A Flexible Framework for Interactive Multiperspective Visualization

<Author names and affiliations withheld for double-blind review>

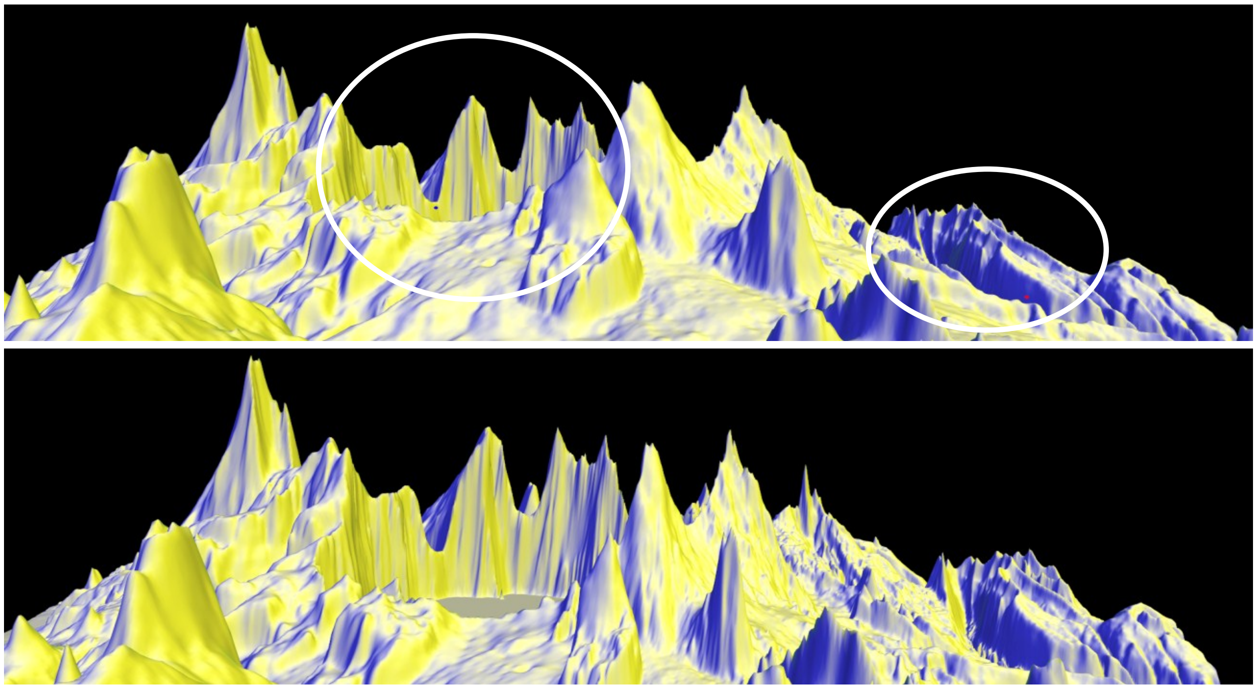
Fig. 1. Conventional visualization of terrain dataset (top) and multiperspective visualization constructed with our framework (bottom). The viewpoint was modified for two regions individually to disocclude a lake (left) and a valley (right).

Abstract—A conventional image can only visualize the parts of a dataset to which there is direct line of sight from the image viewpoint. In complex datasets, occlusions abound, which limits visualization bandwidth. Multiperspective visualization promises to remove this single viewpoint limitation by integrating samples captured from multiple viewpoints into a continuous image. We present a framework for designing multiperspective visualizations with great flexibility by manipulating the underlying camera model. We construct multiperspective visualizations in one of three ways: in target tracking construction, one or several data subsets of interest (i.e. targets) are visualized where they would be seen in the absence of occluders, as the user navigates or the targets move; in top-down construction, the viewpoint is altered for individual image regions to avoid occlusions; in bottom-up construction, input conventional images are connected into a multiperspective image. The multiperspective images are rendered at interactive rates, with the help of the GPU, leveraging a fast projection operation provided by the underlying camera model.

Index Terms—Occlusion management, camera models, multiperspective visualization, interactive visualization

1 Introduction

Most images used in computer graphics and visualization are computed with the conventional planar pinhole camera model, which approximates the human eye. Whereas this is essential in applications such as virtual reality where the goal is to make users believe that they are immersed in the scene rendered, researchers in visualization have recognized that the limitations of conventional images are not always warranted. One such limitation is a reduced field view, which has been addressed with panoramic camera models such as fisheyes. A second limitation is that conventional images sample the dataset uniformly, oblivious to importance variations within the dataset. The focus plus context research direction addresses this limitation by devising mechanisms for allocating more image pixels to data subsets of higher importance.

A third limitation is that a conventional image samples a dataset from a single viewpoint and occlusions limit the visualization capability of the image. One approach for overcoming occlusions is to rely on the user to navigate the viewpoint in order to circumvent occluders and to establish a direct line of sight to each data subset of potential interest. One disadvantage of such a sequential exploration is inefficiency: data subsets are explored one at a time, and the navigation path has to be retraced to achieve a systematic exploration of the entire dataset. A second disadvantage is that the user never sees more than a single data subset at a time and connections between subsets that are far apart in the visualization sequence can be missed. The problem is exacerbated in the case of time varying datasets, where the eloquence of a connection between distant data subsets could be transient. The problem can be alleviated by visualizing the dataset in parallel with multiple conventional images. The user sees several data subsets simultaneously, but the visualization is discontinuous at the borders of the individual images and the user has to examine one image at the time which reduces the benefits of the parallel visualization.

Another approach for overcoming occlusions is multiperspective visualization, which integrates dataset samples captured from

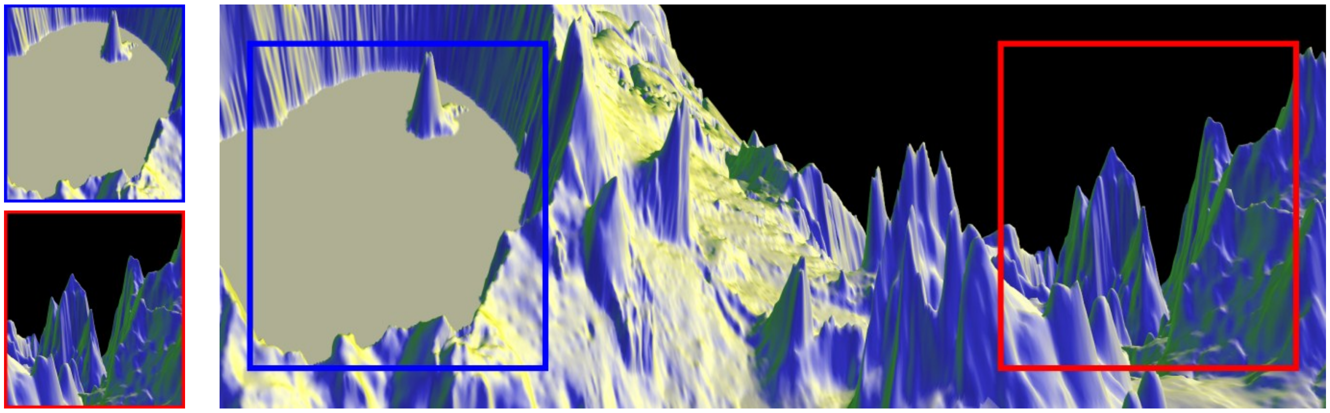
Fig. 2. Conventional images with different viewpoints (left) integrated into a multiperspective visualization (right) of a terrain dataset.

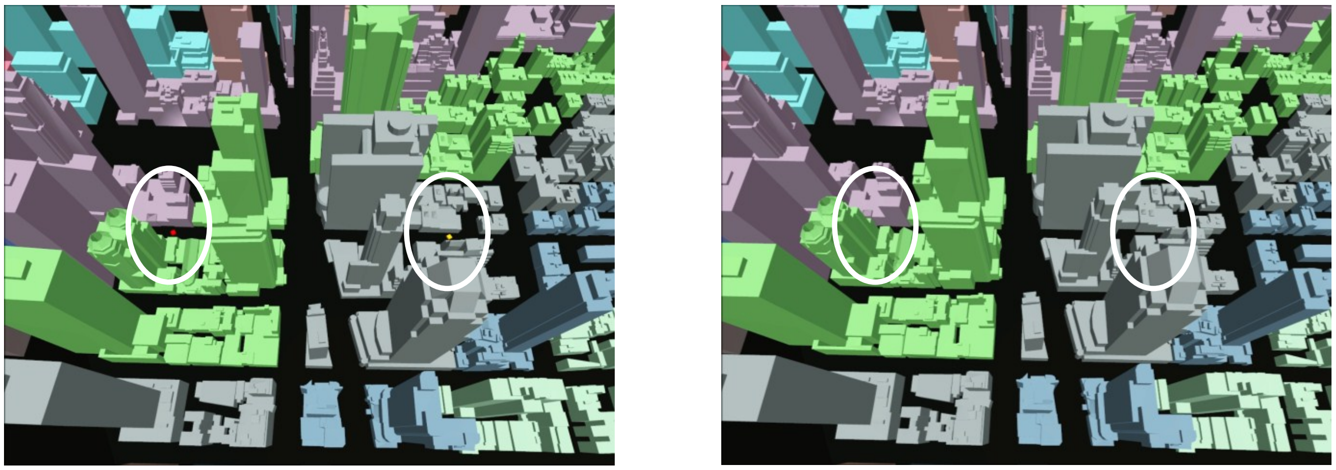
Fig. 3. Multiperspective visualization (left) of an urban dataset showing two targets (red and yellow dots) that are occluded in a conventional visualization (right). The targets are shown where they would be visible in the conventional visualization in the absence of occluders.

multiple viewpoints into a single, continuous “multiperspective” image. The multiperspective image enables parallel visualization without the high cognitive load of the multitude of disparate contexts presented by individual conventional images. Multiperspective visualization can be seen as a generalization of focus plus context visualization. Multiple focus regions are visualized simultaneously connected by continuous context, but without the restriction that all focus regions be visualized from the same viewpoint. The challenge is to construct a multiperspective visualization that provides good control over the multiple viewpoints from where the dataset is sampled, while maintaining image continuity, as needed for visualization efficacy, and while maintaining rendering efficiency, as needed to support interactive visualization and time-varying datasets.

In this paper we present a flexible framework for interactive multiperspective visualization. We also refer the reader to the accompanying video. The multiperspective visualization is constructed in one of three ways.

In *top-down* construction, the multiperspective visualization is obtained by altering the viewpoint for individual regions of an input conventional image (Fig. 1). Consider a scenario where the user visualizes a dataset with a high-resolution, large field of view conventional image. The user can select image regions of interest and modify the viewpoint for each individual region interactively to zoom in and to alleviate occlusions.

In *bottom-up* construction, the multiperspective visualization is obtained by seamlessly integrating two input conventional images. Consider a scenario where a user explores a dataset through conventional interactive visualization; the view parameters of images of interest are saved as they are encountered; the system allows constructing automatically a multiperspective visualization that combines any two of the saved conventional images (Fig. 2). The input images appear undistorted as subregions of the multiperspective image.

In *target tracking*, the visualization is constructed to avoid occlusions to one or several data subsets of interest, i.e. targets. As the targets move, the visualization adapts automatically to keep the targets visible (Fig. 3). The targets are shown where they would be visible in the conventional image in the absence of occluders, which conveys to the user the correct direction to the target. When the targets move to locations where they are visible, the multiperspective visualization reverts automatically to a conventional visualization, as no disocclusion is needed anymore.

The multiperspective image is constructed by rendering the dataset with a camera with piecewise linear rays. Some of the rays are designed to circumvent occluders and to reach the data subsets of interest, while the remaining rays are designed to connect the data subsets of interest with continuous context. The camera is assembled from a small number of *camera segments*, i.e. simple cameras with linear rays, which enables fast construction and fast updates of the camera model. Moreover, the segments provide fast projection which allows rendering the multiperspective image efficiently, on the GPU, by projection followed by rasterization.

2 Prior Work

The more 3D datasets grow in complexity, the more occlusions stemming from the single viewpoint restriction of conventional images become a limiting factor in visualization. Occlusions are an open research problem that has been approached from many directions [7]. We group prior work occlusion management techniques based on the approach taken and we review in greater detail prior multiperspective visualization approaches.

2.1 Transparency, Cutaway, & Explosion Techniques

A straightforward approach for alleviating occlusions is to render occluders transparently (e.g. [9]). The advantage is that the dataset is visualized without distortions. However, transparency techniques are limited to one or two occluding layers after which the blended transparent layers are difficult to discern. For example, visualizing a target occluded by several buildings in an urban dataset by making the occluding buildings partially transparent fails to convey the identity and order of the occluding buildings, as well as the position of the target relative to the occluding buildings. Compared to transparency techniques, our approach displaces occluders, which introduces distortions, but which brings the advantages of visualization clarity and scalability with occlusion complexity.

Another approach is to remove all but the peripheral region of occluders, revealing the data subset of interest that would otherwise be occluded (e.g. [4]). The advantage of such cutaway techniques is a clear view of the subset of interest, but that comes at the cost of an incomplete visualization of the occluding layers. The visualization shows a truncated dataset. In the urban dataset visualization example, removing the occluding buildings reveals the target but it also removes the context needed to locate the target. Our approach "moves" but does not remove occluders, which preserves context.

Explosion techniques subdivide occluders interactively [3] or algorithmically [10], and then move the parts centrifugally away from a data subset of interest to remove occlusions. The process can be repeated two or three times recursively, revealing for example subsystems, assemblies, and parts of a complex mechanical system. The occluders are part of the visualization, but the visualization contributes little more than a sorted inventory of the occluders, as the topology of the dataset is perturbed by the "explosion". Our approach moves occluders without changing the topology of the dataset.

Transparency, cutaway, and explosion techniques have the advantage that they do not require access to the occluded data subset of interest, as they create their own path from the viewpoint to the subset. Therefore they can disocclude subsets that are completely enclosed, such as an internal component of an engine. Deformation and multiperspective techniques, which we discuss next, can only disocclude if there is an access path to the occluded subset of interest, with the benefit of a less intrusive, topology preserving modification of the dataset.

2.2 Deformation and Multiperspective Techniques

Gaining an unobstructed line of sight to a data subset of interest can also be done by deforming the dataset. The idea was first used in the context of 2D data visualization, e.g. in the context of graph visualization [17], and it was subsequently extended to 3D datasets, e.g. in the context of short route [6] or car navigation [15] visualization. The goal is to achieve the desired disocclusion by deforming the dataset as little as possible. The deformation approach is the dual of the multiperspective approach. Visualizing a distorted dataset with a conventional camera can be seen as visualizing the original dataset with a multiperspective camera. For example, the multiperspective image in Fig. 3 could be obtained by distorting the urban dataset and then visualizing it with a conventional camera. The difference is that multiperspective visualizations are constructed by modifying the underlying camera model. The disocclusion effect is designed with greater control directly in the image domain, as opposed to indirectly in the 3D dataset.

Multiperspective visualization originates in art, where the single viewpoint constraint is occasionally abandoned in the interest of artistic expression, effects that were replicated by computer graphics systems [2]. Multiperspective images have also been used to integrate photographs taken from different viewpoints (e.g. street panoramas [1, 14]). In 2D animation, a single multiperspective panorama provides an animation sequence by sliding the frame rectangle over the panorama on a predefined path [18]. Both street and 2D animation panoramas are limited to special scenes.

The generality and flexibility of multiperspective visualization increased through innovations at camera model level. Multiple center of projection images [13] are obtained by rendering a 3D dataset with a one-column camera that slides along a path (i.e. a push-broom camera). Samples from thousands of viewpoints are integrated into a continuous image. The user has good control over viewpoint selection by designing the acquisition path. However, rendering the image is too expensive for interactive visualization or time-varying datasets, as it involves one rendering pass for each image column. Our multiperspective rendering framework relies on a more efficient parameterization of the ray space based on a few (i.e. 10-20) camera segments, which provides the needed disocclusion flexibility without sacrificing rendering performance.

The general linear camera (GLC) [19] is at the other end of the multiperspective camera complexity spectrum by possibly being the simplest non-pinhole camera. The rays of the GLC are obtained by interpolating three input non-concurrent rays. The GLC provides fast projection, which ensures rendering efficiency. Given a 3D point inside the GLC frustum, one can compute the barycentric coordinates of the point's projection on the triangular GLC image by solving linear equations. However, the original parameterization of rays does not provide continuity between adjacent GLC's that share two construction rays. A continuous GLC parameterization has been subsequently proposed [11], which comes at the cost of cubic projection equations. A single GLC doesn't have the disocclusion capability needed in multiperspective visualization, but we use continuous GLC's in our framework to model camera segments, as described in Section 3.

Occlusion cameras are a family of non-pinhole cameras designed to extend the viewpoint of conventional cameras to a view region [20]. The camera rays are bent at occluder silhouettes to capture "barely hidden" samples, which are samples that become visible for small viewpoint translations. The resulting image is a high-quality aggressive solution to the from-region visibility problem. Occlusion cameras generalize the viewpoint to a continuum of nearby viewpoints, whereas what is needed for multiperspective visualization is a generalization of the viewpoint to a small set of distant viewpoints.

The needed viewpoint generalization is provided by the graph camera [12]. Starting from a conventional planar pinhole camera, the graph camera is constructed through a series of frustum bending, splitting, and merging operations applied recursively. The resulting camera is a graph of planar pinhole camera segments. The piecewise linear rays are designed to circumvent occluders and to reach far into the dataset. Compared to the graph camera, the camera employed in our multiperspective framework is built from a mix of planar pinhole camera and continuous general linear camera (CGLC) segments. The CGLC segments enable frustum splitting while maintaining image continuity, thereby overcoming a major shortcoming of the graph camera (see Section 4.1). Graph cameras can only be constructed automatically using a 2D maze with right angle intersections as scaffold, whereas in our framework the multiperspective camera is built automatically, in 3D, to track targets or to integrate two input conventional images. Finally, the graph camera does not allow controlling where a data subset of interest is imaged, whereas our framework allows imaging a subset where it would be imaged in a conventional visualization. This enables the user not only to examine but also to locate the subset of interest.

Inspired by the nonlinear trajectory of light in proximity of large masses, multiperspective visualizations have been proposed based on curved rays [8, 16]. Curved rays have later been used for visualization outside of astronomy using ray segments connected with Bézier arcs [5]. The advantage of curved rays is that the transition from one viewpoint to the next is gradual, which reduces the distortion for objects that are imaged from more than one viewpoint. The cost is a lower rendering performance due to the higher camera complexity. We use piecewise linear rays, with C^0

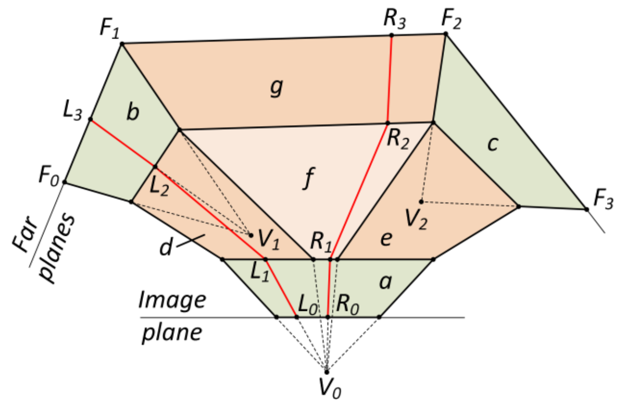

Fig. 4. Camera model that integrates two conventional images with viewpoints V_1 and V_2 into a multiperspective image. The green camera segments are implemented with conventional planar pinhole cameras. The orange camera segments are implemented with continuous general linear cameras. The rays are piecewise linear, e.g. $L_0L_1L_2L_3$ and $R_0R_1R_2R_3$.

continuity, but the C^1 continuous rays of the curved ray camera [5] could be integrated into our framework for applications where the additional cost is warranted .

3 Multiperspective Camera

Our visualization framework is based on a flexible multiperspective camera. In this section, we first describe the multiperspective camera model, we then describe how 3D datasets are rendered with the camera to obtain multiperspective images, and finally we describe how the camera is constructed in one of three ways to support interactive multiperspective visualization.

3.1 Camera Model

A camera model is a function that assigns a ray to each image plane sampling location. We have designed a multiperspective camera model based on the following considerations.

Flexibility; the camera should be able to integrate conventional planar pinhole camera frusta in order to generate a multiperspective image that shows subsets of interest undistorted, each from its own viewpoint.

Continuity; the camera rays should sample the entire space subtended by the regions of interest, without gaps, in order to connect the images of the regions of interest with continuous context.

Projection efficiency; given a 3D point inside its frustum, the camera model should provide a fast method for computing the image plane projection of the point in order to support efficient rendering on the GPU by projection followed by rasterization.

Fig. 4 gives a 2D illustration of our multiperspective camera model. A root planar pinhole camera segment *a* with viewpoint V_0 is used to integrate two leaf planar pinhole camera segments *b* and *c* with viewpoints V_1 and V_2. Segments *b* and *c* sample the data subsets of interest. Segments *b* and *c* are connected to *a* with camera segments *e* and *f*, each implemented with two continuous general linear camera (CGLC) frusta. Fig. 5 illustrates the two CGLC frusta of *d*. The rays of CGLC frustum $A_1B_1C_1A_2B_2C_2$ are defined by linearly interpolating the construction rays A_1A_2, B_1B_2, and C_1C_2.

Using Fig. 4 again, camera segments *f* and *g* sample the dataset in between the subsets of interest to connect the two images with continuous context. Each of them is implemented with two CGLCs. The camera has piecewise linear rays. Ray $L_0L_1L_2L_3$ has three segments, one for each of the camera segments it traverses. Line L_2L_3 passes through V_1 and line L_0L_1 passes through V_0. The far planes F_0F_1, F_1F_2, and F_2F_3 define the far boundary of the camera (here shown closer for illustration compactness).

The camera model allows tuning the percentage of the multiperspective image pixels allotted to each subset of interest (*b*

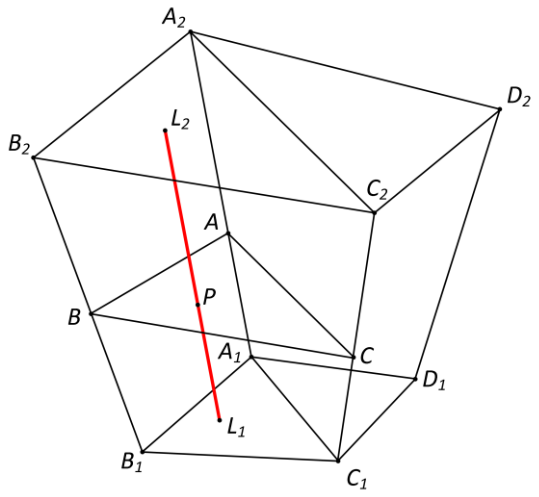

Fig. 5. Pair of adjacent continuous general linear camera frusta $A_1B_1C_1A_2B_2C_2$ and $A_1C_1D_1A_2C_2D_2$ like the ones used to model each of the camera segments *e, d, f,* and *g* in Fig. 4. The endpoints L_1 and L_2 of ray L_1L_2 have the same barycentric coordinates in triangles $A_1B_1C_1$ and $A_2B_2C_2$. 3D point *P* projects at L_1. *A, B,* and *C* split segments A_1A_2, B_1B_2, and C_1C_2 in the same ratio, and *P* has the same barycentric coordinates in *ABC* as L_1 has in $A_1B_1C_1$.

and *c*) and to the context (*g*) by changing how many of the rays of *a* are routed to each of the connecting camera segments *d*, *f*, and *e*. Fig. 4 shows a typical case where the context *g* is sampled at low resolution in favour of the subsets of interest *b* and *c*.

The multiperspective camera can morph to a conventional planar pinhole camera by straightening its piecewise linear rays to become single line segments. This is achieved by gradually translating V_1 and V_2 to V_0 and by aligning the connecting camera segments *d, f,* and *e* with rays from V_0, as shown in Fig. 6.

3.2 Rendering

A 3D dataset modeled with triangles is rendered on the GPU by projection followed by rasterization.

Projection

Given a 3D point *P*, the point is projected with each camera segment until a valid projection is found. Planar pinhole camera segments use the conventional projection. Fig. 5 illustrates CGLC projection. First we find a plane through *P* that splits the CGLC construction ray segments A_1A_2, B_1B_2, and C_1C_2 in the same ratio *t*:

$$t = \frac{A_1A}{A_1A_2} = \frac{B_1B}{B_1B_2} = \frac{C_1C}{C_1C_2} \qquad \text{Eq. 1}$$

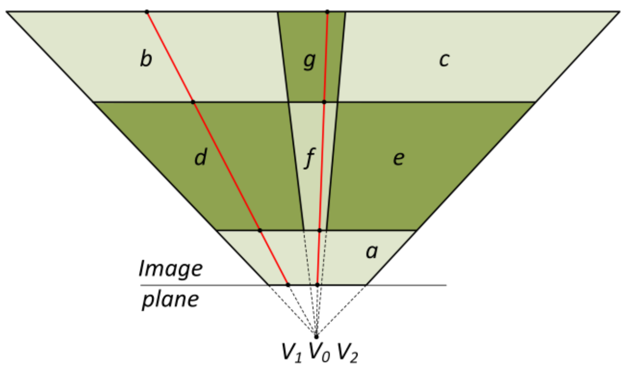

Fig. 6. Multiperspective camera model from Fig. 4 morphed to a conventional planar pinhole camera.

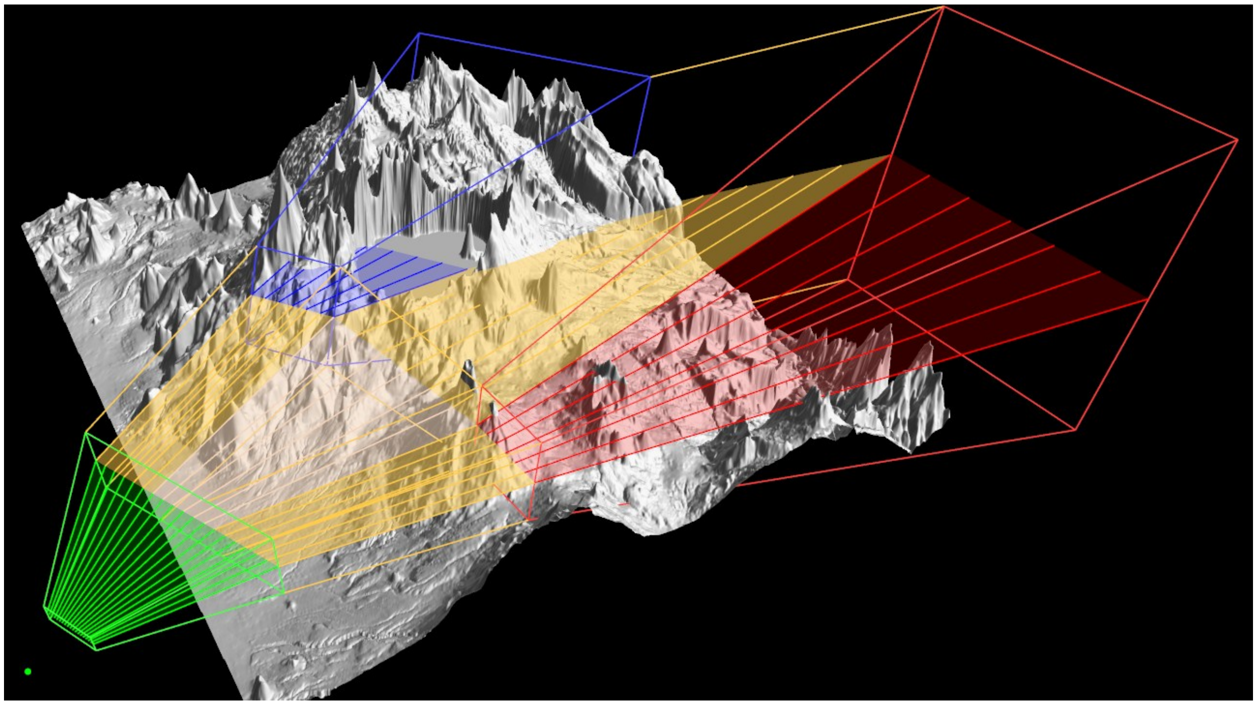
Fig. 7. Multiperspective camera model with the structure shown in Fig. 4 constructed in bottom-up fashion for the scenario shown in Fig. 2. The green, blue, and red frusta correspond to camera segments *a, b,* and *c* in Fig. 4.

Parameter t is computed by solving a cubic equation. Once t is known, points A, B, and C are known, and one can compute the barycentric coordinates (α, β, γ) of P in triangle ABC. (α, β, γ) are found by inverse barycentric interpolation, which implies solving a quadratic. Once (α, β, γ) are known, the projection L_1 of P onto $A_1B_1C_1$ can be computed as:

$$L_1 = \alpha A_1 + \beta B_1 + \gamma C_1 \qquad \text{Eq. 2}$$

However, one *does not* need to project a point P with each camera segment upstream of the segment that contains it. Instead, we map the vertices of the near face of each segment to the output multiperspective image during camera construction (and for each camera update). Using Fig. 5 again, let (u_A, v_A), (u_B, v_B), and (u_C, v_C) be the output image coordinates of A_1, B_1, and C_1. Then the output image projection (u, v) of P is computed as:

$$(u, v) = (\alpha u_A + \beta u_B + \gamma u_C, \alpha v_A + \beta v_B + \gamma v_C) \qquad \text{Eq. 3}$$

Rasterization

The projection of a triangle contained by a CGLC segment has curved edges. We approximate CGLC rasterization with conventional rasterization and we control the approximation error by subdividing any large triangle offline. Visibility is computed as usual through z-buffering. The z value used is the fractional parameter t that locates the 3D point within its camera segment, plus the camera segment index i. Camera segments are depth indexed from the root to the leaf segments. In Fig. 4, a has depth index 0, d, f, and e have depth index 1, and b, g, and c have depth index 2.

3.3 Camera Construction

Bottom-up

In bottom-up construction, the two planar pinhole camera segments b and c are given (Fig. 4), and the system constructs the remaining segments automatically. A precondition for the bottom-up construction algorithm is that the input segments b and c do not intersect. Intersecting segments lead to a redundant multiperspective image with the data subset inside the intersection being imaged multiple times. V_0 is positioned first in front of the near faces of b and c. The field of view of a is chosen to encompass the near faces of b and c, and to capture additional foreground and background according to user specified parameters. Finally, segments f and g are constructed to bridge the gap between (d, b) and (e, c). Fig. 7 illustrates the multiperspective camera model from Fig. 4 specialized for the scenario shown in Fig. 2. Only the central row of rays is shown.

Top-down

In top-down construction, the user starts with a planar pinhole camera and the multiperspective camera is built interactively. First, the user defines regions of interest that creates the partition shown in Fig. 6. Then the user modifies the viewpoints for each individual region. V_1 and V_2 translate away from V_0, and the initial planar pinhole camera morphs into the desired multiperspective camera. The camera model used to render the multiperspective image shown in Fig. 1 is similar to the camera model shown in Fig. 7.

Target Tracking

The construction of the multiperspective camera for tracking a single target is illustrated in Fig. 8. The target T is occluded from V_0. In the absence of the occluder, T would be visible at P. The construction algorithm reroutes the rays of the pixels around P to

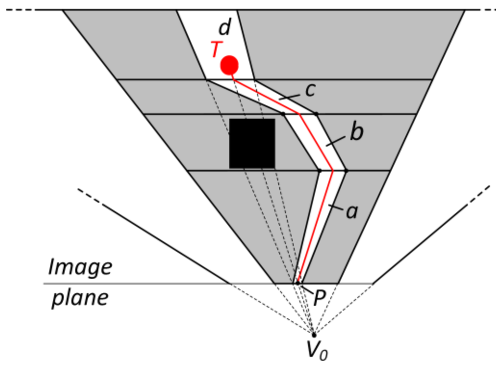

Fig. 8. Multiperspective camera disoccluding target T. Only the part of the camera affected by T is shown for conciseness. T is imaged where it would be seen in a conventional visualization in the absence of the occluder.

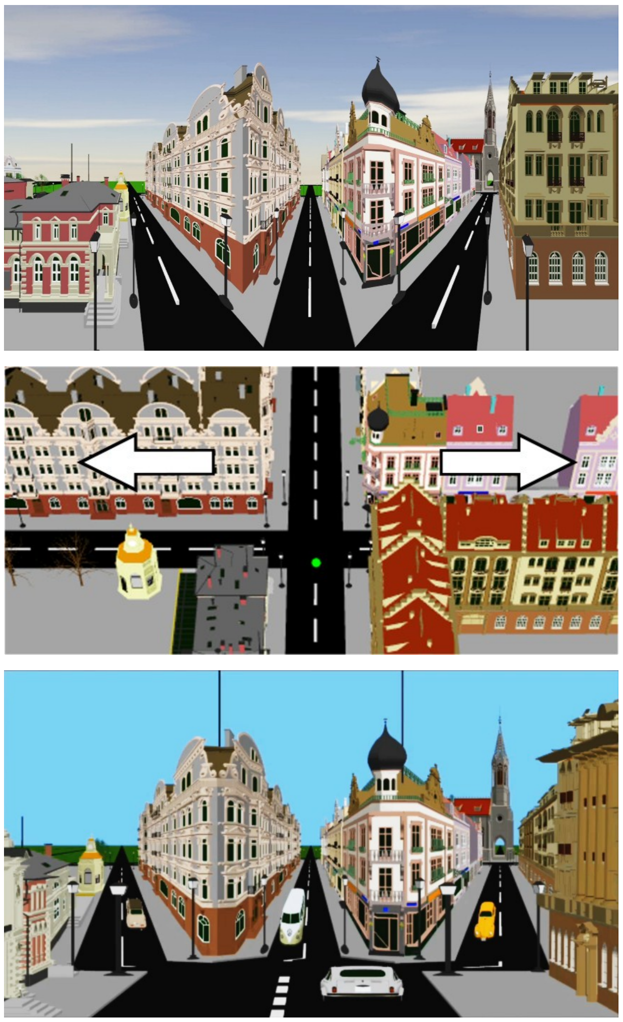

Fig. 9. Multiperspective image rendered with our framework (top) that integrates two opposite views (middle), and image rendered with the prior art graph camera framework for comparison (bottom). Our image is continuous, whereas the graph camera image has two vertical discontinuities over the sky (black lines).

circumvent the occluder. This way the target is visible in the multiperspective image at *P*, and the occluder is "pushed aside" (i.e. to the left). The rays are rerouted using four camera segments *a, b, c*, and *d*. The neighboring camera segments (grey) transition back to the planar pinhole camera, encapsulating the perturbation needed for target tracking. All camera segments have parallel near and far base planes, and plane *ABC* (Fig. 5) is simply constructed parallel to the base planes, which saves having to solve the cubic equation to find parameter *t* (see the *Projection* subsection of Section 3.2).

Multiple targets can be tracked at once, and the targets can converge and then diverge again (see video). We *do not* construct one complex multiperspective camera that disoccludes all targets. Rather we construct one multiperspective camera for each target independently, and compute the final projection of a vertex by projecting the vertex with each camera and by computing a weighted average of the preliminary projections. The weight of a preliminary projection is based on how much the projection is shifted from where the vertex would project with a conventional planar pinhole camera. Larger shifts correspond to larger weights. When targets do not overlap in the image plane, the projection of a vertex is affected by at most one target, and the vertex is projected as in the case of a single target. When two targets overlap, the preliminary and final projections of a vertex at the region of overlap are the same.

The camera models for each of the two targets tracked in Fig. 3 are constructed according to Fig. 8 except that the rays are bent in the vertical plane (and not in the horizontal plane), in order to circumvent the buildings occluding the target. The rays are bent enough to clear the tallest occluder along the line of sight to the target. The tallest occluder is found by projecting and tracing the segment connecting the root eye to the target center onto a height map of the city that is computed as a preprocess.

4 Results and Discussion

We partition the presentation and discussion of our results in three subsections: quality, performance, and limitations.

4.1 Quality

The major design concerns for our multiperspective visualization framework are visualization construction flexibility and image continuity. As shown in the images in this paper and in the accompanying video, flexibility has been achieved through a camera model that allows modifying the viewpoint for individual regions of a given image, connecting two conventional images with continuous context, and tracking one or more targets while avoiding occlusions.

Fig. 9 shows that our framework can connect two conventional images with opposite views into a continuous image, whereas a similar image rendered with the prior work graph camera framework [12] suffers from discontinuities which cannot always be hidden behind geometry. Fig. 10 shows that our framework can reveal a target hidden in a heavily occluded dataset with only minimal image distortions. The framework supports multiple moving targets, which can converge to the same image region, and then diverge again.

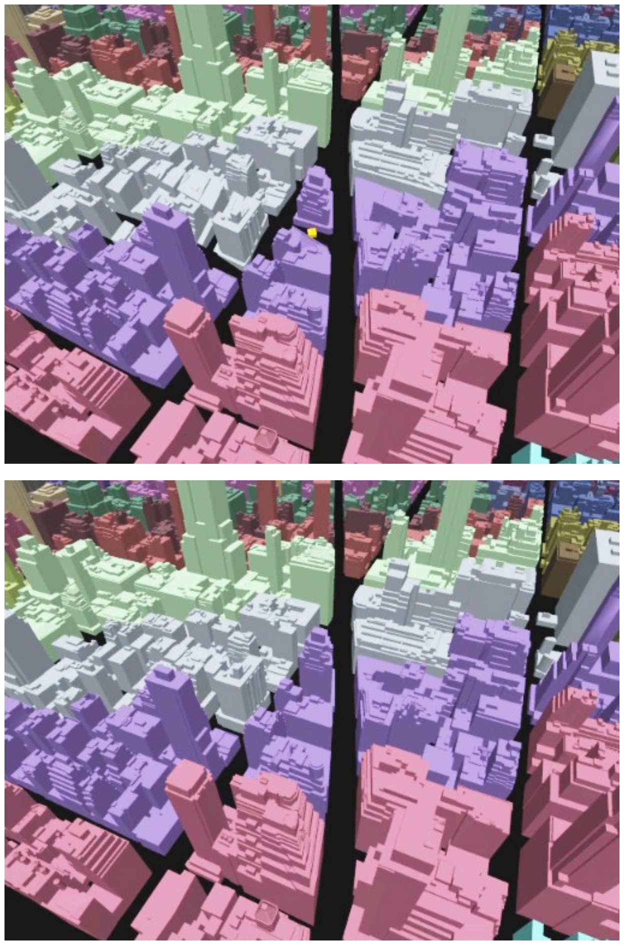

Fig. 10. Multiperspective image rendered with our framework that tracks a target in a heavily occluded dataset (top); conventional image with hidden target, for comparison (bottom).

4.2 Performance

The timing information reported in this paper was collected on an Intel Xeon E5-1660 3.3GHz workstation with 16GB of memory, and with an NVIDIA Quadro K5000 4GB graphics card. The implementation uses OpenGL and Cg GPU shaders. The multiperspective rendering is implemented with a vertex shader that implements the multiperspective projection. The vertex is projected with each camera segment until a valid projection is found. Vertices outside the 3D axis aligned bounding box of the camera segment are trivially rejected. We start with the leaf segments (i.e. *b, c,* and *g* in Figs. 4 and 6), which are the largest and are therefore most likely to contain the vertex. The multiperspective rendering performance of our framework is given in Table 1.

Table 1. Multiperspective rendering performance

Dataset	Tris. x1,000	View	Frame rate [fps]
Terrain	2,120	Fig. 1	28
		Fig. 2	24
Manhattan	3,692	Fig. 3	19
		Fig. 10	39
Eurotown	3,752	Fig. 9	13

The target-tracking multiperspective cameras used on the *Manhattan* dataset are the fastest because the near and far planes of the camera segments are parallel which simplifies projection (see Sections 3.2 and 3.3). When the cubic projection equation has to be solved (i.e. for the top-down and bottom-up constructions), we have found that solving the equation numerically is faster than evaluating the closed-form solution expression.

The nonlinear projection of the multiperspective camera implies that rasterization is not linear as well. In other words, the nonlinear projection applies not only to the vertices of a triangle, but also to its interior. There are two major approaches to nonlinear rasterization, and we have experimented with both of them. One approach is to actually perform nonlinear rasterization in the fragment shader. First one has to derive a method for approximating the image plane axis aligned bounding box of the projection of the triangle. Since the projected triangle now has curved edges, the approximation has to consider more than the vertices of the projected triangles. Then nonlinear rasterization can be performed in 3D, by intersecting the dataset triangle with the ray of each pixel of the axis aligned bounding box and interpolating rasterization parameters using the barycentric coordinates of the intersection point. A second approach is to approximate nonlinear rasterization with conventional rasterization but making sure the projected triangles are small enough. Online subdivision requires a geometry shader that issues a varying and potentially large number of primitives, which is a severe performance bottleneck. Offline subdivision has to be done in view independent way which can result in subdivisions that are either too coarse or to detailed for a particular viewpoint.

We have chosen the approach of offline subdivision. The cost of true nonlinear rasterization is unwarranted for the following three reasons. First, today's complex datasets are modeled with small triangles who can be rasterized conventionally with good results, without subdivision. Second, the focus regions are imaged with planar pinhole camera segments where conventional rasterization is accurate, and nonlinear rasterization is only used for regions whose role is limited to providing context. Third, using conventional rasterization implies that there are no changes at the fragment shader, which makes our framework portable to any already implemented visualization effect.

4.3 Limitations

In all the examples shown in the paper and in the video, the payload of the multiperspective visualizations is limited to *two* regions of interest: the user changes the viewpoint for *two* regions of an input conventional image in top-down construction, *two* conventional images are connected with continuous context in bottom-up construction, and *two* targets are disoccluded in target tracking construction. The framework is general and it allows extending the payload of the multiperspective visualization by constructing a multiperspective camera from any number of camera segments. The CGLC camera segments are powerful building blocks that can be assembled as needed by the application and the dataset, with perfect continuity across shared faces. For example, starting with a cubemap, camera segments could tile the entire space to obtain a complete 360° multiperspective panorama.

Whereas for our examples good performance was obtained by projecting every vertex with every camera segment, scalability with the number of camera segments requires a faster method for determining the camera segment that contains a vertex. The frusta of CGLC segments do not have planar side faces therefore a scheme that subdivides space hierarchically using planes (e.g. a binary space partitioning tree, or a kd-tree) will not separate two adjacent camera segments cleanly. Instead, one should use bounding volume hierarchies like the ones developed for ray tracing acceleration. The goal is to achieve an O(log *s*) projection time, where *s* is the number of camera segments.

Our multiperspective visualizations transition abruptly from one viewpoint to the next, as our camera model employs piecewise linear rays, with C^0 and not C^1 continuity. Objects that are imaged with two viewpoints appear distorted (Fig. 11, video). The graph camera framework uses curved rays modeled with Bézier arcs to transition

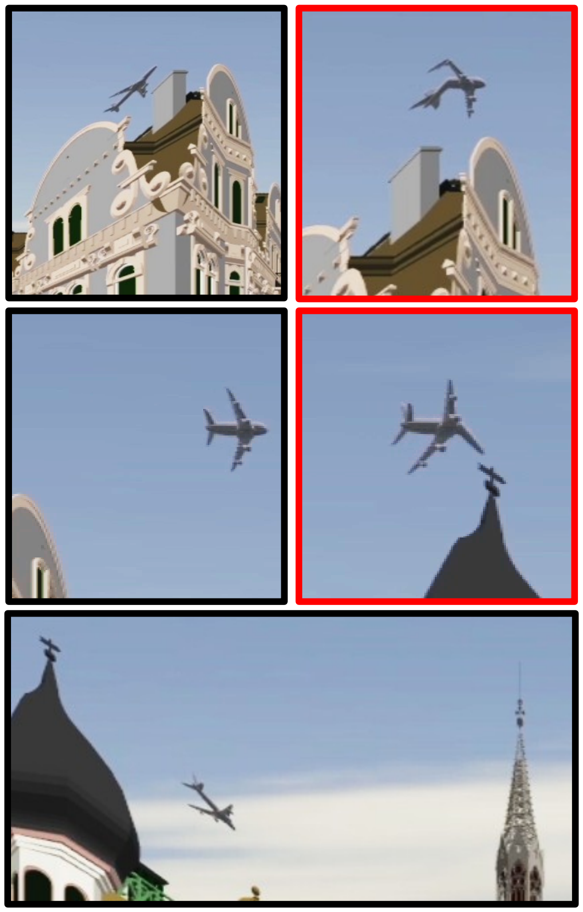
Fig. 11. Fragments of the multiperspective visualization from Fig. 9 showing an airplane flying overt the buildings. The airplane appears distorted as it crosses from one camera segment to the next (images with red border) and it is not distorted while completely contained by one segment (images with black border).

from one viewpoint to the next, solution that can be adapted to our framework. Another solution is to subdivide longitudinally the connective camera segments (i.e. *d, e, f,* and *g* in Fig. 4), in order to achieve a gradual viewpoint change. The rays remain piecewise linear, but the ray segments are shorter which reduces the change in direction from one segment to the next.

5 Conclusions and Future Work

We have presented a framework that advances the state of the art in multiperspective visualization. The framework allows constructing continuous multiperspective visualizations by changing the viewpoint for individual regions of an image, by integrating input images, and by disoccluding moving targets without distorting or displacing the target subimages, all of which are beyond the capabilities of prior art.

The framework relies on a flexible yet fast multiperspective camera. Whereas a conventional camera has a few parameters with which the application can interact directly (e.g. three rotations, three translations, focal length), our multiperspective camera comprises 10-20 camera segments which amounts to hundreds of parameters. The power of our framework comes from the three constructors that set all these parameters automatically to construct the desired multiperspective visualization. The constructors relieve the application from tedious low-level specification of the camera model, in favor of formulating high-level constraints that are satisfied automatically.

We have demonstrated our multiperspective visualization framework in the context of datasets modeled with triangles. The framework can be extended to support other geometric primitives, such as spherical particles, through tessellation. Opacity data can be integrated by the rays of our camera to achieve multiperspective volume rendering.

Another direction of future work is the extension to multiperspective visualization of real-world real-time datasets. Consider an urban scene captured with video cameras mounted at intersections, on cars, and on aircraft. The building geometry is known, for example from off-line LIDAR acquisition and conventional CAD modeling, like is the case for our *Terrain* and *Manhattan* datasets. The goal is to integrate the real-time video feeds into a multiperspective visualization that avoids occlusions for one or more regions of interest.

Our work advocates abandoning the traditional rigidity of the images used in visualization in favor of flexible images that are optimized for each viewpoint, dataset, and application.

Acknowledgements

<Withheld for double-blind review>

References

[1] A. Agarwala, M. Agrawala, M. Cohen, D. Salesin, R. Szeliski. Photographing long scenes with multi-viewpoint panoramas. *ACM Trans. on Graphics*, Vol. 25, No. 3, pp. 853–861, 2006.

[2] M. Agrawala, D. Zorin, T. Munzner. 2000. Artistic Multiprojection Rendering. *In Proc of the EG Workshop on Rendering Techniques*, pp. 125-136, 2000.

[3] S. Bruckner, M. E. Groller, Exploded View for Volume Data. *IEEE TVCG*, Vol 12, No. 5, pp. 1077-1084, 2006.

[4] M. Burns, A. Finkelstein. Adaptive cutaways for comprehensible rendering of polygonal scenes. *In ACM SIGGRAPH Asia*, 2008.

[5] J. Cui, P. Rosen, V. Popescu, and C. Hoffmann 2010. A curved ray camera for handling occlusions through continuous multiperspective visualization. *IEEE Transactions on Visualization and Computer Graphics 16* (November), 1235–1242.

[6] P. Degener, R. Schnabel, C. Schwartz, and R. Klein. Effective visualization of short routes. *IEEE TVCG*, Vol. 14, No. 6, pp. 1452-1458, 2008.

[7] N. Elmqvist, P. Tsigas. A Taxonomy of 3D Occlusion Management for Visualization. *IEEE TVCG.* Vol 14, No. 5, pp. 1095-1109, 2008.

[8] E. Groller. Nonlinear ray tracing: Visualizing strange worlds. *The Visual Computer*, Vol. 11, No. 5, pp. 263-274, 1995.

[9] J. Kruger, J. Schneider, R, Westermann. ClearView: An interactive context preserving hotspot vis technique. *IEEE TVCG*, Vol 12, No. 5, pp. 941-947, 2006.

[10] W. Li, M. Agrawala, B. Curless, D. Salesin. Automated generation of interactive 3D exploded view diagrams. *ACM Trans. Graph*. Vol. 27, No. 3, pp. 1-7, 2008.

[11] V. Popescu, C. Mei, J. Dauble, and E. Sacks 2006. An efficient error-bounded general camera model. In *Proceedings of 3rd International Symposium on Data Processing, Visualization, and Transmission.*

[12] V. Popescu, P. Rosen, N. Adamo-Villani. The Graph Camera. International Conference on Computer Graphics and Interactive Techniques, ACM SIGGRAPH Asia, 2009.

[13] P. Rademacher, G. Bishop. Multiple-center-of-projection images. *In Proc of SIGGRAPH '98*, pp. 199-206, 1998.

[14] A. Román, G. Garg , M. Levoy. Interactive Design of Multi-Perspective Images for Visualizing Urban Landscapes. *In Proc of Visualization '04*, pp. 537-544, 2004.

[15] S. Takahashi, K. Yoshida, K. Shimada, T. Nishita. Occlusion-Free Animation of Driving Routes for Car Navigation Systems. *IEEE TVCG.* Vol 12, No.5, pp. 1141-1148. 2006.

[16] D. Weiskopf, T. Schafhitzel, T. Ertl. GPU-based nonlinear ray tracing. Computer Graphics Forum 23, 3, pp. 625-633. 2004.

[17] N. Wong, M.S.T. Carpendale, S. Greenberg,. EdgeLens: An interactive method for managing edge congestion in graphs. In *Proc of the IEEE Symp. on Info Vis*, pp. 51–58. 2003.

[18] D. N. Wood, A. Finkelstein, J. F. Hughes, C. E. Thayer, D. H. Salesin. Multiperspective panoramas for cel animation. In *Proc of SIGGRAPH '97*, pp. 243-250, 1997.

[19] J. Yu, L. McMillan. General Linear Cameras. *In Proc of the European Conference on Computer Vision (ECCV)*, Vol. 2, pp. 14-27, 2004.

[20] C. Mei, V. Popescu, and E. Sacks. *Computer Graphics Forum,* Vol. 24, issue 3, 2005.