2008

The Graph Camera

Paul Rosen

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

Nicoletta Adamo-Villani

Report Number:
08-005

THE GRAPH CAMERA

Paul Rosen
Voicu Popescu
Nicoletta Adamo-Villani

CSD TR #08-005
January 2008

The Graph Camera

Paul Rosen, Voicu Popescu, Nicoletta Adamo-Villani

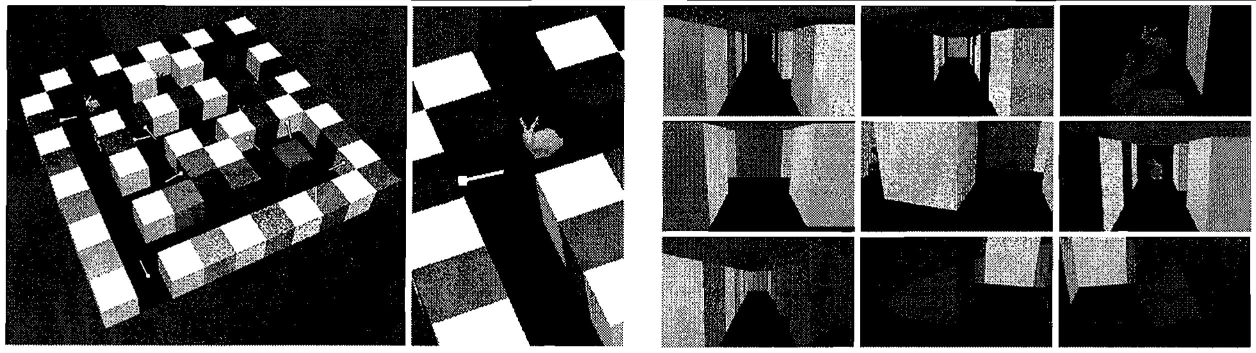


Figure 1 Maze (*left*) rendered in parallel with 9 cameras (*right*). The camera placement is indicated with white pins. The matrix of images does not provide complete coverage and suffers from discontinuities across boundaries of individual images.

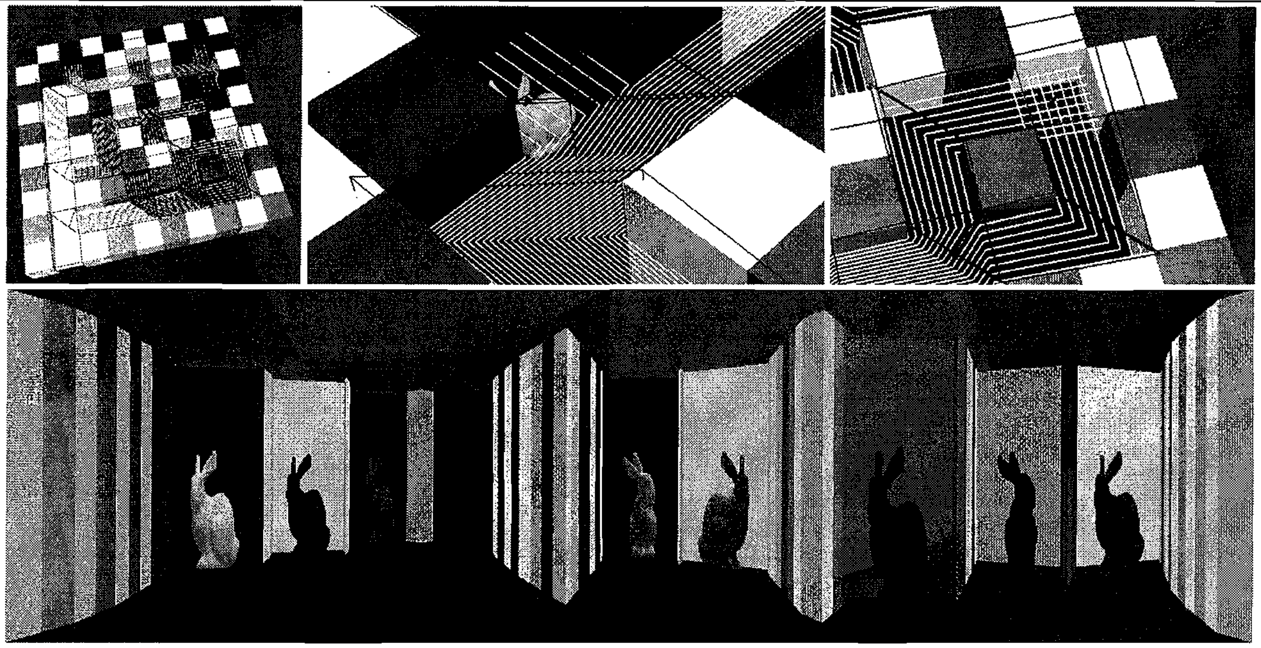


Figure 2 Visualization of graph camera model (*top*), and graph camera image (*bottom*). The graph camera rays (white lines) sample the entire maze recursively starting from the bottom left entrance. The resulting graph camera image has a single layer and it is mostly continuous and non-redundant, yet it shows longitudinally all maze corridors. The graph camera image is rendered at 40fps.

Abstract

In interactive 3-D graphics applications the user typically explores the scene by positioning and orienting a virtual camera. When the experience of actual locomotion in the virtual space is unnecessary, such sequential exploration is undesirable since it is inefficient—the user has to cover large distances in the scene, and ineffective—the user can only see a small fraction of the scene at any given time, which is particularly inadequate for dynamic scenes. The conventional solution is to employ several stationary cameras that render the scene in parallel. However, a large number of cameras is required for adequate scene coverage and there is no continuity between individual images, which requires the user to adapt to a multitude of contexts, one at the time.

We introduce the *graph camera*, a non-pinhole camera with rays that circumvent occluders to sample most or all of a 3-D scene. The graph camera image has a single layer, it is mostly continuous and non-redundant, yet it shows simultaneously all regions of interest in a complex 3-D scene. The graph camera is constructed from a planar pinhole camera through a series of view frustum bending, splitting, and merging operations. The graph camera has tens or even hundreds of frusta, yet rendering is efficient due to a fast projection operation that allows rendering in a single pass and allows resolving visibility automatically.

CR Categories: I.3.m. [Computer Graphics]: Picture/Image Generation– Viewing algorithms.

Keywords: camera model, interactive 3D computer graphics.

1 Introduction

Most interactive 3-D computer graphics applications rely on the virtual navigation paradigm to allow the user to explore a 3-D scene. The user controls a virtual camera which provides visual information about the scene. This sequential mode of exploration has several disadvantages. First, scene exploration becomes inefficient when the user has to cover large distances in the virtual space. Second, guiding the virtual camera such that it optimally reveals a complex scene requires familiarity with non-trivial navigational interfaces. Third, the user is limited at any given time to the small region of the scene captured by the current position and orientation of the camera. This problem is particularly severe when the understanding of the scene depends on establishing connections between remote regions of the scene that are never imaged together during the course of sequential exploration, or when phenomena or features of interest are transient and vanish by the time the virtual camera reaches them.

A possible solution is to render the scene with several cameras simultaneously. However, a large number of cameras is required to achieve satisfactory scene coverage, and the resulting images are poorly integrated. Discontinuities across the boundaries of individual images require the user to examine the images one at a time in order to adapt to each one of the multitude of contexts (Figure 1). The palliative solution of building redundancy into the set of images not only fails to truly solve the problem but it is also expensive.

It is our insight that these limitations are due to the traditional simplicity and rigidity of the camera model (i.e. set of rays captured by the camera) which controls image generation. Most computer graphics applications use the planar pinhole camera (PPC) model. One reason is that the PPC model closely approximates the human eye, producing images familiar to us. Another reason is simplicity—hardware implementations of the PPC model render complex scenes at interactive rates. However, the PPC model is limited. Whereas the field of view limitation has been addressed by innovations such as spherical, cylindrical, or cube map pinhole camera models, relatively little has been done to remove the requirement that all rays pass through a common point—the pinhole. This pinhole constraint limits images to scene regions to which there exists direct line of sight, making images ineffectual in the context of complex scenes with numerous occlusions.

We introduce the *graph camera*, a powerful non-pinhole camera that circumvents occluders to sample simultaneously many or all regions of interest in a 3-D scene. The graph camera image integrates many PPC images into a single-layer, mostly continuous and non-redundant image that allows examining the entire 3-D scene at once, overcoming the disadvantages of a single moving PPC or of a matrix of stationary PPCs (see Figure 2 and accompanying video).

In order to describe the graph camera model we generalize the definition of a camera ray to the set of 3-D points that project at a given image plane location, which allows for rays that are not straight lines. A graph camera ray is a chain of connected line segments. The graph camera is constructed such that its rays circumvent occluders and reach all regions of interest. Construction starts from a regular PPC whose frustum undergoes a series of bending, splitting and merging operations. The resulting graph camera is a graph of many PPCs, each defining a frustum. The graph camera in Figure 2 was automatically constructed starting from the bottom left entrance into the maze. The initial PPC frustum was split recursively at each intersection to cover the entire maze. A breadth first traversal was used here. The graph camera sees farther and farther into the maze to finally sample all corridors.

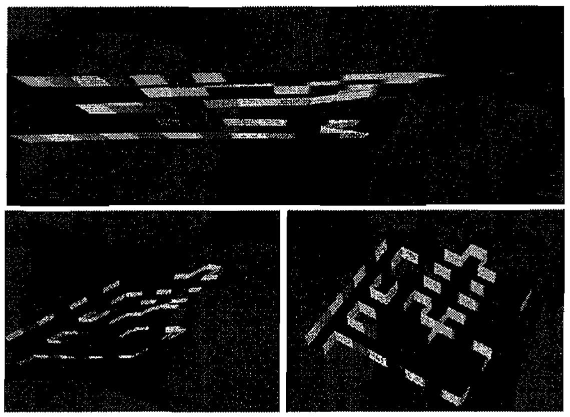


Figure 3 Distorted scene (*top*) corresponding to graph camera image in Figure 2, intermediate image while removing the distortion (*bottom left*), and undistorted scene (*b. right*).

Despite its complexity, the graph camera model provides a fast projection operation that maps a given 3-D point directly to the output image, with consistent depth. The projection operation enables rendering efficiently in a single pass, with consistent visibility, bypassing the need for compositing individual PPC images into the final graph camera image. The graph camera in Figure 2 has 38 frusta and rendering performance exceeds 40 fps at an output resolution of 1,920 by 1,200 pixels.

We define camera model continuity informally as the property that nearby 3-D points are projected at nearby image locations, and camera model non-redundancy as the property that any 3-D point projects to at most one image location. A graph camera is continuous and non-redundant as long as its frusta are disjoint. The graph camera in Figure 2 is continuous and non-redundant with the exception of the intersecting frusta that close the cycle at the bottom right corner of the maze (see top right image in Figure 2), which causes the magenta bunny to appear in duplicate.

The graph camera overcomes occlusions by moving scene regions that compete for the same PPC image location to disjoint graph camera image locations. In essence, the graph camera trades image resolution for depth resolution. For this approach to be effective the graph camera image has to have sufficient resolution to accommodate all scene regions of interest. The approach is supported by current high resolution displays. A single LCD display with WQXGA resolution of 4 million pixels can show a graph camera image with 64 regions of interest with an average foot print of 256 by 256 pixels. A 16 million pixel tiled display can show 256 such regions of interest. The user can change the screen real estate allocation dynamically, emphasizing one or a few regions, while the other regions continue to be rendered, providing context.

The graph camera model can be interpreted as a distortion of the 3-D scene that makes the scene look like the graph camera image when rendered with a conventional PPC. In Figure 3 the distorted scene was created from the graph camera image by connecting groups of 2x2 color and depth pixels to form a 3-D mesh. The inverse of the graph camera projection operation allows recovering the undistorted position of the mesh vertices, creating the undistorted scene. The intermediate image was recovered from a morph of the distorted to the undistorted scene.

The remainder of this paper is organized as follows. Prior work is reviewed next. Section 3 describes the graph camera model and gives construction algorithms. Section 4 describes rendering with the graph camera. Section 5 presents and discusses results. Section 6 concludes and sketches directions for future work.

2 Prior work

Non-pinhole cameras have been developed in the context of image-based rendering, of artistic rendering, and of reflection rendering. We briefly review each of these contexts, discussing the suitability of the camera model developed for creating a non-redundant and continuous image of several regions of interest of a 3-D scene.

Image-based rendering

Several non-pinhole camera models have been developed in the context of image based rendering for the purpose of scene modeling and rendering. The light field [Levoy 1996, Gortler 1996] is a 2D array of PPC images which amounts to a powerful camera model that captures a dense set of rays. However, light fields are ill-suited for the application at hand. First, under the diffuse surface reflectance model assumption, the light field is highly redundant and extracting a single copy of a given set of regions of interest is difficult. Second, (synthetic) light fields have lengthy rendering times: the scene has to be rendered for each of the many PPCs. Ulterior research has reduced the number of redundant rays using surface geometry information (e.g. surface [Wood 2000] and unstructured [Buhler 2001] light fields), but construction remains an offline process. Light fields have to be used as a set of pre-computed color samples rather than as a set of rays, which precludes dynamic scenes.

Layered depth images (LDIs) [Shade 1998] generalize the PPC image by allowing for more than one sample along a ray. Like the graph camera, the LDI camera is a PPC whose rays are broken into several segments, but in the case of the LDI the segments are *collinear*. The application of LDIs is 3D image warping [McMillan 1995] without the problem of disocclusion errors, which are artifacts due to missing samples for surfaces that are visible in the desired view but were not visible in the reference image. The LDI avoids the redundancy of light fields, but it remains difficult to combine many regions of interest in a single-layer output image. Moreover the number of samples stored at each LDI pixel varies widely; therefore it is impractical to construct the LDI by successive rendering passes and by peeling off the nearest layer. Adequate LDIs are built by combining a large number of PPC images rendered from views around the LDI reference view. Like in the case of light fields, LDIs are built offline and the LDI camera is too inefficient to accommodate dynamic scenes.

Occlusion cameras [Mei 2005] are a family of non-pinholes with rays that reach around occluders to gather samples that are barely occluded from the reference viewpoint and thus are likely to be needed to support viewpoint translation without disocclusion errors. Occlusion cameras produce single layer images that show more than what is visible from a single point, and they can be rendered efficiently with hardware support, but they do not offer the ray modeling flexibility required to reach distant regions of a complex 3-D scene.

Multiple center of projection (MCOP) images [Rademacher 1998] collect samples with a vertical slit camera that slides along a user defined path. Possible goals in path selection are good scene coverage or artistic value of resulting image. The great flexibility in defining the rays of the MCOP camera and the resulting single-layer image makes it attractive in the context of our problem of simultaneously capturing several regions of a 3-D scene. An MCOP camera could be used to create an image like the graph camera image shown in Figure 2. However, MCOP cameras are inefficient: images have to be rendered by ray tracing or by rendering the scene in feed-forward fashion for each center of projection along the camera path.

Non-pinhole cameras have also been employed to facilitate the creation of panoramas for cel animation [Wood 1997]. The rays of interest are defined by the desired scene shots. The non-pinhole renders a multiperspective panorama which simulates camera motion in a 3D scene when it is viewed through a rectangular frame sliding on a predetermined path. Like for MCOPs, the panorama is rendered by finely discretizing the 3D camera path and by rendering an image for each position along the path, which amounts to long rendering times.

Artistic rendering

Another application of non-pinhole camera models is in the context of artistic rendering where they are employed to render multiperspective images, similar to the ones produced by the graph camera. In one system, individual PPCs are attached to scene objects, and the resulting sprites are composited in a multi-projection image [Agrawala 2000]. For a small number of objects, the multi-projection image can be updated interactively. The approach of attaching a pinhole to each object has the disadvantages of not scaling with scene complexity, of difficult—sometimes impossible—visibility ordering, and of not supporting multiple perspectives per object.

Another multiperspective rendering system [Yu 2004b] partitions an image plane into general linear camera (GLC) triangular images. A GLC is constructed from three given rays [Yu 2004a] so it offers some flexibility for modeling rays such that they reach the desired regions of a 3-D scene. Moreover, GLCs also have the advantage of fast projection. However, combining several GLCs is non-trivial. The solution adopted by Yu et al. was to blend the rays of neighboring GLCs to provide a continuous ray space which generates an image with smoothly varying perspective. The resulting compound non-pinhole camera model does not provide fast projection and rendering is performed offline by ray tracing.

Multiperspective images of real-world scenes can be constructed by re-sampling a video cube—a stack of images gathered by moving a video camera along a continuous path [Seitz 2003]. The video cube has also been used to support impressionism, cubism, and abstract aesthetic video effects [Klein 2002].

Reflection rendering

The need for non-pinhole cameras also arises in the context of reflection rendering. Curved reflective surfaces perturb the rays "leaving" the pinhole modeling the desired view, thus second and higher order reflected rays are not concurrent and amount to a non-pinhole camera. Indeed, reflections have been modeled with the help of the general linear camera multiperspective rendering framework [Yu 2005]. The sample-based camera reflection rendering method [Popescu 2006] leverages the coherence of small contiguous sets of reflected rays and replaces them with PPCs. The PPCs are stored at the leaves of a binary space partitioning tree which defines a sample-based camera. The graph camera and the sample-based camera are similar in the sense that both are collections of PPCs. However, the sample-based camera constructor focuses on tightly approximating a *given* set of non-concurrent reflected rays, whereas the graph camera constructor

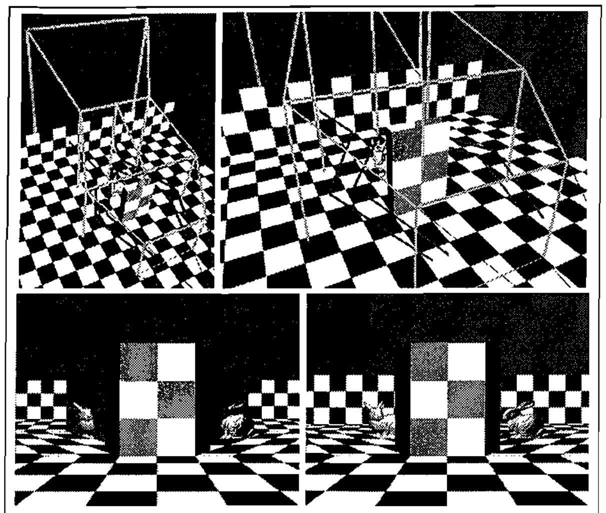


Figure 4 Visualization of graph camera model (*top*), and graph camera images (*bottom*). The graph camera was constructed with a splitting and a merging operation in order to shrink the "shadow" of the vertical block.

aims to provide flexibility for *specifying* rays such that they elude occluders and reach many or all regions of interest in a 3-D scene.

3 The graph camera model

The graph camera is a graph of planar pinhole cameras constructed from an initial planar pinhole camera PPC_0. PPC_0 defines the first segment of the piecewise linear rays of the graph camera and collects the graph camera image.

3. 1 Basic construction operations

The rays of PPC_0 are bent, partitioned, and merged repeatedly in order to capture all regions of interest in the scene to be rendered.

Frustum bending

The bending operation takes a planar pinhole camera PPC_0, a plane p_{01}, and a point P_1, and produces a planar pinhole camera PPC_1 constructed such that it has P_1 as its center of projection (COP) (i.e. pinhole) and p_{01} as its image plane. The rays of PPC_0 are clipped with p_{01}. In the graph camera recursive hierarchy PPC_0 is the parent of a single child PPC_1.

Frustum splitting

The splitting operation takes a planar pinhole camera PPC, a plane p, a set of n points P_i, and a subdivision of p into n disjoint partitions S_i, and produces n planar pinhole cameras PPC_i with centers of projection P_i and image plane p. The rays of PPC_i are restricted to the subset of rays of PPC that intersect partition S_i. The rays of PPC are clipped with p. In the graph camera hierarchy PPC is the parent of n children PPC_i.

Frustum merging

The merging operation is the reverse of splitting. The input consists of n planar pinhole cameras PPC_i, a plane p, and a point P, and the output is a single planar pinhole camera PPC using P and p as its COP and image plane, respectively. The rays of PPC_i are clipped with p. In the graph camera hierarchy all PPC_i have one and the same child PPC.

In Figure 4 the bunny cannot hide behind the vertical block. The non-redundant graph camera variant (*bottom left*) is obtained

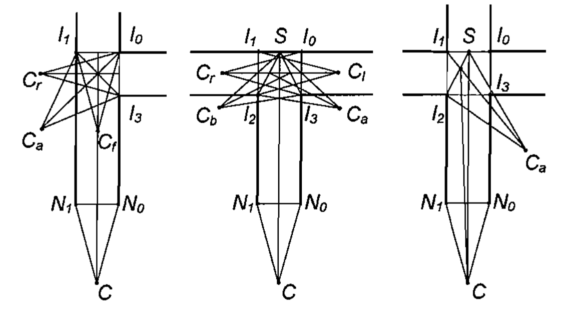


Figure 5 The right & forward (*left*), right & left (*middle*) and right, forward, & left (*right*) intersection cases for the maze graph camera construction algorithm.

using a separator plane between the frusta resulting from the splitting operation. The redundancy is eliminated at the cost of a discontinuity in the image where rays terminate before encountering any geometry. The redundant variant (*bottom right*) allows for frusta to intersect and produce a complete image at the cost of repeating a part of the scene.

3. 2 3-D maze graph camera construction algorithm

For some scenes it is possible and preferable to select the sequence of basic construction operations and their parameters automatically, based on the scene geometry and on a priori knowledge about the regions of interest. In the case of a 3-D maze for example, a graph camera can be constructed automatically to sample longitudinally all maze corridors, regardless of the maze complexity, provided that the display has sufficient resolution to adequately show even the distant parts of the maze.

We have developed a graph camera construction algorithm for mazes that have right angle turns and intersections. The algorithm takes as input the maze geometry as well as the output of a maze traversal algorithm, and produces a graph camera based on the traversal solution provided. If the maze graph does not have cycles, or if the user only desires to visit all intersections (i.e. maze graph nodes) and not also all corridors (i.e. maze graph edges, including back edges that lead to a node already visited), the resulting graph camera is strictly continuous and non-redundant. For the example shown in Figure 1, the redundancy would have been avoided if the graph camera did not have to sample both corridors leading to the magenta bunny.

The construction algorithm proceeds recursively. Each non-terminated corridor is extended to the first intersection. According to the type of intersection, 0 or more exit branches are created, which are followed recursively. A corridor leading to an intersection is handled with one of the following cases: bend left, bend right, split right & forward, split right & left, split forward & left, split right, forward & left, and terminate. The bend left and right cases are trivial. The terminate case occurs when the corridor leads to a dead end or to an intersection that has already been visited. We briefly discuss the remaining cases based on the diagrams in Figure 5.

The right & forward case

The incoming frustum corresponds to a PPC with COP at C. Its frustum is clipped with l_3l_1 to $N_0l_3l_1N_1$ (6 faces in 3-D). The frustum has one child, defined by the auxiliary PPC with center of projection C_a which distributes the rays to the exit planes l_3l_0 and

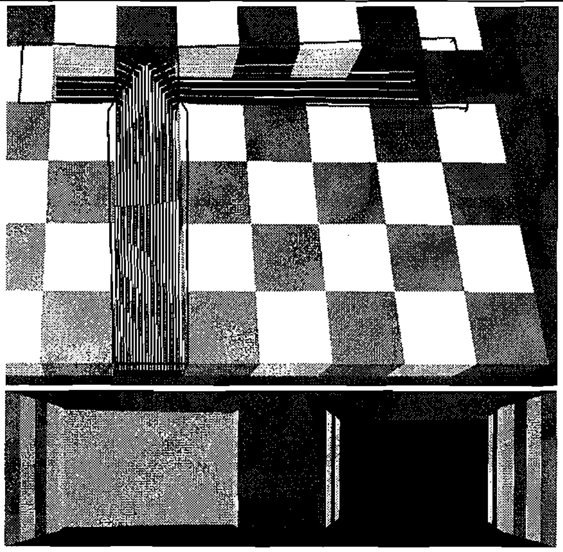


Figure 6 Graph camera model (*top*) and image (*bottom*) for the right & left case with an even split.

I_1I_0. Here the split is even, but it can be controlled to match the importance of the individual exit branches. The auxiliary camera has two children, the forward branch with COP C_f, and the right branch with COP C_r, which are processed recursively. The forward & left case is symmetrical.

The right & left case

In this case two auxiliary cameras are needed with COPs at C_a and C_b to distribute the rays left and right to the exit branches C_l and C_r. The incoming frustum is clipped with two planes defined by SI_2 and SI_3. The resulting frustum has 7 faces in 3-D: 5 quadrilateral faces and two pentagonal faces. The location of S decides the split ratio. The incoming frustum has two children for the auxiliary cameras, each of which has one child for the left and right branches. Figure 6 shows the actual graph camera and its image for a single right & left split. The left branch terminates soon after the intersection, which is reflected in the graph camera image by a shallower depth of the white frontal wall. The frusta here have a 3.5 degree field of view, parameter that controls the tradeoff between sampling the side walls and floor versus sampling deep into the maze.

The right, forward, & left case

Like before, there are two auxiliary cameras, but, for improved readability of the diagram, only the auxiliary camera C_a that generates the left exit branch is shown. The incoming frustum is clipped similarly as in the right & left case. For an even split, the auxiliary camera sends one third of the rays to the left branch (exit plane I_1I_2) and one sixth forward (exit plane SI_1). Since the second auxiliary camera sends one third of the rays to the right and one sixth forward, each exit branch totals one third of the rays. The two halves of the forward branch are merged. The incoming frustum has two children for the auxiliary cameras, each of which has two children, one for the right or left exit branch and one for the forward exit branch.

In this and in all cases the intersection is sampled completely and non-redundantly. Figure 7 visualizes the parts of the scene sampled by the graph camera constructed with the recursive algorithm based on a breadth first traversal of the maze. The color highlights indicate same depth in the traversal.

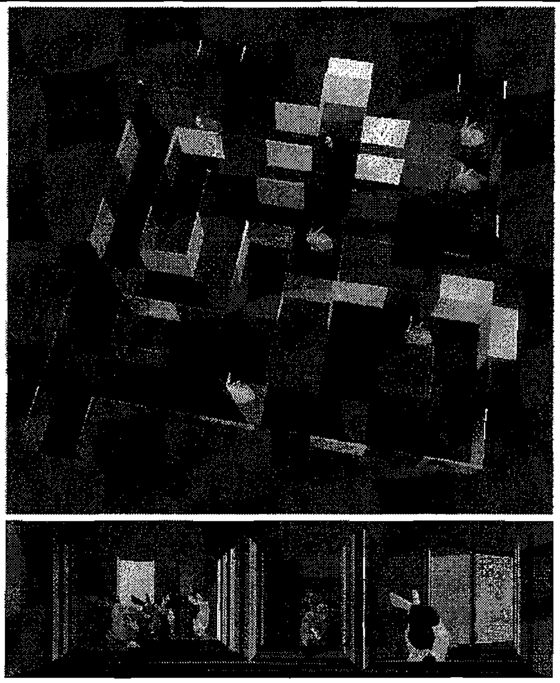


Figure 7 Visualization scene partitioning induced by graph camera (*top*) and graph camera image (*bottom*).

4 Graph camera rendering

A scene modeled with triangles is rendered with a graph camera one planar pinhole camera frustum at the time. For each frustum, the relevant triangles are first found using a conventional hierarchical space subdivision scheme. We use an octree. A triangle t is rendered with a frustum PPC with the following steps:

```
1. Clip $t$ with the faces of $PPC$ to triangles $t_i$
2. For each $t_i$
    2.1. For each vertex $v_j$
        2.1.1. Compute distorted vertex $v_j'$
    2.2. Render distorted triangle ($v_0'$, $v_1'$, $v_2'$) with $PPC_0$
```

PPC_0 is the initial planar pinhole camera that collects the graph camera image. The rendering algorithm essentially computes a distorted scene (Figure 3) which when rendered with PPC_0 produces the same result as when rendering the original scene with the graph camera. Given a vertex V contained by a frustum PPC_k the distorted vertex V' is computed as follows. The projection (u_k, v_k) of V on the image plane of PPC_k is found with the equation:

$$V = C_k + M_k \begin{bmatrix} u_k \\ v_k \\ 1 \end{bmatrix} w_k ,$$

$$\begin{bmatrix} u_k \\ v_k \\ 1 \end{bmatrix} = M_k^{-1}(V - C_k)\frac{1}{w_k}$$

where C_k and M_k are the COP and 3x3 camera matrix of PPC_k. The projection is taken directly to the PPC_0 (output) image plane at coordinates (u_0, v_0) by multiplication with a 3x3 matrix Q_k which concatenates the mappings from the image plane of PPC_k

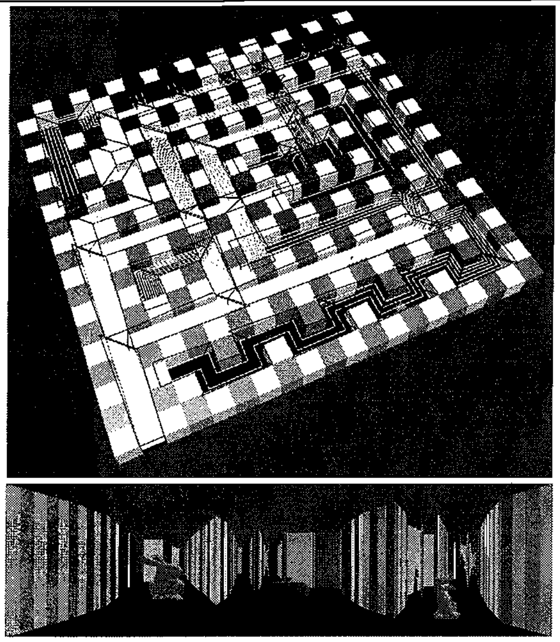


Figure 8 Graph camera model (*top*) and image (*bottom*) for the 20x20 maze.

to that of PPC_{k-1}, from the image plane of PPC_{k-1} to that of PPC_{k-2} and so on all the way to PPC_0 (see Appendix A). The matrix Q_k is pre-computed for each PPC_k during graph camera model construction. Applying the matrix Q_k we obtain:

$$\begin{bmatrix} u_0 \\ v_0 \\ 1 \end{bmatrix} w_0 = Q_k \begin{bmatrix} u_k \\ v_k \\ 1 \end{bmatrix} = Q_k M_k^{-1}(V - C_k)\frac{1}{w_k} \qquad (1)$$

$$\begin{bmatrix} u_0 \\ v_0 \\ 1 \end{bmatrix} w_0 w_k = Q_k M_k^{-1}(V - C_k)$$

The equation above is sufficient to calculate the projection (u_0, v_0) of V with PPC_0. However, using an arbitrary depth to establish the position of V' along PPC_0 leads to an incorrect interpolation of rasterization parameters. What is needed is a depth of V' which ensures that a linear variation in the undistorted space of V matches to a linear variation in the distorted space of V', such that conventional model space interpolation of the distorted triangle produces correct results. In order to find a correct depth of V' we first write the projection of V' with PPC_0 as:

$$V' = C_0 + M_0 \begin{bmatrix} u_0 \\ v_0 \\ 1 \end{bmatrix} w'$$

where w' indicates the unknown depth of V'. It follows that

$$\begin{bmatrix} u_0 \\ v_0 \\ 1 \end{bmatrix} w' = M_0^{-1}(V' - C_0)$$

Examining this equation together with equation (1) it follows that a possible value of w' is the product $w_0 w_k$. Consequently

$$\begin{aligned} & w' = w_0 w_k \\ & M_0^{-1}(V' - C_0) = Q_k M_k^{-1}(V - C_k) \\ & V' = C_0 + M_0 Q_k M_k^{-1}(V - C_k) \\ & V' = (C_0 - M_0 Q_k M_k^{-1} C_k) + (M_0 Q_k M_k^{-1})V \\ & V' = T + RV \end{aligned}$$

In conclusion the distorted vertices are computed by rotating and translating the undistorted vertex V, which can be done with one 4x4 matrix multiplication with the traditional, fixed pipeline. The clip planes of the frustum of *PPC* are also supported by the fixed pipeline. When the number of clip planes of a frustum exceeds the number of clip planes supported, the frustum is split and replaced with frusta with fewer faces. We use OpenGL which allows for 6 clip planes. For the 3-D maze graph camera the only times when this number is exceeded is for the incoming frustum for the right & left, and right, forward, & left cases. Such as frustum is replaced with two 6-face frusta by splitting the frustum in half (i.e. along *CS* in the middle and right drawings in Figure 5).

5 Results and discussion

At low level, the rendering algorithm simply clips and renders triangles conventionally, so there are no artifacts. At high level, the graph camera produces an image that is mostly continuous and non-redundant, with the exceptions discussed earlier. The graph camera is a set of PPCs so a straight line maps to a straight line while the line is contained in a single frustum. Continuity is preserved even as objects pass through the planes that separate frusta and parts of the object are shown with different perspectives. The graph camera image directly conveys the presence or absence of an object anywhere in the maze, and allows judging the relative position of objects with respect to each other and with respect to the viewer.

In addition to the 10x10 maze seen throughout the paper, we have also tested the graph camera approach on a larger 20x20 3-D maze scene (Figure 8), and on the auditorium and David scenes (Figure 9). The graph cameras for the auditorium and the David scenes were constructed with an interactive graphical editor. Rendering performance was measured on a 3.2GHz 2GB Intel Pentium 4 Xeon workstation with an nVidia GeForce 7950 Gx2 graphics card, see Table 1. All scenes are rendered interactively. The output resolution was 1280 by 720 pixels, but experiments indicate that for our scenes the frame rate depends little on the output resolution. The number of camera frusta, the geometry load, and the size of the frusta relative to the geometry are the main factors affecting performance. Many small frusta reduce the

	Triangles	Camera Frusta	Frame Rate
Small maze	25K	38	115 fps
	50K		80 fps
	150K		54 fps
Large Maze	250K	108	19 fps
	500K		12 fps
	700K		6 fps
Auditorium	100K	7	40 fps
David	500K	2	30 fps

Table 1 Rendering performance.

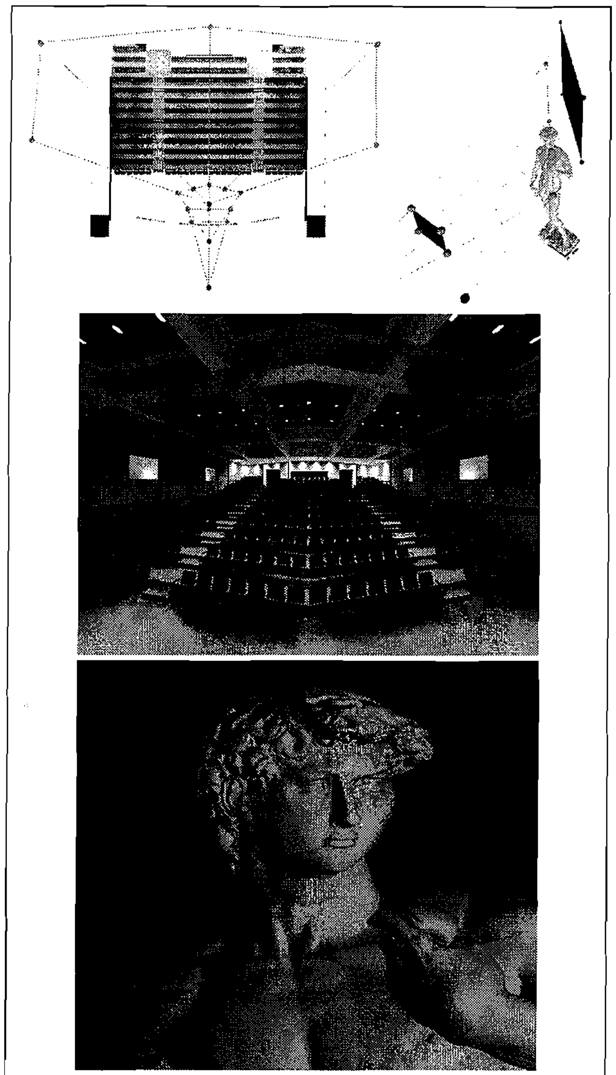


Figure 9 Orthographic view of graph camera models (*top*) and resulting images (*middle* and *bottom*) for the auditorium and David scenes. The auditorium had straight rows of seats. The graph camera merges two perspectives of David's face.

performance of the octree which is essentially responsible for view frustum culling.

6 Conclusions and future work

The literal malleability of the graph camera allows it to sample comprehensively complex 3-D scenes with intricate occlusion patterns. The graph camera has demonstrated a single-layer image that shows an entire 3-D maze. The projection equation is simple, which makes the camera model very efficient.

There are many exciting avenues for future work. One is developing additional constructors. One possibility is a constructor that takes a set of PPC views and builds a graph camera that optimally integrates them into a continuous image. Another possibility is a graph camera that crawls over the 3-D scene, stretching out branches to comprehensively inspect a region. A third possibility is to develop a more powerful interactive graph camera model editor and to place it in a popular animation system such as Maya or 3ds Max, where animators could explore and guide the concretization of the graph camera's potential as a modeling tool.

Another direction of future work is to port the rendering effects developed for conventional cameras to graph cameras. The current rendering algorithm correctly interpolates rasterization parameters and many of existing shaders will simply work with the graph camera, but questions remain such as, for example, how to render shadows, should shadows be rendered in the undistorted or distorted domain, or how to provide view dependent parameters needed by shaders.

We are also interested in building graph camera models from images acquired with video cameras. In order to combine the video feeds into a graph camera image scene geometry is needed, which is a tractable problem, especially in indoor spaces which have the benefit of availability of blue-prints or even CAD data. The main research challenge we foresee is achieving a smooth transition from frustum to frustum for objects for which only coarse geometric proxies are known.

The graph camera can accommodate tens or, as display resolution increases, even hundreds of regions of interest. In order to fully take advantage of the graph camera it will be beneficial to couple the graph camera constructor with image processing, computer vision, and/or data mining algorithms which find the regions of interest automatically and feed them to the constructor.

Another important direction of future work is the study of visual perception from non-pinhole stimuli, an area with little prior work. Understanding the strengths and weaknesses of various non-pinhole image types and deriving guiding principles for visual perception optimization would accelerate the deployment of this novel technology to applications.

The graph camera advocates a departure from the conventional approach of using a simple and rigid camera model for all applications and all datasets, in favor of designing and dynamically optimizing the camera model according to the application and dataset at hand. We foresee that this novel approach will be beneficial for many applications in computer graphics and beyond.

7 Acknowledgments

Withheld for double-blind review.

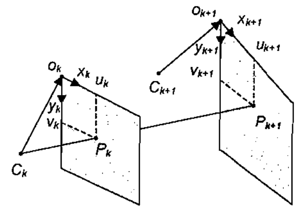
References

[Agrawala 2000] AGRAWALA, M., ZORIN, D., AND MUNZNER, T. 2000. Artistic Multiprojection Rendering. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000* (June 26 - 28, 2000). B. Peroche and H. E. Rushmeier, Eds. Springer-Verlag, London, 125-136.

[Buhler 1999] Buehler, C., BOSSE, M., MCMILLAN, L., GORTLER, S., AND COHEN, M. 2001. Unstructured lumigraph rendering. In *Proceedings of the 28th Annual Conference on Computer Graphics and interactive Techniques* SIGGRAPH '01. ACM, New York, NY, 425-432.

[Gortler 1996] GORTLER, S. J., GRZESZCZUK, R., SZELISKI, R., AND COHEN, M. F. 1996. The lumigraph. In *Proceedings of the 23rd Annual Conference on Computer Graphics and interactive Techniques* SIGGRAPH '96. ACM, New York, NY, 43-54.

[Grossberg 2001] GROSSBERG, D. AND NAYAR, S. 2001. A General Imaging Model and a Method for Finding its Parameters. In *Proceedings of the Eighth IEEE International Conference on Computer Vision, ICCV.*



$$P_{k+1} = \begin{bmatrix} x_{k+1} & y_{k+1} & (C_{k+1}+o_{k+1}) \end{bmatrix} \begin{bmatrix} u_{k+1} \\ v_{k+1} \\ 1 \end{bmatrix}$$

$$P_{k+1} = C_k + \begin{bmatrix} x_k & y_k & o_k \end{bmatrix} \begin{bmatrix} u_k \\ v_k \\ 1 \end{bmatrix} w_k$$

$$R_{k+1} = \begin{bmatrix} x_k & y_k & o_k \end{bmatrix}^{-1} \begin{bmatrix} x_{k+1} & y_{k+1} & (C_{k+1}-C_k+o_{k+1}) \end{bmatrix}$$

$$\begin{bmatrix} u_k \\ v_k \\ 1 \end{bmatrix} w_k = R_{k+1} \begin{bmatrix} u_{k+1} \\ v_{k+1} \\ 1 \end{bmatrix}$$

Figure 10 Derivation of mapping PPC_{k+1} and PPC_k with COP's C_{k+1} and C_k. Vectors *x* and *y* give the row and column direction and are one pixel width and one pixel length long, respectively. Vectors *o* point 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}.

[Gupta 1997] Gupta, R. and Hartley, R. I. 1997. Linear Pushbroom Cameras. *IEEE Trans. Pattern Anal. Mach. Intell.* 19, 9 (Sep. 1997), 963-975.

[Klein 2002] Klein, A. W., Sloan, P. J., Finkelstein, A., and Cohen, M. F. 2002. Stylized video cubes. In *Proceedings of the 2002 ACM Siggraph/Eurographics Symposium on Computer Animation* (San Antonio, Texas, July 21 - 22, 2002). SCA '02. ACM, New York, NY, 15-22.

[Levoy 1996] Levoy, M. and Hanrahan, P. 1996. Light field rendering. In *Proceedings of the 23rd Annual Conference on Computer Graphics and interactive Techniques* SIGGRAPH '96. ACM, New York, NY, 31-42.

[Mei 2005] Mei, C., Popescu, V., and Sacks, E. 2005. *The Occlusion Camera.* In proc. of Eurographics 2005, Computer Graphics Forum, vol. 24, issue 3.

[McMillan 1995] McMillan, L. and Bishop, G. 1995. Plenoptic modeling: an image-based rendering system. In *Proceedings of the 22nd Annual Conference on Computer Graphics and interactive Techniques* S. G. Mair and R. Cook, Eds. SIGGRAPH '95. ACM, New York, NY, 39-46.

[Popescu 2006] Popescu, V., Sacks, E., and Mei, C. 2006. Sample-Based Cameras for Feed Forward Reflection Rendering. *IEEE Transactions on Visualization and Computer Graphics* 12, 6 (Nov. 2006), 1590-1600.

[Rademacher 1998] Rademacher, P. and Bishop, G. 1998. Multiple-center-of-projection images. In *Proceedings of the 25th Annual Conference on Computer Graphics and interactive Techniques* SIGGRAPH '98. ACM, New York, NY, 199-206.

[Seitz 2003] Seitz, S. M. and Kim, J. 2003. Multiperspective Imaging. *IEEE Comput. Graph. Appl.* 23, 6 (Nov. 2003), 16-19.

[Shade 1998] Shade, J., Gortler, S., He, L., and Szeliski, R. 1998. Layered depth images. In *Proceedings of the 25th Annual Conference on Computer Graphics and interactive Techniques* SIGGRAPH '98. ACM, New York, NY, 231-242.

[Wood 1997] Wood, D. N., Finkelstein, A., Hughes, J. F., Thayer, C. E., and Salesin, D. H. 1997. Multiperspective panoramas for cel animation. In *Proceedings of the 24th Annual Conference on Computer Graphics and interactive Techniques* International Conference on Computer Graphics and Interactive Techniques. ACM Press/Addison-Wesley Publishing Co., New York, NY, 243-250.

[Wood 2000] Wood, D. N., Azuma, D. I., Aldinger, K., Curless, B., Duchamp, T., Salesin, D. H., and Stuetzle, W. 2000. Surface light fields for 3D photography. In *Proceedings of the 27th Annual Conference on Computer Graphics and interactive Techniques* International Conference on Computer Graphics and Interactive Techniques. ACM Press/Addison-Wesley Publishing Co., New York, NY, 287-296.

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

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

[Yu 2005] Yu, J., and McMillan, L. 2004. Modelling Reflections via Multiperspective Imaging. In *Proceedings of Eurographics Symposium on Rendering (EGSR).*

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 10. Then we show by induction that $Q_{k+1} = R_1R_2...R_{k+1}$. The base case is verified as follows:

$$\begin{bmatrix} u_0 \\ v_0 \\ 1 \end{bmatrix} w_0 = R_1 \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix}, \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} w_1 = R_2 \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} u_0 \\ v_0 \\ 1 \end{bmatrix} w_0 = R_1 \frac{1}{w_1} R_2 \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = \frac{1}{w_1} R_1 R_2 \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} u_0 \\ v_0 \\ 1 \end{bmatrix} w_0 w_1 = R_1 R_2 \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix}$$

$$Q_2 = R_1 R_2$$

By the induction hypothesis:

$$Q_k = R_1 R_2 ... R_k$$

$$\begin{bmatrix} u_0 \\ v_0 \\ 1 \end{bmatrix} w_0 = R_1 R_2 ... R_k \begin{bmatrix} u_k \\ v_k \\ 1 \end{bmatrix}$$

Using the equations in Figure 10 we obtain:

$$\begin{bmatrix} u_k \\ v_k \\ 1 \end{bmatrix} w_k = R_{k+1} \begin{bmatrix} u_{k+1} \\ v_{k+1} \\ 1 \end{bmatrix}$$

Combining the two equations:

$$\begin{bmatrix} u_0 \\ v_0 \\ 1 \end{bmatrix} w_0 = R_1 R_2 ... R_k \frac{1}{w_k} R_{k+1} \begin{bmatrix} u_{k+1} \\ v_{k+1} \\ 1 \end{bmatrix},$$

$$\begin{bmatrix} u_0 \\ v_0 \\ 1 \end{bmatrix} w_0 w_k = R_1 R_2 ... R_k R_{k+1} \begin{bmatrix} u_{k+1} \\ v_{k+1} \\ 1 \end{bmatrix}$$

which terminates the proof.