Artistic Multiprojection Rendering

Maneesh Agrawala* Denis Zorin† Tamara Munzner*
*Stanford University †New York University

Abstract

In composing hand-drawn images of 3D scenes, artists often alter the projection for each object in the scene independently, thereby generating multiprojection images. We present a tool for creating such multiprojection images and animations, consisting of two parts: a multiprojection rendering algorithm and an interactive interface for attaching local cameras to the scene geometry. We describe a new set of techniques for resolving visibility between geometry rendered with different local cameras. We also develop several camera constraints that are useful when initially setting local camera parameters and when animating the scene. We demonstrate applications of our methods for generating a variety of artistic effects in still images and in animations.

1 Introduction

In computer graphics we typically use a single linear projection – often a perspective projection – to generate a realistic view of a scene. Linear projections achieve this realism at the cost of imposing restrictions on the 2D shape of each object in the image and on the overall composition of the picture. Artists have developed a variety of techniques for composing images of 3D scenes that deviate from the standard perspective projection. One of the most common techniques is to combine multiple projections in a single image.

Artists create such multiprojection images for several reasons, including: expressing a mood, feeling or idea; improving the representation or comprehensibility of the scene; and visualizing information about the spatial relationships and structure of the scene. Multiple projections could similarly enhance computer-generated images and animations, but simple and efficient methods for multiprojection rendering have not been available.

Today, the most common method for creating a multiprojection image requires a combination of 3D rendering and 2D image compositing. The process is a labor intensive cycle that involves rendering multiple views of the scene, transferring the images into a compositing application and then manually merging them into a single multiprojection image. Although some research systems [12, 11] shortcut this cycle by combining rendering and compositing into a single application, resolving visibility between the images remains a manual process. In this paper we present interactive methods for creating multiprojection images and animations. The main technical contributions of our work are new algorithms designed for:

Resolving Visibility: In the multiprojection setting there is no uniquely defined solution to the visibility problem. However, in many cases the user wishes to maintain the visibility ordering of a *master camera* while using different *local cameras* to introduce shape distortions to individual objects. Based on this insight, we propose an algorithm that automatically resolves visibility for most practical cases and allows user adjustments when the automatically computed visibility is not satisfactory.

Constraining Cameras: We suggest a simple and intuitive set of camera constraints allowing the user to choose appropriate projections for a variety of artistic effects. These constraints are particularly effective when initially placing cameras and when animating the scene.

Interactive Rendering: We leverage multipass hardware rendering to achieve interactive rendering rates. The user can immediately see how changing the parameters of any camera or moving any object will affect the final image.

The remainder of this paper is organized as follows. In section 2, we describe artistic uses of multiple projections. After presenting related work in section 3, we describe our multiprojection rendering algorithm with an emphasis on resolving occlusion in section 4. In section 5, we present a set of camera constraints that provide intuitive controls over the camera parameters. Examples of images generated with our system appear in section 6. Finally, section 7 outlines future directions and conclusions.

2 Artistic Uses of Multiple Projections

Using a single projection for an entire scene is restrictive. The ideal projection for one object may not be the best projection for all objects in the scene. Artists have a long history of solving this problem by creating multiprojection images, which generally serve some combination of three functions:

- **Artistic Expression** – Multiple projections help the artist express a mood, feeling or idea.
- **Representation** – Multiple projections improve the representation or comprehensibility of the scene.
- **Visualization** – Multiple projections communicate information about the structure of the objects and spatial relationships in the scene.

In this section, we consider several specific examples of each function. In section 6, we present several images and animations that are based on these examples and were created using our multiprojection rendering tools.

2.1 Artistic Expression

Viewing Anomalies: In Giorgio de Chirico's *The Mystery and Melancholy of a Street*, figure 1(a), the buildings, the van and the ground plane all have different viewpoints. Willats [19] suggests that the melancholy aura is created by the unusual arrangement of the objects which results in an incongruous spatial system. Despite the large disparities between projections, the overall impression of a three-dimensional space remains intact.

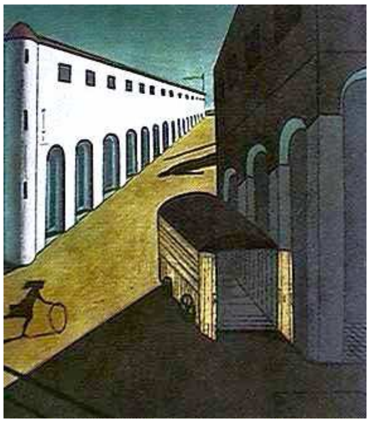
Cézanne similarly incorporates multiple viewpoints in *Still Life with Fruit Basket*, figure 1(b). Loran [13] describes how these viewing distortions generate tension between different planes in the image; the distortions flatten some regions of the picture, while enhancing depth in other regions. He explains that the inconsistencies in projection generate an "emotional nonrealistic illusion of space."

Foreground Elements in Animation: In cel-based animations, moving foreground elements sometimes translate across the image with little to no change in parallax (like cutouts or sprites). Equivalently, as the element translates in the scene, its local camera moves with it, thereby maintaining an identical view of the element from frame to frame. Meanwhile, the background is rendered using its own separate projection. Although fixing the view of foreground elements is done primarily to reuse the same drawing from frame to frame, it has become a stylistic convention in hand-drawn animation since the foreground elements appear flatter and more "cartoony."

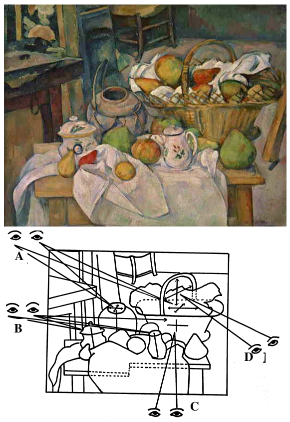
In contrast to the flattened foreground elements, animators often exaggerate the perspective projection of the background to create a deeper, more dynamic environment. However, such a strong perspective could introduce distortions in the foreground of the image. A fixed view of the foreground objects alleviates this problem.

2.2 Representation

Best Views: Certain viewpoints are better than others for comprehending the overall geometric shape of an object. By using multiple viewpoints the artists can present the best views of all objects in the scene. Graham [5] points out that a single viewpoint is often inadequate for large format pictures like murals, frescoes, or billboards. Such images are often placed above standard eye-level and are seen from a much wider range of viewpoints than smaller format pictures. For these reasons, many large format images are created with multiple projections. As

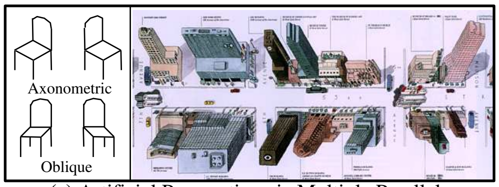


(a) Giorgio de Chirico's *The Mystery and Melancholy of a Street*

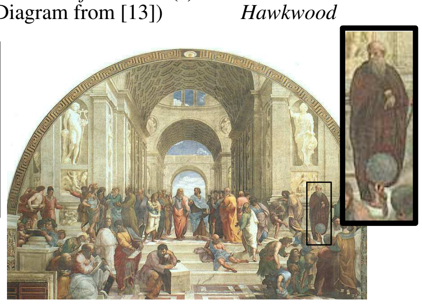


(b) Cezanne's *Still Life with Fruit Basket* (Diagram from [13])

(c) Paolo Uccelo's *Sir John Hawkwood*



(e) Artificial Perspective via Multiple Parallel Projections (Chairs from [2], 53rd Street from [6])



(d) Raphael's *School of Athens*

Fig. 1. Multiprojection examples (clockwise labeling)

(a) A different projection is used for each major structure. Note the difference in vanishing points for the buildings, the van, and the ground plane. An exaggerated perspective projection elongates the white building and the ground plane is placed so that the horizon is high up in the image to create the long receding path.

(b) Cezanne incorporates many viewpoints into this still life, of which four are shown in the diagram. Notice how viewpoint **A** is much higher than the other viewpoints, thus the ginger jar and basket appear to be tipped forward. Differences in viewpoint also cause the table to appear to split under the table cloth.

(c) The projection for the base differs from the projection for the horse and rider. In its original setting viewers stood below and to the left of the fresco, so the entire picture was first created for this viewpoint. While the low−set viewpoint was effective for the base, Uccelo found that it exposed too much of the horse's belly and distorted the rider. He repainted the top of the picture and raised the viewpoint of the horse and rider to eliminate the distortion.

(d) The foreground humans would appear distorted if rendered using the projection of the background architecture. Thus, Raphael altered the projections for the humans to give each one a more central projection. Without the correction, the sphere (inset) would appear elliptical rather than circular[10, 21].

(e) Multiple parallel projections can produce an artificial sense of perspective by pointing the receding parallels towards a central vanishing point. Dubery and Willats[2] argue that this technique works better with oblique projections than with axonometric projections. Each building in the illustration of 53rd Street is projected using a different oblique projection. Note how the receding parallels differ for the buildings at the left, right, top, and bottom of the image.

---

large wall-sized displays become more prevalent [8, 16], we expect that multiprojection rendering techniques will be required to reduce perceptual distortions in images generated for such displays. Paolo Uccelo's fresco of *Sir John Hawkwood* is a well-known large format multiprojection example. He used two projections, one for the base and one for the horse and rider, to produce the best view of both elements, as described in figure 1(c).

Reducing Wide-Angle Distortion: A well known problem with wide-angle, perspective projections is that curved objects located near the edges of the viewing frustum, close to the image plane, can appear unnaturally stretched and distorted. The most common technique for decreasing this distortion is to alter the projection for every object to provide a perceptually "correct" view of each one. The deviations in projection are often subtle and in many cases even the artist, focused on producing a comprehensible image of the scene, may not realize he altered the projections. Kubovy [10] shows that the foreground human figures in Raphael's *School of Athens*, figure 1(d), are inconsistent with the strong central perspective projection of the background architecture. The inconsistencies improve the comprehensibility of the figures and make them easier to recognize.

2.3 Visualization

Artificial Perspective via Multiple Parallel Projections: It is possible to create an artificial sense of perspective using multiple axonometric or multiple oblique projections. The "trick" is to orient the receding parallels of each object towards some pre-chosen vanishing point. In the illustration of 53rd Street [6] (figure 1(e)) the buildings are drawn from above in oblique projection. The receding parallels for each building point towards a central vanishing point, thereby creating the illusion of perspective.

An advantage of such artificial perspective over true perspective projection is that objects do not diminish in size as they recede from the viewer. With oblique projections, one face of each object retains its true shape, allowing the viewer to perform some size and area comparisons. Applying this principle to the 53rd Street example, it is possible to visually compare the 2D area covered by each building. The convergence of true perspective would cause displayed rooftop areas to depend on building height, making such comparisons more difficult.

3 Related Work

In computer graphics, alternatives to standard linear projections have been developed in a variety of contexts. Max [14] shows how to project images onto a curved Omnimax screen, while Dorsey et al. [1] present methods for projecting images onto planar screens for off-axis viewing. Zorin and Barr [21] show how to correct perceptual distortions in photographs by reprojecting them. Glaeser and Gröller [3] use cartographic projection techniques to reduce wide-angle distortions in synthetic images. Both Wood et al. [20] and Rademacher and Bishop [15] describe techniques for smoothly integrating all the views for a given camera path into a single image.

Savranksy et al. [17] describe methods for rendering Escher-like "impossible" scenes. They treat geometric transformations between pairs of objects as constraints on the viewing projection and then solve for a single projection that best meets all the constraints. They compute orthographic viewing projections for several "impossible" scenes. It is unclear how well such an approach would extend to perspective projection. They show that the only way to obtain a single projection for Escher-like scenes is by relaxing the constraints, so they remain valid in the 2D image but not necessarily in 3D object space. In fact, many of the "impossible" scenes only appear impossible from a single viewpoint and animated camera paths would destroy the illusion.

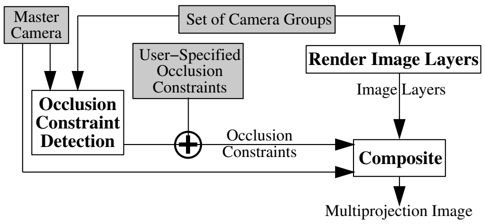
Several systems have been developed to allow more general alternatives to planar-geometric projection. Inakage [9] derives mathematical formulations for three alternatives: curvilinear projection, inverted projection and arbitrary 3D warps. However, he does not consider combining multiple projections within a single image. Löffelmann and Gröller [12] explore the use of curvilinear and non-linear projections to create Escher-like images. Levene [11] investigates artistic uses of curvilinear, inverted, and oblique projections as alternatives to planar perspective. He derives a mathematical framework that unifies several classes of projections. The last two systems are notable because they contain mechanisms for specifying multiple projections within a single image. However, to our knowledge, no system adequately addresses the problem of resolving visibility in the multiprojection setting.

We will show in section 4.1 that resolving visibility is perhaps the most difficult problem in generating multiprojection images because visibility ordering between objects is different under each projection. Our solution is to use a master camera to specify visibility ordering while maintaining shape distortions due to local cameras attached to each object. Although Levene uses the concept of a master camera to compose his multiprojection images, he points out that visibility is not properly resolved with his approach and leaves this as an open problem for future work.

The previous multiprojection rendering systems require the user to directly manipulate the parameters of their generalized projections. Such controls are not always natural and their effect on the final image may be unintuitive. In contrast, we provide several novel camera constraints that allow the user to obtain commonly desired effects with relative ease. Users can also directly specify projection parameters when necessary.

Creating a multiprojection image is far easier with interactive rendering so that the user can immediately see how changing a projection effects the final image. Earlier systems [12, 11] use software ray tracing renderers; therefore, image updates are not interactive for typical image sizes. Our system maintains interactive rendering rates by leveraging graphics hardware. Although this restricts our current implementation to linear planar-projections, our visibility ordering algorithms would work with any invertible projection, including the generalized projection formulations proposed by Inakage, Löffelmann and Gröller, or Levene.

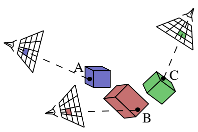
4 Multiprojection Rendering



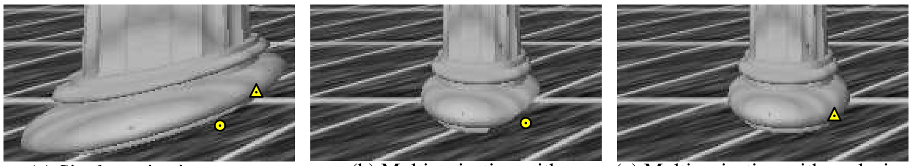
Our multiprojection rendering algorithm includes three computational stages. The input to the algorithm is a set of *camera groups*, each associating a collection of geometric objects with one *local camera*. The user must provide a *master camera* and can optionally specify *object-level occlusion constraints*, both of which are used to resolve visibility. In the block diagram, white boxes represent computational stages of the algorithm, while gray boxes represent user-specified data. The first stage of the algorithm renders each camera group into a separate *image layer*. We then merge the image layers together to form the multiprojection image.

The main difficulty in the compositing stage is the absence of a natural visibility ordering. When visibility ordering differs from camera to camera, there is no unique way to resolve occlusion. Our key observation is that in most multiprojection images, all the local cameras are relatively similar to one another and therefore generate similar visibility orderings. Instead of specifying the occlusion relationship between every pair of objects in the scene, the user simply specifies a master camera (often a local camera doubles as the master). We then use the master camera to resolve visibility through a combination of two automatic techniques: 3D depth-based compositing and standard 2D compositing based on object-level occlusion constraints. If necessary, the user can directly modify the visibility ordering by specifying additional pairwise occlusion relationships between image layers.

4.1 Visibility Ordering



With a single linear projection, visibility is defined unambiguously; the *fibres*, that is, the set of points in 3D space that map to a point on the image surface, are straight lines. For any two points that lie on the same fibre, occlusion is resolved by displaying the point closest to the center of projection. This approach can resolve occlusion whenever the fibres are continuous curves. With multiprojection images, the fibres generally have a more complicated structure since the mapping from 3D space to the image surface can be discontinuous. Suppose, as in the diagram, that points A, B and C project to the same pixel in their local images. The fibre of the multiprojection image at this pixel consists of the union of the three dotted lines. It is difficult to automatically compute a visibility ordering with complicated fibres because no natural ordering exists for the points on different lines.



(a) Single projection master camera view

(b) Multiprojection with depth compositing only

(c) Multiprojection with occlusion constraints and depth compositing

Fig. 2. To reduce the distortion of the column in the single projection image (a) we alter its projection as shown in figure 5(c). In the multiprojection image, the point on the column (triangle) and the point on the floor (circle) coincide. With depth-based compositing alone (b) the floor point "incorrectly" occludes the column point since it is closer to the master camera. However, the column occludes the floor in the master view. Applying this object-level occlusion constraint during compositing yields the desired image (c).

It may be tempting to resolve visibility for the multiprojection image by directly comparing local depth values stored with each image layer. The rendering algorithm would then be quite simple: we could add the local camera projection to the modeling transformation matrix for each object and render the scene using a standard z-buffer pipeline without resorting to layer based compositing. However, this approach would lead to objectionable occlusion artifacts. Suppose our scene consists of a vase sitting on a table. If we simply add the vase's local camera projection into its modeling transform, in most cases the vase will intersect the table. Our algorithm handles these situations more gracefully by using the master camera to impose visibility ordering while employing local cameras to provide shape distortion. In our example, the master camera would be the original projection, in which the vase and table do not intersect, and the local projection would affect only the shape of the vase without affecting visibility.

Given the master camera, we can define an ordering for any set of points in 3D space based on the distance from each point to the master camera viewpoint. To merge the image layers rendered for each camera group, we transform all the pixels in each image-layer into world space using their pixel coordinates and z-values. We then apply the master camera projection to these world space points and use a standard z-buffer test to determine the frontmost point at each pixel of the master camera image. However, figure 2 shows that the results produced by this depth-based approach are not always satisfactory. The problem occurs because visibility is resolved for each pixel independently, yet we want to preserve object-level occlusion constraints, such as "column occludes floor", that occur in the single projection master camera view. In the next section, we describe a simple algorithm for automatically computing these constraints with respect to the master camera. However, we will also show that visibility can not always be resolved using object-level occlusion constraints alone. Thus, the compositing stage of our algorithm combines two approaches. We use object-level occlusion constraints wherever possible and fall back onto depth-based compositing only when the occlusion constraints are ambiguous.

While additional user intervention is not required, the user may explicitly specify occlusion constraints between pairs of objects. These user-defined occlusion constraints are added to the list of constraints computed via the occlusion detection algorithm. Conflicts between user-specified constraints and computed constraints are resolved in favor of the user and the conflicting computed constraints are removed from the list.

4.2 Object-Level Occlusion Constraints

Object-level occlusion constraints are defined for whole objects rather than individual points of the objects. If every point of object A is in front of object B, we say that A occludes B. To compute the occlusion constraints with respect to the master camera, we must determine for each pair of objects A and B whether A occludes B, B occludes A or neither. Our occlusion constraint detection algorithm is based on an algorithm described by Snyder and Lengyel[18].

Occlusion Constraint Detection Algorithm

```
Input: Set of Objects,
       Master Camera
Output: Set of Occlusion Constraints

foreach camera group
  Render object using Master Camera into a separate
  buffer storing (Camera Group ID, depth) per–pixel
foreach pixel
  Sort objects in pixel by depth
  foreach pair of objects in sorted list
    if conflicting occlusion constraint appears in list
      Mark object pair as non–ordered
    else
      Add occlusion constraint to list
```

Object-level occlusion constraints may not provide enough information to merge all the image layers. There are two forms of ambiguity. When the convex hulls of A and B intersect we may find that A occludes B in some regions of the master view, while B occludes A in other regions. Our constraint detection algorithm checks for such binary occlusion cycles and marks the objects as non-ordered. Another type of ambiguity arises when A and B do not occlude one another in the master view at all, but their local image layers do overlap. Our occlusion constraint detection algorithm cannot provide any constraint in this case. We handle these ambiguities during the compositing phase of the algorithm.

4.3 Compositing

The last stage of the algorithm, compositing, combines the multiple image layers produced in the first step into a single image. If two objects map to the same pixel and there is an occlusion constraint between them, we simply use the constraint to determine which object is visible. When no such occlusion constraint is available (as in the cases described in the previous section), we use depth compositing to resolve visibility.

In certain cases, occlusion cycles can pose a problem for our algorithm. Suppose there is a three object cycle[1] in which A occludes B, B occludes C, and C occludes A. Our pixel-by-pixel compositing algorithm will produce the desired multiprojection image regardless of the order in which it considers the objects, unless a single pixel contains all three. While it is possible to detect such loops in the occlusion constraint graph, resolving occlusion in such cases requires an arbitrary choice as to which object in the cycle is visible. In practice, we have found that this problem rarely occurs, and when it does, we provide controls so that the user can choose which object is visible.

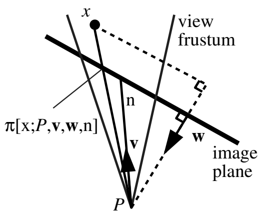
5 Camera Constraints

When creating multiprojection images and animations, it is useful to be able to constrain the motions of an object and its local camera in relation to one another. For example, the user may want to dolly the camera closer to its object without changing the object's size in the image plane. This would produce a close-up, wide angle view of the object, thus increasing its perceived depth without changing its relative size in the image. Such camera constraints are indispensable for multiprojection animations, since direct specification of several dependent cameras is quite difficult. The user might keyframe an object motion and with such constraints in place, the cameras would automatically move in relation to the object to enforce the constraint.

We consider three camera constraints we have found particularly useful. Each constraint is a system of equations, relating the camera parameters and object position before and after modification. The system is typically underconstrained and we can choose some of the camera parameters arbitrarily. Once these parameters are chosen, we use the constaints to determine the other parameters.

Notation: Each equation will be preceded by a label (i.e. pos, dir, size, dist) describing the constraint it imposes. We use $'$ to denote the quantities corresponding to the camera and the objects after modification. We denote the vector length as $\|\mathbf{a}\|$ and the vector dot product as $\mathbf{a}{\cdot}\mathbf{b}$. We use $\pi[x; P, \mathbf{v}, \mathbf{w}, n]$ to denote the world-space coordinates of the projection of a point x into the image plane with a camera defined by parameters $P, \mathbf{v}, \mathbf{w}, n$. The image plane coordinates of the projection of x are denoted $\pi_1[x; P, \mathbf{v}, \mathbf{w}, n]$ and $\pi_2[x; P, \mathbf{v}, \mathbf{w}, n]$, as shown in figure 3.

[1]We remove binary cycles as described in section 4.2.



Camera parameters:
P: center of projection
v: camera view unit vector
w: image plane unit normal
u: world space up unit vector
up: image plane up unit vector
n: distance to image plane along ***v***
height: image plane height
width: image plane width
f: distance to far plane

Projection Equations

World Space Coordinates:

$$\pi[x;P,\mathbf{v},\mathbf{w},n] = (n\mathbf{v}\cdot\mathbf{w})\frac{x-P}{(x-P)\cdot\mathbf{w}} + P$$

Image Plane Coordinates:

$$\pi_1[x;P,\mathbf{v},\mathbf{w},n] = (\pi[x;P,\mathbf{v},\mathbf{w},n] - n\mathbf{v})\cdot\mathbf{up}$$

$$\pi_2[x;P,\mathbf{v},\mathbf{w},n] = (\pi[x;P,\mathbf{v},\mathbf{w},n] - n\mathbf{v})\cdot(\mathbf{up} \times \mathbf{w})$$

Fig. 3. Camera model and parameters.

5.1 Object-Size Constraint

Normally, as a camera moves towards an object, the projected image size of the object grows and the perspective convergence of the object changes. The goal of the object-size constraint is to keep the object's size and position approximately constant while changing its perspective convergence by dollying the camera towards (or away from) it. We characterize the object's size by the distances between two points on the object, x_1 and x_2. The constraint maintains a constant distance between the projections of the two points and is expressed in the following equation:

size : $\|\pi[x_1; P', \mathbf{v}', \mathbf{w}', n'] - \pi[x_2; P', \mathbf{v}', \mathbf{w}', n']\| = \|\pi[x_1; P, \mathbf{v}, \mathbf{w}, n] - \pi[x_2; P, \mathbf{v}, \mathbf{w}, n]\|$

The simplest way to adjust perspective convergence for an object is to change the near parameter n of the camera model while leaving the $height$ and $width$ at the near plane unchanged. Suppose, as in figure 4(a), we move the camera towards the center of the object to a new position $P' = P + (x - P)\Delta t$. We assume that the view vector and image plane normal remain fixed (i.e. $\mathbf{v}' = \mathbf{v}$ and $\mathbf{w}' = \mathbf{w}$), and compute the new near distance n':

$$n' = n\frac{\|\pi[x_1; P', \mathbf{v}, \mathbf{w}, n] - \pi[x_2; P', \mathbf{v}, \mathbf{w}, n]\|}{\|\pi[x_1; P, \mathbf{v}, \mathbf{w}, n] - \pi[x_2; P, \mathbf{v}, \mathbf{w}, n]\|}$$

In the film *Vertigo*, Hitchcock uses this technique to give the audience a sense of the vertigo as experienced by the main character. Hitchcock simultaneously dollies the camera and adjusts its focal length (i.e. near distance) to maintain the object-size constraint. The change in perspective, with little or no change in object size and position conveys the impression of dizziness and falling.

Another interesting use of the object-size constraint is constructing artificial perspective by combining multiple oblique projections as described in section 2.3. If we enforce the object-size constraint, move the camera out to infinity, and also force the view vector **v** to be parallel to the line between the camera center of projection P and the object center x, we obtain an oblique projection. Although there is no vanishing point for the oblique projections, the receding parallels will point towards the vanishing point of the original perspective projection. By separately applying the constraint to each object in the scene, we combine multiple oblique projections to produce an artificial sense of perspective. We used this approach to create plate 3.

5.2 Fixed-View Constraint

Under a perspective projection, as an object translates parallel to the image plane in world space, different sides of the object are exposed. As the object moves from left to right with respect to the camera center of projection, the intra-object occlusions change.

At times we wish to both maintain a fixed view of an object and move its position in the image plane. The result is an object that translates in the image much like a 2D sprite. However, unlike a normal sprite, an object under the fixed-view constraint continues to move in 3D and can therefore rotate and deform as a 3D body.

As shown in figure 4(b), we must maintain the direction vector between the object center and the camera center of projection. In addition, we would like the size and position of the object in the image plane to be approximately the same as if the camera were not changed. That is, we

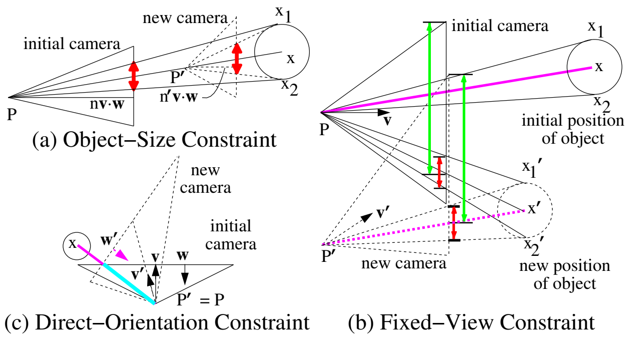


Fig. 4. Camera constraint diagrams. In (a) the image size (red) of the object remains fixed as the camera moves towards it. In (b) as the object translates in world space, its image size (red), image position (green) and the direction between it and the center of projection (magenta) remain fixed. For clarity we introduce a slight separation between the image planes. In (c) the camera is reoriented so the image plane normal $\mathbf{w}'$ matches the direction of the line joining x with P (magenta). The distance (cyan) between the camera and the object's image also remains fixed.

wish to maintain:

$$\begin{aligned}
\textbf{dir}: &\quad \frac{x' - P'}{\|x' - P'\|} = \frac{x - P}{\|x - P\|} \\
\textbf{pos}: &\quad \pi_i[x'; P', \mathbf{v}', \mathbf{w}', n'] = \pi_i[x'; P, \mathbf{v}, \mathbf{w}, n] \quad i = 1, 2 \\
\textbf{size}: &\quad \|\pi[x_1'; P', \mathbf{v}', \mathbf{w}', n'] - \pi[x_2'; P', \mathbf{v}', \mathbf{w}', n']\| = \|\pi[x_1'; P, \mathbf{v}, \mathbf{w}, n] - \pi[x_2'; P, \mathbf{v}, \mathbf{w}, n]\|
\end{aligned}$$

where, as before, x_1, and x_2 are two points that we use to define the size of the object.

We have found that it is convenient to choose x to be the center of the bounding sphere of an object, and x_1 and x_2 to be the two extremal points of the bounding sphere such that the plane formed by x, x_1 and x_2 is parallel to the image plane. Thus, x_1 and x_2 are the endpoints of an interval centered at x, and their positions with respect to x do not change as the object moves. We assume that both $\mathbf{w}$ and the distance from the center of projection to the image plane remain fixed (i.e. $\mathbf{w} = \mathbf{w}'$ and $n\mathbf{v} \cdot \mathbf{w} = n'\mathbf{v}' \cdot \mathbf{w}$). Then the first constraint equation can be solved for P', and the other two can be solved for $\mathbf{v}'$ in order to determine the new position of the image plane.

5.3 Fixed-Position Constraint

The fixed-position constraint is similar to the fixed-view constraint. Instead of maintaining a particular view of the object, the fixed-position constraint maintains the position of the object in the image plane. As the object translates, different sides of it are exposed, but it remains in the same place in the image plane. While the effect is similar to rotating the camera about the object, this constraint actually produces an off-axis projection. The following constraint ensures that the center of the object remains fixed in the image plane:

$$\textbf{pos}: \quad \pi_i[x; P, \mathbf{v}, \mathbf{w}, n] = \pi_i[x'; P', \mathbf{v}', \mathbf{w}', n'] \quad i = 1, 2$$

The desired effect is that as the object moves in world space, the window of the image plane through which we see the object shifts with it. However, the image plane does not move. Thus, P and $\mathbf{w}$ remain fixed (i.e. $P' = P$ and $\mathbf{w}' = \mathbf{w}$), while $\mathbf{v}$ and n change.

As with the fixed-view constraint, the distance from the center of projection to the image plane remains fixed, so $n\mathbf{v} \cdot \mathbf{w} = n'\mathbf{v}' \cdot \mathbf{w}$. Assuming no change in camera parameters, we first compute the displacement d of the object's image and then use this displacement to determine the center of the image plane for the new camera $n'\mathbf{v}'$.

$$d_i = \pi_i[x'; P, \mathbf{v}, \mathbf{w}, n] - \pi_i[x; P, \mathbf{v}, \mathbf{w}, n] \quad i = 1, 2$$
$$n'\mathbf{v}' = n\mathbf{v} + d_1\,\mathbf{up} + d_2\,(\mathbf{up} \times \mathbf{w})$$

5.4 Direct-Orientation Constraint

In section 2.2, we observed that curved objects can appear extremely distorted under a wide-angle projection. There are several ways to reduce this distortion.

The simplest approach is to use a parallel projection for each distorted object. For example, with a scene consisting of a row of columns, figure 5, we would use the same orthographic camera for all the columns to ensure that they all appear to be the same size. The drawbacks of this approach are that the columns appear flat and that we see each column from exactly the same side.

A better approach is to use the direct-orientation constraint. For each object we create a camera pointed directly at this object. Equivalently, the image plane normal for each local camera is parallel to the direction vector to the object. At the same time, we preserve the position of the object in the image plane. Finally, we try to keep the size of the object approximately constant in the image plane (see figure 4(c)). Instead of using a general equation of the type derived for the object-size constraint, we use a simpler condition requiring that the distance from the camera to the image of the object remains fixed. We obtain the following constraint equations for the camera corresponding to the object located at x:

$$\begin{aligned}
\textbf{dir}: &\quad \mathbf{w}' = \frac{x - P}{\|x - P\|} \\
\textbf{pos}: &\quad \pi_i[x, P, \mathbf{v}', \mathbf{w}', n'] = \pi_i[x, P, \mathbf{v}, \mathbf{w}, n] \quad i = 1, 2 \\
\textbf{dist}: &\quad \|\pi[x, P, \mathbf{v}', \mathbf{w}', n'] - P\| = \|\pi[x, P, \mathbf{v}, \mathbf{w}, n] - P\|
\end{aligned}$$

Assuming that the up vector **u** does not change, these equations uniquely determine $\mathbf{v}'$ and n' when **v** and n are known.

6 Results

We have created several images and animations using our multiprojection rendering system as shown in the color plates, figure 5, and the accompanying video[2]. In all our examples, all occlusion relationships were computed by our algorithm. Plate 1 is our reconstruction of de Chirico's *Mystery and Melancholy of a Street*. The scene is modeled as shown in the plan view thumbnail image. The drop shadows from the building are explicitly modeled as polygons. Interactively adjusting the five local cameras took about an hour and a half. Matching the painting took about 20 minutes and the remainder of the time was spent animating the van to move through the scene. We made extensive use of our camera constraints throughout the process. To place the local camera of the brown building, for example, we initially set the image plane of the camera parallel to the front face of the building. We then used the object-size and fixed-position constraints to interactively adjust the vanishing point for its receding faces until they matched the painting. The van was animated to go around the brown building using a combination of object motion and camera motion. Note that as the van moves from being in front of the brown building to going behind it, the occlusion relationship between the two objects is updated automatically.

The multiprojection still life in plate 2 was inspired by Cézanne's *Still Life with Fruit Basket* and adjusting the 10 local cameras took about an hour for this scene. Plate 3 is shows a comparison of an overhead view of a city rendered using multiple oblique projection to create an artificial sense of perspective in (a), and a true perspective projection in (b). There are two advantages to using multiple oblique projections. First, more of the scene is visible; in the true perspective many building extend beyond the field of view the image and there is more occlusion between

[2] http://graphics.stanford.edu/papers/mpr/video

buildings than in the multiple oblique view. Second, it is easier to judge the relative rooftop sizes, and the area of the block that each building covers in the multiple oblique view. Plate 4 shows some frames from an animation we created using the fixed-view constraint. In parts (a) and (b) we show how the fixed-view constraint improves the composition of the scene. The fixed-view constraint only affects translational motions. Unlike traditional sprites, objects can rotate and deform as 3D bodies as shown in the frames on left side of plate 4.

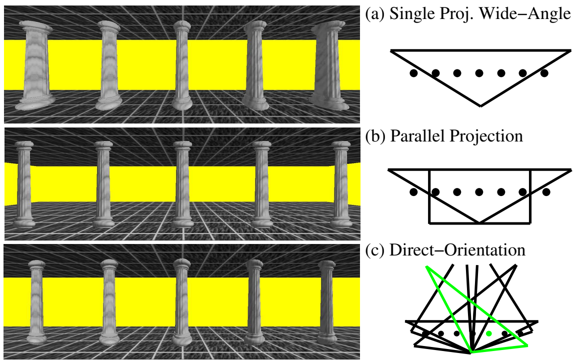


Fig. 5. Reducing the wide-angle distortion of curved objects

The columns in figure 5(a) appear distorted due to a wide-angle camera. If we use a parallel projection for the columns as in figure 5(b) (see camera schematic), the distortion is corrected, but each column appears flat and the lighting is wrong: every column is lit identically. A better approach is to use the direct-orientation constraint to aim each local camera at its column as in figure 5(c). Since the local cameras all have the same center of projection, we see exactly the same parts of each column as in the original view and the lighting is correct because the visibility is unchanged. We have colored one of the columns and its local camera green in the camera schematic to show how the local camera is reoriented.

Our implementation of the multiprojection rendering algorithm uses hardware rendering to maintain an interactive interface. The main bottleneck in the rendering algorithm is this multipass read/write of the framebuffer required before rendering each camera group and resolving visibility. Therefore rendering performance is based on the number of camera groups in the scene and the size of the image. Despite the read/write bottleneck, we are able to achieve interactive framerates for most of our test scenes running on a SGI RealityEngine graphics system. For example, the multiprojection still life contains 10 camera groups and renders at a rate of 3 to 5 frames per second at a resolution of 333 x 256. Other scenes containing fewer cameras render more quickly. The de Chirico scene containing five camera groups renders at 10 - 12 frames per second at a resolution of 402 x 491.

In addition to the interactive interface, we built a script-based animation system for creating keyframed multiprojection animations. Obtaining visually pleasing results with a multiprojection animation is somewhat difficult because animation adds motion parallax as a cue about the spatial relationship between objects. The motion parallax may conflict with the perspective and occlusion cues, disconcerting the viewer. When the entire scene is moved by applying the same transformation to every object in the scene (or equivalently to every camera in the scene), each object will move and scale at a different rate since the local cameras may be very different. We have found that in most cases animating objects in the scene, while holding the background environment fixed, works well. Moving the entire scene tends to only work for relatively small translations and rotations. One notable exception is when all the local cameras have the same center of projection as in the row of columns example in figure 5. In this case, moving the entire scene does not cause any unexpected effects.

7 Future Work and Conclusions

We have presented a multiprojection rendering algorithm as well as a constraint-based camera specification interface. Using this system, it is possible to create a large variety of still and animated visual effects that can not be created with a standard, single projection rendering system.

There are several directions in which to expand this work. As we described in the previous section, not all types of animation produce pleasing results. Moreover, animation requires keyframing the motion over every object by hand. While our camera constraints can make it easier to produce certain types of camera motions, the constraints are designed to produce very specific effects. It would be useful to incorporate a more flexible constraint-based interface for cameras, similar to the design of Gleicher and Witkin [4], in both our camera specification interface and our animation system.

It is unclear how to handle lighting effects like shadows and reflection in the multiprojection setting. We currently assume that all lights are specified in the global scene and each camera group is lit based on its local camera viewpoint. The difficulty arises when we try to cast a shadow from one camera group onto another. Despite these issues, multiprojection images are an effective device for creating a variety of artistic effects.

8 Acknowledgements

Thanks to Pat Hanrahan for giving us insightful feedback throughout the course of this work. Gregory Lam Niemeyer's artistic guidance was invaluable during the early stages of the project.

References

[1] J. O. Dorsey, F. X. Sillion, and D. P. Greenberg. Design and simulation of opera lighting and projection effects. In *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 41–50, July 1991.

[2] F. Dubery and J. Willats. *Perspective and Other Drawing Systems*. Van Nostrand, 1983.

[3] G. Glaeser and E. Gröller. Fast generation of curved perspectives for ultra-wide-angle lenses in VR applications. *The Visual Computer*, 15:365–376, 1999.

[4] M. Gleicher and A. Witkin. Through-the-lens camera control. In E. E. Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 331–340, July 1992.

[5] D. W. Graham. *Composing Pictures*. Van Nostrand Reinhold Company, 1970.

[6] S. Guarnaccia. 53rd Street Map. In N. Holmes, editor, *The Best in Diagrammatic Graphics*, pages 174–175. Quarto Publishing, 1993.

[7] A. Hertzmann. Painterly rendering with curved brush strokes of multiple sizes. In *SIGGRAPH 98 Conference Proceedings*, pages 453–460. ACM SIGGRAPH, Addison Wesley, July 1998.

[8] G. Humphreys and P. Hanrahan. A distributed graphics system for large tiled displays. *IEEE Visualization '99*, pages 215–224, October 1999.

[9] M. Inakage. Non-linear perspective projections. In *Modeling in Computer Graphics (Proceedings of the IFIG WG 5.10 Working Conference)*, pages 203–215, Apr. 1991.

[10] M. Kubovy. *The Psychology of Perspective and Renaissance Art*. Cambridge University Press, 1986.

[11] J. Levene. A framework for non-realistic projections. M.eng. thesis, MIT, 1998.

[12] H. Löffelmann and E. Gröller. Ray tracing with extended cameras. *The Journal of Visualization and Computer Animation*, 7(4):211–227, Oct.-Dec. 1996.

[13] E. Loran. *Cezanne's Composition*. University of California Press, 1943.

[14] N. L. Max. Computer graphics distortion for IMAX and OMNIMAX projection. In *Nicograph '83 Proceedings*, pages 137–159, Dec. 1983.

[15] P. Rademacher and G. Bishop. Multiple-center-of-projection images. In *SIGGRAPH 98 Conference Proceedings*, pages 199–206, July 1998.

[16] R. Raskar, G. Welch, M. Cutts, A. Lake, L. Stesin, and H. Fuchs. The office of the future: A unified approach to image-based modeling and spatially immersive displays. *Proceedings of SIGGRAPH 98*, pages 179–188, July 1998.

[17] G. Savransky, D. Dimerman, and C. Gotsman. Modeling and rendering Escher-like impossible scenes. *Computer Graphics Forum*, 18(2):173–179, June 1999.

[18] J. Snyder and J. Lengyel. Visibility sorting and compositing without splitting for image layer decomposition. In *SIGGRAPH 98 Conference Proceedings*, pages 219–230, July 1998.

[19] J. Willats. *Art and Representation: New Principles in the Analysis of Pictures*. Princeton University Press, 1997.

[20] D. N. Wood, A. Finkelstein, J. F. Hughes, C. E. Thayer, and D. H. Salesin. Multiperspective panoramas for cel animation. In *SIGGRAPH 97 Conference Proceedings*, pages 243–250, Aug. 1997.

[21] D. Zorin and A. H. Barr. Correction of geometric perceptual distortion in pictures. In R. Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 257–264. ACM SIGGRAPH, Addison Wesley, Aug. 1995.

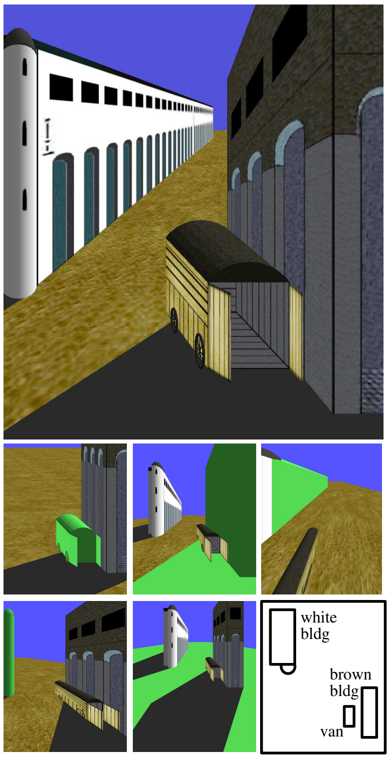


Plate 1. Our reconstruction of Giorgio de Chirico's *Mystery and Melancholy of a Street*. The thumbnails show the 5 local camera views, with attached geometry highlighted in green.

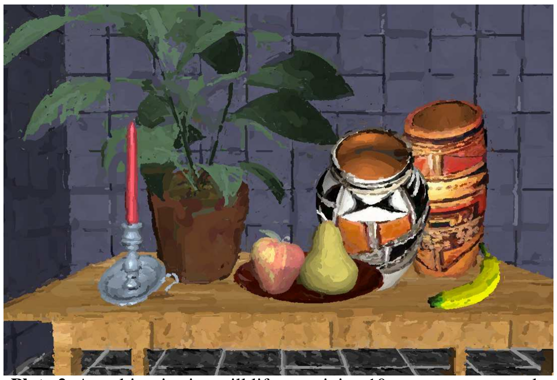
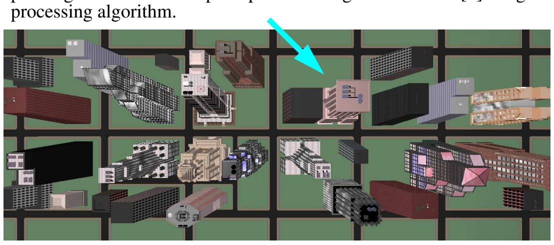
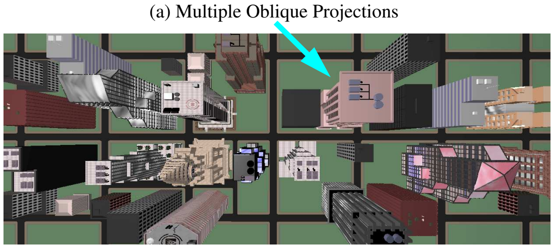


Plate 2. A multiprojection still life containing 10 camera groups took about an hour to create with our system. The impressionist style painting was created in a post−process using Hertzmann's [7] image processing algorithm.

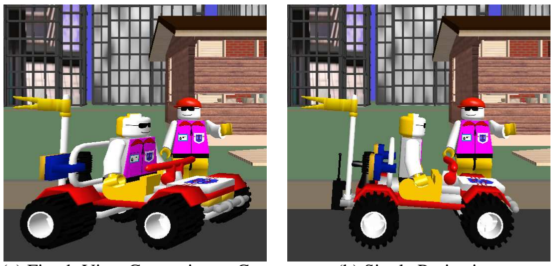


(a) Multiple Oblique Projections



(b) True Perspective Projection

Plate 3.Multiple oblique projections create an artificial sense of perspective, but still allow some area comparisons. In (b) the pink building's rooftop area (arrow) is exaggerated. In (a) it correctly appears to be about the same size as the gray rooftop next to it.



(a) Fixed−View Constraint on Car (b) Single Projection

Plate 4.Our fixed−view constraint can improve composition and give scenes a "cartoony" feel. In (a) it is possible to see the faces of both characters. In (b) the sitting character's face is not visible. The constraint only affects translational motion, so objects can rotate and deform as 3D bodies. In the animation frames to the left, the fixed−view constraint is enforced on both cars. In the first and last frame the views of the cars are the same, but when the blue car turns to pass the red car in the middle two frames, we can see the tires rotate and the blue character's uniform becomes more visible than that of the red character.