unique paths from a node to the root define the projection operation unambiguously.

A scene modeled with triangles is rendered with a graph camera one PPC frustum at the time. The triangles intersecting frustum PPC_i are found using a conventional hierarchical space subdivision scheme; we use an octree. A triangle is clipped with PPC_i, vertices are transformed and projected with the matrix M_{0i} of PPC_i, and the projected triangle is rasterized conventionally. The graph camera image is a collection of PPC pieces, thus conventional linear rasterization can be used within each piece.

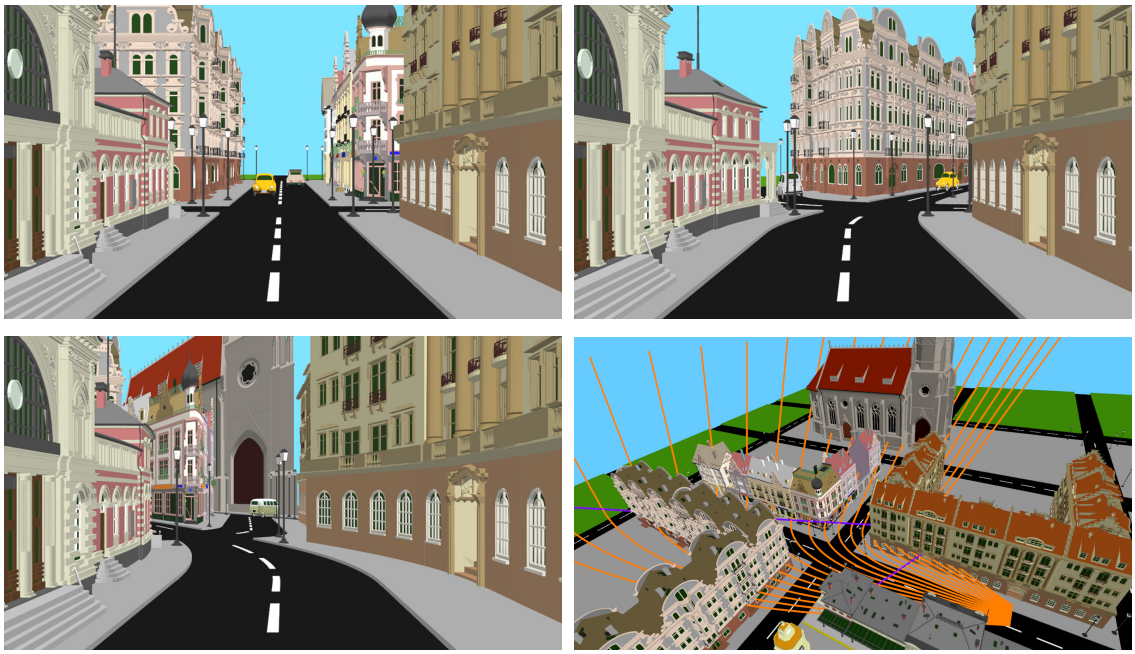


Figure 4.7. A curved ray camera street-level visualization example. A conventional PPC street-level visualization (*top*, *left*), and CRC visualizations showing the left (*top, right)* and right side streets (*bottom, left*) and a visualization of CRC rays (*bottom, right*).

4.2. The Curved Ray Camera

To further address the challenges of multiperspective visualization, we developed a novel multiperspective visualization technique based on the curved ray camera (CRC) which integrates multiple viewpoints seamlessly. While the graph camera has an abrupt C^0 transition between neighboring perspectives, the curved rays of the CRC allow for a progressive transition between one viewpoint and the next. A CRC ray is a sequence of line segments connected by conic curve segments. Each conic connects consecutive line segments with C^1 continuity, which alleviates visualization distortions. The CRC provides a fast projection operation which allows rendering 3-D surface datasets efficiently by projection followed by rasterization, with the help of graphics hardware. The rays of the CRC can be traced inexpensively which enables visualization techniques that require ray

casting, such as volume rendering. We have developed several CRC constructors which further address the graph camera applications of enhanced navigation and scene summarization.

Figure 4.7 illustrates the use of the CRC to alleviate occlusions in visualization. A CRC is used to preview the two side streets without advancing the camera. The CRC samples the main street up to the intersection and then switches to a second viewpoint to sample the side streets. The CRC rays for the bottom left image are also shown. Each ray consists of first a line segment, then a conic curve segment, and finally a second line segment (see purple lines). The first set of line segments converge at the first viewpoint, sampling the main street, and the second set of line segments converge at the second viewpoint, sampling the right side street. The conic curves implement the viewpoint change over a transition region. In Figure 4.8 the car is located in the transition region. The car intersects a relatively small piece of the curved rays. A small piece of the ray is approximately straight, and distortions are minimized (left). Without the transition region, switching directly from the first to the second line segment (i.e. graph camera piecewise linear rays with only C^0 continuity), a disturbing distortion of the car occurs (right).

Figure 4.8. Distortion comparison between the curved ray camera and graph camera. The CRC visualization (*left*) minimizes the distortion through the perspective transition region as opposed to the graph camera which switches abruptly between viewpoints (*right*).

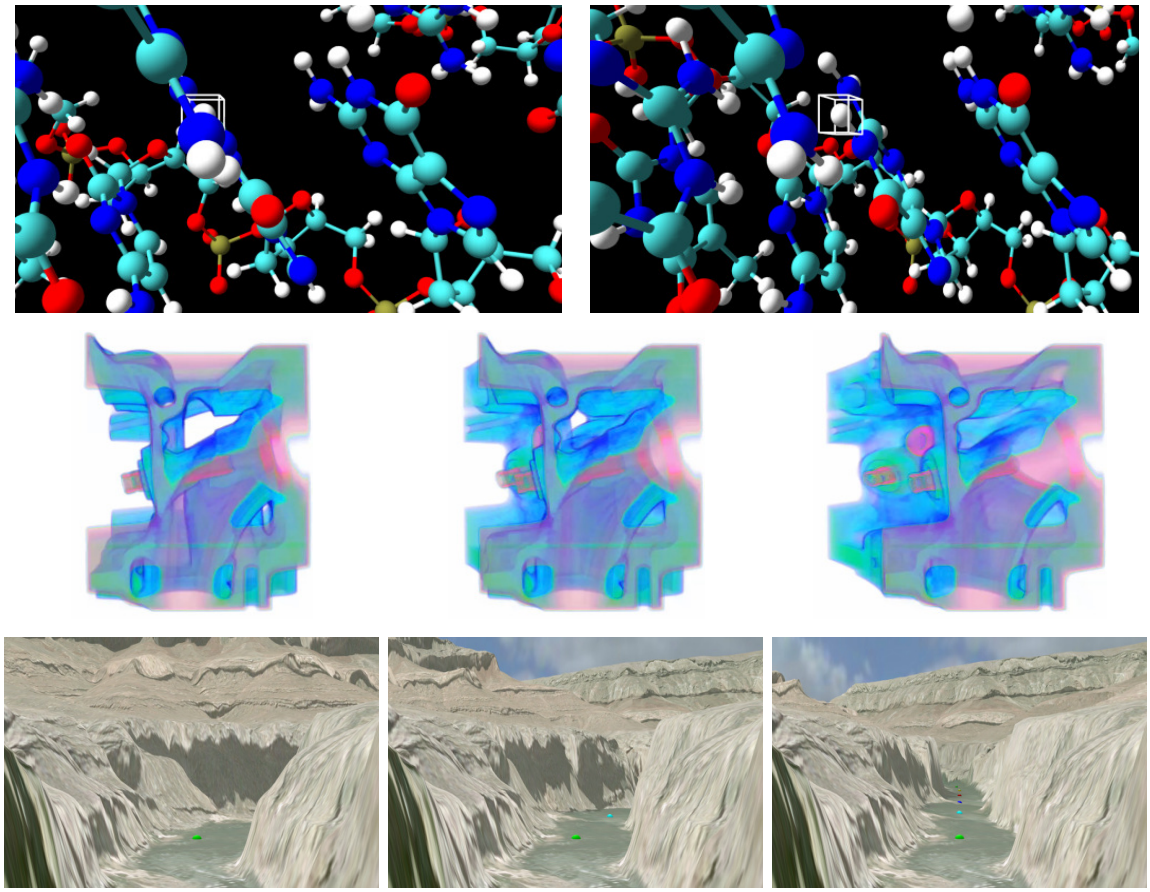


Figure 4.9. A series of curved ray camera examples. *Top:* DNA molecule with target atom shown with wireframe bounding box occluded in the PPC view (*left*) and disoccluded in the CRC view (*right*). *Middle*: engine block dataset volume rendered with a PPC (*left*) and a CRCs with rays of increasing curvature (*middle and right*). *Bottom:* canyon terrain dataset visualized with a PPC (*left*) and CRCs that preview an increasing section of the river bed ahead (*middle and right*).

The first row of Figure 4.9 illustrates CRC target tracking. Target tracking is illustrated in the context of the visualization of a DNA molecule. The user selects an atom as the target, and as the view translates, the algorithm attempts to avoid the occlusion of the target by modifying the parameters of the CRC dynamically. The second row illustrates the CRC in the context of volume rendering. The third row illustrates CRC path previewing in the context of the visualization of a canyon terrain dataset. The path was chosen to correspond to the river. The

CRC effectively linearizes a section of the river. The length of the linearized section is controlled with a user parameter. Evenly-spaced colored markers were added along the path in order to illustrate the path preview effect and to facilitate comparing the images.

4.2.1. Camera Model

The design of the curved ray camera has to allow circumventing occluders and reaching deep into the dataset to disocclude subsets of interest. Given a starting planar pinhole camera PPC_0 with viewpoint C_0, a plane t_1, and a second viewpoint C_1, we want to build a camera model that uses viewpoint C_0 up to t_1 and then switches to C_1. In order to make the transition smooth we use two additional planes t_0 and t_2 that define a transition region where the viewpoint change occurs (Figure 4.10). We connect the C_0 and C_1 rays with a quadratic polynomial Bézier, which is the simplest curve that provides C^1 continuity at both connection points. For example, the CRC ray through P_0 is the C_0 ray up to plane t_0, then the quadratic Bézier with control points P_0, P_1, and P_2, and then the C_1 ray beyond plane t_2.

Given an image plane point P, the CRC camera ray is found by first intersecting ray C_0P with planes t_0 and t_1 to obtain points P_0 and P_1, then ray C_1P_1 is intersected with plane t_2 to obtain point P_2, and then control points P_0, P_1, and P_2 define the ray as explained above. The CRC can be extended by appending additional viewpoints, each with its own transition region. The resulting CRC rays are a sequence of line segments interconnected by quadratic Bézier arcs.

The CRC model can be used directly to support visualization techniques that require ray casting, such as volume rendering. A CRC ray is traced by tracing the sequence of segments and arcs. The arcs are traced by iterating parameter *t* in the Bézier equation below.

$$P = (1-t)^2 P_0 + 2t(1-t)P_1 + t^2 P_2 \qquad \text{Equation 4.1}$$

Equal *t* increments of course do not correspond to arc steps of equal length. For applications where the uniformity of the step is important one could evaluate Equation 4.1 with small steps in *t* and to use a piecewise linear approximation of the length of the arc step. In addition to the ability to trace rays, ray casting also requires the ability to clip a ray with the bounding volume of the dataset. Computing the intersection between a Bézier ray and a plane is easily done by substituting in the expression of a ray point given by Equation 4.1 into the plane equation, which results in a quadratic equation in parameter *t*.

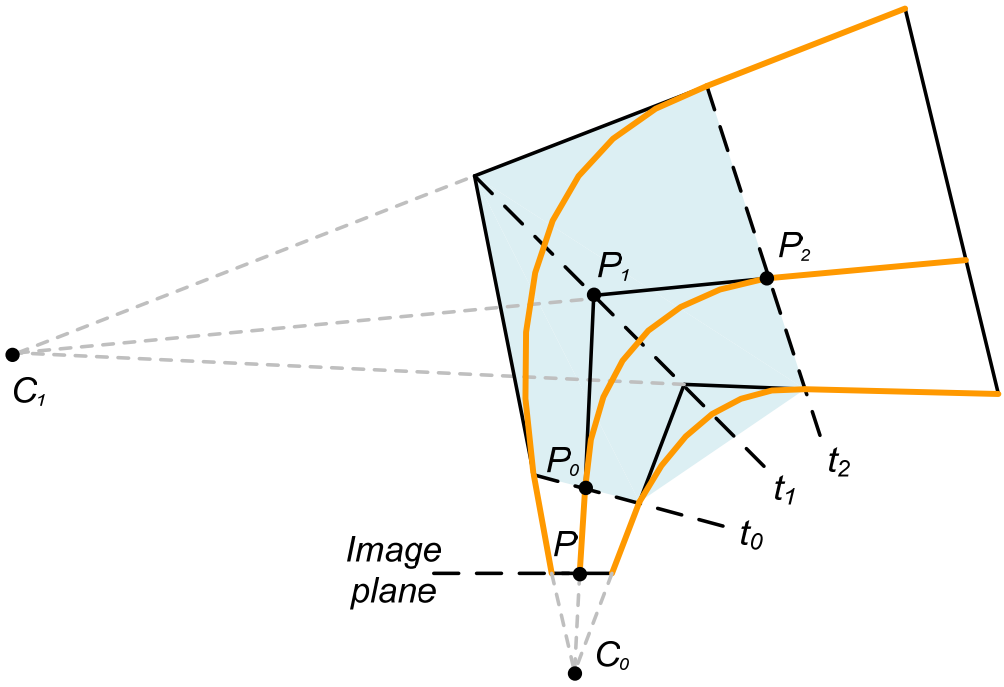


Figure 4.10. A visualization of the curved ray camera model.

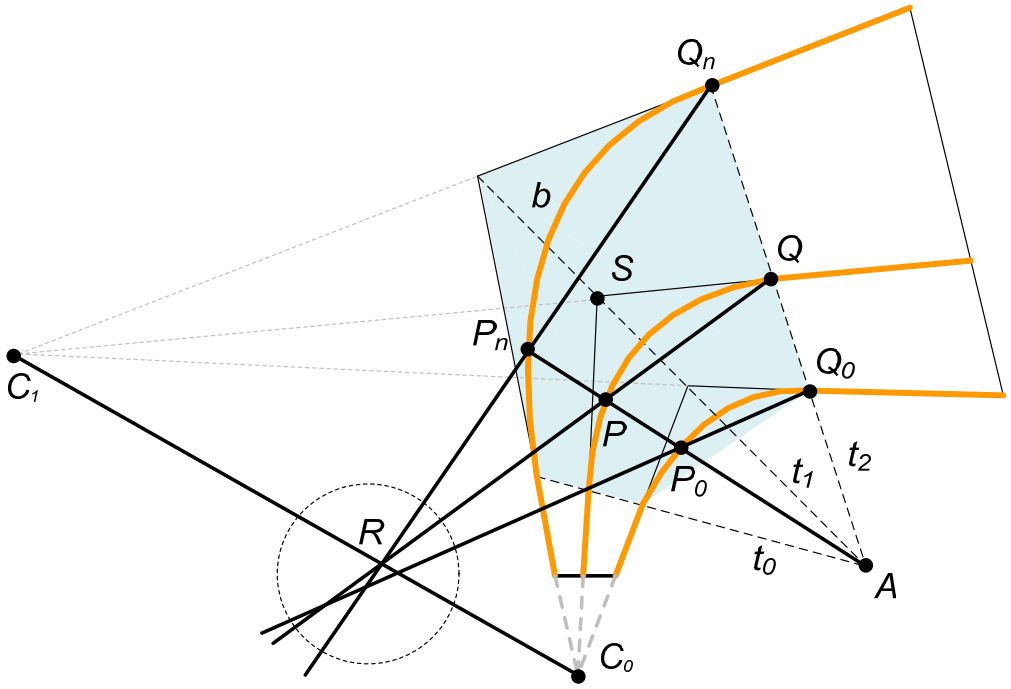


Figure 4.11. Illustration of desired fast projection operation.

It is straightforward to trace the CRC rays as defined, but the approach of choice for interactive visualization of 3-D surface datasets remains feed-forward rendering through projection followed by rasterization. Unfortunately, given a 3-D point P one cannot inexpensively compute the image plane projection of P with the CRC model as defined because the projection equation cannot be solved in closed form and a numerical solution is too expensive. We implemented a numerical solution based on the bisection method and found solutions with sub-pixel accuracy after an average of 10 iterations, too slow for interactive visualization of complex datasets. Fortunately a small modification to the CRC model enables fast, closed-form projection of 3-D points.

4.2.2. Camera Model with Fast Projection

We developed a fast projection operation based on the observation that it is advantageous to choose the three planes t_0, t_1, and t_2 defining the transition region such that they intersect along a common line l. This does not come at a significant loss of generality since planes t_0 and t_2 simply mark the beginning and the end of the transition region. Given a 3-D point P to be projected, let A be the intersection of l with the epipolar plane C_0C_1P, and let P_i be the intersection points of line AP and the bundle of Bézier arcs (Figure 4.11). Let Q_i be the intersection points between the plane t_2 and the same Bézier arcs. Then lines P_iQ_i are almost concurrent, and the near-intersection occurs close to the baseline C_0C_1 (see dotted circle).

4.2.2.1. Desired Projection Operation

If lines P_iQ_i truly intersected at a point on C_0C_1, point P could be projected with the following steps:

1. Intersect AP with the known Bézier arc b through Q_n to obtain point P_n.
2. Intersect P_nQ_n with baseline C_0C_1 to obtain point R.
3. Intersect RP with plane t_2 to obtain Q.
4. Intersect C_1Q with plane t_1 to obtain point S.
5. Intersect C_0S with the image plane.

The first step implies solving a single variable quadratic equation, as discussed above (b is intersected with a plane through AP). All other steps are simple line-plane intersections. Moreover the projection of point Q onto plane t_1 using C_1 (step 4) and the projection of S onto the image plane (step 5) can be combined into a single projection by concatenating the projection matrices of PPC_1 and PPC_0. PPC_1 is defined by viewpoint C_1 and image plane t_1. Points inside the frustum of PPC_0 but outside the transition region are simply projected with PPC_0.

Points inside the frustum of PPC_1 but outside the transition region are projected with the concatenated projection matrices of PPC_1 and PPC_0. The cost of projection does not increase as additional turns are added to the rays of the CRC. The projection matrices corresponding to the additional turns are simply concatenated such that points are projected directly to the image plane, in one step.

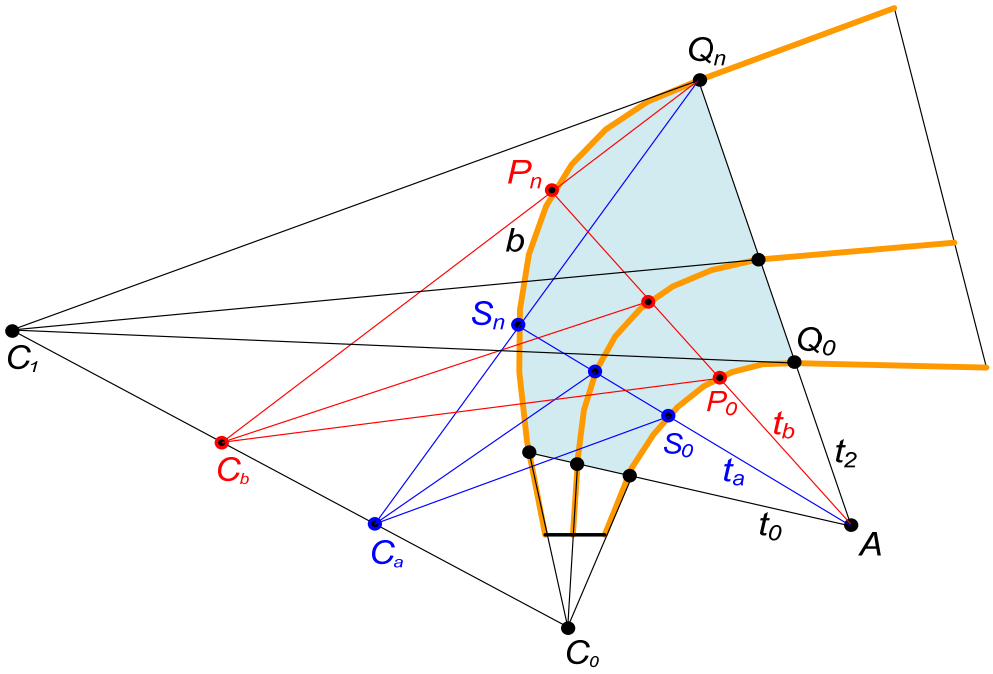


Figure 4.12. The modified curved ray camera model for fast projection.

4.2.2.2. Camera Model Modification to Enable Fast Projection Operation

We slightly modify the CRC model in order to have a precise convergence of lines P_iQ_i on baseline C_0C_1, such that the projection algorithm sketched above applies. A modified CRC ray is redefined as follows (refer to Figure 4.11 again). Given a point on the image plane we compute points S and Q as before, using rays through viewpoints C_0 and C_1, respectively. The left-most ray in plane C_0C_1Q remains the same as before, a quadratic Bézier arc b. Then the CRC ray in the transition region is defined by rotating a line through point A from plane t_0

to plane t_2. For each intermediate position, the line is intersected with b to define point P_n, and then Q_nP_n is intersected with C_0C_1 to find point R. The ray point P is defined by the intersection of RQ and line AP_n.

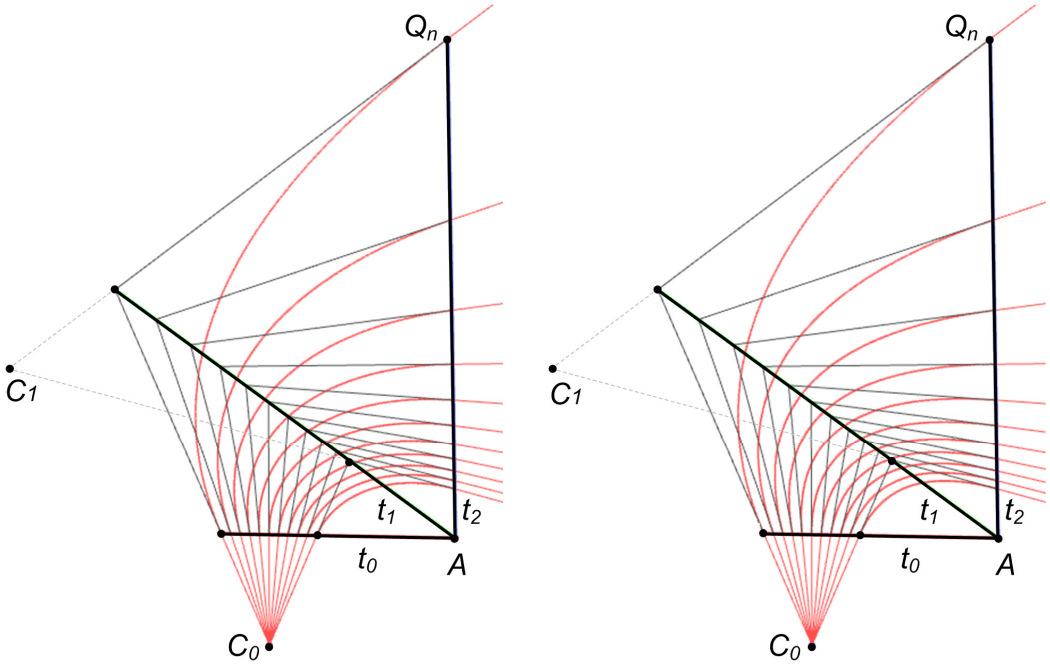


Figure 4.13. Comparison between the modified and original curved ray camera models. The rays of the modified CRC model (*left*) are virtually identical to the original (*right)* CRC model with the advantage of fast projection.

The modified CRC model switches gradually from C_0 to C_1 through a continuum of intermediate viewpoints on the baseline segment C_0C_1 (Figure 4.12). Each intermediate viewpoint C_i is used to image all 3-D points located in a plane t_i of the transition region. The plane t_i is defined by the line l common to planes t_0, t_2, and t_1 (not shown) which define the transition region, and by a point X_n. X_n can be defined in any epipolar plane e through C_0C_1 as the intersection between line C_iQ_n and the Bézier arc b in plane e. For example intermediate viewpoints C_a and C_b with the image planes t_a and t_b. In other words, given a point P, the viewpoint to use depends on where line PA intersects the Bézier arc b. If the intersection occurs close to Q_n, the viewpoint to use is closer to C_1.

Figure 4.13 shows the effect of the camera model modification. The left most ray in each epipolar plane is the same as before, i.e. the ray through Q_n remains a Bézier arc; the other rays change slightly. As we will show these rays become conic arcs. Lines P_iQ_i of the modified model converge on the baseline C_0C_1 by construction and hence the projection algorithm above applies. The rays of the modified CRC model are C^1 continuous at both ends of the transition region, and they do not intersect, as shown in Appendix B.

4.3. Graph Camera Applications

4.3.1. Scene Exploration

In conventional 3-D scene exploration the user positions and orients a PPC interactively in order to gain direct line-of-sight to various regions of the scene. When a discovered region is uninteresting, the effort of navigating to the region and back is wasteful. We propose exploring 3-D scenes with a graph camera that continually adapts to the scene in order to reveal not only what is visible from the current position but also adjacent scene regions to which there is no direct line-of-sight. The additional information displayed by the graph camera image assists the user in selecting a navigation path that is more likely to reveal regions of interest, increasing navigation efficiency. We have developed several graph camera constructors which, given a 3-D scene and a current user position, define rays that circumvent occluders to sample multiple scene regions. Most constructors can be used in combination but are presented here individually for clarity.

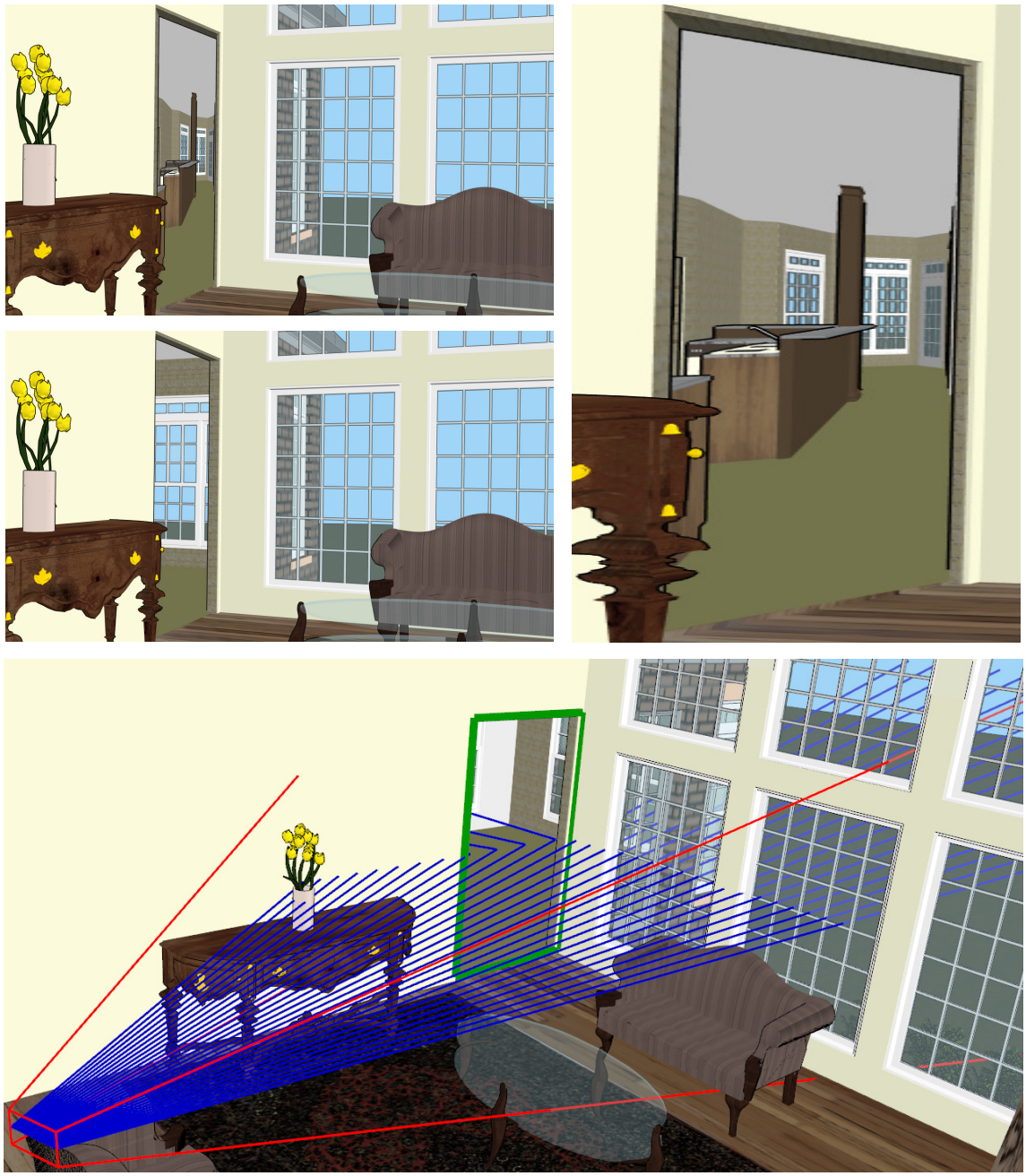


Figure 4.14. A portal-based graph camera image. A comparison of a portal-based graph camera image (*top, left* and fragment *right*) and PPC image (*middle, left*) along with a visualization of the portal-based graph camera (*bottom*).

4.3.1.1. Portal-Based Constructor

Portals such as doors, windows, and spaces between buildings in indoor and outdoor architectural scenes are natural gateways between neighboring scene regions. We have developed a graph camera constructor that allows the user to see directly into adjacent scene regions. In Figure 4.14 the graph camera captures a good view of the adjacent kitchen while the PPC sees only an uninteresting corner through the portal.

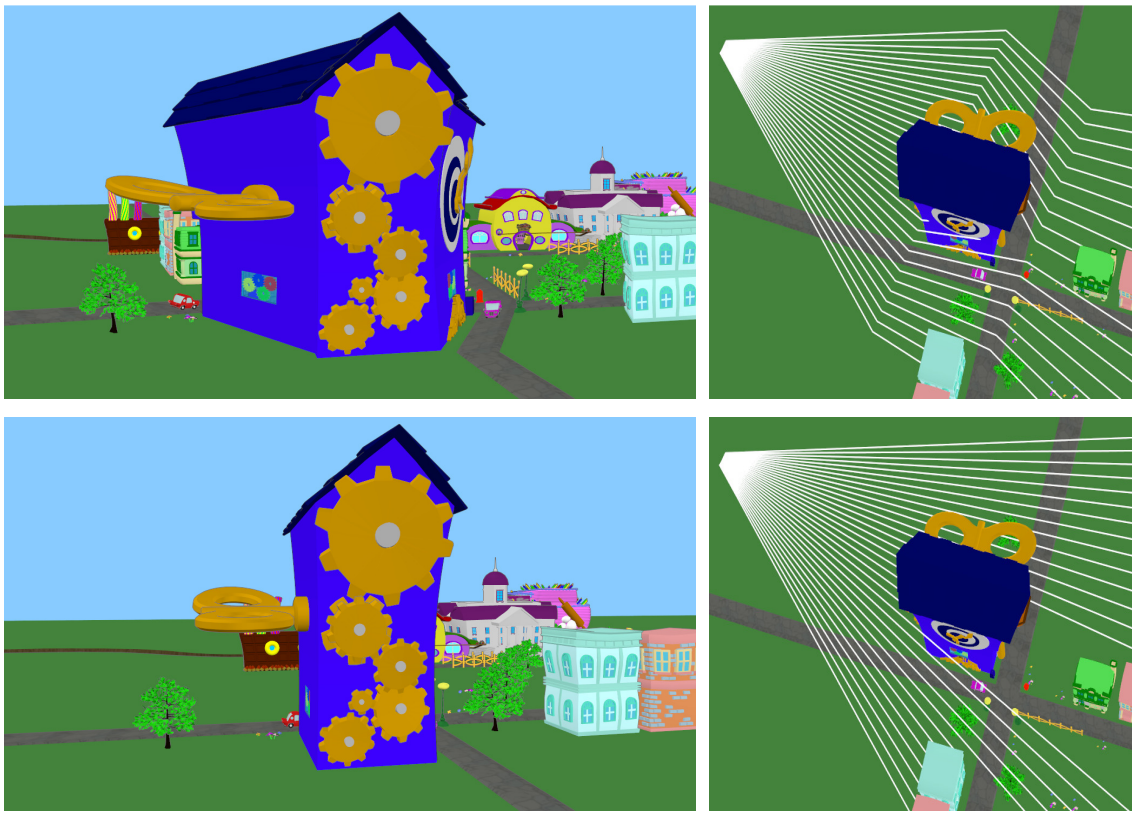


Figure 4.15. Occluder-based graph camera image. The occluder-based graph camera image (*top, left*) is compared to a PPC image (*bottom, left*). A ray visualization (*right*) of each is also shown.

Given a 3-D scene with predefined portals and a current view PPC_0, the portal-based constructor builds a graph camera with a root frustum PPC_0 and one leaf frustum for each portal visible in PPC_0. Each leaf frustum PPC_i is defined by

bending the rays of PPC_0 that sample the portal frame such that PPC_i has a view direction normal to the portal plane. In Figure 4.14 a single portal is visible, shown with green. The root frustum PPC_0 is shown with red and the graph camera rays are shown with blue. As the user navigates new portals become visible. The additional perspectives are introduced gradually by morphing the new view direction from the view direction of PPC_0 to the normal of the portal plane. As the user approaches a portal, the leaf frustum view direction is morphed back to the PPC_0 view direction, which maintains visualization continuity as the user transitions through the portal.

4.3.1.2. Occluder-Based Constructor

The converse of a portal is an occluder—a large opaque object that hides potentially interesting scene regions. We have developed a graph camera constructor that allows the user to see behind an occluder. Given a 3-D scene with a predefined occluder bounding box B and a current view PPC_0, the occluder-based constructor builds a graph camera with rays that reach around B. The graph camera is constructed with a split followed by a merge (Figure 4.6). In Figure 4.15 the graph camera shrinks the occlusion shadow of the clock store to capture buildings and streets hidden in the PPC image, as well as the side faces of the clock store.

4.3.1.3. Target Tracking Constructor

The occluder-based constructor disoccludes around a particular object, without any concern as to what object is revealed behind the occluder. We have designed a constructor that uses the disocclusion capability of the CRC to modify the camera in order to maximize the visibility of an object of interest, i.e. target, as the view, the target, and/or other objects move. The CRC is controlled with 3 parameters: the depth z_0 from the first viewpoint C_0 where rays should start to

bend, a translation amplitude *a* that defines the maximum lateral offset of the second viewpoint C_1 with respect to C_0, and a fraction *f* between -1 and 1 that modulates the maximum offset. When *f* is -1/+1 the CRC rays are bent all the way left/right. When *f* is 0 rays are straight and the CRC is a PPC. We set both z_0 and *a* as half the distance from C_0 to the target. The ray bending fraction *f* is updated dynamically for every frame with the following algorithm.

1. If the target is visible for $f = 0$, set *f* to 0.
2. Else if the target is visible with current *f*, keep current *f.*
3. Else search for a new *f* such that the target is visible.

Visibility of the target for a given value of *f* is determined efficiently at bounding box level: axis aligned bounding boxes are fitted to the objects and target, the bounding boxes are projected on the CRC image, and the image aligned bounding boxes of the projections are tested for intersection. The target bounding box is enlarged a user chosen number of pixels to achieve a clear disocclusion of the target.

The rays are bent only if needed and the same bending factor is kept if possible. The search for a new *f* value that disoccludes the target starts at the current *f* value and iteratively tests *f* values left and right at increasing distance. If the search succeeds *f* is updated to the new value, else *f* is set to 0 (i.e. the target cannot be disoccluded). The frame to frame change of *f* is capped to avoid abrupt changes in the visualization. The gradual change of *f* comes at the cost of occluding the target for a few frames as the rays swing to the new value of *f* that disoccludes the target. Figure 4.16 illustrates target tracking on a test dataset comprising 700 randomly placed and sized blocks. The algorithm finds a CRC that disoccludes the red target. The only state data maintained by the algorithm is the ray bending factor *f*, so it directly supports dynamic occluders, dynamic targets, and dynamic views.

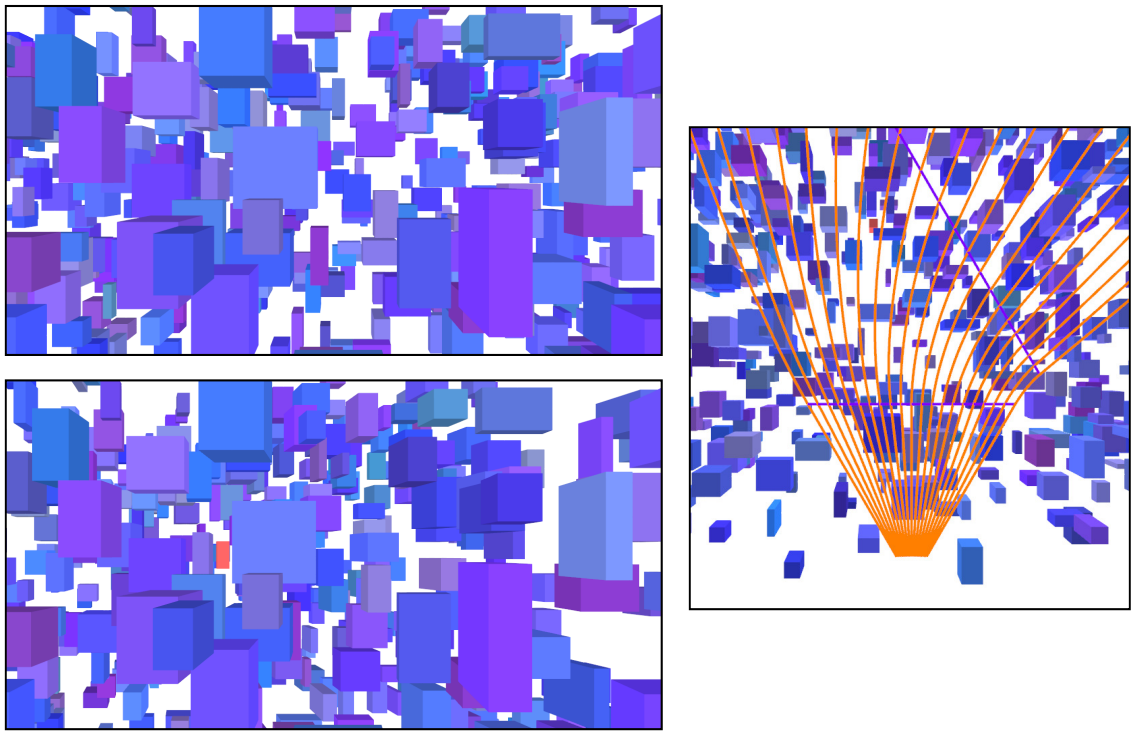


Figure 4.16. Curved ray camera used to disocclude a red target object. The red object is not visible with the PPC (*top*), but the CRC (*bottom*) disoccluded the red target by bending the CRC rays (*right*).

4.3.1.4. Maze-Based Constructor

The previously mentioned graph cameras have good disocclusion capability when the current user position is relatively close to a portal or an occluder. However, when the current position is far away, the root frustum projection of the portal or occluder is small, which limits the output image size of the visualization for the disoccluded regions. Consider for example the bottom right image in Figure 4.1. The left and right side street openings are too small to accommodate a good visualization of the side streets.

We have developed a powerful graph camera constructor that achieves a good, user-controllable visualization of adjacent regions, independent of how big or far the portal is. As a preliminary step, the set of possible desired user paths is

modeled with a maze. The maze defines virtual tunnels through which the user navigates during scene exploration. The maze also serves as virtual scaffolding for constructing a graph camera for each user position. The maze is built by fitting rectangular boxes to path intersections (Figure 4.17). The boxes, which can have different sizes, are connected by tunnels with four planar faces. The cross sections of tunnels are modulated with rings to allow modeling streets with building facades that are not aligned.

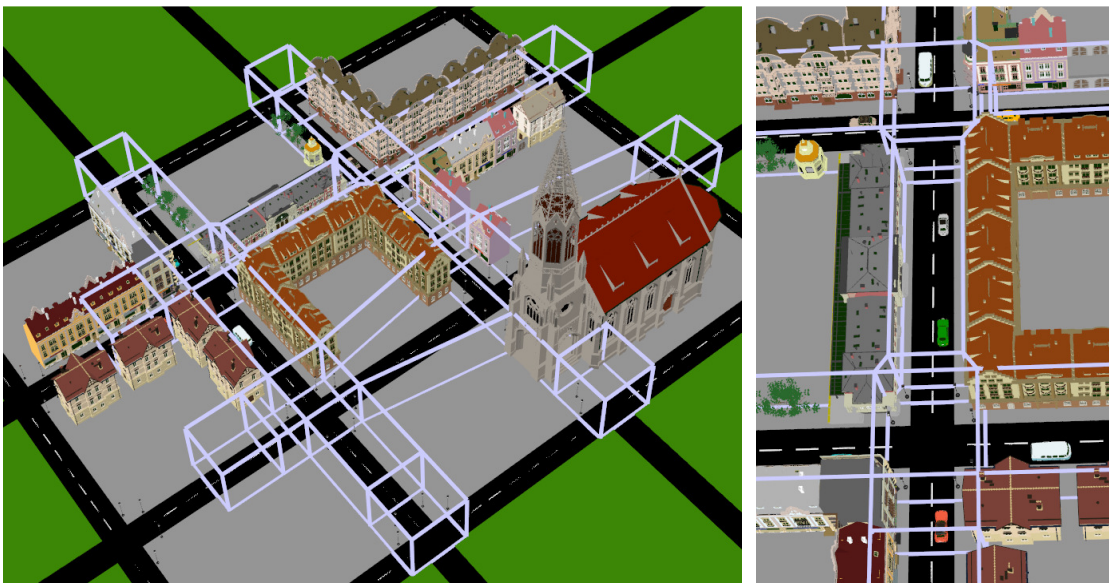


Figure 4.17. Visualization of a maze construction. The maze was used to construct the graph camera from Figure 4.1.

Given a 3-D scene enhanced with a maze and a user position inside the maze, a graph camera is constructed that allows the user to see from the current position to the first intersection ahead, as well as along the virtual tunnels emanating from the first intersection. In order to allow the user to see behind, a similar graph camera is constructed pointing backwards. In Figure 4.18 top, the user is located at the quad Q_u. The forward and the backward graph cameras each have 6 PPC frusta. The graph for the forward graph camera is shown Figure 4.18, bottom, left.

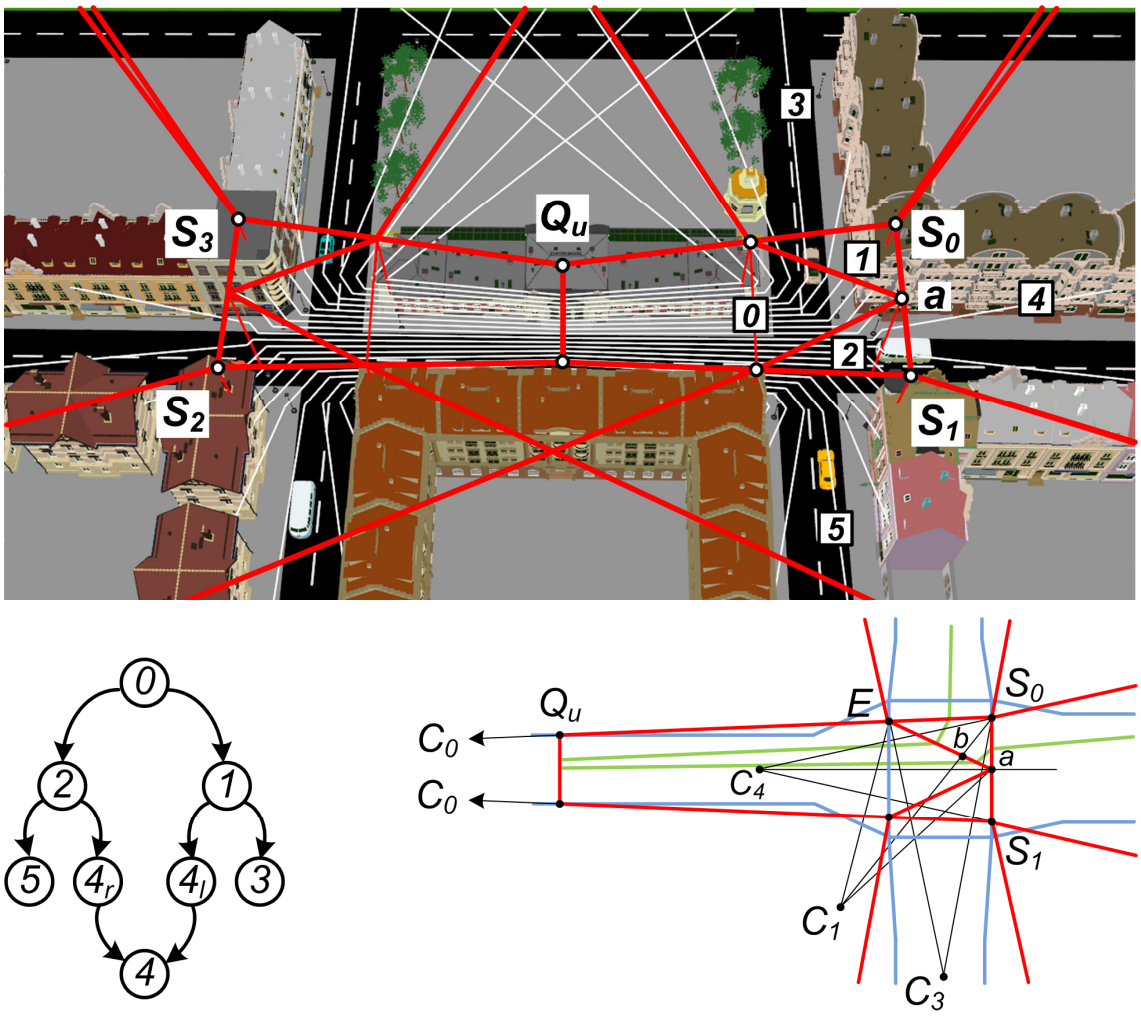


Figure 4.18. Visualization of the maze-based graph camera. *Top*: Visualizations of forward and backward graph cameras from Figure 4.1. *Bottom*: Graph of PPC frusta for forward graph camera (*left*) and construction details (*right*).

The forward graph camera is constructed using the maze (blue lines in Figure 4.18) starting from the user quad Q_u. The root frustum PPC_0 with COP C_0 is constructed using the user quad and a field-of-view parameter. Points E and S_0 are defined by the intersection of PPC_0 frustum edge C_0Q_u with the near and far intersection box faces. Frustum PPC_1 is constructed using a field-of-view parameter along with the points E, a, and b. PPC_0 rays that hit between E and b are directed to the left corridor, and rays between a and b are directed to the forward corridor (see green lines). Points a and b are derived from ray split ratio

parameters. For example, for an even (1/3, 1/3, 1/3) split, a needs to be half way between S_0 and S_1 and *Eb* needs to capture the left 1/3 of the rays of PPC_0. PPC_3 is constructed using a field-of-view parameter and the points *E* and S_0. PPC_{4l} is built using a field-of-view parameter and the points S_0 and *a*. The right half of the graph camera is constructed similarly.

The raw image produced by the forward graph camera is shown in Figure 4.19, where the forward-right perspective is emphasized using a split ratio of (1/6, 1/6, 2/3). The raw images are texture mapped to a display surface that rotates the back image 180° while maintaining the connection to the top image. The surface also splits the image along discontinuities to obtain the final image shown in Figure 4.1. The final image shows in front (right panel) and behind (left panel) the user. Each panel has a left, a center, and a right perspective. The discontinuities between perspectives are defined by the visible parts of the vertical split lines through points S_0, S_1, S_2, and S_3 (Figure 4.18) and are shown as grey vertical lines in Figure 4.19. At a conceptual level, the two step process first achieves the desired disocclusion effect and then rearranges the image regions optimizing visualization eloquence.

Figure 4.19. Top half of uneven split raw graph camera image.

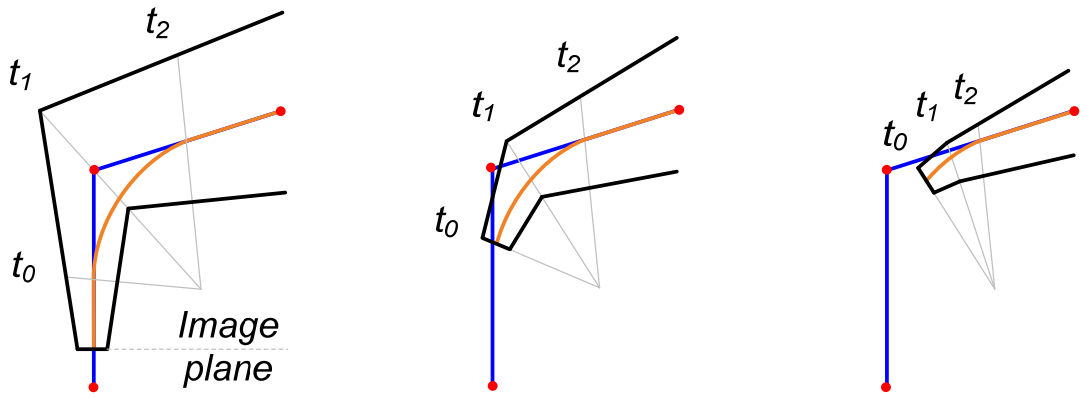


Figure 4.20. Illustration of curved ray camera construction for path tracking. As the current position advances the transition planes are collapsed.

The graph cameras are rebuilt for every frame. Since the visibility of split lines can change abruptly from frame to frame, the depth of each split is averaged over 30 frames. When the user reaches an intersection the graph cameras turn in 4 steps. Let's assume that the user desires to turn right. First, the lateral perspectives of the backward camera are retracted, after which the backward graph camera becomes a simple PPC with a view direction aligned with the current street. Then the forward left and forward center perspectives are retracted in favor of the chosen forward right perspective. In step 3 the backward camera turns to show the old forward left street (the new backward center street). Finally the lateral perspectives of the forward and backward cameras are redeployed. Retracting and deploying perspectives is done by changing split ratio parameters.

4.3.1.5. Path Previewing

We have developed an even more flexible constructor for visualizally exploring heavily occluded scenes using the CRC. The constructor alleviates occlusions along a given visualization path through a 3-D dataset. Visualization paths can be defined in a variety of ways, including by leveraging inherent properties of the dataset (e.g. a river cutting through a canyon, a road, a blood vessel), by

following an object that moves through the dataset, or by finding and saving a sequence of relevant views through interactive visualization. As the turns in the path are typically chosen to circumvent occluders, following a path with a PPC limits the visualization to the first turn in the path. We have developed an algorithm for constructing a CRC that previews forthcoming parts of the path. The CRC linearizes the path locally which allows the user to see beyond one or several turns in the path (Figure 4.21). The algorithm takes two parameters as input: the current position along the path, and how much of the path should be linearized, i.e. the preview length. Both parameters are under user control. The user can advance the current position and increase or reduce how far ahead the visualization shows.

For a given position and preview length, the CRC is constructed using control points (red dots in Figure 4.20) along the path (blue line). Three consecutive control points that are not collinear define a turn. The current position corresponds to the image plane. When the image plane is about to enter a transition region, the planes t_0, t_1, and t_2 defining the transition regions are collapsed progressively (see middle and right figures). Conversely, as the current position advances, new control points enter the linearization region and their transition planes are deployed progressively.

4.3.2. Scene Summarization

Summarization aims to capture important regions of the scene and to arrange them on the canvas in an eloquent composition. The resulting image should inventory, explain, and/or advertise the scene. The summarization image is *not* a map, in the sense that global spatial relationships are *not* preserved. Summarization images break free from single perspective rules to achieve comprehensive scene visualization and optimal composition.

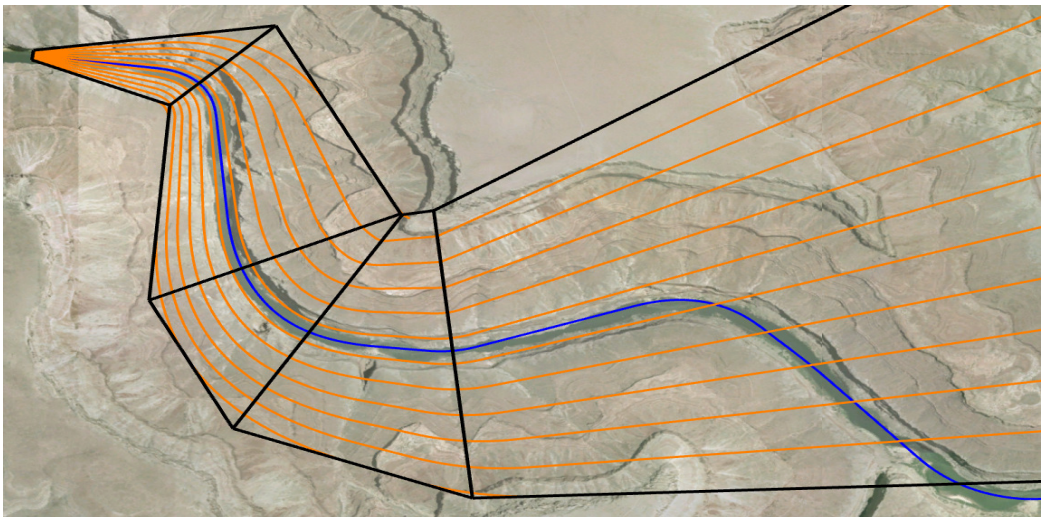


Figure 4.21. Visualization of a curved ray camera with multiple bends. The camera was used to render the bottom right image in Figure 4.9. The blue line shows the path that follows the river. The CRC conforms to a section of the path using 4 turns.

A first conventional method for summarization is drawing. The artist represents only certain scene regions and scales and rearranges them at will. However, the method is laborious and requires drawing talent. A summarization image can also be assembled from photographs or computer graphics renderings of the desired scene regions. Sophisticated digital image editing tools allow collating the ingredients seamlessly. However, the method is time consuming. We asked a computer graphics student to replicate the graph camera cartoon town summarization image from Figure 4.2. The resulting collage (Figure 4.22, left) was assembled in 8½ hours from 14 rendered shots. Another disadvantage of collages is lack of 3-D continuity: a car driving down the streets of the cartoon town would move discontinuously, jumping from shot to shot. A third conventional method for creating summarization images is to rearrange the 3-D models of important scene regions in a new 3-D scene which is then rendered with a conventional PPC (Figure 4.22, right). The method has the advantage of 3-D continuity—objects can move in the consistent 3-D space of the auxiliary scene. However, the method requires remodeling the infrastructure that anchors

the original scene regions in the new configuration. This considerable change is reflected in the summarization image and the identity of the original scene is somewhat lost.

The graph camera is well suited for designing summarization images quickly and with good continuity. We have developed two methods for constructing graph cameras for scene summarization: interactive construction, and recursive maze-based construction.

Figure 4.22. Collage and 3-D remodeling summarization images.

4.3.2.1. Interactive Construction

We have developed an interactive graph camera constructor that allows the user to add, configure, bend, split, and merge PPC frusta through a graphical user interface. The fast graph camera rendering algorithm allows for changes to the camera model to be propagated to the graph camera image in real time. Figure 4.23 shows the graph camera constructed interactively to make the cartoon town summarization image in Figure 4.2. Construction took less than 5 minutes, a great improvement over the 8½ hours for the comparable collage. Moreover, the graph camera offers continuity: a car driving from the bakery (pink building) to the clock store moves continuously in the graph camera image.

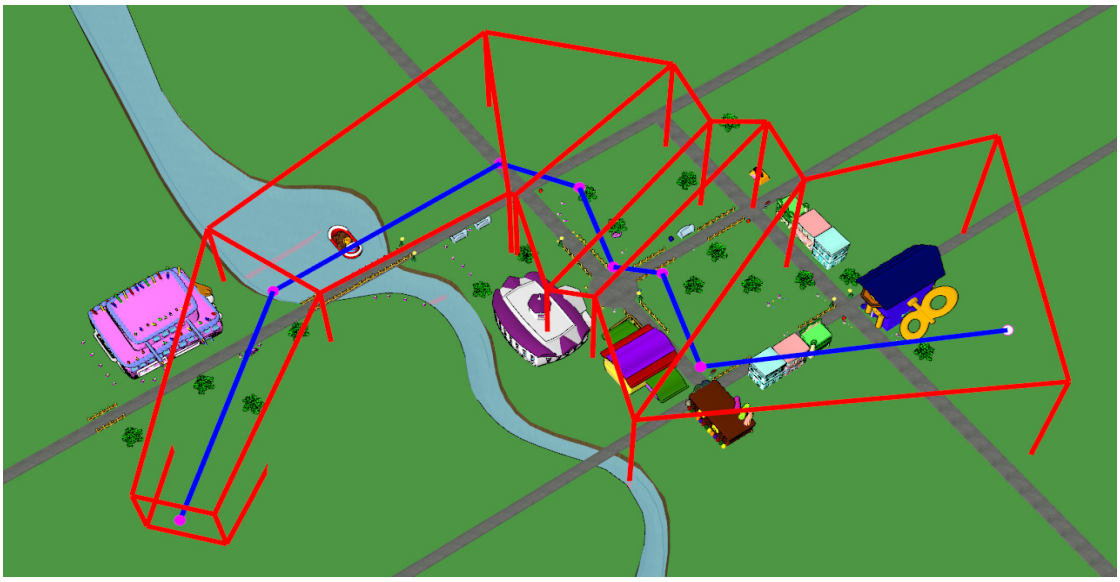


Figure 4.23. Graph camera model visualization for multiple bends. The camera was used to render Figure 4.2.

We have also developed an interactive CRC constructor that allows the user to set all components of a two viewpoint CRC through a graphical user interface. The position of the second viewpoint C_1 and the transition region controls how much and which way the CRC rays should bend, and the size of the transition region controls how abrupt the transition between the two viewpoints should be. The CRC components are manipulated and visualized in an overhead view of the scene while the user is also shown the corresponding CRC image for immediate feedback. The image in Figure 4.24 was rendered with a CRC that was designed with the interactive constructor to capture the entire forward and right street branches.

4.3.2.2. Recursive Maze-Based Construction

Another method for building powerful 3-D scene summarization graph cameras is to use a maze as scaffolding that connects important scene regions, similar to the maze used in scene exploration. Instead of stopping graph camera construction at the first intersection, the construction algorithm proceeds

recursively until the graph camera covers the entire maze. Many maze traversal algorithms are possible, we used breadth first search. In addition to the RFL (right, front, and left) type of intersection described earlier, the algorithm similarly handles simpler R, L, RF, FL, and RL intersections. The resulting graph camera image shows all scene regions traversed by the maze. The graph camera in Figure 4.25 has its root frustum at *O* and samples the cake conveyor belt to the left (*a*), outside the bakery through the door (*b*), the adjacent room through the tunnel (*c*), and then the door once again (*d*).

Figure 4.24. Curved ray camera image that captures all of the forward and right streets.

4.3.3. Real-World Scene Visualization

Creating a physical graph camera brings the benefits of comprehensive single-image visualization to real-world scenes. Each PPC frustum is implemented with a conventional video camera. Several problems need to be overcome.

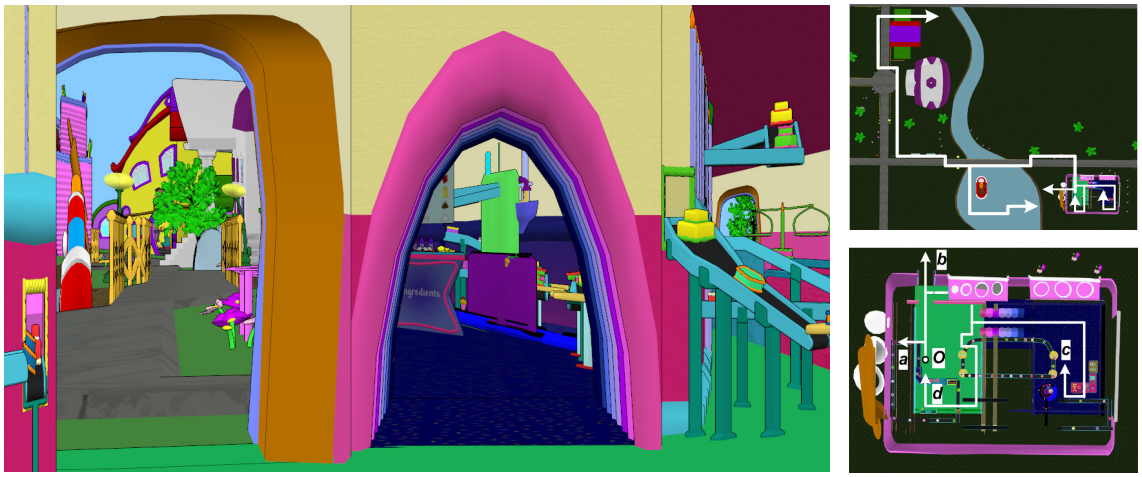


Figure 4.25. Maze-based summarization graph camera. An example summarization image is rendered with a graph camera constructed using the recursive maze-based algorithm (*left*). The camera model visualizations (*right*) show the shape of the camera overall and inside the bakery.

First, the graph camera rendering algorithm needs scene geometry. Providing high-resolution per-pixel depth for the video streams is a challenging problem which does not have a robust real-time solution yet. Fortunately the rendering algorithm can be amended such that video streams can be combined into a graph camera without the need of per-pixel depth. Given a frustum PPC_i, the first projection of the sequence of projections to the root frustum PPC_0 is implemented by the physical video camera. This first projection "flattens" the 3-D scene to a 2-D image which is projected successively by the subsequent frusta. Consequently, rendering with a frustum PPC_i of a physical graph camera is done without needing to know scene geometry by simply rendering a texture mapped quad using frustum PPC_{i-1}. The quad corresponds to the near face of frustum PPC_i and the texture corresponds to the video frame acquired by the video camera of PPC_i.

The second problem is extrinsic calibration of the cameras, which we achieve by registration to a coarse geometric scene model (i.e. a proxy). For the scene in Figure 4.4 we used a simple box model of the hallways. Although the video

cameras could be registered without a proxy using only 2-D correspondences, the proxy makes the registration robust and globally consistent.

The third problem is finding a placement of the PPCs that is physically realizable. A narrow field-of-view is also desirable for the PPCs of a physical graph camera like the one used in Figure 4.4 in order to control the amount of perspective foreshortening along the corridors. A large field-of-view wastes pixels on nearby side walls to the detriment of regions farther along the corridors. Camera *A* was placed as far back as possible, against the opposite wall, yet its field-of-view is still slightly larger than desired, exaggerating the importance of the side walls of the central hallway. A possible solution to this problem would be to push the camera back in software, which requires scene geometry. The proxy would be adequate for the corridor walls but estimating the depth for a subject in the corridor is more challenging.

The fourth problem is implementing the clipping planes which separate the PPC frusta. We simulate near clipping planes using background subtraction, as illustrated in Figure 4.26. The subject is located in the left branch of the corridor so the correct graph camera image should not show the subject in the right branch. The video camera monitoring the right branch does not have the correct near clipping plane so it samples the back of the subject. We simulate the correct image by erasing the subject using pre-acquired background pixels (Figure 4.4). Since the pixels erased are not live, some applications could prefer that they be highlighted in red to mark the uncertainty (Figure 4.26, bottom). Deciding from which video feed to erase the subject is done based on the subject's location, which is inferred by triangulation between two or more cameras. A non-leaf video camera also needs far clipping planes, which are implemented by background subtraction, similar to near clipping planes.

The fifth and final challenge specific to the physical implementation of graph cameras is the visualization of an object moving between connected frusta. A

correct graph camera image requires intersecting the moving object with the plane separating the frusta, which in turn requires per pixel depth. We use the approximate solution of fading the object out of the video frame of the old frustum and into the video frame of the new frustum as the object traverses the separation plane.

Figure 4.26. Illustration of near clip plane by background subtraction. The back of the subject is sampled by the camera that is supposed to sample the right branch of the corridor (*top*). The back of the subject is erased using a background image of the corridor acquired before the subject appeared (*bottom*).

4.3.4. Geometry Approximations for High-Quality Rendering Effects

Section 3.5.2 introduced the idea of replacing PPC depth images with occlusion camera depth images in order to obtain higher quality rendering effects such as reflections, refractions, and relief texture mapping at little additional rendering cost. In certain cases, replacing PPC depth images with graph camera depth images, similar results can be achieved. Using the graph camera, we have applied this approach to reflections, refractions, and ambient occlusion. A graph camera geometric approximation can be used as an environment impostor for rendering reflections and refractions since the graph camera image captures more of the environment than a traditional environment map and therefore remains valid for many frames. A graph camera image can also be used as geometric approximation for image space ambient occlusion where it produces more accurate results than a planar pinhole camera approximation.

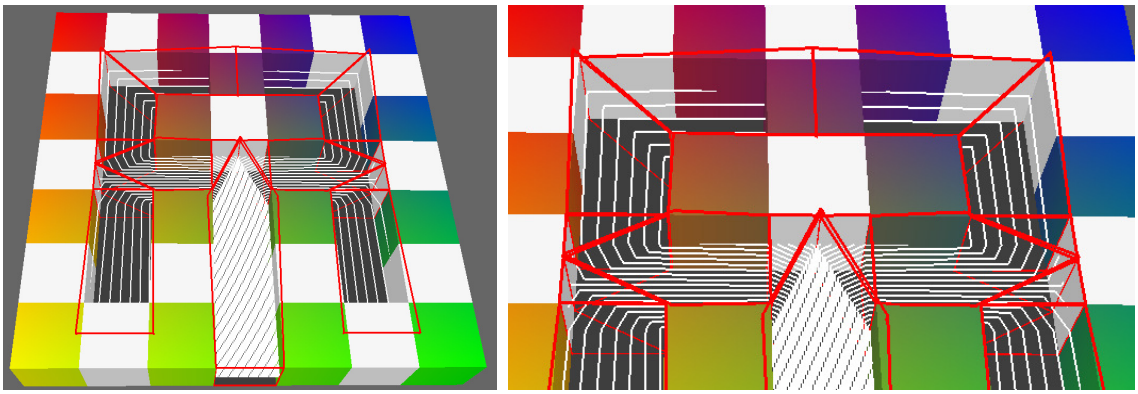


Figure 4.27. Graph camera model visualization. The frusta are shown in red and a few rays are shown in white.

4.3.4.1. Graph Camera Depth Images

The graph camera is constructed from a planar pinhole camera through a series of bending, splitting, and merging operations. The result is a graph of planar pinhole camera frusta. 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 rays of the graph camera for the maze in Figure 4.28 are shown in Figure 4.27. The construction used a breadth first traversal starting from the entrance.

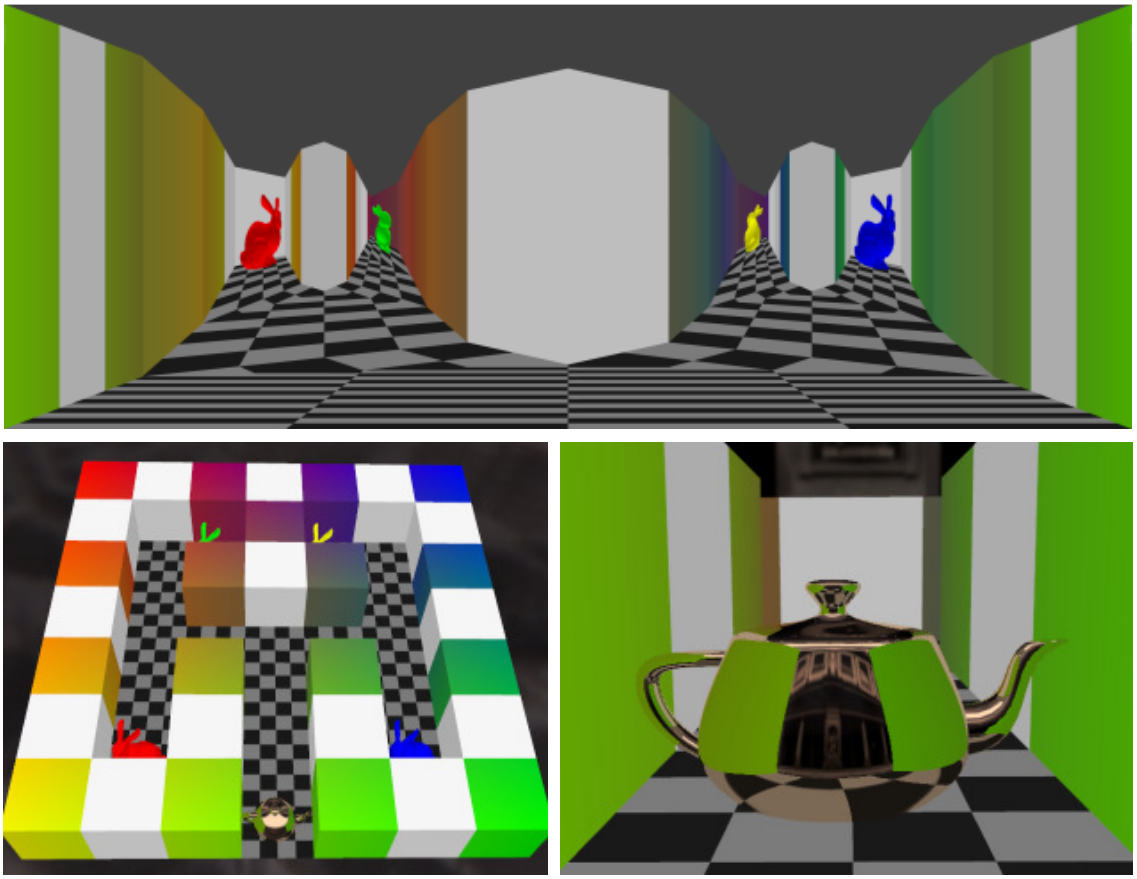


Figure 4.28. Graph camera environment map used for reflections. The graph camera depth image (*top*) captures an entire 3-D maze (*bottom, left*) and a reflection is rendered using it (*bottom, right*).

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.

4.3.4.2. 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 (Section 3.5.2.2). However, since each graph camera frustum is a pinhole, the projection of a given ray is piecewise linear (Figure 4.29), 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 which requires small 3-D steps just to model the breaking points of the piecewise linear ray 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, in a similar manner to the generic algorithm.

4.3.4.3. Reflections and Refractions

Rendering reflections and refractions with a graph camera geometric approximation is virtually identical to the process described in Section 3.5.2.3.1. The graph camera can encode a complex environment into a single graph camera depth image producing more accurate reflections and refractions than those possible with a planar pinhole camera depth image (Figure 4.28).

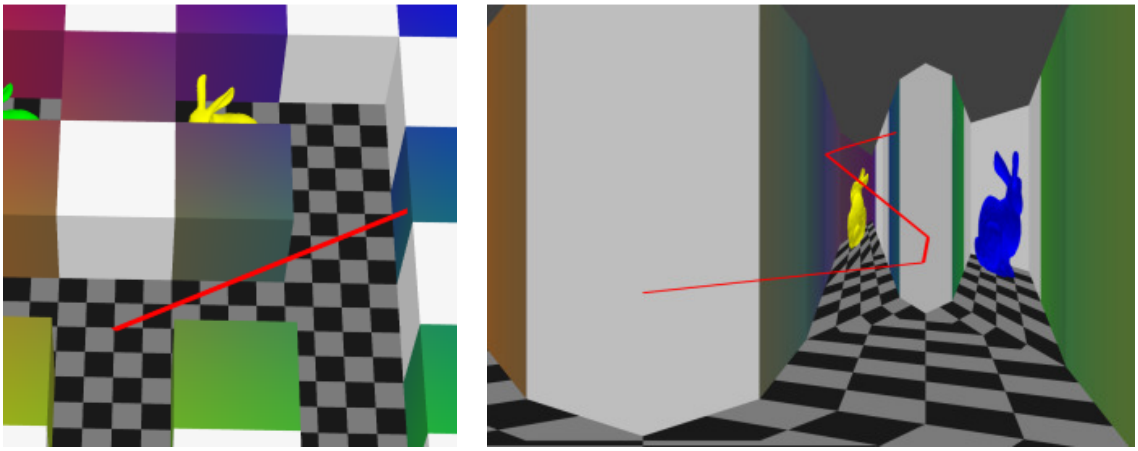


Figure 4.29. Visualization of a 3-D ray and its projection onto a graph camera image. The 3-D ray (*left*) produces a piecewise linear projection on the graph camera image (*right*).

4.3.4.4. Ambient Occlusion

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 by nearby geometry. The computational cost is high since a ray has to be cast from each point in all directions. Early implementations precomputed ambient occlusion in model space off-line [176], which precludes dynamic scenes. Initial efforts of enrolling the GPU in accelerating ambient occlusion resorted to a large number (i.e. 128-1,024) of spherical shadow maps of the scene [120]. Our graph camera depth buffer ambient occlusion method builds upon an image-space technique introduced by Bavoil et al [14]. Their technique approximates the amount of ambient occlusion at an output pixel using the output image depth buffer. They noticed that in order to sample occlusion at a pixel for an entire half plane it is sufficient to traverse one depth 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 contains all of the geometry in the

scene, which it does not, causing missing and unstable ambient occlusion artifacts.

The horizon-based screen-space ambient occlusion algorithm [14] is fast because one can estimate occlusion in a half plane by traversing a single output image depth buffer segment. In Figure 4.30 left, *p* is an arbitrary output image pixel. Occlusion on the *q* side of *ep* is estimated by simply traversing *pq*, and casting of the rays r_i is *not* needed. This is enabled by the property that at any output pixel *p,* any plane through the pixel ray *ep* will project to a line. This property needs to be maintained when porting the algorithm to non-pinhole depth buffers. We have designed a graph camera model that enhances the output image depth buffer with samples visible from two nearby viewpoints and also exhibits the desired property. 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 and the property is maintained. Using Figure 4.30 left again, *pq* remains a line after subsequent projections.

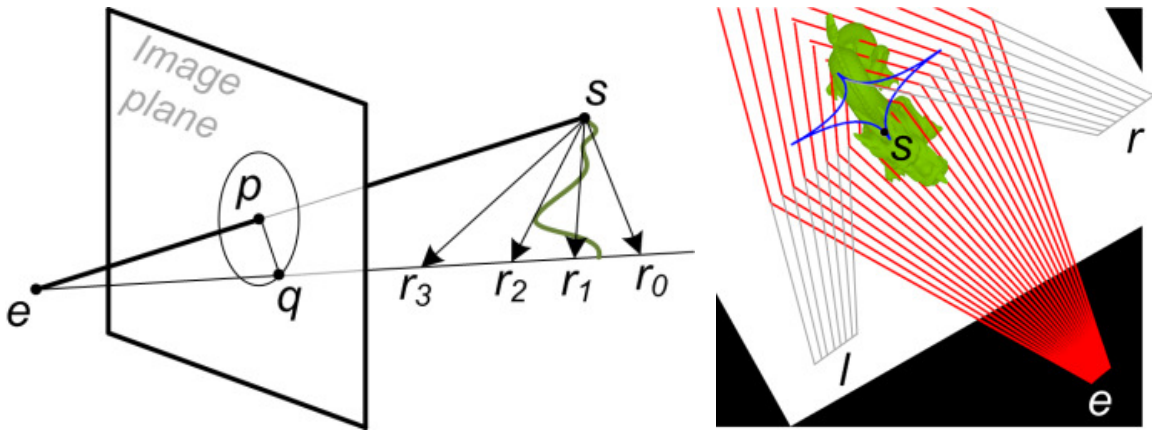


Figure 4.30. Screen-space ambient occlusion. Shown are the algorithm (*left*) and visualization of graph camera model (*right*) used in Figure 4.3.

Figure 4.30 right visualizes the rays of the graph camera used to render the non-pinhole depth buffer shown in Figure 4.3. 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. The splitting point moves on the blue curve as the output image view revolves around the dragon. 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 depth buffer suffices.

4.4. Discussion

The problem of occlusions can be solved by eliminating the occluder (i.e. cutaway techniques), by seeing through the occluder (i.e. transparency techniques), or by seeing around the occluder (i.e. multiperspective techniques). Even with the small-scale distortions of the CRC, the graph camera family does perturb global spatial relationships, which can lead to confusion. This limitation is inherent to all dataset distortion and multiperspective visualization techniques. Therefore the graph camera is not suited for applications where images are supposed to mimic the images the user would actually see in the corresponding real-world 3-D scene, such as virtual reality, CAD, or interior design.

Similar to the occlusion cameras, the graph camera family also needs a way, albeit indirect, of accessing the occluded data subset through which to route its rays. They cannot, for example, disocclude the engine of a car with the hood closed. One option is to combine the CRC with other disocclusion management techniques.

4.4.1. The Graph Camera

Performance: Graph camera images are rendered efficiently in feed-forward fashion leveraging a fast projection operation. Performance was measured on a

Quad Core Xeon 3.16 GHz with 4 GB of RAM and an nVidia 280 GTX graphics card, for 1,280x720 output resolution (Table 4.1). The frame rate remains interactive even for the graph camera with 35 frusta. For the real world graph cameras the dominant computational task is background subtraction, which is implemented using CUDA 1.2.

Limitations: The current maze-based constructor assumes straight maze edges and right angle 2-way, 3-way, or 4-way intersections. As discussed, limitations specific to real-world graph cameras relate to video camera placement, clip planes, and to objects crossing between frusta. Also the real-world graph camera depends on the robustness of the background subtraction algorithm. Each application should choose settings that produce an appropriate balance between false positives and false negatives. For example, a false negative at the far plane incorrectly hides an object whereas a false negative at the near plane incorrectly shows an object from another frustum.

Table 4.1. Graph camera rendering frame rates.

Scene	Triangles	Constructor	Illustration	Frusta	FPS
House	958K	Portal	Figure 4.14	1-6	27
House	958K	Interactive	Figure 4.32	13	6.7
Cartoon town	971K	Occluder	Figure 4.15	5	31
Cartoon town	971K	Interactive	Figure 4.2	7	25
Cartoon town	971K	Maze	Figure 4.25	35	5.2
City	4,123K	Maze	Figure 4.1	12	12
Real world 1	N/A	Proxy	Figure 4.4	3	9.0
Real world 2	N/A	Proxy	Figure 4.31	4	9.5

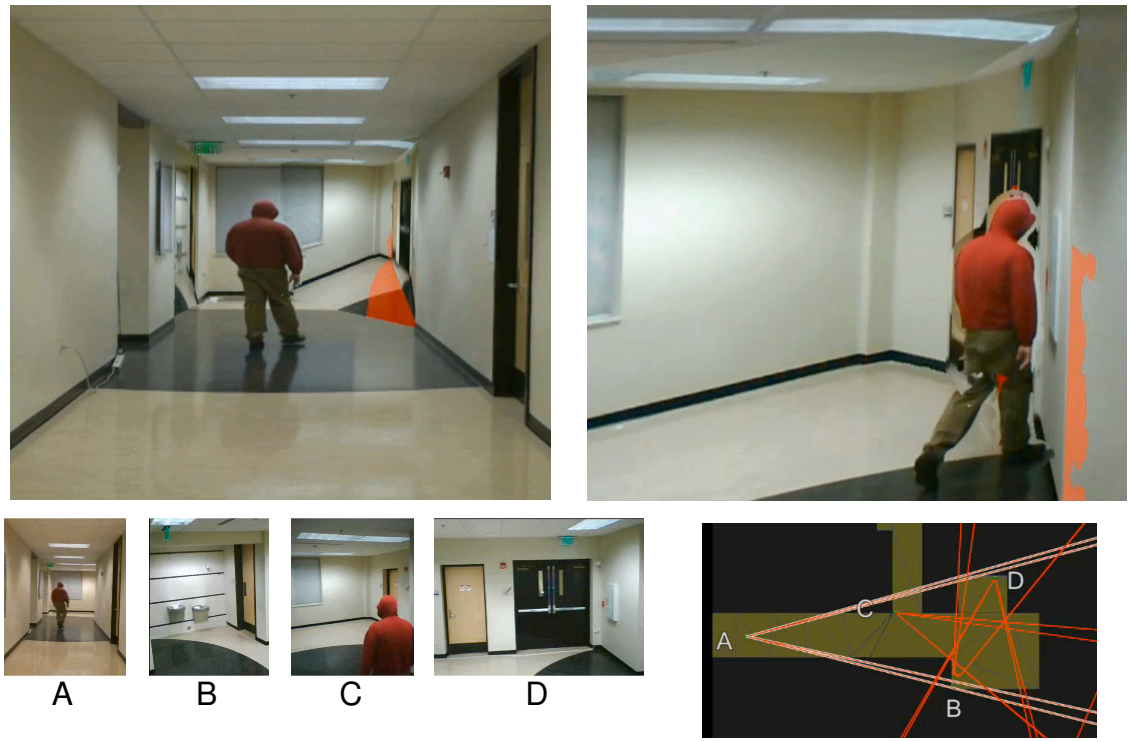


Figure 4.31. A real-world graph camera zooming example. The video feeds provide sufficient resolution to zooming in on regions of interest.

4.4.2. The Curved Ray Camera

We have tested CRC visualization on 5 datasets: a city model (Figure 4.7, Figure 4.8, and Figure 4.24), a ball and stick DNA molecule model [159] (Figure 4.9, middle row), the Engine Block volume dataset [41] (Figure 4.9, top row), a set of random blocks (Figure 4.16), and a canyon terrain dataset corresponding to a section of the Colorado river [157] (Figure 4.9, bottom row, and Figure 4.21).

Quality: The CRC succeeds at alleviating occlusions in complex datasets while minimizing distortions. Objects that traverse the region where rays transition between one viewpoint and the next deform only very little, resulting in a visualization effect comparable to that of a rigid body transformation. The GUI-based constructor allows the user to design the disocclusion effect interactively. The user can explore distant, occluded data subsets without modifying the

visualization context of nearby data. The target tracking constructor keeps the target visible as the target, the view, and/or occluders translate, as long as a solution exists given the maximum ray curvature allowed by the user. The path previewing constructor allows the user to visualize a pre-recorded path without being limited by the occlusions brought by the next few turns in the path.

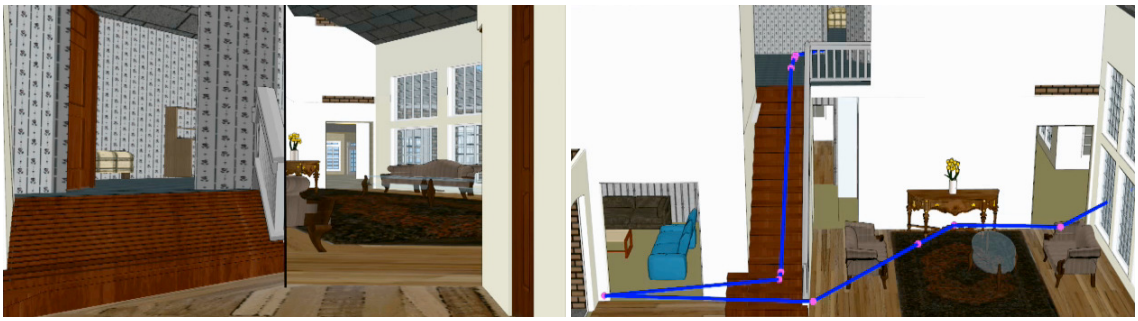


Figure 4.32. A graph camera summarization which compresses the stairs to show two levels of the house.

Performance: All performance numbers reported were measured on a Dual Xeon 3.2 GHz Intel processor with 4 GB of RAM and an nVidia GTX 285 graphics card. The interface to the graphics hardware was implemented using OpenGL and the Cg shading language. The output resolution was 1,280x720.

All datasets except for the Engine Block are 3-D surface datasets and are rendered by projection followed by rasterization. CRC projection is implemented in a vertex program. The program first determines the CRC sub-frustum that contains the vertex. If the vertex is inside a conventional PPC sub-frustum, the vertex is projected directly to the output image by multiplication with a 4-D matrix pre-computed by concatenating the modelview and projection matrices of all the viewpoints from the current viewpoint to the root. If the vertex is inside a transition region sub-frustum, the vertex is projected. For datasets where the triangles are small, conventional rasterization is a good approximation of the non-linear rasterization mandated by the non-linear projection of the CRC. Datasets with large triangles are subdivided as a pre-process. In our case the only dataset

that required subdivision was the city dataset due to large triangles used to model the ground (e.g. sidewalks, roads). We have also implemented a geometry program that performs on demand subdivision but performance is lower than in the case of off-line uniform subdivision, which we attribute to the primitive emission bottleneck of GPU geometry programs.

Table 4.2. Curved ray camera rendering performance for datasets.

Dataset	Vertices x1,000	Triangles x1,000	Frame rate (Hz)		
			PPC	Graph Camera	CRC
Blocks	90	43	153	136	137
DNA	2,170	4,112	48	32	23
Canyon	4,287	2,168	99	56	52
City	29,892	10,351	2.8	2.8	2.8
Engine	256x256x110 vol. res.		7.9	6.7	4.1

The side faces of the CRC frusta are planes. For the Canyon and City datasets we perform view frustum culling at primitive group level using a simple uniform 3-D grid. We avoid triangle level clipping by enlarging the frustum of the CRC and discarding any triangle for which at least one of the vertices is outside the CRC frustum. This is done with a simple geometry program that emits 1 or 0 triangles per incoming triangle. Triangles do not need to be clipped with the separation planes between CRC sub-frusta; a triangle crossing such a plane is simply handled by projecting each one of its vertices with the projection function of its sub-frustum.

Rendering performance is given in Table 4.2. The geometry load figures correspond to primitives that pass view frustum culling. We compare the CRC to a PPC (i.e. rendering the dataset with the root PPC_0 of the CRC), and to a camera with C^0 continuous rays that switches abruptly between the viewpoints of

the CRC. The CRC renders all datasets comfortably except for the City dataset which is too large to fit on the graphics card, and which none of the 3 methods render quickly. The ratio between PPC and CRC performance is at most 2.08; between the graph camera and CRC it is at most 1.39. The CRC vertex projection performance is at least 12, 50, 222, and 83 million vertices per second for the first 4 datasets in the table, respectively. These figures were computed by multiplying the frame rate by the number of vertices, which counts the entire frame time as projection time. As such, the figures give a lower bound on projection performance.

We measured performance for the Canyon dataset for CRCs with an increasing number of viewpoints. Performance remains virtually unchanged even for as many as 8 viewpoints, which is expected since the few additional dot products needed to classify the vertex sub-frusta are a small cost compared to the vertex projection operation as a whole.

The Engine Block dataset is volume rendered by ray casting. For the transition region we derive the number of steps from the sum of the length of the original viewpoint C_0 and C_1 rays (i.e. $P_0P_1 + P_1P_2$ in Figure 4.10). The numbers reported in the table correspond to the left Block Engine image in Figure 4.9 for the PPC and to the right image for the CRC. The difference in performance is due to the fact that the volume covers a considerably larger fraction of the image for the CRC, and that the CRC rays travel longer through the volume.

4.4.3. Geometry Approximations for High-Quality Rendering Effects

Quality: Graph camera depth images enable quality reflections, refractions, and ambient occlusion. Like for all sample-based methods, the quality of the results obtained with graph camera depth images is contingent upon adequate sampling. The graph camera sampling rate is not uniform: it is higher closer to the initial frustum and is lower for 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 4.28). 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 4.33 the teapot is in the top left corner of the maze (see Figure 4.28), 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 1,920x1,175. 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.

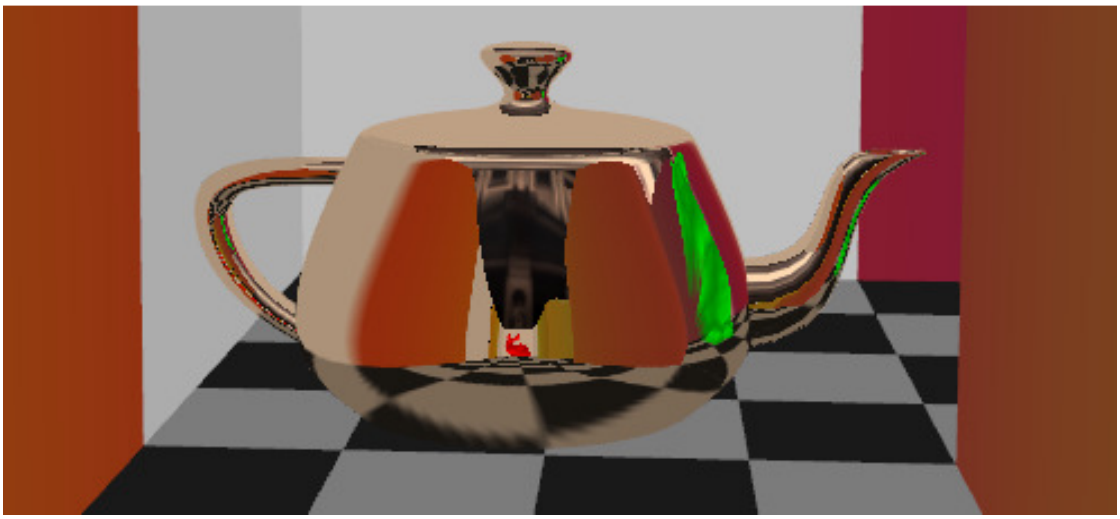


Figure 4.33. Undersampling artifacts on the floor reflection.

Performance: The timing information reported here was collected on a 3.4 GHz Intel Xeon PC with 2 GB of RAM and an nVidia 8800 Ultra graphics 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. Coarse stepping reduces the number of steps for the graph camera depth image

as can be seen in Figure 4.34, 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 image is constructed at over 100 fps. The average, minimum, and maximum frame rates for the path that follows the teapot through the maze are 45.5, 30, and 105 fps without antialiasing, and 26.8, 20, and 42 fps with 8x MSAA, respectively.

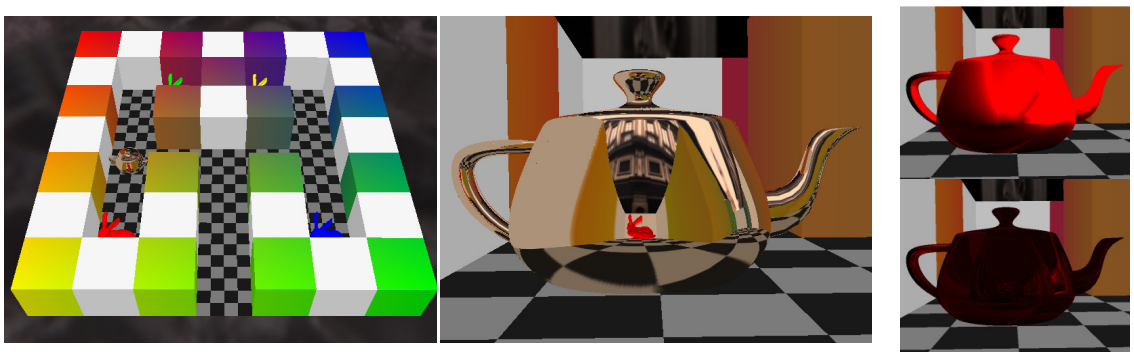


Figure 4.34. Graph camera image used to calculate a reflection. Teapot location (*left*) and reflection (*middle*) alongside a visualization (*right*) of the number of steps with (*bottom*) and without (*top*) coarse stepping. Brighter red indicates a large number of steps being taken.

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 $1{,}024^2$ in both cases. Average performance for the dragon scene (Figure 4.3) is 35 and 16 fps for the regular and fine settings, respectively. Figure 4.3 uses the fine sampling setting—the regular setting produces a noisier ambient occlusion, an issue orthogonal to the use of the graph camera depth buffer. Rendering the locally illuminated output image, the graph camera depth buffer, and adding the ambient occlusion effect takes 5.9, 16, and 6 ms for the regular setting and 5.9, 16, and 40 ms 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 5.9 ms for the regular setting and 5.9 and 32 ms for the fine setting, respectively. Sampling occlusions in the graph camera depth buffer as opposed to the conventional depth buffer brings a 25% penalty, and most performance loss is brought by having to render the graph camera depth buffer of the 890K triangles scene. However, in some scenes it is unnecessary to update the graph camera depth buffer for every frame. The graph camera depth buffer does see more than what is visible in the output image which greatly extends its resiliency to output view changes. For example the depth buffer shown in Figure 4.3 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 3-D model when computing the graph camera depth buffer.

CHAPTER 5. GENERAL PINHOLE CAMERA

One of the significant limitations of the planar pinhole camera not yet addressed is the uniform sampling of the image plane. The general pinhole camera (GPC) addresses this problem by providing a flexible sampling rate. Of course, one could chose to resample a conventional PPC image to any desired set of sampling locations, but such an approach is only approximate since it computes the desired samples by interpolation and not by actually sampling the data to be visualized. The approximation is particularly poor for large sampling rate variations when an adequate approximation by interpolation requires a prohibitively high resolution for the input image.

We present the general pinhole camera (GPC) model that supports any set of sampling locations on the image plane. The GPC rays are defined by a pinhole and the desired image plane sampling locations. The GPC image is rendered by directly sampling the data to be visualized at the desired sampling locations. GPC visualization is versatile—it supports many types of data, including surface geometry, volume, and image data. GPC visualization is also efficient—complex datasets are rendered interactively with the help of graphics hardware. Moreover, if the application demands it, a GPC image can be resampled at little cost into a conventional PPC image. We demonstrate the advantages of the non-uniform sampling afforded by the GPC in three contexts: remote visualization, focus-plus-context visualization, and antialiasing.

In order to visualize a dataset at a site other than the site where it was computed or acquired, one approach is to transfer the data. However, data transfers become more and more challenging as data size increases continue to outpace

networking bandwidth increases. Moreover, replicating visualization capabilities at all user sites also scales poorly. A second approach overcomes these disadvantages by computing the desired visualization image at the remote site, followed by transferring the image to the local user site. No dataset transfer or replication of visualization capabilities is required. However, such a remote visualization approach suffers from reduced interactivity. Even though the bandwidth requirement for transferring an image is greatly reduced compared to transferring a large dataset, the image has to be transferred in real time and bandwidth remains a bottleneck, affecting the frame rate. The solutions of reducing image resolution or aggressive compression are only palliative.

We propose to improve interactivity in remote visualization by transferring GPC images instead of conventional PPC images. The idea is to reuse a GPC image at the local site to compute several high-quality output frames, without the need of any additional data from the remote site. The GPC is designed to produce images with a sampling rate higher at the center and lower at the periphery. The resulting image has size and coherence similar to those of a PPC image, thus the transfer costs are comparable. At the local site the GPC image is resampled into a conventional PPC output frame at interactive rates. The higher sampling rate at the GPC image center allows the user to zoom in with little quality loss. Once the output frame sampling rate exceeds that provided by the GPC image, a new GPC image is requested and transferred from the remote site. The GPC also has a larger field-of-view than the output frame, which allows the user to rotate the view direction without the need of transferring new images. Like the PPC, the GPC is a pinhole so a GPC reference image cannot accurately support viewpoint translation due to occlusions. However, the approximate translation support provided is sufficient to allow the user to select a novel viewpoint for which a new GPC image is rendered and transferred.

In Figure 5.1 the PPC and GPC reference images have the same size (800x480) and field-of-view (90^{o}). The output frames are 600x360 in size and have a

smaller, variable field-of-view. The GPC allows zooming in with good results. Frames are computed from a GPC at the rate of 525 frames per second.

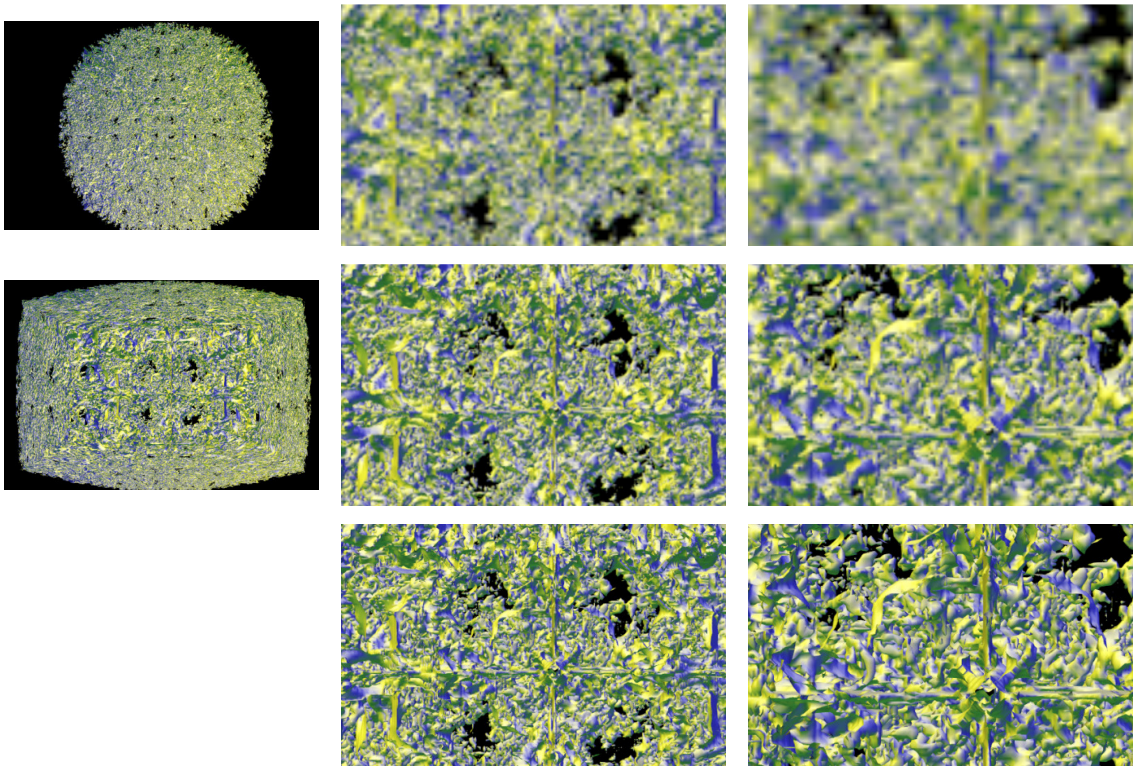


Figure 5.1. General pinhole camera used for remote visualization. *Top:* Conventional PPC reference image and two output frames resampled from it. *Middle:* The GPC reference image and corresponding frames. *Bottom: S*ame frames rendered from actual geometry data for comparison. The GPC image has a higher sampling rate at the center which produces higher quality frames when zooming in.

The second context in which the GPC infrastructure pays dividends is focus-plus-context visualization. Many methods have been developed in visualization and computer-human interaction that capitalize on the perceptual advantages of visualizing a region of interest in detail (i.e. focus) while keeping it seamlessly integrated into the surrounding area (i.e. context). The GPC naturally supports higher sampling rates for one or several focus regions. We describe a GPC

variant suitable for focus-plus-context visualization with advantages that include sampling *rate* continuity from focus region to surrounding context, distortion free focus regions, efficiency, and support for many types of data. In Figure 5.2 the model has 1.3 million triangles, the GPC has 2 focus regions with cubic distortion, and the GPC image is rendered at 7 Hz, which allows the user to modify focus region parameters interactively.

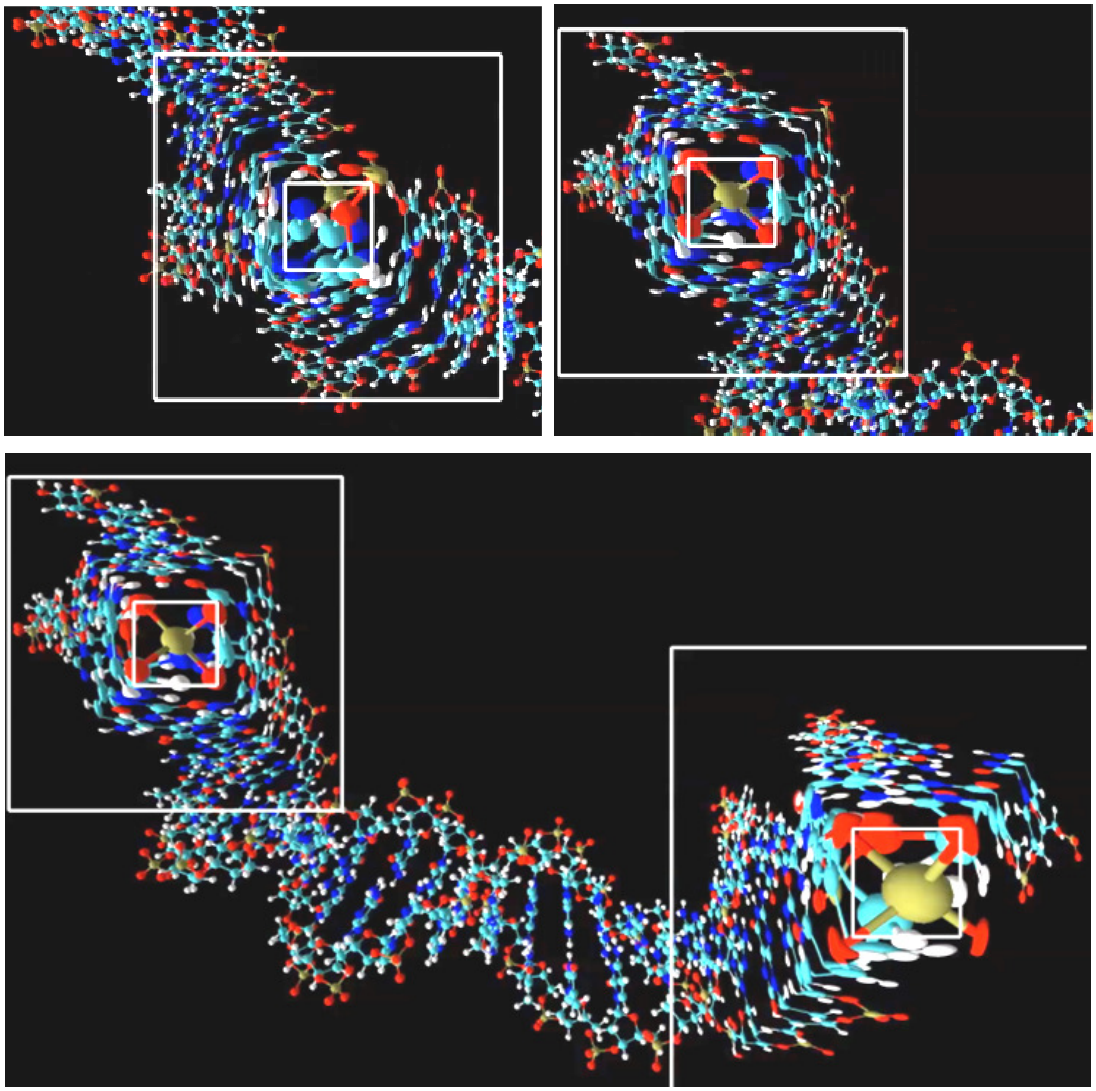


Figure 5.2. General pinhole camera focus-plus-context visualization of a DNA molecule. The bottom GPC image has two focus regions with 3x and 6x magnification factors, respectively.

The GPC proves to be a powerful tool in a third context—antialiasing. As the acuity of measuring instruments and the sophistication of simulations increase, visualization will face the increasing challenge of alleviating mismatches between output image resolution and dataset resolution. Antialiasing techniques can prevent the disturbing visual artifacts that occur when the dataset resolution exceeds the output image resolution. Antialiasing for image data is a solved problem. Antialiasing for 3-D data is more challenging, requiring the computation of multiple color samples for each output pixel. When the resolution mismatch is severe, the required supersampling factor is large, making conventional full-frame antialiasing prohibitively expensive. A possible approach is to address the resolution mismatch offline by precomputing lower levels of detail (LoDs), which is challenging and delays the visualization of real-time datasets.

Figure 5.3. General pinhole camera used for extreme antialiasing. GPC-based 361x extreme antialiasing (*left*) and conventional hardware supported 16x antialiasing (*right*). Images are magnified at 600% for illustration purposes.

We describe a GPC variant that capitalizes on the fact that extreme supersampling levels might only be needed in small regions of the output image. The GPC allows supersampling locally at very high levels without substantially increasing the overall memory or rendering cost. In Figure 5.3, left, a GPC renders the 32^2 pixel screen area covered by the tree with 361 color samples per pixel using an off-screen framebuffer tile which produces a high quality output image, bypassing the need for challenging LoD computation. Note that the tree

model has only 9K triangles so rendering load is not a factor. With conventional antialiasing serious minification artifacts occur even for a high 16x setting.

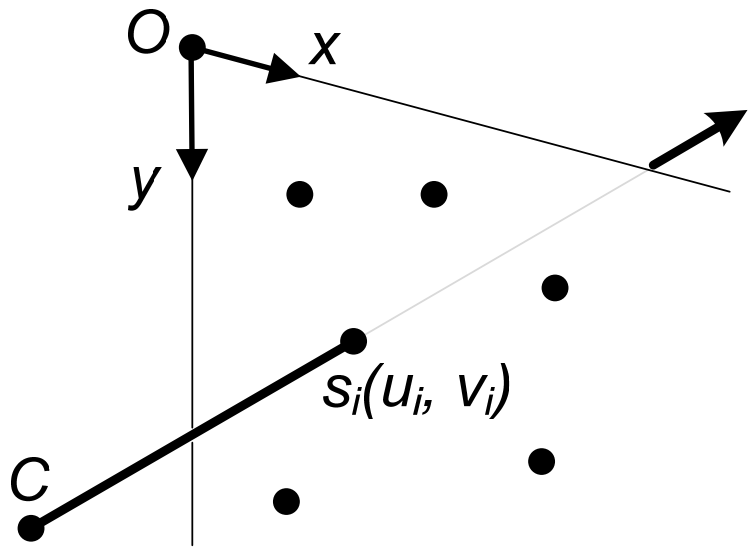


Figure 5.4. The generic general pinhole camera model.

5.1. Camera Model

The general pinhole camera model is defined by a center-of-projection *C* (Figure 5.4), an image plane specified by a coordinate system with origin *O* and axes *(x, y)*, and a set *S* of *N* sampling locations $s_i(u_i, v_i)$, where u_i and v_i are the image plane coordinates of sample s_i. A GPC ray is defined by the ordered pair $(C, O+xu_i+yv_i)$. This generic model is tailored to an application in three steps:

- *Sampling location selection*: The actual image plane sampling locations are specified based on the application's goal.
- *Rendering*: Rendering algorithms are devised based on the sampling location pattern. The considerations are efficiency and support for a broad range of data types.
- *Display*: A mapping between the uniform pixels of the display and the non-uniform GPC sampling locations is specified.

5.2. General Pinhole Camera Applications

5.2.1. Remote Visualization

In conventional remote visualization the images received from the server are ephemeral—any view change requested by the user renders them obsolete and new images have to be transferred. It is our goal to design images which contain information sufficient for several visualization frames in order to reduce the frequency of image transfers and thereby improve interactivity.

5.2.1.1. Prior Work

The motivation for remote visualization [69] includes immediate access to remote datasets without off-line downloads; visualization of complex datasets without the need for local high-end storage and visualization capabilities; and visualization without full disclosure of the datasets [75].

One general remote visualization strategy is to reduce the visualization dataset on the server to a size that can be transmitted by the network and visualized by the client. The dataset reduction is performed using one or a combination of techniques, including multiresolution and level-of-detail [22, 92], progressive refinement [21, 61], feature extraction [57, 89], occlusion culling [49, 117], and data compression [64, 88]. Unique strengths of this strategy include the ability to leverage client visualization capabilities, and the ability to accumulate data at the client, which, over time, leads to reduced dependence on the network. Disadvantages include the reliance on visualization capabilities at the client, the need of good a priori estimates for the values of the parameters of the data reduction techniques employed, and the need for data type specific tools at both the server and client.

A second general remote visualization strategy is to relieve the client of all visualization duties. The client transmits the desired visualization view and parameters, the server renders, compresses, and sends the visualization frame, which is received, decompressed, and displayed by the client. The strategy is appealing because of its portability—it gives access to visualization to any client that can display an image (i.e. a terminal), and because of its generality—any type of data and any visualization algorithm is supported as long as it is supported by the server. Leveraging X Window System [109] transport infrastructure and VNC [134] application portal technology, generic remote visualization systems have been developed by academia [73, 78, 150] and industry (e.g. Vizserver [113]) to support thousands of heterogeneous clients. The main challenge is the network bandwidth bottleneck which limits frame rate and resolution. The bottleneck is alleviated by high performance image compression / decompression schemes which run in parallel [94], or with the help of graphics hardware [149].

The importance of optimal work-load partitioning between server and client was also recognized by Luke et al. who propose a remote visualization framework [93] that provides a third, hybrid, partitioning scheme: the client receives images enhanced with per-pixel depth which are rendered locally for higher frame rate and lower latency. Another example of such a hybrid strategy is the Visapult system [15], where the final image is composited in sort-last fashion on the client, but this time the motivation is to lower the computational burden at the server.

The GPC assisted remote visualization approach we propose falls in this hybrid category: the image received by the client is resampled to produce several output visualization frames. The GPC produces a 2-D image that can be compressed or composited with algorithms devised for conventional images, making it suitable for straight-forward integration within existing remote visualization frameworks.

5.2.1.2. Sampling Location Selection

The sampling locations are chosen such that the GPC image anticipates view changes requested by the user. View rotations are easily supported using a larger field-of-view and a higher resolution for the reference image than for the output frame. Another common view change in visualization applications is zooming. Zooming in by resampling a conventional PPC image leads to blurriness as the original sampling rate is exceeded. In order to alleviate the quality loss as the user zooms in, the GPC image needs to have a higher sampling rate at the center, anticipating the zoom in operation. Compared to a PPC image of same size, the higher density of samples at the center of the GPC has to come at the cost of a lower sample density at the periphery, which is reasonable since the user is likely to concentrate on the center of the image.

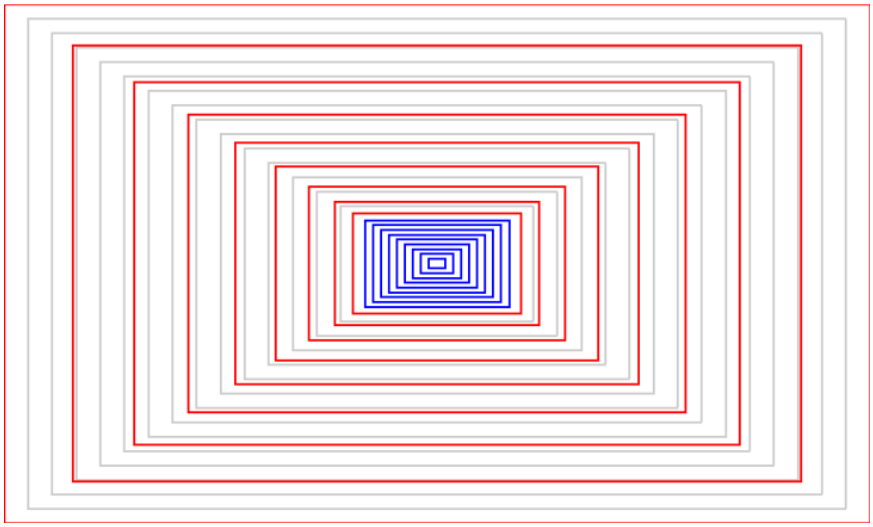


Figure 5.5. Sampling pattern used for the general pinhole camera remote visualization application.

We choose a sampling pattern with a constant and higher sampling rate at the center of the image (*blue* in Figure 5.5) and a lower sampling rate at the periphery (*red*). The samples are specified as a distortion of an original PPC image (*grey*). Denser samples produce a magnification and sparser samples produce a compression of the resulting GPC image (Figure 5.6). The distortion

is specified with 3 parameters *w, h*, and *zf* which define the size and zoom factor of the central region C. Larger *w, h,* and *zf* values allow zooming in longer with good quality at the cost of a lower quality at the periphery of the GPC image. Here *w*, *h,* and *zf* were chosen as *W/2*, *H/2*, and 3, where *W* and *H* are the dimensions of the GPC image.

It remains to specify the samples at the transition region. The constraints are continuity with the central region at the inner boundary and the image frame at the outer boundary. C^0 continuity at both boundaries ensures that all of the original field-of-view is sampled. C^1 continuity at the inner boundary ensures that the sampling *rate* is continuous from the transition to the central region. The simplest expression that satisfies these three conditions is a quadratic. Each of the four regions L, B, R, and T defined by the diagonals of the central rectangle has its own distortion expression. For L the horizontal distorted coordinate u_d is given by:

$$u_d(u) = a_0u^2 + a_1u + a_2 \qquad \text{Equation 5.1}$$

where *u* is the undistorted horizontal coordinate (Figure 5.7). The coefficients a_i are the same for the entire region and are computed by solving a linear system of three equations with three unknowns:

$$u_d(u_l) = u_l$$

$$u_d(u_r) = \frac{W}{2} + \left(u_r - \frac{W}{2}\right)/zf \qquad \text{Equation 5.2}$$

$$u'_d(u_r) = \frac{1}{zf}$$

The first two equations ensure sampling continuity at the outer and inner boundaries of the transition region. The third equation ensures sampling rate

continuity between the transition and central regions. Once u_d is known, the vertical distorted coordinate v_d is computed using the constraint that the sample be distorted on a line towards the center of the image. The sampling rate increases from the outer to the inner edge of the transition region ($u_d''(u)$ is a positive constant). The lowest sampling rate is $u_d'(u_l)$, which is 0.333 for this example, as can be verified in Figure 5.5 where the grey rectangles are three times as dense as the red rectangles at the periphery. The highest lower bound on the sampling rate is of course achieved when the sampling rate is maintained constant across the transition region. This would yield here a minimum sampling rate of 0.652, but abandoning the sampling rate continuity requirement causes visual artifacts.

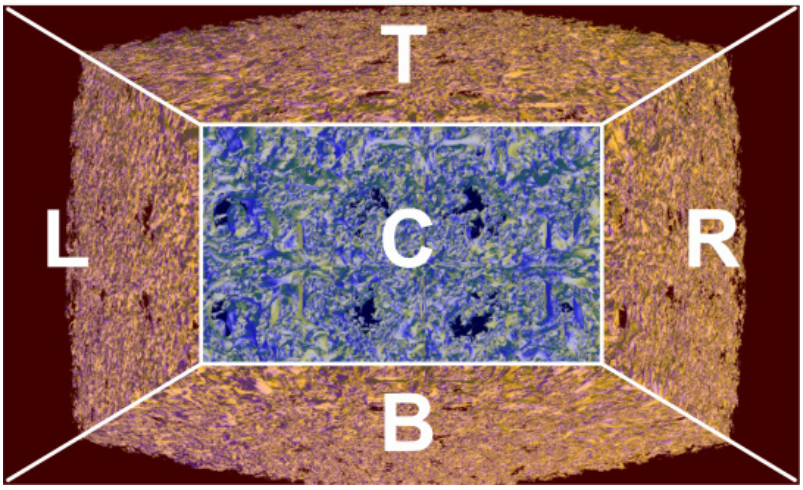


Figure 5.6. Correspondence between general pinhole camera samples and image.

In summary, a GPC with *W*x*H* sampling locations is defined by a PPC with resolution *W*x*H*, a central rectangular region with a magnification factor, and 4 quadratic distortion expressions, one for each of the sub-regions of the transition region.

5.2.1.3. Rendering

5.2.1.3.1. Ray casting

Each sampling location defines a GPC ray. Ray casting or ray tracing with the GPC is straightforward. Instead of following the rays of a conventional PPC, the rendering algorithm follows the GPC rays. Like Wang et al. [161], we leverage this fact to provide GPC volume rendering support by ray casting on the GPU (Figure 5.8). Similarly, surface geometry data can be rendered with a GPC using a ray tracer. As new graphics architectures become more programmable [140], ray tracing is likely to become a serious competitor to feed-forward rendering. For now, rendering by projection followed by rasterization remains the preferred approach in interactive rendering.

5.2.1.3.2. Feed-Forward Rendering

Feed-forward GPC rendering has to overcome two challenges: projection and rasterization. GPC projection is straightforward. A 3-D point *P* is first projected to coordinates (u_d, v_d) using the PPC associated with the GPC. Then GPC image coordinates (*u, v*) are computed by inverting the distortion. For the central rectangular region (C in Figure 5.6), the distortion is a simple linear scaling, thus inverting it implies solving a linear equation. For the transition region inverting the distortion implies solving a quadratic. Using Equation 1 again, for region L the quadratic equation that provides the GPC image coordinate *u* of *P* is:

$$a_0 u^2 + a_1 u + a_2 - u_d = 0 \qquad \text{Equation 5.3}$$

As before, the GPC image coordinate *v* of *P* is obtained using the fact that the line defined by (*u, v*) and (u_d, v_d) passes through the center of the rectangular region C.

The construction of the GPC causes rasterization to become non-linear. The challenges posed by non-linear rasterization affect all camera models described up until now. Details of how these challenges are overcome will be presented in Chapter 6.

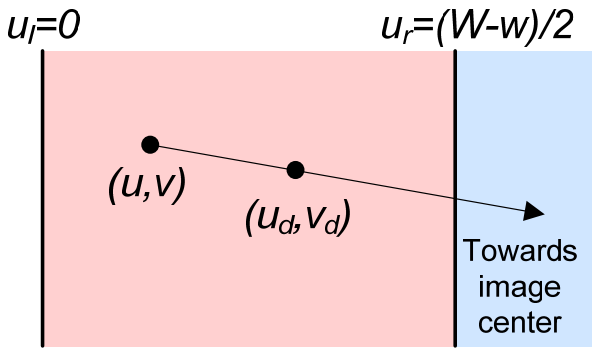


Figure 5.7. General pinhole camera distortion at the transition.

5.2.1.4. Display

Once the GPC image is rendered, it is ready to be compressed conventionally and to be sent to the client site. Once received, the client application uses the GPC image to reconstruct visualization frames for the user at interactive rates. Rotations and zoom changes are handled by resampling the GPC image to the PPC, modeling the view of the current frame. Each pixel *p* of the current frame is set in three steps; the corresponding 3-D point *P* on the PPC image plane is computed first, then *P* is projected onto the GPC image plane, and finally *p* is set to the GPC image color at the projection location. The resampling cost is small and bounded by the output frame resolution. Resampling the GPC image produces correct results because the center-of-projection of the output frame PPC coincides with the center-of-projection of the GPC. View dependent rendering effects are supported (e.g. transparency, reflections, and refractions).

Like any pinhole, the GPC does not support viewpoint translations. In order to allow the user to change the viewpoint we provide approximate translation

support in one of two ways. The less expensive but also the more approximate way is to assume all GPC samples lie in a vertical plane. There is no motion parallax within the 3-D dataset, but the approximation enables selecting the next viewpoint. The second way is based on 3-D image warping [102]—the GPC image is enhanced with per pixel depth, which is then used to reproject the GPC samples during viewpoint translation. 3-D warping comes at the cost of the additional channel for the GPC image. Moreover, 3-D warping reuses color as is, thus effects like reflections are not handled correctly, and when new surfaces are exposed disocclusion errors occur (Figure 5.9). 3-D warping does however provide correct motion parallax, a strong visual cue in 3-D data visualization. During translation a red frame border indicates that the visualization is only approximate. The user can request a new GPC image at any time.

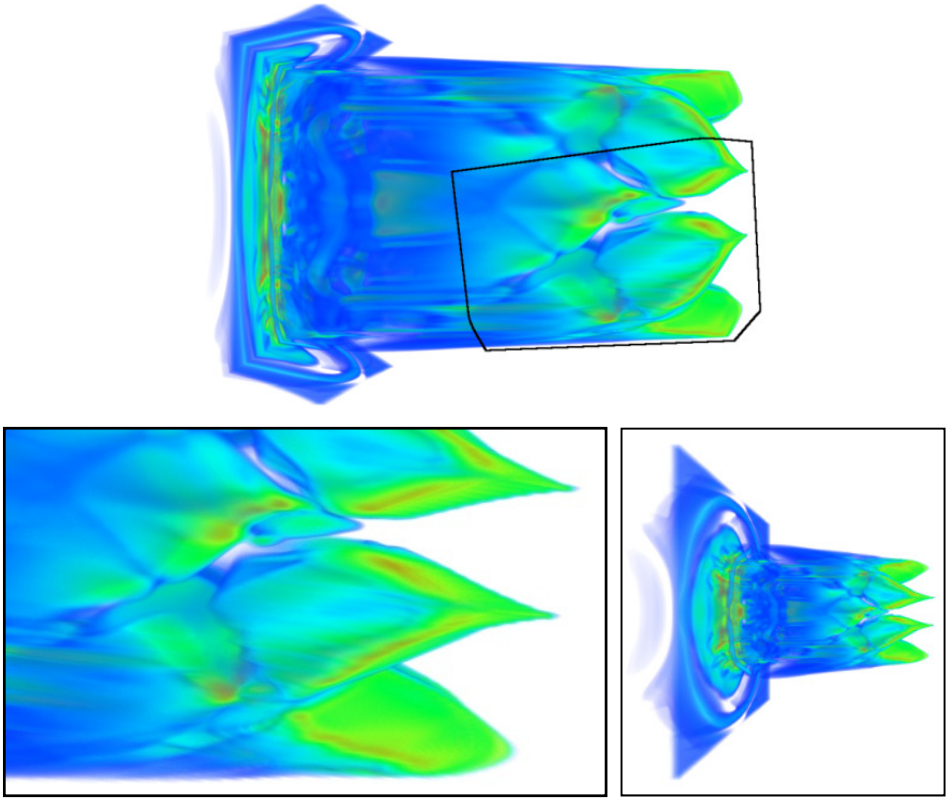


Figure 5.8. General pinhole camera volume rendering example. Volume rendering GPC image (*top*) and two frames resampled from it (*bottom*).

5.2.2. Focus-Plus-Context Visualization

5.2.2.1. Prior Work

Focus-plus-context visualization uses an inequitable screen real estate allocation favoring data subsets deemed more important, an idea introduced by the fisheye views [48]. The approach is supported by the way our visual system works, both at high level—we concentrate on a small part of what we see and rely on the background for general situational awareness, and at low level—the density of receptors on the retina is non-uniform. Focus-plus-context has been applied to 2-D image data, either acquired or rendered from 2-D primitives, including hierarchies [79], graphs [168], and maps [122]. While these techniques apply a 2-D magnification lens to emphasize the focus region, the Mélange system [38] resorts to mapping the 2-D image data to a 3-D surface designed to emphasize several focus regions, while compressing less interesting connecting context.

The magnification of the focus region implies an increased sampling rate. However, most computer displays are designed for a uniform sampling rate and mapping a focus-plus-context image to such a conventional display requires distortion. An alternative is to build displays with a variable pixel density that can display a focus-plus-context image directly [12]. The challenges are difficulty in changing pixel density, abrupt pixel density changes, and bulkiness.

Focus-plus-context techniques for 3-D surface geometry data typically apply a 3-D space distortion followed by conventional visualization [23]. The distortion has the potential to reveal subsets of interest otherwise occluded [37], but has the disadvantages of difficult distortion design and distortion of the focus regions. These difficulties are avoided by distorting the camera rays as opposed to the geometry data, as demonstrated by Wang et al. for volume data [161]. The context and focus regions are essentially rendered with conventional cameras with various resolutions while the distortion is confined to the transition area. Our

GPC focus-plus-context technique is similar to the Wang et al. approach in that the distortion is applied at camera model level. However, the GPC is a general method that supports many types of data, including surface geometry data and that offers great flexibility for designing the transition region, including for achieving sampling rate continuity. Compared to space distortion techniques the GPC is a pinhole thus it has no disocclusion capability, but the GPC avoids the disadvantages of difficult distortion design and distortion of the focus regions.

5.2.2.2. Sampling Location Selection

A GPC model similar to the one described for remote visualization can also serve for focus-plus-context visualization. The magnified central rectangular region is equivalent to a focus region. A focus-plus-context GPC is obtained by extending the remote visualization GPC model as follows:

- The focus region should not necessarily be centered, which adds two parameters u_0 and v_0 defining the rectangle center,
- The transition to context region should not necessarily extend all the way to the edge of the image, which adds another parameter *tw* to encode the width of the transition region,
- More than a single focus region should be allowed, however the focus regions will be kept disjoint and their number will be a small constant, which facilitates interactive rendering.

Since the focus region is now surrounded by context, it is of interest to maintain sampling rate continuity at the outer edge of the focus region. To accommodate this additional constraint a fourth coefficient is needed for the distortion equation (Equation 5.1):

$$u_d(u) = a_0u^3 + a_1u^2 + a_2u + a_3 \qquad \text{Equation 5.4}$$

The 4 coefficients are found with the following linear system:

$$u_d(u_l) = u_l$$

$$u_d(u_r) = u_0 + \frac{u_r - u_0}{zf}$$

Equation 5.5

$$u'_d = (u_r) - \frac{1}{zf}$$

$$u'_d(u_l) = 1$$

Figure 5.10 illustrates the sampling rate at the transition region: it starts out at 1.0 (same spacing between first two red lines and the grey lines), then decreases (minimum value here is 0.59), and then increases to match the sampling rate of the central region (here 3.0).

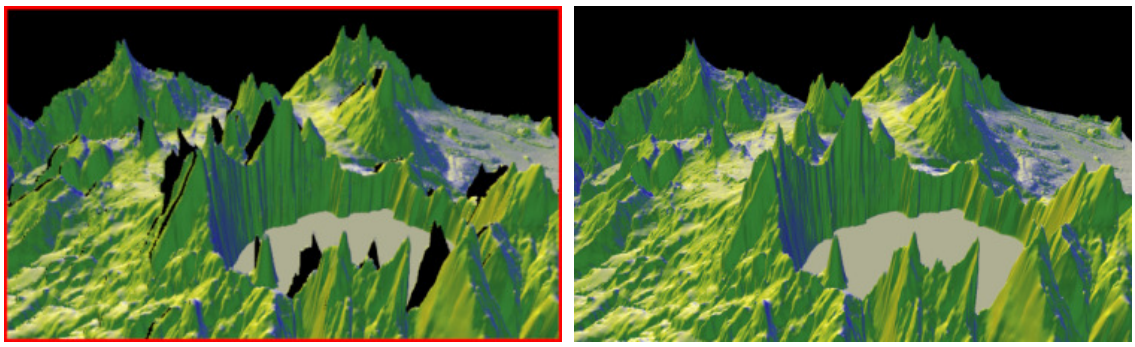


Figure 5.9. General pinhole camera used for 3-D warping. Approximate translation support by 3-D warping (*left*) and ground truth for comparison (*right*).

5.2.2.3. Rendering and Display

Once the GPC is built, ray casting for volume rendering or for surface rendering can proceed like before. For feed-forward rendering the projection operation is slightly more complicated. Like before, a 3-D point *P* is first projected with the

PPC to *p*. Then the focus regions are consulted sequentially and if one contains *p*, the GPC image coordinates are computed by solving the cubic Equation 5.4 with unknown *u*. Since the focus regions are disjoint, at most one cubic equation is solved per projection. Rasterization does not change. Adaptive subdivision works well for a focus-plus-context GPC since the context region does not require subdivision. Whereas in the remote visualization case the GPC image was an intermediate data structure used to create the output frame presented to the user, for focus-plus-context the GPC image is displayed as is.

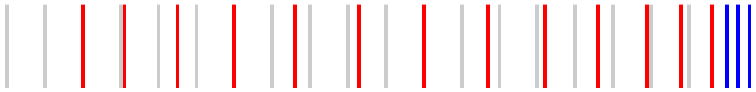


Figure 5.10. Sampling rate continuity at the transition region edges.

5.2.3. Extreme Antialiasing

When the resolution of the data to be visualized exceeds the resolution of the output image aliasing artifacts occur. Regularly sampled data (e.g. 2-D image data or 3-D volume data) can be easily downsampled to match the output resolution. Reducing the resolution of geometry is more challenging. Many techniques for computing lower complexity versions of a 3-D geometric model have been proposed, but no perfect solution exists. LoD computation is slow, which is inadequate in real-time visualization, it depends on the specifics of the data, which requires specialized algorithms and manual fine tuning of parameter values, and transitions between LoDs often cause abrupt output image changes.

Instead of reducing geometric complexity to fit the output image, an obvious alternative is to temporarily increase the output image resolution to capture the complex geometry well, and then to convert the auxiliary image into the desired output image. The approach is appealing because the resolution mismatch is resolved in the image domain, where it is straightforward. The approach has two

implications. First, the geometry is rendered at full complexity. Whereas employing LoDs also reduces rendering load in addition to improving visual quality, graphics hardware can now handle many datasets at full geometric complexity. For such datasets this first implication does not pose problems.

The second implication is that the framebuffer needs to be supersampled to accommodate the complex geometry. Conventional full-screen antialiasing cannot and should not be used for this purpose. Consider a dataset where high geometric complexity is confined to a few clusters and let's assume that the largest supersampling factor demanded by any of the clusters is 1,000 color samples per pixel. A 1,000x supersampling of the entire output image is and will remain prohibitive for the foreseeable future. We have developed a GPC variant that enables supersampling locally with extreme supersampling factors at small and controllable additional framebuffer memory and rendering costs.

5.2.3.1. Prior Work

Mipmapping [166] is an effective hardware supported technique for avoiding minification artifacts when rendering from image data. Antialiasing surface geometry however remains challenging. By combining supersampling with multisampling, which only supersamples coverage and shades once per fragment, today's ultra high-end nVidia [108] and ATI [9] GPUs achieve a full-screen total antialiasing level of 64x (i.e. 8x8). While this is adequate for smoothing triangle edges in most cases, it is insufficient for avoiding minification artifacts when rendering triangles with a small screen footprint. For such triangles a higher level of true supersampling is needed. Full-screen true supersampling at extreme levels (e.g. 1,024x) will remain impractical for the foreseeable future. However, extreme supersampling is only needed for the screen areas with extreme complexity. The GPC enables feed-forward rendering with adaptive supersampling, a practice reserved so far for ray tracing [51].

Finally we note that both nVidia and ATI have exposed functionality for per primitive antialiasing setting, but the supersampling level cannot be locally adapted.

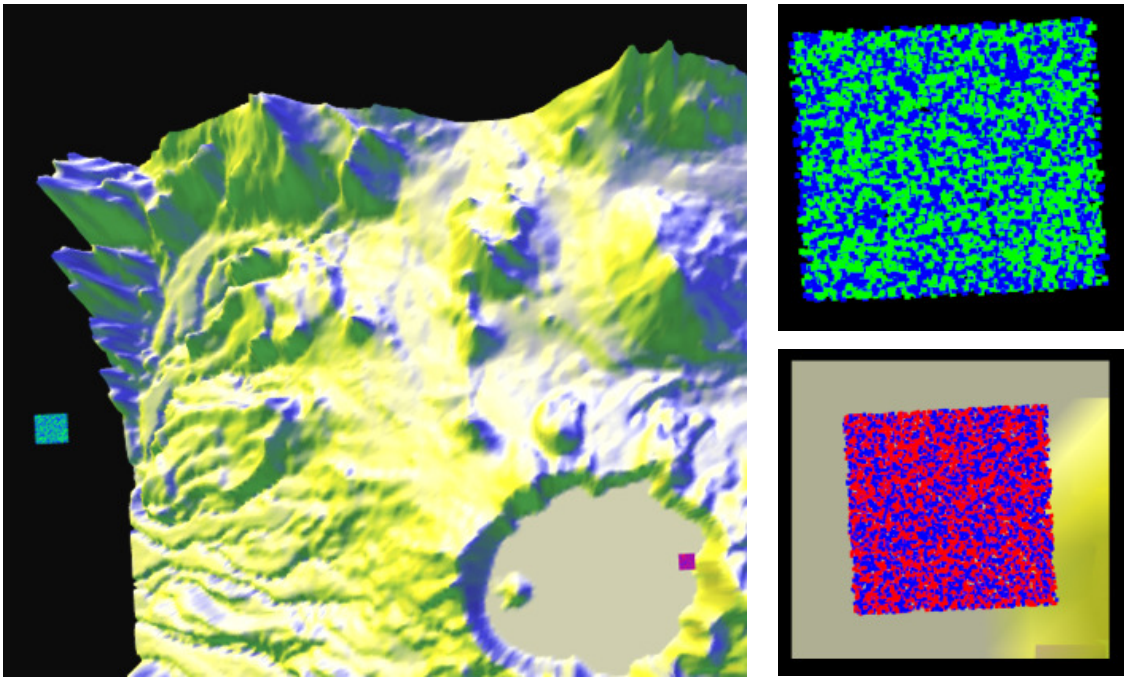


Figure 5.11. Image rendered using general pinhole camera extreme antialiasing and off-screen framebuffer tiles.

5.2.3.2. Sample Location Selection

Our algorithm assumes that the clusters of high complexity are known. For example the test scene in Figure 5.11 has two clusters of 10,000 square particles each. The particles are placed randomly and close together in the cluster volume. Conventional hardware supported antialiasing is not sufficient even at high 16x settings. The screen regions where extreme supersampling levels are needed are defined by the output image projections of the bounding volumes of the clusters. The supersampling factor is determined for each region independently based on cluster geometric complexity, on screen region size, and on available resources (i.e. framebuffer memory and rendering power). For the

extreme antialiasing (XAA) GPC there is no transition region. For Figure 5.11 the XAA GPC camera model has two supersampled regions, one for the green-blue cluster and one for the red-blue cluster. The supersampling factors are 256x (i.e. 16x16) for the green-blue cluster and 1,024x (i.e. 32x32) for the red-blue cluster. When the view changes or if the clusters are dynamic, the GPC is updated to track the clusters.

5.2.3.3. Rendering

The rendering algorithm uses off-screen framebuffer space. A rectangular tile is assigned to each supersampled region. A triangle *t* is rendered as follows:

1. Project *t* to *t'* conventionally
2. If *t'* intersects the output image, rasterize conventionally
3. For each supersampled region SSR_i intersected by *t'*
    a. If t' is completely inside SSRi, map *t'* to the off-screen tile T_i of SSR_i and rasterize *t'* in T_i
    b. If *t'* crosses the border of SSR_i,, intersect bounding box of *t'* with SSR_i to obtain *bb*, compute rasterization expressions $E_i(t)$ at the four corners of *bb* as well as the barycentric matrix of *t*, *BM(t)*, and finally rasterize *bb* using $E_i(t)$ and *BM(t)*.

In Figure 5.11, right, both tiles are 600x720 pixels in size which is sufficient to host the supersampled projections of the cluster volumes. A triangle is rendered at most *n*+1 times, where *n* is the number of supersampled regions. The algorithm clips with a tile at bounding box level and edge sidedness is enforced during rasterization, which is advantageous since most primitives that map to a tile have a small footprint. The red-blue particles are rasterized conventionally and only a few large background triangles (yellow terrain) are rasterized as clipped bounding-box quads.

5.2.3.4. Display

The XAA GPC image is an intermediate representation from which the final image is computed. The screen pixels covered by the clusters are reconstructed by filtering the supersampled tiles. A straightforward approach is to use mipmapping. A more accurate approach is to actually convolve with a high resolution kernel, which remains efficient since the number of output pixels covered by the clusters is small. We use flat kernels with 1 pixel support, which means that an output pixel is reconstructed as the average of *k* color samples, where *k* is the supersampling factor for the cluster region.

5.3. Results and Discussion

We tested our approach on several datasets including a Rayleigh-Taylor instability surface (Figure 5.1) [81], a Ionization Front volume dataset (Figure 5.8) [164], the Crater Lake terrain dataset (Figure 5.9) [157], and a DNA molecule triangle mesh (Figure 5.2) [159]. All performance numbers were measured on a 3.16 GHz Intel Xeon workstation with 4 GB of RAM and an nVidia GeForce GTX 280 graphics card.

5.3.1. Remote Visualization

5.3.1.1. *Rendering Performance*

All visualization algorithms described run on the GPU. The adaptive subdivision GPC rendering algorithm uses multiple geometry shader passes to circumvent the bottleneck at emitting primitives. At each pass a triangle is subdivided into at most 4 triangles. The triangle data is piped in from one pass to the next using two vertex buffers. Triangles that do not need subdivision are simply passed through. The subdivision ends if a pass does not subdivide any triangle, or after

a maximum number of passes (e.g. 6). The resulting triangles are rasterized with a single fragment shader pass.

We have compared the performance of the GPC algorithms for rendering geometry data on several resolutions of the Crater Lake dataset obtained by downsampling the original resolution (Table 5.1). Nonlinear Rasterization is consistently outperformed by Adaptive Subdivision due to bounding box estimation, geometry to fragment shader communication, and overdraw costs. Compared to the approximate algorithm of GPC projection followed by Conventional Rasterization, the Adaptive Subdivision quality advantage comes at a smaller relative cost for finer datasets.

Table 5.1. Performance for various resolutions of the Crater Lake dataset.

Triangles (x10^6)	**Nonlinear Rasterization (Hz)**	**Adaptive Subdivision (Hz)**	**Conventional Rasterization (Hz)**
0.12	3.2	15	55
0.5	1.1	9.1	18
2	0.33	5.3	6.7

For volume rendering, the cost of distorting the PPC ray to obtain the GPC ray is dwarfed by the cost of stepping along the ray. For Figure 5.8, the GPC image resolution is 1,200x720, the volume resolution is 700x496x496, the number of ray steps is capped at 1,024, and the GPC image is rendered at 3.51 Hz. For a comparable PPC the frame rate is 3.93 Hz, difference due to a higher average number of steps for the GPC (rays focused at volume center).

Resampling a GPC image to the output frame runs at hundreds of frames per second on the GPU and it is even sufficiently light weight to run interactively on the CPU (13 Hz for frames in Figure 5.1). Like resampling, 3-D warping has cost bounded by the resolution of the GPC. On the GPU the 864K points of a

1,200x720 GPC are reprojected at over 60 fps. Even in software the GPC is warped at 7 fps using simple 1x1 splats for reconstruction. The GPU can easily handle more sophisticated reconstructions such as connecting the GPC samples in a triangle mesh.

5.3.1.2. Image Data

Building GPC images from image data is straight forward. Each pixel in the GPC is found in the input PPC by evaluating the second order distortion polynomial given in Equation 5.1. Figure 5.12 shows a GPC image of the Chicago area which allows the user to zoom in at the center without the pauses needed to download the next level-of-detail which are common to current applications.

5.3.1.3. Communication Performance

We have integrated the GPC framework into an application that sends images between a visualization server and a visualization client via TCP/IP. For a typical path through the Crater Lake dataset, the user requested two GPC reference images, which were used at the client to create 1,800 output frames. The GPC resolution was 2,560x1,440 and the output frame resolution was 1,280x720 (i.e. 720p). The average GPC reference image size was 8 MB for color and 7 MB for depth, using non-lossy LZW compression. Over the campus 100 Mb network the average time for transferring a GPC reference image with depth is 5.04 seconds. All transfer times are measured from the time the client issues the request to the time the client has finished uncompressing the received image. Once the client receives the GPC the frame rate exceeds 60 fps.

We have compared the GPC to the conventional approach of sending individual visualization frames. We tested several compression modes: non-lossy LZW and JPEG with 3 quality factors. The results are summarized in Table 5.2.

Figure 5.12. General pinhole camera used on image data. GPC reference image and frame resampled from it (*top*) and corresponding PPC image and frame for comparison (*bottom*).

When sending each frame from the server the frame rate is below 2 fps even when the frame is aggressively compressed (Figure 5.13), and even with the high bandwidth network. The GPC approach on the other hand provides a frame rate of over 60 fps at the cost of the two 5 second pauses needed to transfer the two GPC reference images. The GPC provides great quality from the viewpoint of the reference image and good quality from translated viewpoints.

We have also tested our system using standard residential broadband connectivity (i.e. cable modem, ~300 KB/s average download rate). Transferring the GPC image took on average 105 s. Once the transfer completed the laptop on which the client was running was resampling and warping the GPC at over 20 fps. For the conventional approach lossless compression is impractical since the frame rate is below 0.1 Hz. JPEG compression achieved 0.5-1.0 fps.

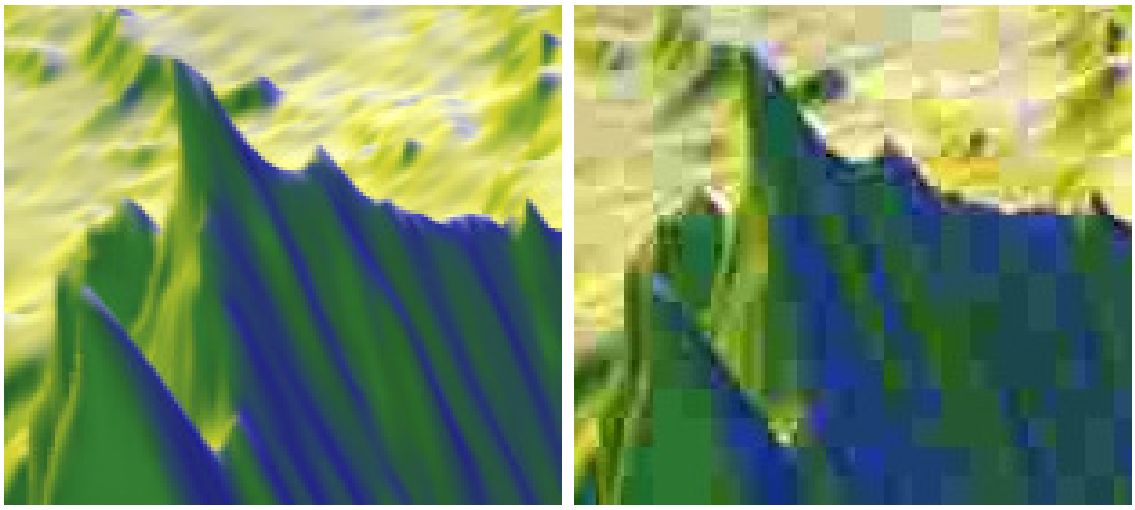


Figure 5.13. General pinhole camera frame compared to JPEG frame. Frame resampled from GPC (*left*) and frame compressed with JPEG with a quality factor of 10% (*right*).

5.3.1.4. Discussion

Compared to a single conventional PPC image, a GPC image takes longer to render because of the non-linear projection operation. However, using fast feed-forward rendering, the GPC rendering time is negligible compared to network transfer times. Moreover in typical remote visualization scenarios rendering capabilities are distributed asymmetrically between the server and the client, in favor of the server, and the GPC is rendered on the server. A GPC image takes up more space than a single PPC image because it is larger and because it also stores depth per pixel. Assuming that the GPC image is twice as big as the PPC image in each direction and charging 32 bits per depth sample, a GPC image is typically 8 times larger than a conventional PPC image. Even so, bandwidth savings are considerable since hundreds or even thousands of output frames are computed from a single GPC image. In our example, 2 GPC images were used to create 1,800 frames.

Table 5.2. Average per frame communication performance for *conventional remote visualization* at various compression settings.

	LZW	**JPEG**		
		10%	**50%**	**90%**
Data (MB)	2.15	0.0338	0.0804	0.184
Time (s)	1.54	0.70	0.74	0.93

Rendering quality GPC images requires antialiasing, which is not more complicated than in the case of conventional images. Like for conventional antialiasing, multiple samples are computed per pixel and the samples are blended to produce the final image. The feed-forward algorithms described readily produce antialiased GPC images. If the GPC image is rendered by ray tracing, antialiasing proceeds as usual, with the exception that the additional rays per pixel are defined using the GPC model and not the PPC model, a negligible cost compared to actually tracing the additional rays.

Like in any image-based rendering method, the GPC approach implies an additional resampling step, which reduces the quality of the output image. The quality loss can be reduced by not antialiasing the GPC image, which allows reconstructing the output frame from point samples. On a GPU the resampling step is essentially free from the performance standpoint (i.e. in our experiments resampling took less than 2ms). Even when no hardware support is present, resampling can be executed on the CPU at interactive rates (e.g. 13 Hz in our case). We will test our system on additional client platforms in the future. For platforms such as cell phones the lower compute power is compensated by a lower screen resolution, so software resampling at interactive rates probably remains tractable. Moreover, the main concern for such platforms is the low connectivity bandwidth (e.g. 3G), which makes the GPC approach even more appealing.

GPCs support blurriness-free zoom up to a user chosen factor. Higher zoom factors come at the cost of a lower resolution at the periphery of zoomed out frames (Figure 5.14). The examples shown here were constructed under the assumption that the most likely zoom center is the center of the image. This assumption is supported by the fact that in interactive visualization the view typically changes smoothly. The view is a function $V(f, t_x, t_y, t_z, \varphi, \theta, \rho)$ where f is the focal length, t_x, t_y, and t_z are the translation parameters, φ, θ, and ρ are the rotation parameters, and the frame resolution is assumed to be constant. Smooth navigation implies that these parameters change continuously thus views following a reference view will have similar view directions, and thus zooming occurs close to the center of the reference view. The GPC image anticipates small variations of each of the 7 view parameters: the depth channel allows warping the samples to nearby viewpoints, a larger field-of-view allows panning, tilting and zooming out, and a higher resolution at the center allows zooming in. The GPC essentially covers a 7-D volume of views centered at the reference view. The output frame reconstruction quality decreases towards the periphery of this view volume (i.e. increasing severity of disocclusion errors and increasing blurriness). The GPC buys time to transfer a new reference image. Unlike conventional remote visualization, the GPC approach scales well with the frame rate at the client. The higher the frame rate at the client, the higher the benefit of the GPC. A high frame rate implies a dense sampling of the view volume covered by the GPC image, and more output frames are reconstructed from a single GPC image than in the case of a small frame rate.

The overall size of the view volumes covered by a GPC and a PPC image is comparable, since the two images have the same number of samples. However, the shape of the view volume covered by the GPC is better suited for remote visualization. The reference view is not at the center of the PPC volume of views, but rather at its periphery: the PPC does not support zooming in or forward translation. The GPC volume of views is more evenly distributed around

the reference view, at the cost of reducing the pan-tilt range, which is a desirable trade-off. The GPC image is an image-based model that allows rendering multiple output images. Like for all models, image-based or not, the utility of the model is not synonymous to the user rendering all possible images of the model. If the user never zooms in or never translates forward, the higher resolution at the center of the GPC remains unutilized. However, zooming in and translation forward are likely operations in interactive visualization since they are the mechanisms that allow the user to increase the level-of-detail.

Let's compare the GPC approach to a simple approach of zooming in mipmapping fashion. Initially the output frame is transferred from the server along with a PPC reference image that has the same view but twice the resolution (4 times the number of pixels). The client reconstructs the output frame by trilinear interpolation between the two images. Once the resolution of the reference PPC image is surpassed, a new image is requested from the server and so on. If the output frame resolution is w x h, the approach transfers $4wh$ pixels for each 2x zoom in sequence. Now let's consider a GPC reference image of w x h resolution with a central region of size $w/2$ x $h/2$ and a zoom factor of 2 (Figure 5.15). The amount of data transferred is reduced 4 fold to wh, at the cost of a lower peripheral sampling rate. Using Equation 5.1, we find that the minimum sampling rate for the first (F_1) and the last (F_n) frame of the zoom in sequence is 0.4 and 0.9, respectively. The sampling rate requirement (e.g. 1.0 for F_1 and 2.0 for F_n) is met at the center of the frame throughout the zoom in sequence. The GPC is a flexible camera model that allows trading off periphery sampling rate for zoom in support. For example one could choose a sampling rate of 4.0 at the central region of the GPC image, which suffices for *two* 2x zoom in sequences, reducing the transfer cost 8 fold compared to the mipmapping approach. Another application might choose to enforce a minimum sampling rate of 1.0 for F_1 by increasing the resolution of the GPC image to $1.6w$ x $1.6h$. In this case the data reduction factor is 4.0/2.56, without any decrease of sampling rate

at the periphery. This is of course possible since the mipmapping approach uses a 2.0 sampling rate at the periphery which is never used.

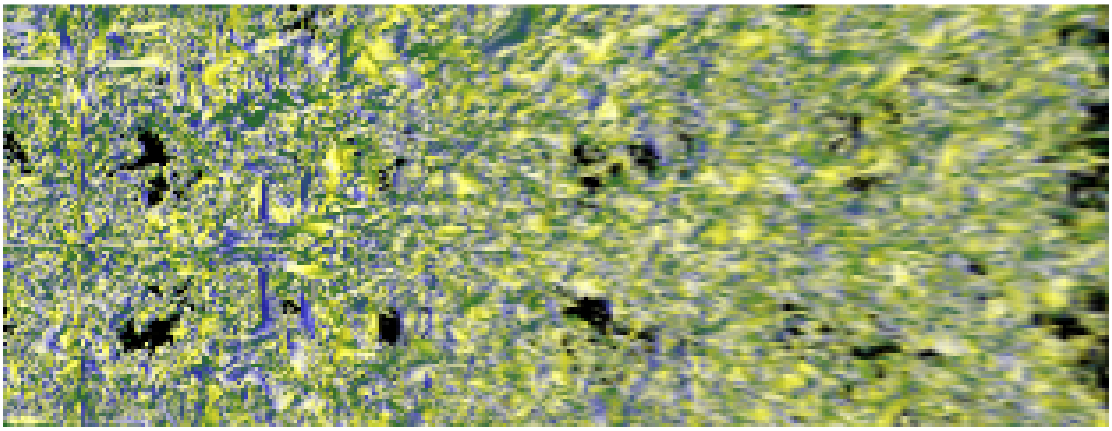


Figure 5.14. Illustration of the decrease in resolution from center to periphery of a frame resampled from general pinhole camera image.

If regions of interest are known for a dataset, the GPC should be built to allocate more samples to those regions, similar to the GPC images constructed for the focus-plus-context application. Many datasets have intrinsic regions of interest. For example, in a computational molecular dynamics simulation where the goal is to investigate the therapeutic potential of a designed molecule (ligand), the visualization is likely to focus on the biomolecule receptor sites which reveal the quality of the fit between the ligand and the biomolecule. These receptor sites are known a priori and can be marked as regions of interest. In a computational fluid dynamics application an algorithm for extracting features (e.g. separation surfaces, vortices) produces geometry of variable complexity, and regions with high complexity are likely to be examined by the user at a higher resolution. These regions should be marked as regions of interest anticipating the need for additional samples. Similarly a CAD model of a complex system can have known regions of interest such as complex components, components that are known to fail, or components that are likely to have failed based on diagnostic tests performed by a technician in the field.

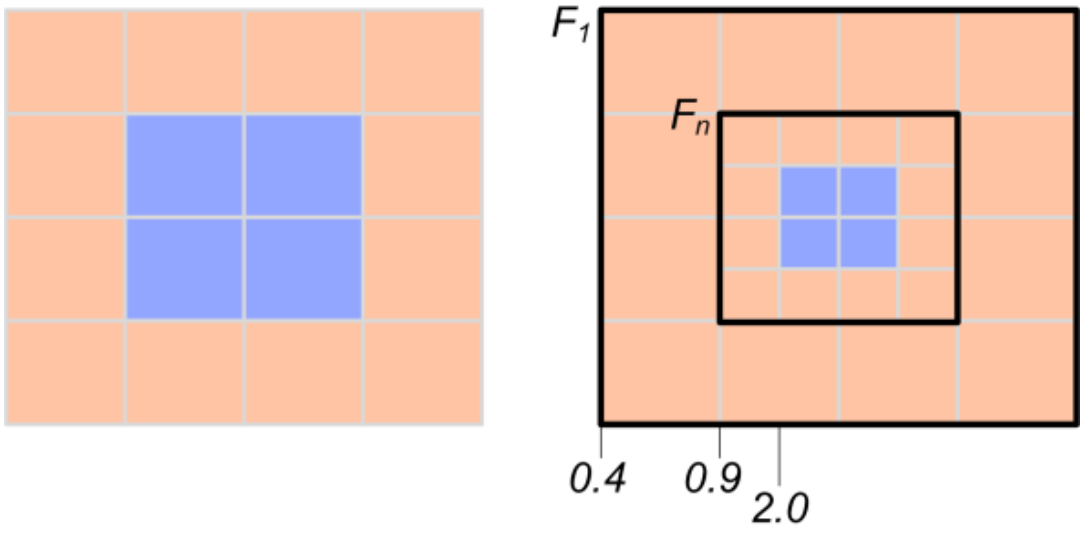


Figure 5.15. General pinhole camera sampling rate visualization. The black rectangles show the samples used by the first and last visualization frame in a 2x zoom in sequence.

A current limitation of our GPC-based remote visualization system is the relatively long GPC transfer times. Several strategies can be employed to reduce this time including progressive refinement of color and depth, not transmitting depth at all and supporting translation by texture mapping the frame onto a quadrilateral, compressing color with various quality factors, or simply letting the user navigate through the current GPC while the next GPC is being transferred.

5.3.2. Focus-Plus-Context Visualization

5.3.2.1. Rendering Performance

For focus-plus-context GPCs, the cubic equation per distorted vertex is an important performance factor. GPC rendering performance increases for Figure 5.2 from 7 Hz to 10 Hz or 11 Hz if the cubic distortion is replaced with a quadratic or linear distortion.

5.3.2.2. Discussion

The GPC provides a focus-plus-context technique with the important advantages of simplicity, versatility, interactive rendering performance, and sampling rate continuity between focus and context regions. The interactive rendering performance not only supports dynamic scenes, but it also allows the user to modify the focus region parameters at interactive rates. If the focus regions are known a priori, or once the focus regions are found, the user can lock them and continue interactive exploration. A unique strength of our method is the robust and efficient handling of surface geometry data.

5.3.3. Extreme Antialiasing

5.3.3.1. Rendering Performance

The image in Figure 5.11 is rendered at 21.2 Hz. Increasing the number of particles in each cluster from 10^4 to 10^5 and then to 10^6 (>4 million triangles), the frame rate becomes 14.5 Hz and 3.76 Hz, respectively. The bulk of the time is spent rendering the GPC image—the kernel-based reconstruction takes negligible time.

5.3.3.2. Quality

Figure 5.16 shows GPC XAA at work in 3 visualization frames. The camera translates forward so the tree has a larger and larger footprint. The size of the off-screen tile is 640x720 for all three frames, which is sufficient to render the tree well with a single color sample per pixel. The supersampling factor is chosen as to maximize the utilization of the off-screen tile, and it decreases as the tree footprint grows bigger. The size of the tree in the off-screen tile remains approximately the same. The biggest change in size is recorded when the

supersampling factor changes from one discrete level to the next (e.g. from 144 to 121). The correct reconstruction prevents transmitting the abrupt change to the output image, where the cluster regions are smooth, have accurate borders, and exhibit good frame to frame stability. On the other hand 16x hardware antialiasing alone does not render the correct foliage volume, even when the camera is closest to the tree.

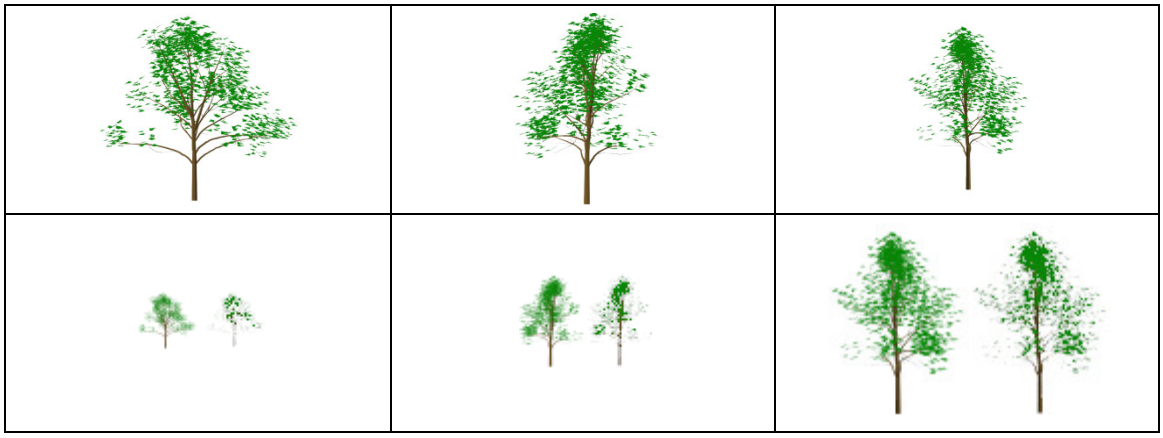


Figure 5.16. Extreme antialiasing compared to hardware antialiasing. Off-screen framebuffer tiles (*top*) and (*bottom*) comparison between GPC XAA + MSAA 16x (*left*) and MSAA 16x alone (*right*). The XAA factors are 361x, 144x, and 25x.

5.3.3.3. Discussion

The GPC allows for a flexible management of framebuffer resources enabling local supersampling factors that are out of reach for full-frame antialiasing. Complex datasets are visualized directly, bypassing problematic offline simplification. GPC XAA works under the assumption that extreme geometric complexity is concentrated in a few screen regions. For example, a city scene with a few complex trees or a flow dataset with a few complex turbulences should be rendered using GPC XAA whereas for a forest conventional LoD approaches are still needed. In general any dataset with great complexity variation is suitable for GPC XAA whereas datasets with uniform complexity, low or high, are not.

A current limitation of the GPC XAA algorithm is that the supersampled regions have to be disjoint. In the current implementation overlapping regions have to be merged into a single region with a single supersampling factor, which can lead to aliasing when the original regions required widely different supersampling factors. This can probably be addressed with an improved reconstruction algorithm that takes into account the depth channel of the off-screen tile to depth-composite the tile into the output image. Another limitation is that the off-screen tiles are uniformly sampled which considerably weakens the antialiasing capability of the algorithm for thin nearly horizontal or nearly vertical features. In such cases a large number of samples can change sidedness from a frame to the next causing temporal aliasing. The solution to this problem is well known and we foresee that the algorithm can be adapted to use jittered sampling locations in the off-screen tile.

CHAPTER 6. RASERIZATION FOR NON-LINEAR PROJECTION

All of the camera models discussed have been designed to overcome occlusions or to non-uniformly sample 3-D scenes. The great advantage from a computational standpoint of the planar pinhole camera model is that it is constructed and rendered from a set of linear equations efficiently on graphics hardware. The problem of rendering cameras with non-linear projection at interactive rates is a broad and sometimes challenging one, but all camera models that have been introduced were designed specifically to address occlusion or non-uniform sampling while rendering interactively.

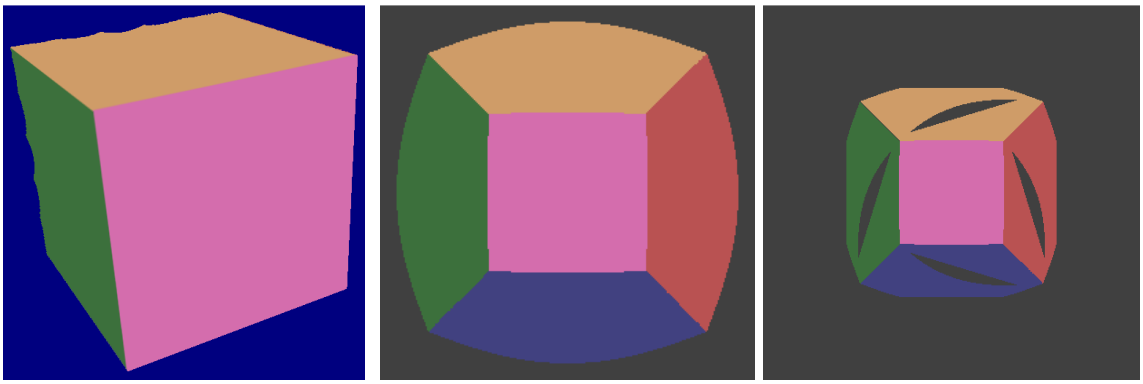


Figure 6.1. Example single pole occlusion camera image of a box. The straight lines of the box (*left*) are no longer straight in the SPOC image (*middle*) which can lead to cracks when rendering (*right*).

The great challenge in rasterizing when using cameras with non-linear projection is that straight lines in 3-D space do not necessarily project to straight lines on the image plane anymore (Figure 6.1). This implies that the edges of a polygonal primitive (i.e. triangle) are not necessarily line segments after projection, and that the bounding box of the projected triangle is not necessarily the bounding box of

its projected vertices anymore. If not properly accounted for, this can lead to cracks when rendering (Figure 6.1). Moreover, the color and depth parameters of interior pixels of polygonal primitives can no longer be calculated as linear systems.

No single solution exists for rendering camera models with non-linear projection, but the solutions tend to break down into a few available techniques.

6.1. Rendering Techniques

6.1.1. Raycasting

Raycasting is the process by which an image is broken down into a set of camera rays or ray segments and each individual camera ray is intersected with the 3-D scene. This level of generalization makes raycasting an option for any camera model composed of straight ray segments. The rays of all three occlusion cameras, the graph camera, and the general pinhole camera are piecewise linear segments. The rays of the cameras can easily be distilled into individual segments and later be recombined making raycasting a viable solution for rendering images. By subdividing curved segments into small piecewise linear segments, the curved ray camera can also be rendered using raycasting.

Raycasting is less well suited for interactive rendering than other approaches. In its naïve implementation, every ray must be tested for intersection with every piece of geometry within a 3-D scene. More advanced implementations which use spatial hierarchies to improve performance, still end up performing large numbers of intersection tests just to find the closest triangle that intersects with the ray. These large numbers of intersection tests preclude interactive rendering in scenes of non-trivial complexity. Instead a solution must be chosen which takes advantage of the efficiency of the feed-forward graphics pipeline.

6.1.2. Point-Based Rendering

A straight forward way of rendering images of 3-D scenes is to simply ignore the connectivity data attached to the scene and render the scene as a set of points. Alternatively, the scene triangles could be rendered using a conventional planar pinhole camera and then have all the samples distorted to the final output image plane locations. These types of sample-based approaches often suffer from 'holes' in the output image locations where too few samples project. Both of these approaches need to address the problem of maintaining surface continuity to guarantee adequate output image coverage.

This problem has been studied extensively in the world of image-based rendering [102, 119, 128, 139] where a number of possible solutions exist. One possible solution is to use splatting, where point-based samples are replaced with overlapping surface elements. Splatting can give good scene coverage without the need for any explicit connectivity. Such techniques are not ideal as they are only a surface approximation and end up simply discarding the connectivity data that already enumerates the surface within the 3-D scene dataset.

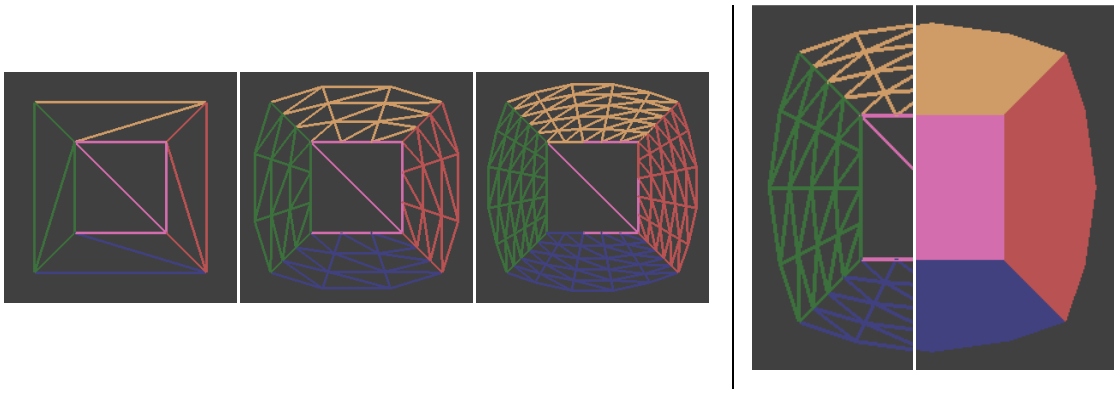


Figure 6.2. Wireframe single pole occlusion camera image constructed with subdivision factors of 1, 3, and 5.

6.1.3. Triangle-Based Rendering

An alternative method to point-based rendering which uses 3-D scene connectivity data is triangle subdivision. In this approach the triangles of the scene are subdivided to a fine-grain level, projected using the relevant camera model, and rasterized using the conventional feed-forward graphics pipeline.

The general algorithm used with this method is:

1. Project the vertices of the triangle *t*
    a. If the length of an edge is great than epsilon
        - Subdivide triangle *t* into triangles $t_0' \ldots t_n'$
    b. Else
        - Pass through triangle *t*

The algorithm takes a triangle as input. It projects the vertices of the triangle and subdivides the triangle based upon the projected edge lengths. The algorithm can be run for a fixed number of iterations or until the maximum edge length is below some epsilon. Figure 6.2 shows an example SPOC with various subdivision levels. The smaller the output triangles, the more accurate the SPOC rendering becomes.

This method has the advantage of addressing the limitations of the previous two methods. Like raycasting, this method uses the connectivity data to produce a crack-free output image. Like point-based rendering, this approach fits into the standard feed-forward graphics pipeline. These two reasons make this approach the best compromise between speed and quality.

However, this method does come with its own shortcomings. This approach is in reality an approximation to point-based rendering. In order to produce an output image that accurately represents the scene, the triangles of the scene need to be

subdivided to a very fine level, about 1 output pixel in size. Otherwise, the triangle pixels may rasterize to incorrect image plane locations.

The fine grain subdivision needed leads to a few problems. The subdivision of the scene is for the most part view dependent. If the subdivision is pre-computed, the maximum level of potential subdivision must be selected, resulting in large quantities of data. Modern graphics cards now have primitive level programmability which enables in-stream triangle subdivision. Using this capability has severe performance implications, which should be overcome by the introduction of primitive level tessellation in future architectures. The other problem with this approach is that the increased number of triangles requires many more projection operations be performed than are normally required to render a planar pinhole camera image.

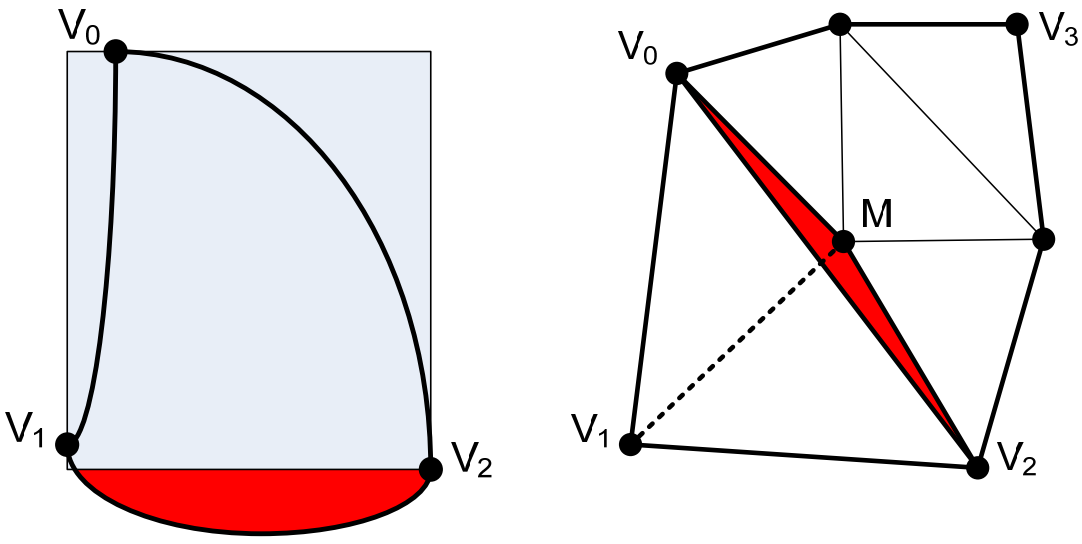


Figure 6.3. Illustration of incorrect bounding box. The incorrect bounding box of curved projected triangle (*left*) leads to cracks between adjacent projected triangles (*right*).

6.1.3.1. Hybrid Raycasting

A final means of rendering with non-pinhole cameras is a hybrid approach which comes full circle back to raycasting. While traditional raycasting tests whether each ray intersects with each triangle, hybrid raycasting performs the same ray triangle intersection test per triangle on a smaller, yet conservative set of rays (only those who could potentially intersect the triangle). One of the more powerful aspects of this approach is that it runs one triangle at a time, fitting into the feed-forward graphics pipeline. The resulting images have the same quality and correctness of standard raycasting, but the performance begins to approach that of the triangle and point-based feed-forward techniques.

The basic algorithm for raycasting approach is as follows:

For each triangle *t* in the scene

1. Calculate a conservative screen space bounding box
2. For every pixel within that bounding box
    a. Calculate the ray *r* that pixel represents
    b. Test for intersection between *r* and *t*
        - If they intersect
            - Compute barycentric coordinates of the intersection
            - Compute rasterization parameter values (i.e. color and depth)
        - Else discard sample

Step 1 is responsible for calculating a compact but conservative region in which a ray could potentially intersect the current triangle. If this calculation is too aggressive, rays might never be tested against triangles they intersect. If the region is overly conservative, computation will be wasted on rays that have no chance of intersecting the triangle.

Step 2 tests each of the rays within the bounding box for intersection with the individual triangle. Rays which do intersect the triangle have their parameters calculated and the equivalent color and depth buffer locations set. Rays which do not intersect the triangles are simply discarded.

This approach produces high quality results with a few limitations. The efficiency and correctness of this method requires that a conservative footprint of all triangles can be calculated. If the triangle footprint is too small, this can lead to holes in the output image (Figure 6.3 and Figure 6.4). Alternatively, if the footprint is too large, computation can be wasted testing for ray-triangle intersections that will never occur.

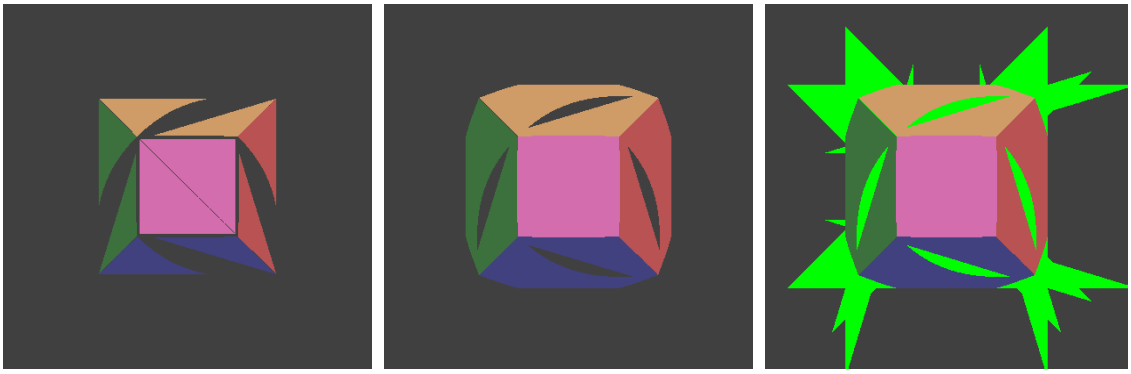


Figure 6.4. A single pole occlusion camera image computed with hybrid raycasting. The triangles of the left and middle images are not properly enlarged leading holes to appear in the output image. The right image highlights the triangle footprint required for images with no holes.

CHAPTER 7. VISUAL PERCEPTION OF NON-PLANAR PINHOLE CAMERA STIMULI

As computational power increases, the size and complexity of datasets increase in stride. These datasets invariably have complex composition of structures leading to occlusions when some data subsets hide other important data subsets. These occlusions limit how much is visible from any single viewpoint. This, for example, increases the complexity for a user to find subsets of interest within larger datasets. It also limits the ability of a user to simultaneous examine discrete regions.

The most common approach to overcoming occlusion is to simply use navigation. Here, the user is expected to move the camera around a complex dataset in order to identify any data of interest. One challenge of navigation is developing intuitive user interfaces. Most desktop applications navigate by using the keyboard and mouse—an interface which does not map well to natural forms of human navigation, such as walking. Another challenge is that almost all navigation is performed using the planar pinhole camera (PPC). The PPC requires direct line-of-sight to the objects it is sampling. This means that any occlusions within a scene will force the user to manually navigate the PPC to a new location in order to view the space hidden by the occluder.

An alternative to navigating a PPC for dataset exploration is to use a collection of multiple PPCs simultaneously. Using multiple PPCs has the advantages of potentially being comprehensive and therefore requiring no navigation, but the approach also has significant restrictions. Such visualizations produce a collection of views which are redundant and discontinuous. When a user views

the visualization, sequential examination and mentally aligning and connecting the various views is required.

In order to make a multiple PPC visualization comprehensive, a large number of PPCs might be required. The number of PPCs needed for complex datasets will likely be impractical for display by any conventional means. If the number of PPCs is reduced to a more modest level, portions of the dataset will be unsampled and an alternative means for viewing those regions is required.

A number of visualization approaches also exist which enhance a single PPC view in order address the problem of occlusion. Transparency attempts to solve the problem of occlusion by making outer occluding layers less opaque, thus exposing inner layers, but these approaches are limited to simple occlusions with very few layers. Cutaway addresses some of the transparency limitations by simply removing outer occluding layers. Unfortunately in the process, important data can be discarded as well. Space distortions try to rearrange data subsets such that they are all visible from a single point of view. These approaches often require a scene hierarchy and distort the spatial relationships between neighboring objects.

Using multiperspective images (MPIs) for scene exploration presents an intriguing and promising alternative to the existing approaches. The graph camera is an MPI framework which enhances a PPC with multiple perspectives by applying bending, splitting, and merging operations to a PPC frustum. The flexibility of the graph camera allows for MPIs which span a continuum of exploration methods. They span cameras which are simply PPCs enhanced with views around corners all the way to seamless and non-redundant comprehensive views of 3-D scenes. The flexibility of MPIs allows designing exploration techniques that can specifically cater to the needs of an individual application.

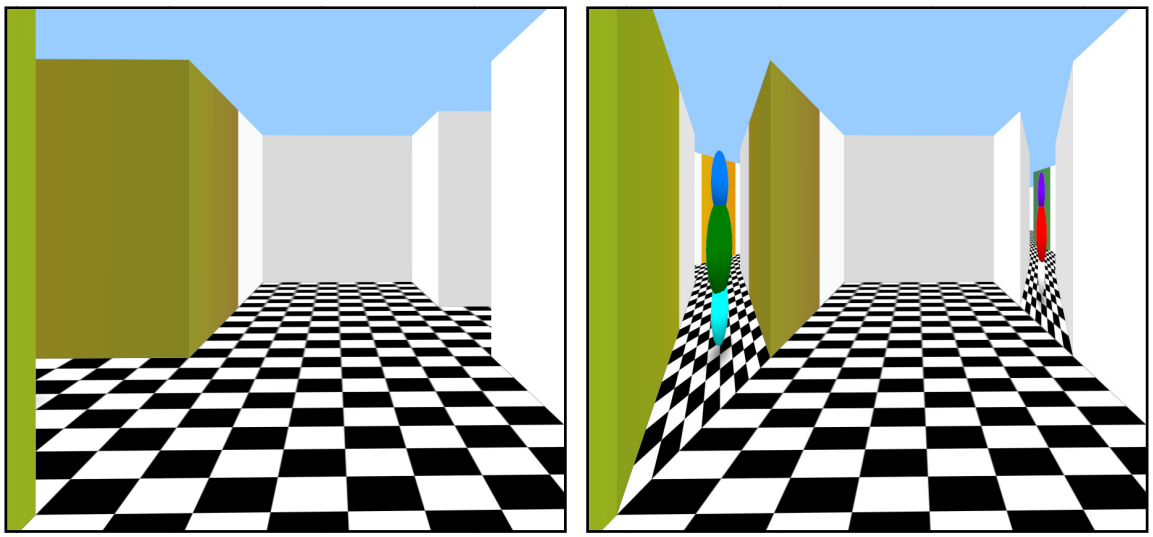


Figure 7.1. Example of the object finding experiment. The PPC (*left*) does not show the side corridors while the GC (*right*) reveals the hidden objects.

We have conducted an extensive user study to investigate and quantify the potential benefits of MPIs in scene exploration. Our study was conducted using 47 subjects whose accuracy and speed were tested for a number of tasks commonly performed on 3-D datasets. For the first set of tasks, subjects were asked to navigate a GC or PPC around a 3-D scene in order to find an object of interest as quickly as possible (Figure 7.1). The second set of tasks asked subjects to count the total number of objects within a 3-D scene when given a comprehensive set of PPCs or a comprehensive GC (Figure 7.2). The final task tested the subjects understanding of spatial orientation by playing back a prerecorded path using either a GC or PPC and asking the subject to identify a corresponding path layout.

The results of our experiments show a high potential for the application of MPIs in assisting users to perform tasks more efficiently by overcoming occlusions when viewing datasets. Our experiments showed that by using a GC, users were able to located stationary objects on average 34.8% faster compared to using a PPC. When counting stationary objects, using the comprehensive GC resulted in an accuracy of 90.2% compared to only 43.6% when using a

comprehensive set of PPCs. For the orientation task, users performed virtually identically with an accuracy of 66.7% and 69.8% for the PPC and GC, respectively.

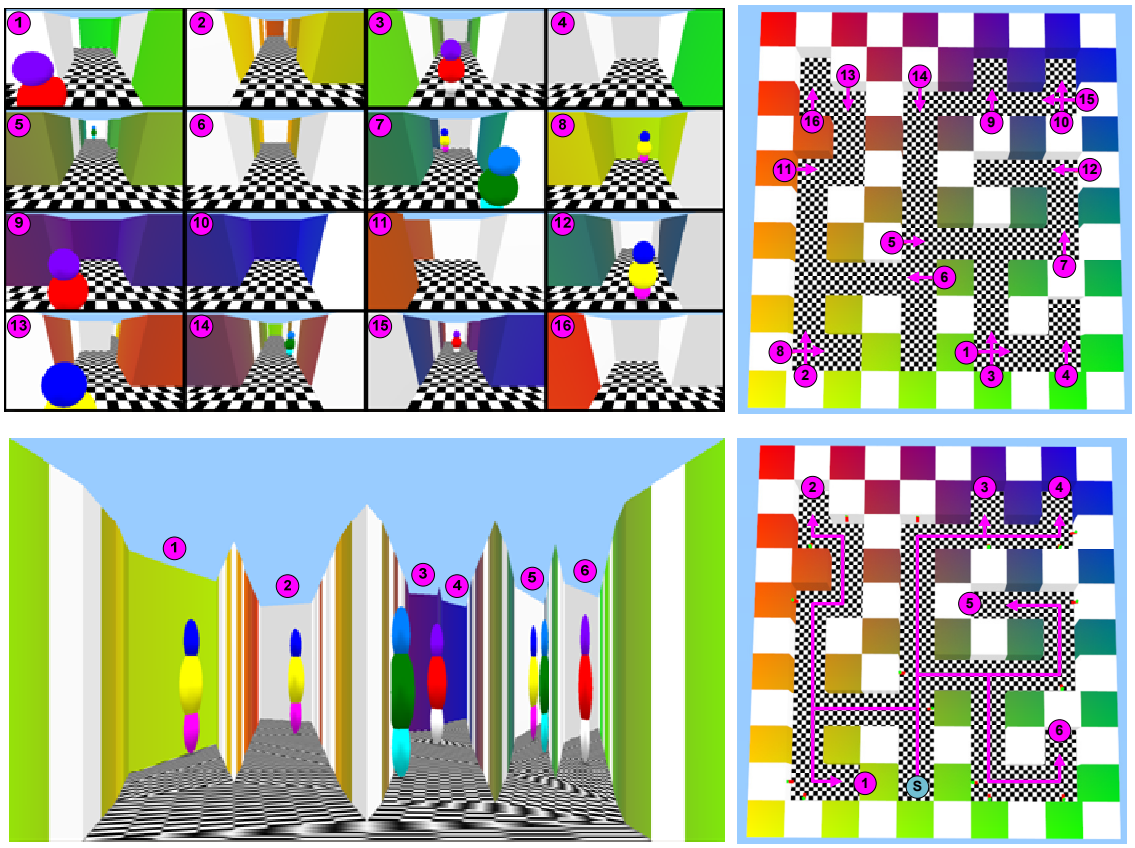


Figure 7.2. A comparison of comprehensive visualization methods. *Left:* A comprehensive matrix of PPCs (*top*) and GC (*bottom*) capture the same scene containing 7 objects. *Right:* The layout of the associated cameras.

7.1. Prior Work

To the best of our knowledge, this is the first user study on the benefits of using MPIs to explore 3-D scenes. One significant reason for that is most MPI techniques are not amenable to navigation. The most relevant prior work is in multiperspective imagery (Section 2.2), model modification and dataset distortion (Section 2.3) and navigation.

7.1.1. Navigation

Free-form, 6 degree-of-freedom 3-D navigation is difficult for users to master. Constraining navigation [11, 40, 52, 60] is the approach most frequently used to improve the user's navigation experience. This also includes keeping a safe distance from objects for the purpose of having the camera at a natural height for architectural navigation [148], providing safe navigation [46] which avoids walls and other distractions, or simply hovering [71] at a constant distance for object inspection. Simplified point-and-click controls [58, 74] or gesture-based controls [104, 121, 143, 175] are also effective at improving the user's navigation experience.

The speed of camera navigation is another concern for user comfort. A number of systems [96, 101, 155, 162] have addressed the problem of navigation in scenes of multiple scales. Here, fast and smooth transitions between environments ranging from the size of the earth down to a single room or even to a microscopic level are desired.

Identifying a path, or wayfinding, through unfamiliar 3-D space is a task which can be challenging for users. By employing tools, such as maps and signs, wayfinding through 3-D virtual environments is made easier [27]. By providing additional context, like using proxies and tethers to link maps with ground level views, subject navigation performance can be further improved [123]. Visit Wear [146] addresses the lesser wayfinding problem of revisitation (following a path already visited) by adding a history mechanism to fisheye views using a distortion-based technique. It improves performance by changing the task from one of remembering to one of visual search. It does not provide the previewing capability of the GC making it ineffective in tasks like searching and counting.

Work flow visualization tools begin to provide a multiview interface for 3-D dataset exploration. Some systems [76] provide mechanisms to review

navigation and visualization tasks in a serialized pattern (undo and redo mechanisms). Other systems [13] allow branching of parallel tasks allowing multiple views simultaneously.

There are a wide variety of full multiview systems [68, 135] and spreadsheet-like interfaces [67, 83] which exist. These techniques allow varying the visualization viewpoint or fixing the viewpoint and varying visualization parameters. Like traditional spreadsheets, the multiview spreadsheets are tabular allowing them to be treated like flipbooks, but they can also be moved, rotated, or scaled. These interfaces can provide a wide variety of views simultaneously, but lack the user interfaces for good interactive exploration of datasets.

7.2. The Graph Camera

The Graph Camera (GC) is a multiperspective framework designed around interactive rendering and a continuum of navigation modes. The navigation supported ranges from PPCs enhanced with additional perspectives, all the way to fixed position comprehensive images. This wide variety of navigation modes has led us to use the GC in this study.

The construction of a GC begins with a regular PPC frustum. That PPC frustum then undergoes 1 of 3 operations. A bend operation takes one PPC and bends the frustum producing one child PPC. The split operation takes one PPC frustum, splits it into two separate pieces producing two child frusta. The merging operation takes two PPCs and combines them to produce a single child frustum. Each child frustum is itself a PPC allowing the operations of the GC to be further applied to those frusta recursively. The result is a graph of PPC frusta.

The GC calculates a closed-form projection for each of the sub-frusta allowing for interactive rendering using the feed-forward graphics pipeline. The images of the GC are non-redundant and C^0 continuous across the changing perspectives.

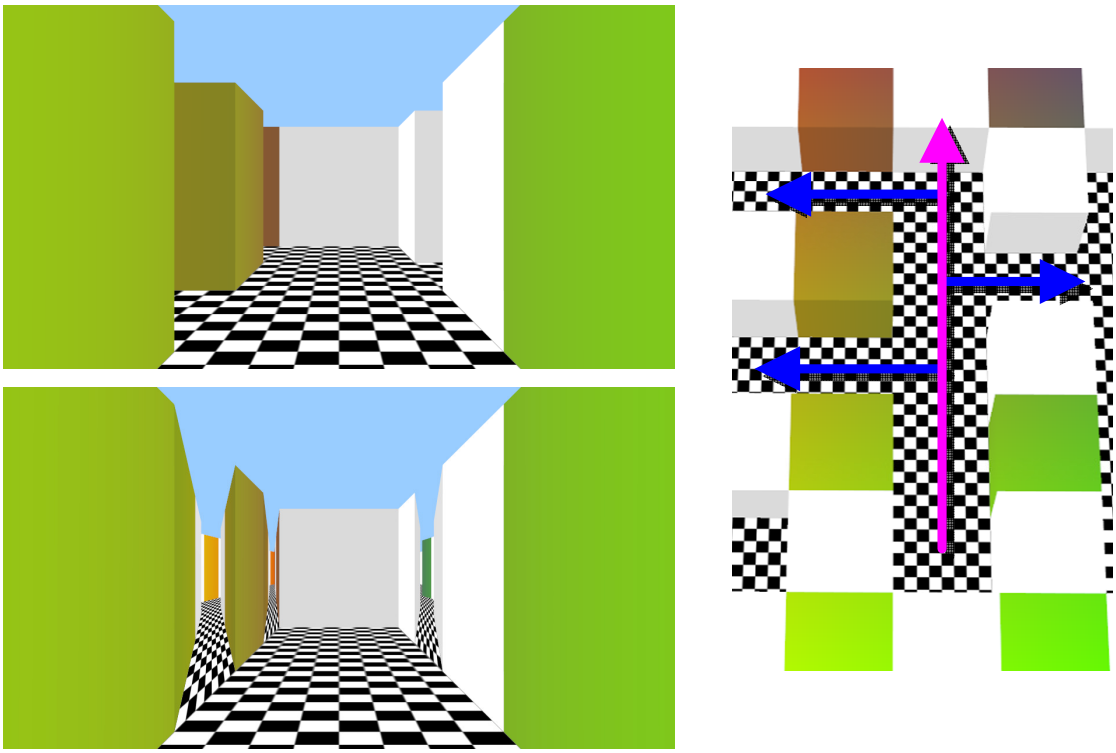


Figure 7.3. Planar pinhole camera and portal-based multiperspective image comparison. The PPC (*top*) only follows the magenta path. The portal-based GC (*bottom*) follows the magenta path plus the blue paths down the adjacent corridors.

In order to build an individual GC, an application specific construction function needs to be developed. This function will determine all of the parameters of the camera model (similar to setting the focal length and view direction of a PPC). Two construction methods are used in this study.

The first method to build the GC is a portal-based constructor. Given a PPC, the constructor automatically identifies portals (perpendicular hallways in our case) and performs a GC bend operation down those portals (Figure 7.3). The view direction of the bend is straight down the hallway. In order to facilitate a smooth transition between hallways, the portals are collapsed into the root PPC as the user approaches them.

The other construction method used is a comprehensive GC that captures the entire scene into a single image (Figure 7.2). The construction is a recursive method which begins from a pre-selected location and searches the scene in depth first order. At straight corridors, the GC does nothing. At turns, the GC simply bends around the corner. At 3-way intersections, the GC splits 2-ways in the outgoing directions. For 4-way intersections, the GC splits 3-ways for the outgoing directions. The algorithm could be generalized to n-way splits, but we chose to limit the scene to a simple grid-based layout. If the GC reaches a region which has already been visited, the search along that branch terminates. The resulting GC is a tree which provides a non-redundant view of the entire space.

7.3. Experiments

We have tested subjects' performance in three tasks to better understand what tasks in general might benefit from MPIs.

7.3.1. Finding Objects

The first task users performed was to navigate the 3-D scene in order to find an object (Figure 7.4, left). Using the mouse and keyboard, the subject was given 3 degrees-of-freedom (forward/backward, left/right, and pan) for navigation. The subject was asked to, as quickly as possible, move throughout the scene in order to locate and click on the object. Subjects were limited to 60 seconds of search time. In this experiment the subjects' performance was compared for a single PPC to that of the portal-based GC (Figure 7.1). The subjects were tested against both stationary and moving objects.

7.3.2. Object Counting

The next task subjects were asked to perform involved counting the number of objects visible within a scene. Here a comparison was performed between a comprehensive GC and a matrix of PPCs (Figure 7.2). The PPCs were hand chosen to completely cover the scene with as little redundancy as possible. The subjects were shown the scene which contained between 4 and 7 objects. The subjects were given 5, 10, or 20 seconds to count the number of objects before they were hidden. The objects could have either duplicate or unique colors. As with the previous experiment, subjects were tested against both stationary and moving objects.

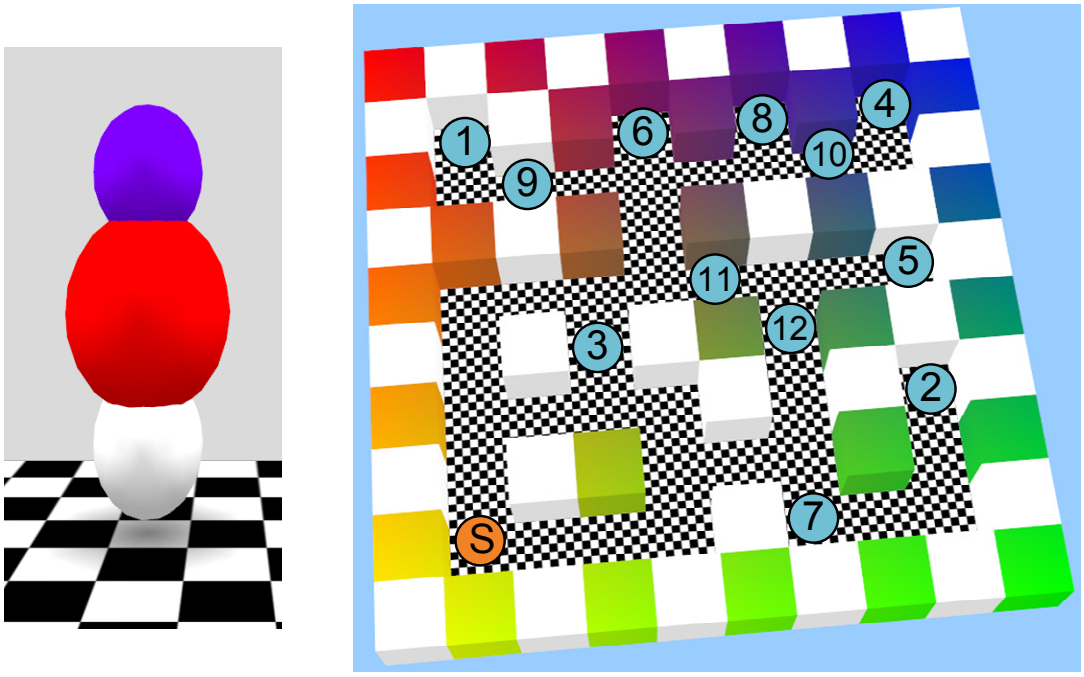


Figure 7.4. Sample object and object locations used for our user study. The 3-D scene (*right*) used for object (*left*) finding. The users' starting location is marked with an orange S and the object locations are identified by the blue circles.

7.3.3. Path Matching

The final task we asked subjects to perform was a passive test of spatial orientation. The subjects were asked to watch a video playback of a randomly selected path. The subjects were then shown diagrams of 3 paths (1 correct path and 2 randomly generated incorrect ones) and asked to match the path they just watched play back. This test was performed comparing a single PPC to the portal-based GC.

7.4. Results and Discussion

All of our experiments were performed on computers with minimum configurations of Intel Xeon 2.4 GHz, 4 GB RAM, and nVidia GeForce 280 GTX graphics cards. All experiments used Dell 24” monitors with 1,920x1,200 resolution. The PPC and GC are both capable of rendering at hundreds of frames per second, but we enabled vertical sync for a refresh rate of 60 Hz.

7.4.1. Subject Pool

We had a total of 47 subjects participate in at least one battery of tests. Subjects were recruited from our research lab and from computer science and computer technology courses. The ages of the subject pool ranged from 18 to 38 years old and consisted of 37 males and 10 females. The subjects self-reported their level of 3-D navigation experience ranging from none to very high, though the majority, 33, reported their level as high or very high. Subjects were asked to participate in 1 to 3 sessions. Of the 47 subjects, 25 subjects chose to participate in a total of 3 testing sessions.

Of the 47 total subjects, single session results are reported for only 45 subjects. One subject’s results were excluded because they became motion sick about 10

minutes into the experiment. Another subject was excluded because they suffered from an unspecified visual impairment (color-blindness, cataracts, etc).

Subjects were not financially compensated for their time. Students recruited from courses were offered extra credit. All testing was performed in accordance with the policies of our Institutional Review Board.

7.4.2. Finding Objects

The first series of tests consisted of finding stationary objects within a 3-D maze. For each test, an object was placed at one of 12 locations (Figure 7.4) within the maze. The subjects had objects placed at each of the locations one time in a preselected order. The view mode alternated between the PPC and GC for a total of 24 tests.

For all tests, there were exactly 33 (6.35%) cases of time expiring for both the PPC and GC. These outliers were removed from the results in the top of Figure 7.5. As the numbers indicate, using the GC, subjects outperformed the PPC for almost all cases. Performance improved by as much as 90.7% for case #3. On average, the subjects found objects in 20.4 seconds with the PPC and 13.3 seconds with the MPI, an improvement of 34.8%.

Case #3 was the case in which using the GC most outperformed using the PPC. This was a particularly easy case for the GC. The extra perspective of the portal-based GC made the object visible from the initial viewpoint and view direction. This required the subjects perform almost no navigation. For the PPC case however, the subjects needed to translate forward slightly and look right to uncover the object.

Case #8 was the only case which showed the PPC outperforming the GC. The GC can sometimes introduce new occlusions into a scene which were not

originally there. As shown in Figure 7.6, the change of perspective in the GC in case #8 introduces a new occlusion which delays the subject finding the object.

A second series of tests used the same 3-D maze and the first 8 object locations from Figure 7.4. Here the object location only served as a starting point for moving objects. As time progressed, the object would translate around the map in random directions. Like previously, the view mode alternated between the PPC and GC resulting in a total of 16 tests.

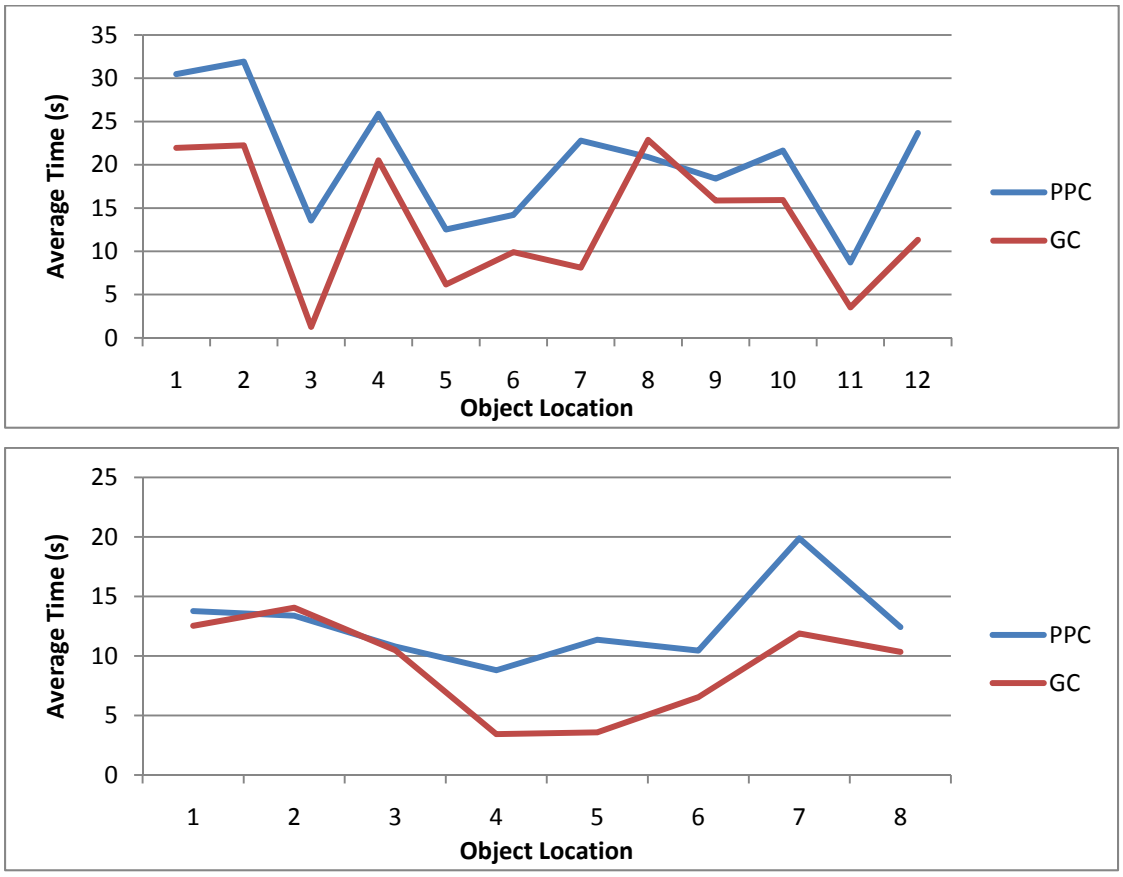


Figure 7.5. User performance in finding objects. User performance finding stationary objects (*top*) and dynamic objects (*bottom*) compares the PPC to the GC for each object.

For moving objects, the PPC had a total of 7 (1.90%) outliers (time expired) while the GC only had 1 outlier (0.27%). The bottom of Figure 7.5 compares the

results. For the first 3 cases, the PPC and GC performed virtually identically. For the remaining cases the GC once again outperforms the PPC. For all cases, the subjects found the object in an average of 12.6 seconds with the PPC and 9.11 seconds with the GC for a speedup of 27.8%.

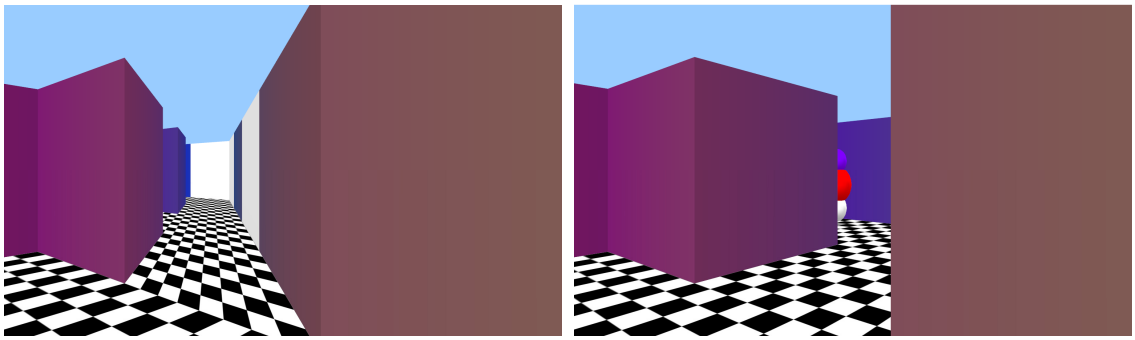


Figure 7.6. Example of a multiperspective image inducing new occlusions. The GC (*left*) has introduced new occlusions not visible with the PPC (*right*) delaying the subject finding the object.

Subjects were tested for multiple sessions to identify whether or not learning played a factor in their ability to use the GC. The average time to find objects is recorded in Table 7.1. For stationary objects, subjects' performance improved consistently for both the GC and PPC. The GC consistently outperformed the PPC on average 34.8%, 30.6%, and 35.4% for sessions 1, 2, and 3, respectively. For the moving objects, the PPC performance improved substantially with every session. The GC performance however did not, implying that subjects' performance was near the peak from the outset.

In virtually all cases, the subjects' performance with the GC was at least as good as or better than the PPC. This should not come as a surprise since the GC is a PPC enhanced with additional viewpoints. The additional viewpoints assist the subject in two valuable ways. First, the preview of upcoming regions reduces the navigation required for an exhaustive search by eliminating empty dead-end hallways more quickly. With a PPC, the subject would need to navigate all the

way up to the hallway to check for a dead-end. With the MPI, the subject simply previewed the hallway and eliminated it without needing to approach it. The second benefit of the additional viewpoints is that the user will see the object earlier. With the PPC the subject navigates all the way to the hallway of interest to look for an object. When using the GC, the subject can simply look one intersection ahead to the upcoming hallway and click on the object without the need for any additional navigation.

Table 7.1. Average time to find objects for subjects participating in three testing sessions.

	Stationary Objects		**Moving Objects**	
	PPC	**GC**	**PPC**	**GC**
Session #1	19.7 s	12.9 s	11.6 s	8.6 s
Session #2	15.7 s	10.9 s	10.8 s	8.5 s
Session #3	14.3 s	9.2 s	9.0 s	8.2 s

7.4.3. Object Counting

The next series tested subjects' ability to count objects. For these tests subjects were shown between 4 and 7 objects for various lengths of time and asked to select the correct number that they counted in the scene. The testing alternated between a comprehensive matrix of PPCs and a comprehensive GC for a total of 20 tests for stationary objects and 20 for moving objects. For stationary objects, subjects selected the correct number of objects 90.2% of the time with the GC and only 43.6% with the PPC. For moving objects, the results showed an accuracy of 92.4% for the GC and only 40.4% for the matrix of PPCs.

The results for stationary objects were tested under a variety of conditions. The first of which varied the amount of time subjects were shown objects (Table 7.2,

left). Not surprisingly, increasing the available time did increase accuracy for both the PPC and GC. Though, the increase in performance for the GC was less pronounced between 5 and 20 seconds (6.1%) than that of the PPC (15.2%) due to the high level of starting accuracy for the GC.

Table 7.2. Accuracy counting stationary objects under various conditions. Results compare user performance for different times (*left*) and duplicate coloring conditions (*right*).

	5 s.	**10 s.**	**20 s.**	**Dup.**	**No Dup.**
PPC	41.3%	37.6%	56.5%	40.9%	45.5%
GC	84.7%	93.0%	90.8%	84.9%	94.4%

Another condition we tested for involved object coloring. For some testing, at least 2 objects in the scene would have the same coloring. Subjects were made aware of the potential for this situation, but were not informed when it was the case. Table 7.2, right, shows the subject performance when duplicate coloring was introduced. For both the PPC and GC, performance went when down with duplicates at a rate essentially proportional to the case without duplicates.

Finally, we tested subjects in multiple sessions (Table 7.3). Using the PPC, subjects' performance went up at a good pace improving with each session. For the GC, performance peaked during the second session. Our interpretation of that result is that for the counting task in particular, the learning required to use the GC is *less* than the learning required for the PPC.

Our results show that the GC significantly outperforms the PPC for the counting task. We see 3 features of the GC which have the most significant contributions to this result. The GC is completely non-redundant. For the PPCs, the redundancy, although chosen to be minimal, still leads objects to be visible multiple times within the matrix of PPCs. That can cause one object to be

incorrectly counted multiple times by subjects. The GC is completely continuous making tracking objects very simple. The lack of continuity between the views of the matrix of PPCs can make it difficult to track objects as they transition between various views. Finally, the layout of the GC is very natural for counting. This reduces the cognitive effort required for the task by allowing the subject to simply read the image left to right counting along the way. The layout of the matrix of PPCs is less natural requiring the subject to bounce their eyes from view to view in unnatural patterns. These features simplify the counting task by transforming the task from one of visual search plus spatial alignment (for tracking and removing duplicate objects) to simply one of visual search.

Table 7.3. Accuracy counting objects for subjects participating in three testing sessions.

	Stationary Objects		**Moving Objects**	
	PPC	**GC**	**PPC**	**GC**
Session #1	38.8%	89.6%	39.6%	90.8%
Session #2	50.4%	98.4%	46.0%	98.0%
Session #3	58.4%	98.4%	47.6%	95.6%

7.4.4. Path Matching

Our path matching experiment was designed to test basic orientation understanding. For each test, we generate a random path with 5 turns. That path was then played back with either a PPC or portal-based GC. Play back lasted between 20 and 40 seconds. The process was repeated 10 times alternating between the PPC and GC. The results showed that users selected the correct path at a rate of 66.7% for the PPC and 69.8% for the GC.

For subjects who participated in 3 sessions, their performance for the PPC was 60.8%, 72.0%, and 78.4% for sessions 1, 2, and 3, respectively. The performance for the GC was 64.0%, 73.6%, and 76.0% for sessions 1, 2, and 3.

These results show that when it comes to the very basic understanding of orientation, the GC and PPC perform similarly. The maze we used only had 90° turns throughout. In the end, this only tested the subjects' ability to determine whether they were going straight, turning left, or turning right. As with all multiperspective techniques, spatial relationships are disturbed. If we had asked the subjects more probing questions, such as guess the distance traveled or the exact angle of a particular turn (given non-90° turns), the performance of the GC would likely have been lower.

7.4.5. Subject Survey

At the conclusion of testing, subjects were surveyed for their opinion on whether the PPC or GC was easier to use for the tasks described. They were asked to respond on a scale of 1 to 5, with the PPC marked as 1, the GC marked as 5, and 3 marked as about the same. The results of the survey seem to align with the numbers collected. For object finding, the average response was 4.5, making the GC the strongly preferred method. For object counting, the average was 4.98 (1 subject selected 4, all others selected 5) making the GC the clear standout. The results for the path matching were more balanced but slightly favoring the PPC at 2.7 probably indicating a higher comfort level with the familiar PPC images.

CHAPTER 8. CONCLUSION

8.1. Summary

We have shown Camera Model Design to be a viable problem solving paradigm in a wide variety of applications. By adhering to the principles of Camera Model Design, we can construct cameras which improve the sampling of 3-D scenes when compared to that of the planar pinhole camera. Removing the planar pinhole camera constraints of convergent rays and uniform sampling allows us to capture samples not visible from a reference viewpoint in non-uniform patterns. Camera models can then be designed for a particular application and adapted to the data they are sampling allowing for the optimal construction to be used. Finally, cameras are carefully designed to have fast projection allowing for interactive rendering using the feed-forward graphics pipeline.

The occlusion cameras work locally to improve sampling by bending rays around the silhouettes of objects. They capture the samples needed to reconstruct a scene from a reference viewpoint and a cluster of nearby viewpoints. All of the occlusion cameras have fast projection operations making them efficient to render. Many occlusion camera applications were presented including using occlusion camera images as a geometry replacement for high-quality rendering effects and volumetric display acceleration, along with as a means of compression for a dense set of photographs. These are just a small sampling of the applications which could benefit from using the occlusion cameras.

The graph cameras allow for comprehensive views deep into heavily occluded scenes. They do so by bending, splitting, and merging a planar pinhole camera

frustum. The graph cameras have efficient projection functions for interactive rendering. Some of the applications for the graph camera include enhanced surveillance, 3-D scene summarization, and enhanced dataset exploration.

The general pinhole camera removes the planar pinhole camera constraint of uniform sampling of the image plane. This allows the general pinhole camera to adapt the sampling to correspond with the complexity of the dataset being sampled. The general pinhole camera was also designed with efficient projection function allowing it to be rendered quickly. We demonstrated its application to remote visualization, focus-plus-context visualization, and extreme antialiasing, though versions of the camera model could be applied to any problem involving non-uniformly sampled datasets.

We have also shown that some camera models can be designed to improve human perception. In particular, we have shown that the graph camera can improve user performance in a number of tasks commonly performed on 3-D datasets. These include searching and counting, which showed dramatic improvement over the planar pinhole camera.

Given the evidence presented, I believe I have proven my thesis statement:

> *In order to create images that better sample 3-D scenes, I propose abandoning the constraints of the conventional planar pinhole camera model by no longer requiring that rays be straight, converge, or sample space uniformly. Camera models can then be designed for specific applications and optimized dynamically for each 3-D scene or dataset so as to achieve adequate sampling. At the same time, camera models should also be designed to preserve image computation efficiency in order to support interactive rendering of dynamic scenes.*

8.2. Camera Model Design in Graphics, Visualization, and Vision

Many avenues of future work exist for Camera Model Design. They span the fields of computer graphics, visualization, computer vision, and computer-human interaction.

An intriguing direction of work for the occlusion cameras would be to extend the epipolar occlusion camera to higher dimensions. The epipolar occlusion camera has extended the 1-D viewpoint to a 2-D viewsegment. Further extension of the view to a 3-D view triangle or quadrilateral would allow for a single image to contain an entire lightfield. The view could be further extended to a 4-D view tetrahedron effectively capturing all the data needed for 5-D plenoptic function into a single image. Extending the camera even further to encode time varying data into that image would allow for a single image to contain the data for a 6-D plenoptic function.

Further exploration of the graph camera's application to navigation is important. This includes the need for new, more automatic constructors. For example, a top-down constructor could begin from the current viewpoint and automatically detects regions which are unsampled. The graph camera could then automatically bend, split, or merge the frustum in order to generate a comprehensive view of the scene. An alternative bottom-up constructor could take as input a collection of desired views. It would then need to automatically generate graph camera components which would combine those independent views into a single graph camera.

Another broader avenue of future work for the graph camera involves its application to modeling and artwork. To further develop such applications, new more intuitive graph camera constructors would need to be invented. Then, collaborations with artists would be required to refine the constructors to their needs.

There are many applications in computer vision which depend upon integrating and displaying data from a cloud of sensors (i.e. a large set of cameras). We have already shown the graph camera to be useful in integrating a few fixed position cameras into a single view. Further extension of these techniques to integrating a higher number of cameras with non-fixed positions has potential application in both surveillance and situational awareness applications.

The current implementation of the general pinhole camera only supports one or a few discrete simply shaped regions of variable sampling. Extending the camera model to support more generalized non-uniform sampling would allow for applications in many areas. For example, shadow mapping could benefit from such an approach with higher sampling rates at object silhouettes. This type of constructor could also be used to compress height maps for remote visualization. Regions with high variation in height could receive a greater number of samples than other regions. This approach could be extended to higher-dimensional datasets as well. For example, large 3-D volumetric data could be compressed in a similar method to improve volume rendering performance.

There are also domain specific challenges for Camera Model Design. Almost certainly, a camera model designed for use on one type of domain specific data will not be optimal for data from other domains. For example, a camera model designed for searching through a microbiologist's dataset will not have any use to a civil engineer searching within a finite element dataset. Instead, a new camera model should be designed or an existing camera model modified for each domain specific application.

8.3. Camera Model Design and Perception

We have shown that Camera Model Design can be used to improve user performance on a few generic tasks. A more detailed understanding of the

influence of perception on Camera Model Design remains one of the most open of the problems we have addressed.

The quantity and effects of the distortion introduced in multiperspective images remains a significant unknown. Within the context of the graph camera, simply testing the ability of a user to estimate the angle between 2 crossing non-perpendicular hallways might prove quite interesting. For tasks where good spatial understanding is required, multiperspective images might never be an acceptable solution. There are however many tasks, like searching and counting, which do not require good spatial understanding in all contexts.

A final path of future work with Camera Model Design involves perceptually driven camera models. Much the way a magician uses slight-of-hand to cause the audience to see only what he intends for them to see, we can use Camera Model Design in the same way. Camera models can be designed to change the user's understanding of the space or automatically focus the user's attention on key pieces of data.

LIST OF REFERENCES

LIST OF REFERENCES

[1] ADELSON, E. H. AND BERGEN, J. R. The plenoptic function and the elements of early vision. In *Computational Models of Visual Processing* (1991), MIT Press, pp. 3–20.

[2] AGARWALA, A., AGRAWALA, M., COHEN, M., SALESIN, D., AND SZELISKI, R. Photographing long scenes with multi-viewpoint panoramas. *ACM Transactions on Graphics 25*, 3 (2006), 853–861.

[3] AGRAWALA, M., ZORIN, D., AND MUNZNER, T. Artistic multiprojection rendering. In *Rendering Techniques 2000: Proceedings of the 11th Eurographics Workshop on Rendering Techniques* (London, UK, 2000), Springer-Verlag, pp. 125–136.

[4] AILA, T. AND LAINE, S. Alias-free shadow maps. In *Rendering Techniques 2004: Proceedings of the 15th Eurographics Workshop on Rendering Techniques* (2004), pp. 161–166.

[5] ALIAGA, D. AND CARLBOM, I. A spatial image hierarchy for compression in image-based-rendering. In *ICIP 2005: IEEE International Conference on Image Processing* (2005), vol. 1, pp. I – 609–12.

[6] ALIAGA, D., COHEN, J., WILSON, A., BAKER, E., ZHANG, H., ERIKSON, C., HOFF, K., HUDSON, T., STUERZLINGER, W., BASTOS, R., WHITTON, M., BROOKS, F., AND MANOCHA, D. MMR: An interactive massive model rendering system using geometric and image-based acceleration. In *I3D '99: Proceedings of the Symposium on Interactive 3D Graphics* (1999), pp. 199–206.

[7] ALIAGA, D. G. AND CARLBOM, I. Plenoptic stitching: A scalable method for reconstructing 3D interactive walk throughs. In *SIGGRAPH '01: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2001), ACM, pp. 443–450.

[8] ALIAGA, D. G., FUNKHOUSER, T., YANOVSKY, D., AND CARLBOM, I. Sea of images: A dense sampling approach for rendering large indoor environments. *IEEE Computer Graphics and Applications 23* (2003), 22–30.

[9] ATI, AMD CORPORATION, 2010.

[10] BAHMUTOV, G., POPESCU, V., AND SACKS, E. Depth enhanced panoramas. In *VIS '04: Proceedings of the Conference on Visualization* (Los Alamitos, CA, USA, 2004), IEEE Computer Society Press.

[11] BARES, W. H. AND LESTER, J. C. Intelligent multi-shot visualization interfaces for dynamic 3D worlds. In *IUI '99: Proceedings of the 4th International Conference on Intelligent User Interfaces* (New York, NY, USA, 1999), ACM, pp. 119–126.

[12] BAUDISCH, P., GOOD, N., AND STEWART, P. Focus plus context screens: Combining display technology with visualization techniques. In *UIST '01: Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology* (New York, NY, USA, 2001), ACM, pp. 31–40.

[13] BAVOIL, L., CALLAHAN, S. P., SCHEIDEGGER, C. E., VO, H. T., CROSSNO, P. J., SILVA, C. T., AND FREIRE, J. Vistrails: Enabling interactive multiple-view visualizations. *VIS '05: Proceedings of the Conference on Visualization* (2005), 18.

[14] BAVOIL, L., SAINZ, M., AND DIMITROV, R. Image-space horizon-based ambient occlusion. In *SIGGRAPH '08: ACM SIGGRAPH 2008 Talks* (New York, NY, USA, 2008), ACM.

[15] BETHEL, W., TIERNEY, B., LEE, J., GUNTER, D., AND LAU, S. Using high-speed WANs and network data caches to enable remote and distributed visualization. In *Supercomputing '00: Proceedings of the ACM/IEEE Conference on Supercomputing* (Washington, DC, USA, 2000), IEEE Computer Society Press, p. 28.

[16] BJORKE, K. *Image-based lighting, GPU Gems*. nVidia Corp., 2004.

[17] BLINN, J. F. AND NEWELL, M. E. Texture and reflection in computer generated images. *Communications of the ACM 19*, 10 (1976), 542–547.

[18] BRUCKNER, S. AND GROLLER, M. E. Exploded views for volume data. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 1077–1084.

[19] BUEHLER, C., BOSSE, M., MCMILLAN, L., GORTLER, S., AND COHEN, M. Unstructured lumigraph rendering. In *SIGGRAPH '01: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2001), ACM, pp. 425–432.

[20] BURNS, M. AND FINKELSTEIN, A. Adaptive cutaways for comprehensible rendering of polygonal scenes. In *SIGGRAPH Asia '08: ACM SIGGRAPH Asia 2008 Papers* (New York, NY, USA, 2008), ACM, pp. 1–7.

[21] CALLAHAN, S. P., BAVOIL, L., PASCUCCI, V., AND SILVA, C. T. Progressive volume rendering of large unstructured grids. *IEEE Transactions on Visualization and Computer Graphics 12* (2006), 1307–1314.

[22] CALLAHAN, S. P., COMBA, J. L. D., SHIRLEY, P., AND SILVA, C. T. Interactive rendering of large unstructured grids using dynamic level-of-detail. In *VIS '05: Proceedings of the Conference on Visualization* (Los Alamitos, CA, USA, 2005), IEEE Computer Society Press, p. 26.

[23] CARPENDALE, M. S. T., CARPENDALE, T., COWPERTHWAITE, D. J., AND FRACCHIA, F. D. Distortion viewing techniques for 3-dimensional data. In *INFOVIS '96: Proceedings of the IEEE Symposium on Information Visualization* (Washington, DC, USA, 1996), IEEE Computer Society Press, p. 46.

[24] CHAN, B. AND WANG, W. Geocube – GPU accelerated real-time rendering of transparency and translucency. *The Visual Computer 21*, 8-10 (Sept. 2005), 579–590.

[25] CHANG, C.-F., BISHOP, G., AND LASTRA, A. LDI tree: A hierarchical representation for image-based rendering. In *SIGGRAPH '99: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 291–298.

[26] CHEN, S. E. Quicktime VR: An image-based approach to virtual environment navigation. In *SIGGRAPH '95: Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1995), ACM, pp. 29–38.

[27] DARKEN, R. P. AND SIBERT, J. L. Wayfinding strategies and behaviors in large virtual worlds. In *CHI '96: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 1996), ACM, pp. 142–149.

[28] DEBEVEC, P., YU, Y., AND BOSHOKOV, G. Efficient view-dependent image-based rendering with projective texture-mapping. Tech. rep., Berkeley, CA, USA, 1998.

[29] DECKERT, C. Eye design book, 2001.

[30] DÉCORET, X., DURAND, F., SILLION, F. X., AND DORSEY, J. Billboard clouds for extreme model simplification. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers* (New York, NY, USA, 2003), ACM, pp. 689–696.

[31] DEGENER, P. AND KLEIN, R. A variational approach for automatic generation of panoramic maps. *ACM Transactions on Graphics 28*, 1 (2009), 1–14.

[32] DEGENER, P., SCHNABEL, R., SCHWARTZ, C., AND KLEIN, R. Effective visualization of short routes. *IEEE Transactions on Visualization and Computer Graphics 14* (2008), 1452–1458.

[33] DELLAERT, F., SEITZ, S., THORPE, C., AND THRUN, S. Structure from motion without correspondence. In *CVPR '00: IEEE Conference on Computer Vision and Pattern Recognition* (2000), vol. 2, pp. 557–564.

[34] DIEPSTRATEN, J., WEISKOPF, D., AND ERTL, T. Transparency in interactive technical illustrations. *Computer Graphics Forum 21* (2002).

[35] DIEPSTRATEN, J., WEISKOPF, D., AND ERTL, T. Interactive cutaway illustrations. In *Proceedings of Eurographics 2003, Computer Graphics Forum* (2003), pp. 523–532.

[36] DIMENSION TECHNOLOGIES, INC., 2010.

[37] ELMQVIST, N. BalloonProbe: Reducing occlusion in 3D using interactive space distortion. In *VRST '05: Proceedings of the ACM Symposium on Virtual Reality Software and Technology* (New York, NY, USA, 2005), ACM, pp. 134–137.

[38] ELMQVIST, N., HENRY, N., RICHE, Y., AND FEKETE, J.-D. Melange: Space folding for multi-focus interaction. In *CHI '08: Proceeding of the 26th Annual SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2008), ACM, pp. 1333–1342.

[39] ELMQVIST, N. AND TSIGAS, P. A taxonomy of 3D occlusion management for visualization. *IEEE Transactions on Visualization and Computer Graphics 14*, 5 (2008), 1095–1109.

[40] ELMQVIST, N., TUDOREANU, M. E., AND TSIGAS, P. Evaluating motion constraints for 3D wayfinding in immersive and desktop virtual environments. In *CHI '08: Proceeding of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2008), ACM, pp. 1769–1778.

[41] ENGINE BLOCK VOLUME DATASET, STANFORD UNIVERSITY, 2010.

[42] EVERITT, C. Interactive order-independent transparency. Tech. rep., nVidia Corp., 2002.

[43] FAVALORA, G. E., NAPOLI, J., HALL, D. M., DORVAL, R. K., GIOVINCO, M., RICHMOND, M. J., AND CHUN, W. S. 100-million-voxel volumetric display. In *Proceedings of SPIE: Cockpit Displays IX: Displays for Defense Applications* (2002), pp. 300–312.

[44] FEINER, S. K. AND SELIGMANN, D. D. Cutaways and ghosting: Satisfying visibility constraints in dynamic 3D illustrations. *The Visual Computer 8*, 5-6 (Sept. 1992), 292–302.

[45] FERNANDO, R., FERNANDEZ, S., BALA, K., AND GREENBERG, D. P. Adaptive shadow maps. In *SIGGRAPH '01: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2001), ACM, pp. 387–390.

[46] FITZMAURICE, G., MATEJKA, J., MORDATCH, I., KHAN, A., AND KURTENBACH, G. Safe 3D navigation. In *I3D '08: Proceedings of the Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2008), ACM, pp. 7–15.

[47] FUJII, T., KIMOTO, T., AND TANIMOTO, M. A new flexible acquisition system of ray-space data for arbitrary objects. *IEEE Transactions on Circuits and Systems for Video Technology 10*, 2 (Mar. 2000), 218 –224.

[48] FURNAS, G. W. Generalized fisheye views. *SIGCHI Bulletin 17*, 4 (1986), 16–23.

[49] GAO, J. AND SHEN, H.-W. Parallel view-dependent isosurface extraction using multi-pass occlusion culling. In *IEEE Symposium on Parallel and Large-Data Visualization and Graphics* (2001), IEEE Computer Society Press, pp. 67–152.

[50] GERSHO, A. AND GRAY, R. M. *Vector quantization and signal compression*. Kluwer Academic Publishers, Norwell, MA, USA, 1991.

[51] GLASSNER, A. S. *An introduction to ray tracing*. Academic Press Ltd., London, UK, UK, 1989.

[52] GLEICHER, M. AND WITKIN, A. Through-the-lens camera control. *SIGGRAPH Computer Graphics 26*, 2 (1992), 331–340.

[53] GORTLER, S. J., GRZESZCZUK, R., SZELISKI, R., AND COHEN, M. F. The lumigraph. In *SIGGRAPH '96: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1996), ACM, pp. 43–54.

[54] GOTZ, D., MAYER-PATEL, K., AND MANOCHA, D. IRW: An incremental representation for image-based walkthroughs. In *MULTIMEDIA '02: Proceedings of the 10th ACM International Conference on Multimedia* (New York, NY, USA, 2002), ACM, pp. 67–76.

[55] GROSSBERG, M. D. AND NAYAR, S. K. A general imaging model and a method for finding its parameters. In *ICCV '01: IEEE International Conference on Computer Vision* (Los Alamitos, CA, USA, 2001), vol. 2, IEEE Computer Society, p. 108.

[56] GUPTA, R. AND HARTLEY, R. Linear pushbroom cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence 19*, 9 (Sept. 1997), 963–975.

[57] GYULASSY, A. AND NATARAJAN, V. Topology-based simplification for feature extraction from 3D scalar fields. In *VIS '05: Proceedings of the Conference on Visualization* (2005), IEEE Computer Society Press, pp. 535 – 542.

[58] HACHET, M., DECLE, F., KNODEL, S., AND GUITTON, P. Navidget for easy 3D camera positioning from 2D inputs. *3D User Interfaces* (2008), 83–89.

[59] HALLE, M. Autostereoscopic displays and computer graphics. *ACM SIGGRAPH Computer Graphics 31*, 2 (1997), 58–62.

[60] HANSON, A. J. AND WERNERT, E. A. Constrained 3D navigation with 2D controllers. In *VIS '97: Proceedings of the 8th Conference on Visualization* (Los Alamitos, CA, USA, 1997), IEEE Computer Society Press, p. 175ff.

[61] HOPPE, H. Progressive meshes. In *SIGGRAPH '96: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1996), ACM, pp. 99–108.

[62] HUANG, J. AND CARTER, M. B. Interactive transparency rendering for large CAD models. *IEEE Transactions on Visualization and Computer Graphics 11*, 5 (2005), 584–595.

[63] ISAKSEN, A., MCMILLAN, L., AND GORTLER, S. J. Dynamically reparameterized light fields. In *SIGGRAPH '00: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 297–306.

[64] ISENBURG, M., LINDSTROM, P., AND SNOEYINK, J. Streaming compression of triangle meshes. In *SGP '05: Proceedings of the 3rd Eurographics Symposium on Geometry Processing* (Aire-la-Ville, Switzerland, Switzerland, 2005), Eurographics Association, p. 111.

[65] ISO/IEC INTERNATIONAL STANDARD 15444. Final committee draft, March 2000.

[66] IVES, H. E. A camera for making parallax panoramagrams. *Journal of the Optical Society of America 17*, 6 (1928), 435–437.

[67] JANKUN-KELLY, T. AND MA, K.-L. Visualization exploration and encapsulation via a spreadsheet-like interface. *IEEE Transactions on Visualization and Computer Graphics 7*, 3 (2001), 275–287.

[68] JANKUN-KELLY, T. J., MA, K. L., AND GERTZ, M. A model for the visualization exploration process. In *VIS '02: Proceedings of the Conference on Visualization* (Washington, DC, USA, 2002), IEEE Computer Society, pp. 323–330.

[69] JOHNSON, C. R., MOORHEAD, R., MUNZNER, T., PFISTER, H., RHEINGANS, P., AND YOO, T. S. NIH-NSF visualization research challenges report. IEEE Press, 2006.

[70] JOHNSON, G. S., LEE, J., BURNS, C. A., AND MARK, W. R. The irregular z-buffer: Hardware acceleration for irregular data structures. *ACM Transactions on Graphics 24*, 4 (2005), 1462–1482.

[71] KHAN, A., KOMALO, B., STAM, J., FITZMAURICE, G., AND KURTENBACH, G. Hovercam: Interactive 3D navigation for proximal object inspection. In *I3D '05: Proceedings of the Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2005), ACM, pp. 73–80.

[72] KLEIN, A. W., SLOAN, P.-P. J., FINKELSTEIN, A., AND COHEN, M. F. Stylized video cubes. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2002), ACM, pp. 15–22.

[73] KLIMECK, G., MCLENNAN, M., BROPHY, S. P., ADAMS III, G. B., AND LUNDSTROM, M. S. nanoHUB.org: Advancing education and research in nanotechnology. *Computing in Science and Engineering 10*, 5 (2008), 17–23.

[74] KNÖDEL, S., HACHET, M., AND GUITTON, P. Navidget for immersive virtual environments. In *VRST '08: Proceedings of the ACM Symposium on Virtual Reality Software and Technology* (New York, NY, USA, 2008), ACM, pp. 47–50.

[75] KOLLER, D., TURITZIN, M., LEVOY, M., TARINI, M., CROCCIA, G., CIGNONI, P., AND SCOPIGNO, R. Protected interactive 3D graphics via remote rendering. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers* (New York, NY, USA, 2004), ACM, pp. 695–703.

[76] KREUSELER, M., NOCKE, T., AND SCHUMANN, H. A history mechanism for visual data mining. In *INFOVIS '04: IEEE Symposium on Information Visualization* (2004), pp. 49–56.

[77] KUTHIRUMMAL, S. AND NAYAR, S. K. Multiview radial catadioptric imaging for scene capture. *ACM Transactions on Graphics 25*, 3 (2006), 916–923.

[78] LAMBERTI, F. AND SANNA, A. A streaming-based solution for remote visualization of 3D graphics on mobile devices. *IEEE Transactions on Visualization and Computer Graphics 13*, 2 (Mar.-Apr. 2007), 247 –260.

[79] LAMPING, J. AND RAO, R. The hyperbolic browser: A focus+context technique for visualizing large hierarchies. *Journal of Visual Languages & Computing 7*, 1 (1996), 33 – 55.

[80] LAND, M. AND NILSSON, D.-E. *Animal Eyes*. Oxford Animal Biology series, Oxford University Press, 2002.

[81] LANEY, D., BREMER, P.-T., MASCARENHAS, A., MILLER, P., AND PASCUCCI, V. Understanding the structure of the turbulent mixing layer in hydrodynamic instabilities. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (Sept.-Oct. 2006), 1053 –1060.

[82] LE GALL, D. MPEG: A video compression standard for multimedia applications. *Communications of the ACM 34*, 4 (1991), 46–58.

[83] LEVOY, M. Spreadsheets for images. In *SIGGRAPH '94: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1994), ACM, pp. 139–146.

[84] LEVOY, M. AND HANRAHAN, P. Light field rendering. In *SIGGRAPH '96: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1996), ACM, pp. 31–42.

[85] LI, W., AGRAWALA, M., CURLESS, B., AND SALESIN, D. Automated generation of interactive 3D exploded view diagrams. *ACM Transactions on Graphics 27*, 3 (2008), 1–7.

[86] LI, W., RITTER, L., AGRAWALA, M., CURLESS, B., AND SALESIN, D. Interactive cutaway illustrations of complex 3D models. In *SIGGRAPH '07: ACM SIGGRAPH 2007 Papers* (New York, NY, USA, 2007), ACM, p. 31.

[87] LIGHTSPACE TECHNOLOGIES, 2010.

[88] LIPPERT, L., GROSS, M. H., AND KURMANN, C. Compression domain volume rendering for distributed environments. In *Proceedings of Eurographics '97* (1997), pp. 95–107.

[89] LIVNAT, Y., PARKER, S. G., AND JOHNSON, C. R. Fast isosurface extraction methods for large image data sets. *Handbook of Medical Imaging* (2000), 731–745.

[90] LOUVAIN, U. C. D., AND KUSAK, E. Desargues theorem in projective 3-space.

[91] LUCENTE, M. Interactive three-dimensional holographic displays: Seeing the future in depth. *ACM SIGGRAPH Computer Graphics 31*, 2 (1997), 63–67.

[92] LUEBKE, D., WATSON, B., COHEN, J. D., REDDY, M., AND VARSHNEY, A. *Level of Detail for 3D Graphics*. Elsevier Science Inc., New York, NY, USA, 2002.

[93] LUKE, E. AND HANSEN, C. Semotus visum: A flexible remote visualization framework. In *VIS '02: Proceedings of the Conference on Visualization* (2002), IEEE Computer Society Press, pp. 61–68.

[94] MA, K.-L. AND CAMP, D. M. High performance visualization of time-varying volume data over a wide-area network status. In *Supercomputing '00: Proceedings of the 2000 ACM/IEEE Conference on Supercomputing* (Washington, DC, USA, 2000), IEEE Computer Society Press, p. 29.

[95] MACIEL, P. W. C. AND SHIRLEY, P. Visual navigation of large environments using textured clusters. In *I3D '95: Proceedings of the 1995 Symposium on Interactive 3D Graphics* (New York, NY, USA, 1995), ACM, p. 95ff.

[96] MACKINLAY, J. D., CARD, S. K., AND ROBERTSON, G. G. Rapid controlled movement through a virtual 3D workspace. In *SIGGRAPH '90: Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1990), ACM, pp. 171–176.

[97] MAGNOR, M. AND GIROD, B. Data compression for light field rendering. *IEEE Transactions on Circuits and Systems for Video Technology*, 10 (2000), 338–343.

[98] MARK, W. R., MCMILLAN, L., AND BISHOP, G. Post-rendering 3D warping. In *I3D '97: Proceedings of the 1997 Symposium on Interactive 3D Graphics* (New York, NY, USA, 1997), ACM, p. 7ff.

[99] MATUSIK, W. AND PFISTER, H. 3D TV: A scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers* (New York, NY, USA, 2004), ACM.

[100] MAX, N. AND OHSAKI, K. Rendering trees from precomputed z-buffer views. In *Rendering Techniques '95: Proceedings of the 5th Eurographics Workshop on Rendering Techniques* (1995), pp. 45–54.

[101] MCCRAE, J., MORDATCH, I., GLUECK, M., AND KHAN, A. Multiscale 3D navigation. In *I3D '09: Proceedings of the Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2009), ACM, pp. 7–14.

[102] MCMILLAN, L. AND BISHOP, G. Plenoptic modeling: An image-based rendering system. In *SIGGRAPH '95: Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1995), ACM, pp. 39–46.

[103] MEI, C., POPESCU, V., AND SACKS, E. The occlusion camera. In *Proceedings of Eurographics 2005, Computer Graphics Forum* (2005), vol. 24, pp. 139–143.

[104] MINE, M. R., BROOKS, JR., F. P., AND SEQUIN, C. H. Moving objects in space: Exploiting proprioception in virtual-environment interaction. In *SIGGRAPH '97: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1997), ACM Press/Addison-Wesley Publishing Co., pp. 19–26.

[105] MO, Q., POPESCU, V., AND WYMAN, C. The soft shadow occlusion camera. In *PG '07: Proceedings of the 15th Pacific Conference on Computer Graphics and Applications* (Washington, DC, USA, 2007), IEEE Computer Society, pp. 189–198.

[106] NAYAR, S. K. Catadioptric omnidirectional camera. In *CVPR '97: IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (Washington, DC, USA, 1997), IEEE Computer Society Press, pp. 482–488.

[107] NOMURA, Y., ZHANG, L., AND NAYAR, S. Scene collages and flexible camera arrays. In *Rendering Techniques 2007: Proceedings of the Eurographics Symposium on Rendering* (Jun. 2007).

[108] NVIDIA CORPORATION, 2010.

[109] NYE, A. *X Protocol Reference Manual*. Thomson Learning, 1994.

[110] OFEK, E. AND RAPPOPORT, A. Interactive reflections on curved objects. In *SIGGRAPH '98: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1998), ACM, pp. 333–342.

[111] OLANO, M. AND GREER, T. Triangle scan conversion using 2D homogeneous coordinates. In *HWWS '97: Proceedings of the ACM SIGGRAPH/Eurographics Workshop on Graphics Hardware* (New York, NY, USA, 1997), ACM, pp. 89–95.

[112] OLIVEIRA, M. M., BISHOP, G., AND MCALLISTER, D. Relief texture mapping. In *SIGGRAPH '00: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 359–368.

[113] OPENGL VIZSERVER 3.0, SILICON GRAPHICS, INC. Application-transparent remote interactive visualization and collaboration, 2003.

[114] PAJDLA, T. Geometry of two-slit camera. Tech. Rep. CTU–CMP–2002–02, Czech Technical University, 2002.

[115] PERLIN, K., PAXIA, S., AND KOLLIN, J. S. An autostereoscopic display. In *SIGGRAPH '00: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 319–326.

[116] PERSPECTA, ACTUALITY SYSTEMS, 2010.

[117] PESCO, S., LINDSTROM, P., PASCUCCI, V., AND SILVA, C. T. Implicit occluders. In *IEEE Symposium on Volume Visualization and Graphics* (Los Alamitos, CA, USA, 2004), IEEE Computer Society Press, pp. 47–54.

[118] PETER, I. AND STRASSER, W. The wavelet stream: Interactive multi resolution light field rendering. In *Rendering Techniques 2001: Proceedings of the 12th Eurographics Workshop on Rendering Techniques* (2001), p. 127–138.

[119] PFISTER, H., ZWICKER, M., VAN BAAR, J., AND GROSS, M. Surfels: Surface elements as rendering primitives. In *SIGGRAPH '00: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 335–342.

[120] PHARR, M. AND GREEN, S. *Ambient Occlusion, GPU Gems*. 2004.

[121] PIERCE, J. S., FORSBERG, A. S., CONWAY, M. J., HONG, S., ZELEZNIK, R. C., AND MINE, M. R. Image plane interaction techniques in 3D immersive environments. In *I3D '97: Proceedings of the Symposium on Interactive 3D Graphics* (New York, NY, USA, 1997), ACM, p. 39ff.

[122] PIETRIGA, E. AND APPERT, C. Sigma lenses: Focus-context transitions combining space, time and translucence. In *CHI '08: Proceeding of the 26th Annual SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2008), ACM, pp. 1343–1352.

[123] PLUMLEE, M. AND WARE, C. An evaluation of methods for linking 3D views. In *I3D '03: Proceedings of the Symposium on Interactive 3D Graphics* (New York, NY, USA, 2003), ACM, pp. 193–201.

[124] POLICARPO, F. AND OLIVEIRA, M. M. Relief mapping of non-height-field surface details. In *I3D '06: Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2006), ACM, pp. 55–62.

[125] POLICARPO, F., OLIVEIRA, M. M., AND COMBA, JO A. L. D. Real-time relief mapping on arbitrary polygonal surfaces. *ACM Transactions on Graphics 24*, 3 (2005), 935–935.

[126] PONCE, J. What is a camera? In *CVPR '09, IEEE Conference on Computer Vision and Pattern Recognition '09* (2009), pp. 1526 –1533.

[127] POPESCU, V. AND ALIAGA, D. The depth discontinuity occlusion camera. In *I3D '06: Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2006), ACM, pp. 139–143.

[128] POPESCU, V., EYLES, J., LASTRA, A., STEINHURST, J., ENGLAND, N., AND NYLAND, L. The WarpEngine: An architecture for the post-polygonal age. In *SIGGRAPH '00: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 433–442.

[129] POPESCU, V. AND LASTRA, A. The vacuum buffer. In *I3D '01: Proceedings of the 2001 Symposium on Interactive 3D Graphics* (New York, NY, USA, 2001), ACM, pp. 73–76.

[130] POPESCU, V., LASTRA, A., ALIAGA, D., AND DE OLIVEIRA NETO, M. Efficient warping for architectural walkthroughs using layered depth images. In *VIS '98: Proceedings of the Conference on Visualization* (Los Alamitos, CA, USA, 1998), IEEE Computer Society Press, pp. 211–215.

[131] PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T., AND FLANNERY, B. P. *Numerical recipes in C (2nd ed.): The art of scientific computing*. Cambridge University Press, New York, NY, USA, 1992.

[132] QU, H., WANG, H., CUI, W., WU, Y., AND CHAN, M.-Y. Focus+context route zooming and information overlay in 3D urban environments. *IEEE Transactions on Visualization and Computer Graphics 15* (2009), 1547–1554.

[133] RADEMACHER, P. AND BISHOP, G. Multiple-center-of-projection images. In *SIGGRAPH '98: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1998), ACM, pp. 199–206.

[134] RICHARDSON, T., STAFFORD-FRASER, Q., WOOD, K., AND HOPPER, A. Virtual network computing. *IEEE Internet Computing 2*, 1 (Jan. 1998), 33–38.

[135] ROBERTS, J. C. Multiple-View and Multiform Visualization. In *Proceedings of SPIE Visual Data Exploration and Analysis VII* (2000), R. Erbacher, A. Pang, C. Wittenbrink, and J. Roberts, Eds., vol. 3960, IS&T and SPIE, pp. 176–185.

[136] ROGER, D., ASSARSSON, U., AND HOLZSCHUCH, N. Whitted ray-tracing for dynamic scenes using a ray-space hierarchy on the GPU. In *Rendering Techniques 2007: Proceedings of the Eurographics Symposium on Rendering* (Jun. 2007), The Eurographics Association, pp. 99–110.

[137] ROMAN, A., GARG, G., AND LEVOY, M. Interactive design of multi-perspective images for visualizing urban landscapes. In *VIS '04: Proceedings of the Conference on Visualization* (Washington, DC, USA, 2004), IEEE Computer Society Press, pp. 537–544.

[138] ROMÁN, A. AND LENSCH, H. P. A. Automatic multiperspective images. In *Rendering Techniques 2006: Proceedings of the Eurographics Symposium on Rendering* (2006), Eurographics Association, pp. 161–171.

[139] RUSINKIEWICZ, S. AND LEVOY, M. QSplat: A multiresolution point rendering system for large meshes. In *SIGGRAPH '00: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 343–352.

[140] SEILER, L., CARMEAN, D., SPRANGLE, E., FORSYTH, T., ABRASH, M., DUBEY, P., JUNKINS, S., LAKE, A., SUGERMAN, J., CAVIN, R., ESPASA, R., GROCHOWSKI, E., JUAN, T., AND HANRAHAN, P. Larrabee: A many-core x86 architecture for visual computing. *ACM Transactions on Graphics 27*, 3 (2008), 1–15.

[141] SEITZ, S. M. AND KIM, J. Multiperspective imaging. *IEEE Computer Graphics and Applications 23*, 6 (2003), 16–19.

[142] SHADE, J., GORTLER, S., HE, L.-W., AND SZELISKI, R. Layered depth images. In *SIGGRAPH '98: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1998), ACM, pp. 231–242.

[143] SHIN, M. C., TSAP, L. V., AND GOLDGOF, D. B. Towards perceptual interface for visualization navigation of large data sets. In *CVPRW '03: Conference on Computer Vision and Pattern Recognition Workshop* (2003), vol. 5, pp. 48–48.

[144] SHORT, N. The photographic process, the remote sensing tutorial, Apr. 2010.

[145] SHUM, H.-Y. AND HE, L.-W. Rendering with concentric mosaics. In *SIGGRAPH '99: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 299–306.

[146] SKOPIK, A. AND GUTWIN, C. Improving revisitation in fisheye views with visit wear. In *CHI '05: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2005), ACM, pp. 771–780.

[147] SMYTHE, R. H. *Vision in the Animal World*. Macmillan Press, 1975.

[148] STEED, A. Efficient navigation around complex virtual environments. In *VRST '97: Proceedings of the ACM Symposium on Virtual Reality Software and Technology* (New York, NY, USA, 1997), ACM, pp. 173–180.

[149] STEGMAIER, S., DIEPSTRATEN, J., WEILER, M., AND ERTL, T. Widening the remote visualization bottleneck. In *ISPA 2003: Proceedings of the 3rd International Symposium on Image and Signal Processing and Analysis* (2003), vol. 1, pp. 174–179.

[150] STEGMAIER, S., MAGALLÓN, M., AND ERTL, T. A generic solution for hardware-accelerated remote visualization. In *VISSYM '02: Proceedings of the Symposium on Data Visualization* (Aire-la-Ville, Switzerland, Switzerland, 2002), Eurographics Association, p. 87ff.

[151] STERN, A. AND JAVIDI, B. Ray phase space approach for 3-D imaging and 3-D optical data representation. *Journal of Display Technology 1* (2005), 141.

[152] SZELISKI, R. AND SHUM, H.-Y. Creating full view panoramic image mosaics and environment maps. In *SIGGRAPH '97: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1997), ACM Press/Addison-Wesley Publishing Co., pp. 251–258.

[153] SZIRMAY-KALOS, L., ASZÓDI, B., LAZÁNYI, I., AND PREMECZ, M. Approximate ray-tracing on the GPU with distance impostors. *Computer Graphics Forum 24*, 3 (2005), 695–704.

[154] TAKAHASHI, S., YOSHIDA, K., SHIMADA, K., AND NISHITA, T. Occlusion-free animation of driving routes for car navigation systems. *IEEE Transactions on Visualization and Computer Graphics 12* (2006), 1141–1148.

[155] TAN, D. S., ROBERTSON, G. G., AND CZERWINSKI, M. Exploring 3D navigation: Combining speed-coupled flying with orbiting. In *CHI '01: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2001), ACM, pp. 418–425.

[156] TRAUB, A. C. Stereoscopic display using rapid varifocal mirror oscillations. *Applied Optics 6* (1967), 1085–1087.

[157] UNITED STATES GEOLOGICAL SURVEY, 2010.

[158] VETTERLI, M. AND KOVACEVIC, J. *Wavelets and Subband Coding*. Prentice Hall PTR, 1995.

[159] VISUAL MOLECULAR DYNAMICS, UNIVERSITY OF ILLINOIS AT URBANA CHAMPAIGN, 2010.

[160] WALLACE, G. K. The JPEG still picture compression standard. *Communications of the ACM 34*, 4 (1991), 30–44.

[161] WANG, L., ZHAO, Y., MUELLER, K., AND KAUFMAN, A. The magic volume lens: An interactive focus+context technique for volume rendering. In *VIS '05: Proceedings of the Conference on Visualization* (2005), IEEE Computer Society Press, pp. 367–374.

[162] WARE, C. AND FLEET, D. Context sensitive flying interface. In *I3D '97: Proceedings of the Symposium on Interactive 3D Graphics* (New York, NY, USA, 1997), ACM, p. 127ff.

[163] WEXLER, D., GRITZ, L., ENDERTON, E., AND RICE, J. GPU-accelerated high-quality hidden surface removal. In *HWWS '05: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware* (New York, NY, USA, 2005), ACM, pp. 7–14.

[164] WHALEN, D. AND NORMAN, M. L. Competition data set and description. *IEEE Visualization Design Contest* (2008).

[165] WHITTED, T. An improved illumination model for shaded display. *Communications of the ACM 23*, 6 (1980), 343–349.

[166] WILLIAMS, L. Pyramidal parametrics. *ACM SIGGRAPH Computer Graphics 17*, 3 (1983), 1–11.

[167] WILSON, A., MAYER-PATEL, K., AND MANOCHA, D. Spatially-encoded far-field representations for interactive walkthroughs. In *MULTIMEDIA '01: Proceedings of the 9th ACM International Conference on Multimedia* (New York, NY, USA, 2001), ACM, pp. 348–357.

[168] WONG, N., CARPENDALE, S., AND GREENBERG, S. EdgeLens: An interactive method for managing edge congestion in graphs. In *INFOVIS '03: IEEE Symposium on Information Visualization* (Los Alamitos, CA, USA, 2003), IEEE Computer Society Press.

[169] WOOD, D. N., AZUMA, D. I., ALDINGER, K., CURLESS, B., DUCHAMP, T., SALESIN, D. H., AND STUETZLE, W. Surface light fields for 3D photography. In *SIGGRAPH '00: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 287–296.

[170] WOOD, D. N., FINKELSTEIN, A., HUGHES, J. F., THAYER, C. E., AND SALESIN, D. H. Multiperspective panoramas for cel animation. In *SIGGRAPH '97: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1997), ACM Press/Addison-Wesley Publishing Co., pp. 243–250.

[171] WYMAN, C. An approximate image-space approach for interactive refraction. *ACM Transactions on Graphics 24*, 3 (2005), 1050–1053.

[172] YU, J. AND MCMILLAN, L. A framework for multiperspective rendering. In *Rendering Techniques 2004: Proceedings of the 15th Eurographics Workshop on Rendering Techniques* (2004), pp. 61–68.

[173] YU, J. AND MCMILLAN, L. General linear cameras. In *ECCV '04: The 8th European Conference on Computer Vision* (2004), vol. 2, pp. 14–27.

[174] YU, J. AND MCMILLAN, L. Modeling reflections via multiperspective imaging. In *CVPR '05: IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2005), vol. 1, pp. 117–124.

[175] ZELEZNIK, R. AND FORSBERG, A. Unicam: 2D gestural camera controls for 3D environments. In *I3D '99: Proceedings of the Symposium on Interactive 3D Graphics* (New York, NY, USA, 1999), ACM, pp. 169–173.

[176] ZHUKOV, S., IONES, A., AND KRONIN, G. An ambient light illumination model. In *Rendering Techniques '98: Proceedings of the 9th Eurographics Workshop on Rendering Techniques* (1998), pp. 45—56.

[177] ZIV, J. AND LEMPEL, A. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory 23*, 3 (May 1977), 337–343.

[178] ZOMET, A., FELDMAN, D., PELEG, S., AND WEINSHALL, D. Mosaicing new views: The crossed-slits projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence 25*, 6 (June 2003), 741–754.

APPENDICES

Appendix A.

We derive the mapping Q_{k+1} of a point on the image plane of planar pinhole camera *k+1* (PPC_{k+1}) to PPC_0 by first establishing the mapping R_{k+1} between PPC_{k+1} and PPC_k as shown in Figure A.1. Then we show by induction that $Q_{k+1} = R_1R_2\ldots R_{k+1}$. The base case is verified as follows:

$$\begin{bmatrix} u_0 \\ v_0 \\ 1 \end{bmatrix} w_0 = r_1 \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix}, \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} w_1 = r_2 \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} u_0 \\ v_0 \\ 1 \end{bmatrix} w_0 = R_1 \frac{1}{w_1} R_2 \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = \frac{1}{w_1} R_1R_2 \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix}$$

Equation A.1

$$\begin{bmatrix} u_0 \\ v_0 \\ 1 \end{bmatrix} w_0w_1 = R_1R_2 \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix}$$

$$Q_2 = R_1R_2$$

By the induction hypothesis:

$$Q_k = R_1R_2 \ldots R_k$$

$$\begin{bmatrix} u_0 \\ v_0 \\ 1 \end{bmatrix} w_0 = R_1R_2 \ldots R_k \begin{bmatrix} u_k \\ v_k \\ 1 \end{bmatrix}$$

Equation A.2

Using the equations in Figure A.1 we obtain:

$$\begin{bmatrix} u_k \\ v_k \\ 1 \end{bmatrix} w_k = R_{k+1} \begin{bmatrix} u_{k+1} \\ v_{k+1} \\ 1 \end{bmatrix}$$

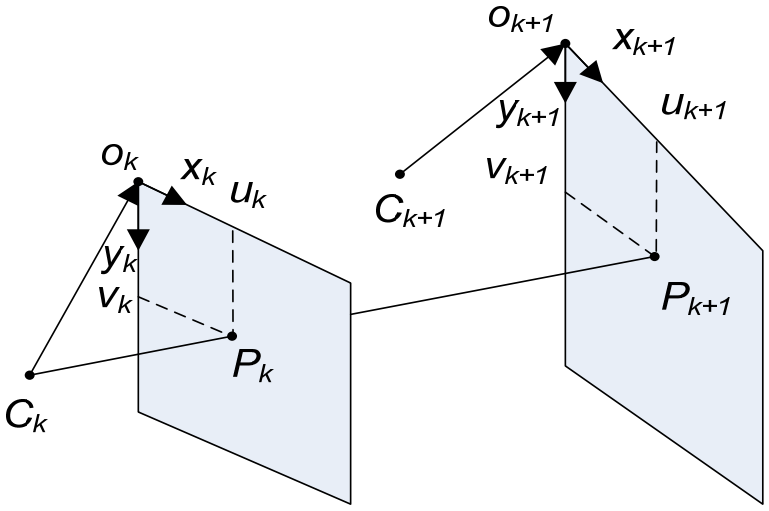
Equation A.3

Combining equations A.2 and A.3 terminates the proof:

$$\begin{bmatrix} u_0 \\ v_0 \\ 1 \end{bmatrix} w_0 = R_1 R_2 \ldots R_k \frac{1}{w_k} R_{k+1} \begin{bmatrix} u_{k+1} \\ v_{k+1} \\ 1 \end{bmatrix}$$

Equation A.4

$$\begin{bmatrix} u_0 \\ v_0 \\ 1 \end{bmatrix} w_0 w_k = R_1 R_2 \ldots R_k R_{k+1} \begin{bmatrix} u_{k+1} \\ v_{k+1} \\ 1 \end{bmatrix}$$



$$P_{k+1} = [x_{k+1} \quad y_{k+1} \quad (C_{k+1} + o_{k+1}] \begin{bmatrix} u_{k+1} \\ v_{k+1} \\ 1 \end{bmatrix}$$

$$P_{k+1} = C_k + [x_k \quad y_k \quad o_k] \begin{bmatrix} u_k \\ v_k \\ 1 \end{bmatrix} w_k$$

$$R_{k+1} = [x_k \quad y_k \quad o_k]^{-1} [x_{k+1} \quad y_{k+1} \quad (C_{k+1} - C_k + o_{k+1})]$$

$$\begin{bmatrix} u_k \\ v_k \\ 1 \end{bmatrix} w_k = R_{k+1} \begin{bmatrix} u_{k+1} \\ v_{k+1} \\ 1 \end{bmatrix}$$

Figure A.1. Derivation of mapping PPC_{k+1} and PPC_k with centers of projection C_{k+1} and C_k. Vectors *x* and *y* give the row and column direction and are one pixel in width and one pixel in length, respectively. Vector *o* points from the COP to the top left corner of the image. Point P_{k+1} on the image plane of PPC_{k+1} is mapped to point P_k on the image plane of PPC_k through matrix R_{k+1}.

Appendix B.

We first show that the modified arcs are conic sections that connect the component rays from the transitioned PPCs with C^1 continuity. Consider Figure B.1 that shows the modified rays of the CRC model in a single epipolar plane through C_0 and C_1. T_n, S_n, and Q_n are the control points of the Bézier arc b that aids with the projection of all points in the epipolar plane. Triangles $T_nS_nQ_n$ and $T_kS_kQ_k$ satisfy Desargues' Theorem [90]. Since the intersections of planes t_0, t_1, and t_2 with the epipolar plane are concurrent, all such triangles are perspective, as are triangles $P_nS_nQ_n$ and $P_kS_kQ_k$. Here, A is the center and line C_0C_1 is the axis of perspectivity. By construction, lines Q_kP_k intersect on the axis of perspectivity in a common point R_k and therefore points P_k are also perspective.

Now the Bézier curve b is the projected intersection of a quadratic cone in 3-D, with vertex A, and a plane through a line that projects onto the axis of perspectivity. Since triangles $T_kS_kP_k$ are perspective, the other transition curves are also conic sections. Again in 3-D, the plane containing lines AT_n and AS_n is tangent to the cone, in a line projecting onto line AT_n and so the conic arcs are all C^1 continuous with the lines through C_0. Likewise, lines through C_1 are C^1 continuous with the conic arcs on the other end.

Having established that the transition curves are tangent continuous and are conics, we now establish that different transition arcs cannot intersect in the transition region.

Let $P_kS_kQ_k$ and $P_jS_jQ_j$ be two triangles defining the intersecting transition curves. Since the control points P_kP_j and Q_kQ_j are distinct, the intersection must be in the interior of the arcs, say at X. The lines Q_kX and Q_jX are distinct and intersect in X. Thus they do not intersect on the line C_0C_1 and so the point X is not perspective on the two curves, yet the line XA establishes projective

correspondence. Hence the two arcs cannot intersect in the interior. An alternative argument is as follows. There must be perspective points Y_k and Y_l on the intersecting curves such that X is on opposite sides of the line Y_kY_lA that establishes the correspondence. Since this is impossible, there cannot be an intersection in the interior of the arcs either.

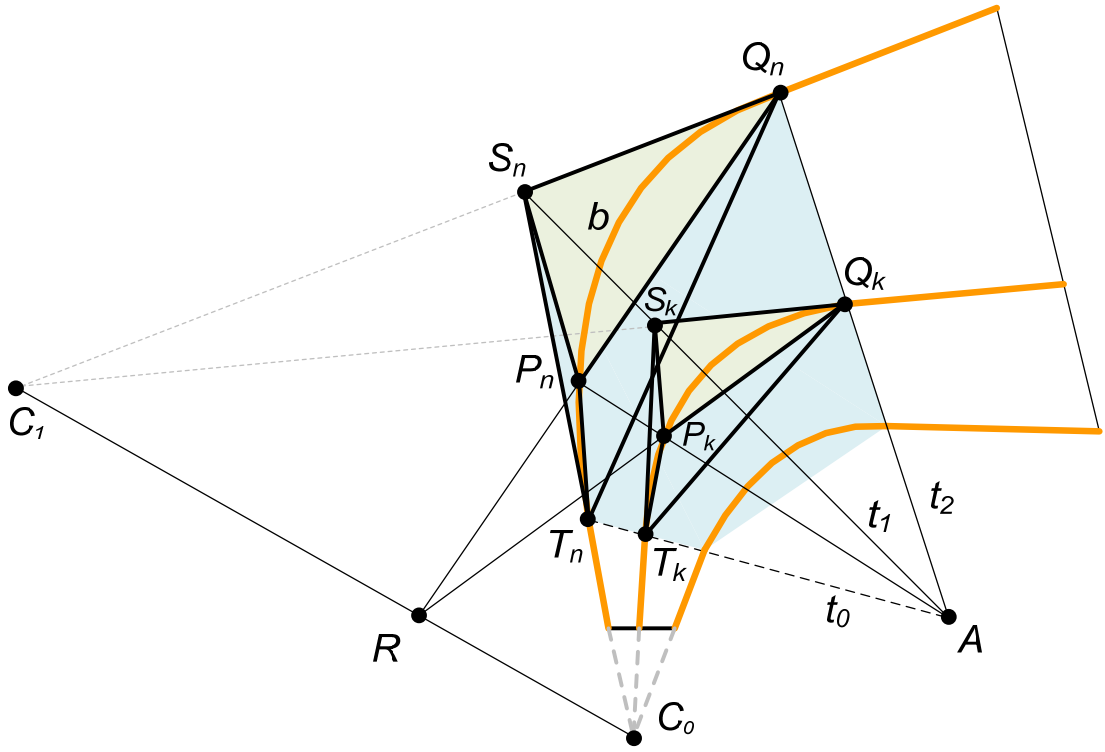


Figure B.1. Modified curved ray camera model.

VITA

VITA

Paul Rosen received his B.S., M.S., and Ph.D. from the Computer Science Department of Purdue University. His research interests lie primarily in the application of Camera Model Design to computer graphics and visualization, but also to computer vision and computer-human interaction.

In addition to his dissertation project in Camera Model Design, Dr. Rosen has participated in a wide variety of projects which include perception of 3-D display imagery, urban modeling and visualization, and graphics hardware assisted constraint solving.

Dr. Rosen also participated in a project focused on high-fidelity visualization of large scale simulations. As part of the project he was a key member of the team which modeled, simulated, and visualized the September 11, 2001 attack on the World Trade Center North Tower. The video produced for the simulation generated significant publicity for Purdue University. The video, in addition to appearing on many news and educational television programs, has been downloaded over 9 million times.

After the completion of his Ph.D., Dr. Rosen joined the Scientific Computing Institute at the University of Utah as a Post Doctoral Research Associate.