The Graph Camera

Voicu Popescu
Purdue University
popescu@cs.purdue.edu

Paul Rosen
Purdue University
rosen@cs.purdue.edu

Nicoletta Adamo-Villani
Purdue University
nadamovi@purdue.edu

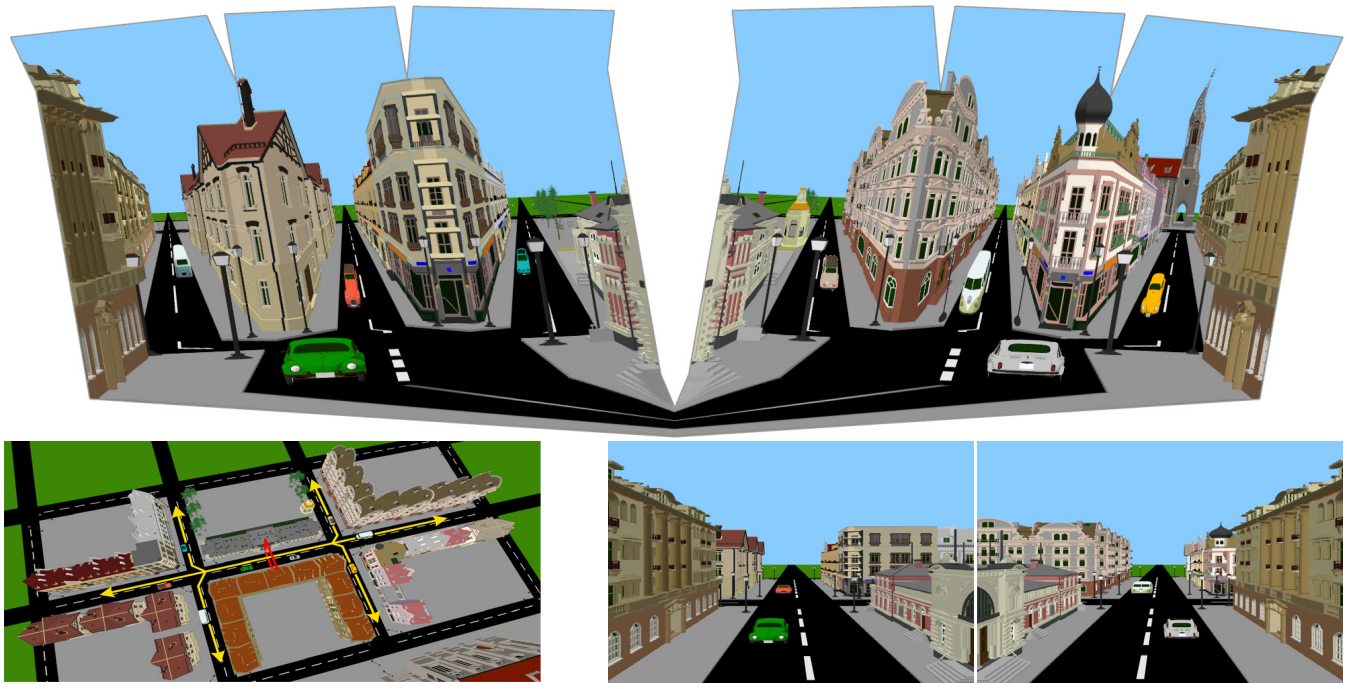

Figure 1 Enhanced virtual 3-D scene exploration. The graph camera image (*top*) samples longitudinally the current street segment as well as the 3 segments beyond the first intersections (*bottom left*). The 4 side streets are occluded in conventional images (*bottom right*).

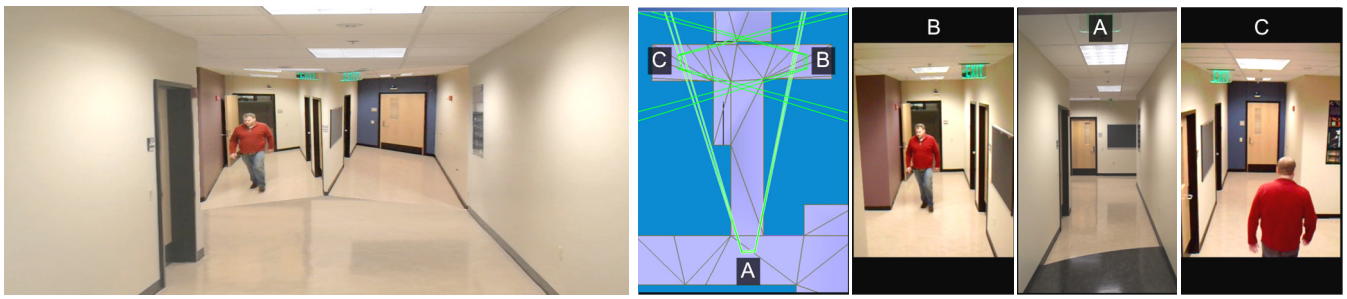

Figure 2 Single-image comprehensive visualization of real-world scenes. The graph camera image (*left*) seamlessly integrates 3 video feeds (*right*) and shows all 3 branches of the T corridor intersection.

Abstract

A conventional pinhole camera captures only a small fraction of a 3-D scene due to occlusions. We introduce the graph camera, a non-pinhole with rays that circumvent occluders to create a single layer image that shows simultaneously several regions of interest in a 3-D scene. The graph camera image exhibits good continuity and little redundancy. The graph camera model is literally a graph of tens of planar pinhole cameras. A fast projection operation allows rendering in feed-forward fashion, at interactive rates, which provides support for dynamic scenes. The graph camera is an infrastructure level tool with many applications. We explore the graph camera benefits in the contexts of virtual 3-D scene exploration and summarization, and in the context of real-world 3-D scene visualization. The graph camera allows integrating multiple video feeds seamlessly, which enables monitoring complex real-world spaces with a single image.

CR Categories: I.3.3 [Computer Graphics] Picture/Image Generation—Viewing algorithms, I.4.m [Image Processing and Computer Vision] Miscellaneous.

Keywords: camera models, non-pinholes, panoramas, image-based rendering, video integration, interactive rendering.

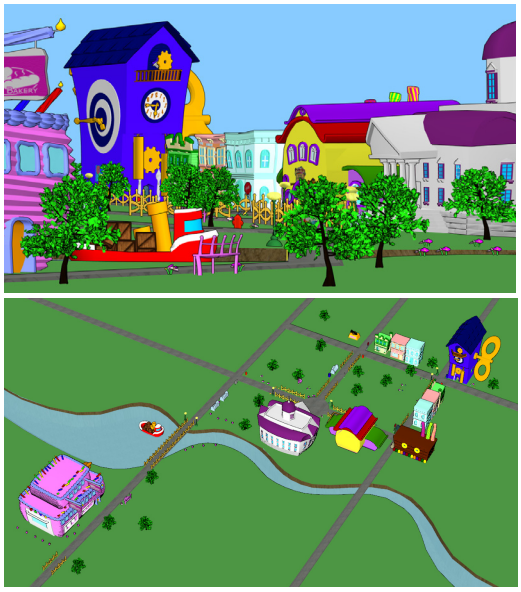

Figure 3 Graph camera image that summarizes a cartoon town scene (*left*) and conventional image for comparison (*right*).

1 Introduction

Most 3-D computer graphics applications rely on the planar pinhole camera (PPC) model to compute images of a 3-D scene. One reason is that the PPC model is simple, enabling efficient software and hardware implementations that render complex scenes at interactive rates. Another reason is that the PPC model approximates the human eye well, producing familiar images. However, the PPC model has limitations such as a uniform sampling rate, a limited field of view, and a single viewpoint. In this paper we address the single viewpoint limitation. By definition, all rays of a PPC pass through a common point—the pinhole. This limits images to scene regions to which there exists direct line of sight. In scenes with complex occlusions, a PPC image captures only a small fraction of the scene.

We introduce the graph camera, a non-pinhole 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.

The flexibility of the graph camera model makes it useful in many contexts. In this paper we explore 3 applications. 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 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.

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 3 the graph camera was constructed to sample most building façades. Sampling the same buildings with a PPC leads to poor image space utilization.

Finally, the graph camera is also useful 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 2 monitoring the hall way 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.

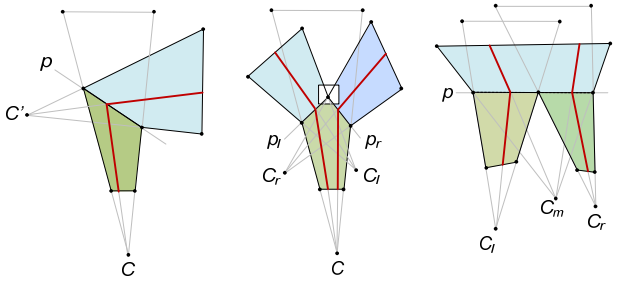

Figure 4 Basic construction operations: bending (*left*), splitting (*middle*) and merging (*right*). Sample rays are shown in red.

2 The Graph Camera Model

We relax the definition of a camera ray to the locus of 3-D points that project at a given image location, which allows for rays that are not a straight line. We define 3 basic construction operations on the frustum of a PPC (Figure 4). 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 an oriented graph with PPCs at its nodes (Figure 5). The graph is constructed starting from a root PPC through a sequence of bending, splitting, and merging operations. For each operation, graph arcs 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 5).

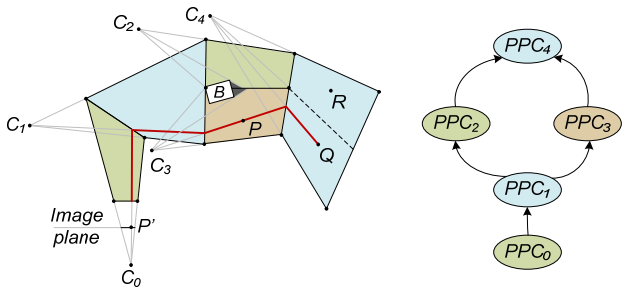

Figure 5 Graph camera with 5 PPC frusta. The PPC_1 is split into convergent PPC_2 and PPC_3 to shrink the occlusion shadow of the rectangular object B. P and Q are projected at P'.

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 the line at the intersection of planes p_l and p_r is hidden by geometry (white rectangle) which avoids a discontinuity in the graph camera image. The graph camera image is non-redundant as long as the PPC frusta are disjoint. Unlike most camera models, the graph camera is defined based on the actual 3-D scene it is called upon to visualize. Its rays are designed to circumvent occluders and 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 5 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.

Special care needs to be taken when deriving the projection matrix for frusta downstream from a merging operation. For example in Figure 5 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.

3 Prior Work

3. 1 Image-based rendering

Several non-pinhole camera models have been developed for scene modeling and rendering. The light field [Levoy 1996, Gortler 1996] essentially combines multiple PPC images which amounts to a powerful camera model that captures a dense set of rays. Layered depth images [Shade 1998] allow for more than one sample along a PPC ray. Neither light fields nor LDIs can be easily transformed into single layer, continuous and non-redundant images with multiple perspectives.

Panoramas are single-layer images [Chen 1995] that capture virtual or real-world 3-D scenes in all directions, but the single layer comes at the cost of a single viewpoint. The single viewpoint constraint is relaxed by mosaicing techniques which still require *nearly* coincident COPs [Szeliski 1997]. Multiple viewpoints are desired rather than an acquisition artifact in urban landscape photography. Facades along a street are captured with a vertical push-broom camera and additional perspectives are integrated into the panorama where the proxy façade plane is discontinued [Roman 2004 and 2006, Agarwala 2006]. A precursor of street panoramas are multiple center of projection (MCOP) images [Rademacher 1998], which also collect samples along a user defined path. Compared to graph camera images, street panoramas and MCOP images provide a smooth perspective change which is possible due to the large number of push-broom acquisition viewpoints. A graph camera changes perspective with C^0 continuity which keeps the number of viewpoints low. This enables feed-forward rendering for virtual scenes, and acquisition with only a few cameras for real-world scenes.

Non-pinhole cameras have also been used in cel animation [Wood 1997]. Camera motion in a 3-D scene is simulated by viewing a multiperspective panorama through a sliding frame. Like MCOP images, cel panoramas are rendered by finely discretizing the camera path, which is slow.

Occlusion cameras [Mei 2005, Popescu 2006, Rosen 2008] are a family of non-pinholes that enhance a PPC image with barely hidden samples to alleviate disocclusion errors when the viewpoint translates. A barely hidden sample is a sample that projects close to the silhouette of an occluder and is thus likely to become visible even for small viewpoint translations. Occlusion cameras and the graph camera have the common goal of creating a 2-D image of a 3-D scene that shows more than what is visible from a single viewpoint. However, occlusion cameras are designed to enhance a given PPC with a few extra samples, whereas the graph camera is designed to integrate multiple PPCs. Occlusion cameras do not offer the ray modeling flexibility required to reach distant regions of a complex 3-D scene and therefore they do not support graph camera applications (e.g. scene summarization, video feed integration).

3. 2 Artistic rendering and computer vision

Non-pinhole cameras are also used in multiperspective artistic rendering. In one system, PPCs are attached to individual scene objects, and the resulting sprites are composited in a multi-projection image [Agrawala 2000]. The approach has the disadvantages of not scaling with scene complexity, of difficult visibility ordering, and of not supporting multiple perspectives per object. Another system [Yu 2004b] partitions an image plane into general linear camera (GLC) triangular images. A GLC is constructed from three given rays [Yu 2004a, Ponce 2009] so it offers some ray modeling flexibility. However, combining several GLCs is non-trivial. The solution adopted [Yu 2005] 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 been used for impressionism, cubism, and abstract aesthetic video effects [Klein 2002]. The collage artistic photography technique has also been reproduced computationally [Nomura 2007]. Individual PPC images are only roughly aligned, resulting in a multi-perspective image with visible seams, which are kept intentionally in order to convey subject or camera motion. Like in most previous multiperspective work, we strive to avoid seams; seams are visible only when an object moves between two frusta in a real-world graph camera, or when there is no geometry to hide a frustum split line.

Computer vision research has also been devoted to overcoming the limitations of the PPC model. Whereas early efforts targeted producing omnidirectional cameras with the emphasis on preserving ray concurrency, more recent efforts developed non-pinholes. For example, the disparity embedded in a single image acquired with a catadioptric non-pinhole has been used for scene capture [Kuthirummal 2006]. Although the first physical implementation of the graph camera described here relies exclusively on conventional video cameras we will consider integrating reflective surfaces in the future for added flexibility.

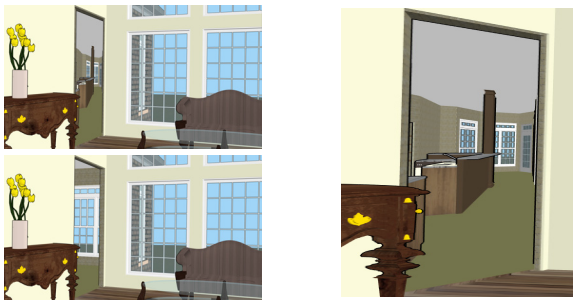

Figure 6 Portal-based graph camera image (*top left* and fragment *right*) and PPC image for comparison (*bottom left*).

3. 3 Visualization

The graph camera addresses the important visualization problem of alleviating occlusions in complex datasets in order to increase the visualization payload. One approach is to distort the dataset such that entities of interest are given preference and map to disjoint image locations (e.g. Generalized Fisheye Views [Furnas 1986], the Hyperbolic Browser [Lamping 1996], the EdgeLens [Wong 2003], the Balloon Probe [Elmqvist 2005]). Although the graph camera model can be described as a distortion of 3-D space that makes the scene look like the graph camera image when rendered with a conventional PPC, designing the visualization directly in image space provides greater flexibility for overcoming occlusions and for achieving continuity. In technical illustration occlusions are handled using transparency and cutaway [Diepstraten 2002, 2003] techniques which respect global spatial relationships, but visualize outer layers with little detail.

4 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 to explore 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. The constructors can be used in combination but are presented here individually for clarity.

4. 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 6 the graph camera captures a good view of the adjacent kitchen while the PPC sees only an uninteresting corner through the portal.

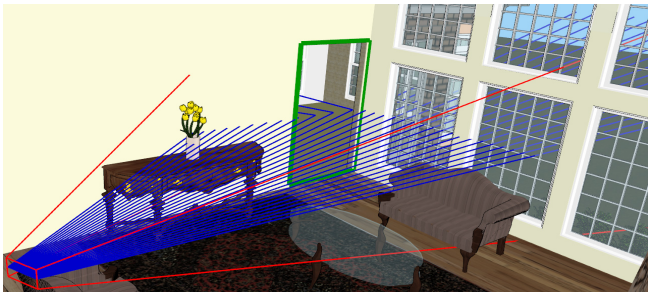

Figure 7 Visualization of portal-based graph camera from Figure 6.

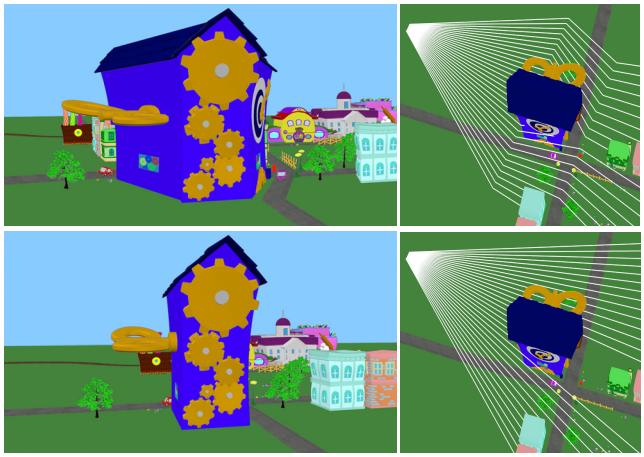

Figure 8 Occluder-based graph camera image (*top left*), PPC image for comparison (*bottom left*), and ray visualizations (*right*).

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 7 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 enters through the portal.

4. 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 5). In Figure 8 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 Maze-based constructor

Portal and occluder based graph cameras have good disocclusion capability when the current user position is relatively close to the portal or occluder. However, when the current position is far away, the root frustum projection of the portal or occluder is small, which limits the quality of the visualization of the disoccluded regions. Consider for example the bottom right image in Figure 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 how far the portal is. As a preliminary step, the set of possible desired user paths is modeled

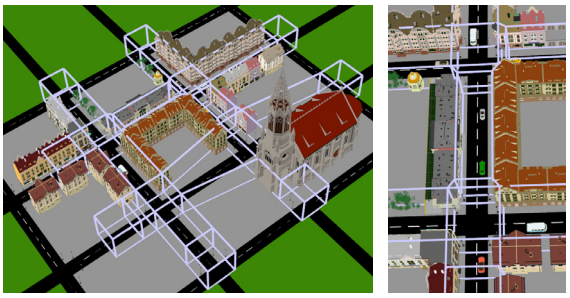

Figure 9 Visualizations of maze used to construct graph camera.

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 9). 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.

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 10 (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 10, bottom left.

The forward graph camera is constructed using the maze (blue lines in Figure 10) 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 points E, a, b, and a field of view parameter. PPC_0 rays that hit between E and b are directed to the left corridor, and rays between b and a 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

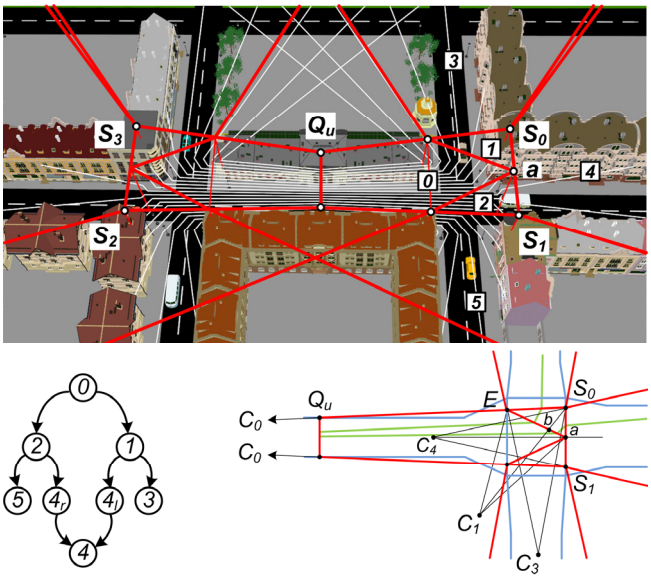

Figure 10 Visualization of forward and backward maze-based graph cameras for Figure 1 (*top*), graph of PPC frusta for forward graph camera (*bottom left*) and construction details (*bottom right*).

Figure 11 Top half of uneven split raw graph camera image.

constructed using points E and S_0 and a field of view parameter. PPC_{4l} is built using points S_0 and a, and field of view parameter. The right half of the graph camera is constructed similarly.

The raw image produced by the forward graph camera is shown in Figure 11, 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, and that splits the image along discontinuities to obtain the final image shown in Figure 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 10) and are shown as grey vertical lines in Figure 11. At a conceptual level, the two step process first achieves the desired disocclusion effect and then rearranges the image regions optimizing visualization eloquence.

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 (also see video). 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.

5 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.

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 3. The resulting collage (Figure 12, 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,

Figure 12 Collage and 3-D remodeling summarization images.

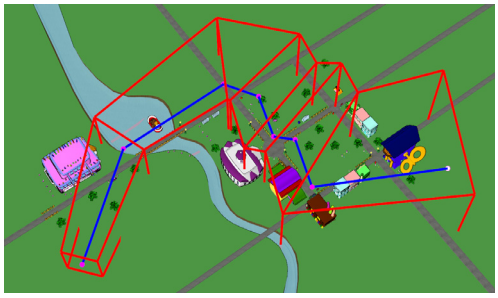

Figure 13 Graph camera model visualization for Figure 3.

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 12, 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.

5. 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 13 shows the graph camera constructed interactively to make the cartoon town summarization image in Figure 3. 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.

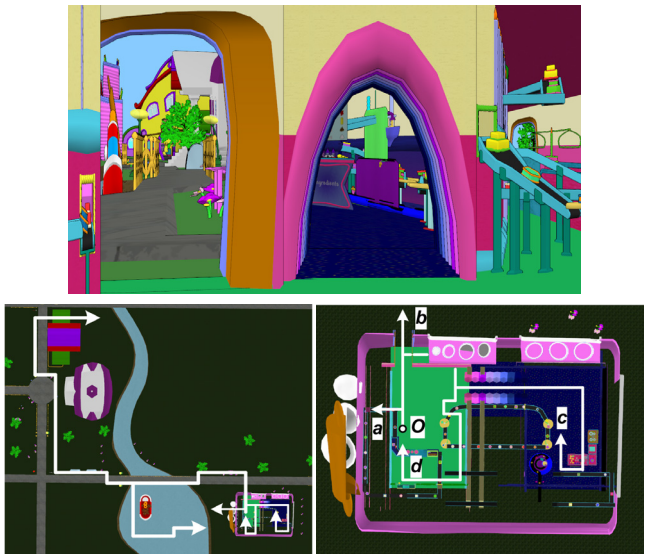

Figure 14 Summarization image rendered with a graph camera constructed with the recursive maze-based algorithm (*top*), camera model visualizations (*bottom*), overall and inside bakery.

5. 2 Recursive maze-based construction

Another method for building powerful 3-D scene summarization graph cameras is to use a maze that connects important scene regions, similar to the maze used in scene exploration (Section 4). 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 use breadth first search. In addition to the RFL (right, front, 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 14 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*).

6 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.

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 2 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. Narrow fields of view are also desirable for the PPCs of a physical graph camera like the one used in Figure 2 in order to control the amount of perspective foreshortening along the corridors. Large fields of view waste 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

Figure 15 Illustration of near clip plane simulation by background subtraction. The back of the subject is sampled by the video 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 right corridor acquired before the subject appeared (*bottom*).

background subtraction, as illustrated in Figure 15. 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 2, left). Since the pixels erased are not live, some applications could prefer that they be highlighted in red to mark the uncertainty (Figure 15, 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, similarly 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 (see video).

7 Discussion

Contributions. We have presented a framework for designing and rendering multiperspective images based on a non-pinhole that integrates several conventional pinhole cameras into a C^0 continuous and non-redundant image. Graph camera construction takes into account the targeted 3-D scene. We have developed an interactive constructor as well as constructors that take advantage of portals, occluders, and user path mazes in the scene. The graph camera benefits are illustrated in the context of scene exploration, of scene summarization, and of real-world scene visualization.

Performance. Graph camera images are rendered efficiently in feed-forward fashion leveraging a fast projection operation. Performance was measured on a 4 Core Xeon 3.16GHz 4GB workstation with an nVidia 280 GTX graphics card, for 1,280x720 output resolution (Table 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 graph camera image is not C^1 continuous, which translates into abrupt perspective changes between

Scene	Tris	Constructor	Illustration	Frusta	FPS
House	958K	Portal	Fig. 6, video	1-6	27
House	958K	Interactive	Fig. 15, video	13	6.7
Cartoon town	971K	Occluder	Fig. 8, video	5	31
Cartoon town	971K	Interactive	Fig. 3, video	7	25
Cartoon town	971K	Maze	Fig. 17, video	35	5.2
City	4,123K	Maze	Fig. 1, video	12	12
Real world 1	N/A	Proxy	Fig. 2, video	3	9.0
Real world 2	N/A	Proxy	Video	4	9.5

Table 1 Graph camera rendering frame rates.

connected frusta (see video). Another limitation is that the current maze-based constructor assumes straight maze edges and right angle 4-way intersections. As discussed, limitations specific to real world graph cameras relate to video camera placement, to z 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.

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). The graph camera is a multiperspective technique, and, like all other such techniques, it sees around occluders by introducing distortions that perturb global spatial relationships. 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.

8 Future Work

8. 1 Potential graph camera improvements

One area of future work is improving the quality of transitions between frusta. Whereas currently a transition is defined by a plane, using a transition region with a non-zero volume and interpolating the projection of points in the transition region should achieve a smooth transition. Interpolating the projections of the two frusta is equivalent to curving the rays of the graph camera. The projection cost is only marginally higher but the non-linear projection raises non-linear rasterization issues, which can be mitigated by subdivision. For real-world scenes a first goal is to achieve C^0 transitions. Whereas per-pixel depth would be sufficient, fortunately it is not also necessary: all that is needed is the line of pixels along which to connect the two video frames.

We have introduced an additional stage to the multiperspective rendering pipeline that maps the raw image to a display surface increase the eloquence of the final image. We believe that further improvements are possible by using 3-D display surfaces as well as by adding annotations to the image (e.g. animated arrows).

Shading for multiperspective rendering also needs further attention. One challenge is posed by view-dependent effects, since now there are multiple "eyes". Should a car reflect the side streets exposed by the graph camera? Another challenge are contradictory shadows in the graph camera image. Graph camera rendering can be described as rendering a distorted world with a conventional camera, which presents the opportunity to experiment with consistent viewpoint and lighting.

We will investigate developing graph camera constructors that do not require a predefined scaffolding to guide the construction. We will consider a top-down constructor which starts from the root frustum and then automatically finds openings in the scene

through which to extend rays. We will also consider a bottom-up constructor which starts from a predefined set of regions of interest. We will also investigate building graph cameras from moving video cameras, with challenges that include tracking the cameras and updating the graph camera topology.

The benefits of current and future graph cameras will be quantified through user studies with tasks such as finding static, moving, or transient targets, and with variables such as graph camera complexity, graph camera intrinsic parameter values, and display surface design. We hope that the flexible multiperspective rendering framework offered by the graph camera will also be used for generating images to study visual perception from non-pinhole stimuli for which virtually no previous work exists.

8. 2 Future directions and applications

Whereas in the applications explored in this paper the graph camera image is directly presented to the user, the graph camera image can also serve as an intermediate scene representation from which the final output image is computed efficiently. For example, a graph camera image with per pixel depth could provide a high-fidelity environment map for accelerating costly effects such as reflection or ambient occlusion. If the 6 PPCs corresponding to the faces of the cube map are replaced with graph cameras, rays could be designed as to sample all surfaces that are likely to be mirrored by the reflector. A key observation is that intersecting a graph camera depth map with a ray is not fundamentally more expensive than intersecting a conventional depth map: the intersection search space is still one-dimensional, defined by the piecewise linear projection of the ray.

Another example is the use of graph camera images to alleviate the bandwidth bottleneck in remote rendering. A conventional PPC image sent from the server has a lifespan of a single frame at the client. A graph camera depth image however can be used to render several high-quality frames without any additional data from the server. Indeed, a graph camera image stores more samples than those visible from the viewpoint of the root frustum, which helps reconstruct frames as the viewpoint translates. A graph camera image also has good coherence thus conventional compression algorithms apply. Challenges that need to be overcome in this context include graph camera construction to minimize image size and maximize viewpoint translation range and encoding of view dependent color into the graph camera image for high-quality frame reconstruction at the client.

As dataset sizes increase, scientific visualization pipelines shield the user from the full complexity of the dataset through automated preprocessing steps that identify potential data subsets of interest. The graph camera is well suited for visualizing several such data subsets simultaneously in a single image which could accelerate the process of weeding out false positives. The graph camera could also help uncover relationships between distant data subsets, which is crucial in dynamic contexts where the saliency of such relationships quickly degrades over time.

We have begun exploring the use of the graph camera as a tool for integrating video feeds for surveillance. The graph camera can also offer a comprehensive occlusion-free view of the scene to the operator of a robot or of a weapon system. The operator can trivially select a target directly in the graph camera image and the guidance system, which is aware of the geometry of the graph camera, executes the maneuver transparently.

The graph camera advocates a departure from the conventional approach of using a simple and rigid camera model in favor of designing and dynamically optimizing the camera model for each application, for each 3-D scene, and for each desired view. We foresee that this novel paradigm will benefit many other applications in computer graphics and beyond.

References

AGRAWALA, M., ZORIN, D., AND MUNZNER, T. 2000. Artistic Multiprojection Rendering. *EG Workshop on Rendering Techniques 2000*, 125-136.

AGARWALA, A. ET AL., 2006. Photographing long scenes with multi-viewpoint panoram. *SIGGRAPH* 2006, 853-861.

CHEN, E. QuickTime VR. *SIGGRAPH '95*, 29-38.

DIEPSTRATEN, J., WEISKOPF, D., AND ERTL, T., 2002. Transparency in interactive technical illustrations. *Computer Graphics Forum*, 21(3):317–325.

DIEPSTRATEN, J., WEISKOPF, D., AND ERTL, T., 2003. Interactive cutaway rendering. *EUROGRAPHICS 2003*, 523–532.

ELMQVIST, N., 2005. BalloonProbe: Reducing Occlusion in 3D using Interactive Space Distortion. *ACM Symposium on Virtual Reality Software and Technology 2005*, 134-137.

FURNAS, G. W, 1986. Generalized fisheye views. *ACM CHI'86 Conference on Human Factors in Computer Syst.*, 16–23.

GORTLER, S. J., GRZESZCZUK, R., SZELISKI, R., AND COHEN, M. F. 1996. The lumigraph. *SIGGRAPH '96*, 43-54.

KLEIN, A. W., SLOAN, P. J., FINKELSTEIN, A., AND COHEN, M. F. 2002. Stylized video cubes. *Siggraph/Eurographics Symposium on Computer Animation* SCA '02, 15-22.

KUTHIRUMMAL S. AND NAYAR S. Multiview Radial Catadioptric Imaging for Scene Capture. *SIGGRAPH '06*.

LAMPING, J., AND RAO, R., 1996. The Hyperbolic Browser: A focus + context technique for visualizing large hierarchies. *Journal of Visual Languages and Computing*, 7(1):33–35.

LEVOY, M. AND HANRAHAN, P. 1996. Light field rendering. *SIGGRAPH '96*, 31-42.

MEI, C., POPESCU, V., AND SACKS, E. 2005. *The Occlusion Camera*. *Eurographics 2005*, 335-342.

NOMURA, Y., L. ZHANG, AND S. NAYAR 2007. Scene collages and flexible camera arrays. *EG Symp. on Rendering*, 2007.

PONCE, J. 2009. What is a camera. *IEEE Conference on Computer Vision and Pattern Recognition,* 2009.

POPESCU, V., ALIAGA, D. 2006. The depth discontinuity occlusion camera. *ACM Symp. on Interactive 3-D Graphics*, 139-143.

RADEMACHER, P. AND BISHOP, G. 1998. Multiple-center-of-projection images. *SIGGRAPH '98*, 199-206.

ROMAN, A., GARG, G., AND LEVOY, M. 2004. Interactive Design of Multi-Perspective Images for Visualizing Urban Landscapes. *Visualization'04*, 537-544.

ROMAN, A., LENSCH, H. P. A. 2006. Automatic Multiperspective Images. *Eurogr. Symposium on Rendering*, 537-544.

ROSEN, P., AND POPESCU, V. 2008. The epipolar occlusion camera. *ACM Symp. on Interactive 3-D Graphics*, 2008, 115-122.

SEITZ, S. M. AND KIM, J. 2003. Multiperspective Imaging. *IEEE Comput. Graph. Appl.* 23, 6 (Nov. 2003), 16-19.

SHADE, J., GORTLER, S., HE, L., AND SZELISKI, R. 1998. Layered depth images. *SIGGRAPH '98,* 231-242.

SZELISKI, R., AND SHUM, H.-Y. 1997. Full view panoramic image mosaics and texture-mapped models. *SIGG '97*, 251-258.

WONG N., CARPENDALE, M. S. T., AND GREENBERG, S., 2003. EdgeLens: An interactive method for managing edge congestion in graphs. *IEEE Symp. on Info. Vis. 2003*, 51–58.

WOOD, D. N., FINKELSTEIN, A., HUGHES, J. F., THAYER, C. E., AND SALESIN, D. H. 1997. Multiperspective panoramas for cel animation. *SIGGRAPH '97*, 243-250.

YU, J., AND MCMILLAN, L. 2004. General Linear Cameras *European Conf. on Computer Vision 2004*, Vol. 2, 14-27.

YU, J., AND MCMILLAN, L. 2004. A Framework for Multiperspective Rendering. *Eurographics Symposium on Rendering 2004,* 61-68.

YU, J., AND MCMILLAN, L. 2005. Analyzing Reflections via Multiperspective Imaging. *Computer Vision and Pattern Recognition (CVPR)*, 2005, 117-124.