In Proc. 10th IEEE International Conference of Computer Vision, Beijing, October 2005.

Realtime IBR with Omnidirectional Crossed-Slits Projection

Doron Feldman Daphna Weinshall
School of Computer Science & Engeineering, Hebrew university, Jerusalem, Israel 91904
{doronf, daphna}@cs.huji.ac.il

Abstract

The Crossed-Slits (X-Slits) projection can be used to generate new views of a scene from a sequence of perspective images. Compared with other image-based rendering (IBR) techniques, X-Slits image generation is simple and requires a relatively small number of input images, which makes it suitable for realtime IBR. In this paper we extend this model to omnidirectional cameras and a circular slit. We show how it can be used for realtime image-based rendering of omnidirectional images, and how to optimize it for speed and quality. We analyze the inherent geometric distortions of the circular X-Slits projection, and describe a normalization mechanism to reduce distortions, creating a realistic virtual environment. Essentially the same mechanism is used to augment the X-Slits images with artificial objects, when using standard graphics tools which assume perspective projection.

1. Introduction

New view generation is an emerging application which can benefit from both image-based techniques and graphics. The traditional approach to new view generation is to render a 3D model of the scene from different viewpoints. Unless the model is known apriori, this approach requires the recovery of the scene structure, which is a hard task. Moreover, the realistic synthesis of optical effects such as specularity, reflections and transparency is an involved problem.

Another recent approach, called Image-Based Rendering (IBR), advocates the use of raw images instead of 3D models. New views of the scene are generated based on a sequence of images, without a model of the scene, by sampling light rays. If the set of input images is dense, then the rays necessary for the synthesized image can be sampled from the input images, without knowledge about the scene and without attention to optical effects. However, the input images must be very precisely calibrated, and together should contain all possible rays of the scene.

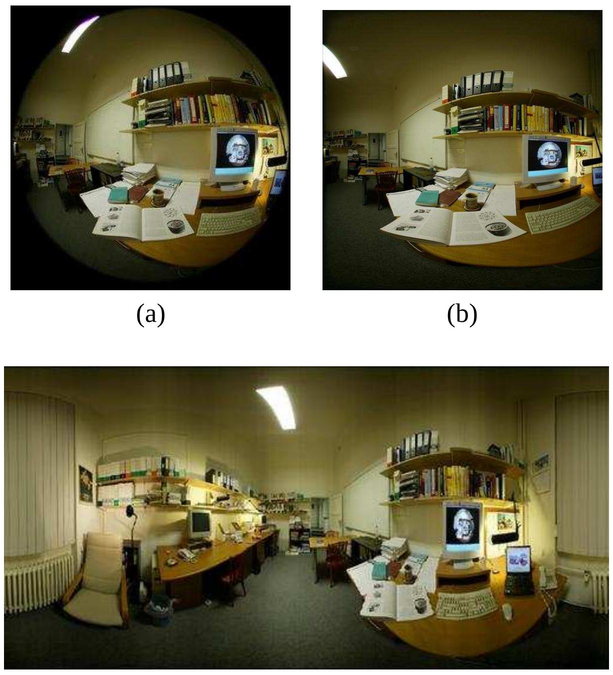
The set of all rays, as the plenoptic function [1], can be reasonably represented as a four-dimensional function. Using the full plenoptic function would require a very large amount of input data to produce synthetic views with good quality. The amount of data can be reduced using information about the scene or with some restrictions on the viewer's movement [4, 6–9]. However, a much larger reduction (from 4D to 3D) can be obtained by using X-Slits camera models.

A *Crossed-Slits camera* [15] is defined as a camera where all rays intersect in two lines ("slits") (rather than one point, as in the perspective projection model). Each scene point defines a plane with each of the slits, and the intersection of these planes is a line which passes through the point and both slits – hence, each scene point has a single ray passing through it.

The Crossed-Slits projection model offers another method for image-based rendering. If, rather than generating perspective new views of the scene, we can settle for X-Slits views, we can do IBR with a much smaller plenoptic function. Specifically, in [15] the input is taken from a perspective camera moving sideways along a line, and thus only a 3D subset of the 4D plenoptic function is sampled – all rays that intersect the camera path. X-Slits views of the scene can be rendered, with the horizontal slit at the input camera path, and the vertical "virtual" slit moving with the viewer. Generating X-Slits images from an input sequence is now a simple matter of mosaicing strips of pixels, so it can be done very efficiently. Although the images are not perspective, the sense of depth and occlusions is realistic and appealing.

In order to create a complete virtual environment, it is necessary to have rays in all directions, which leads to the choice of a circular camera path and panoramic perspective cameras (see Fig. 1). We describe a setup in which a calibrated panoramic camera rotates off-axis in a circular path, so the set of rays thus collected is sufficient for generating X-Slits views with the vertical slit at *any* location inside the circle [3, 11–14]. This is described in Section 2.

Since X-Slits images are not perspective, they may appear distorted (as analyzed in [5]). The main difference between X-Slits images and perspective images is that in the former the aspect ratio of fronto-parallel surfaces is not preserved, but rather it depends on depth. Thus scene objects may appear elongated or condensed depending on their depth and the virtual slit's location. This can be reduced by normalization – the image is transformed so as to cancel


(a) (b)

(c)

Figure 1: **Spherical images. (a) An image from the input sequence, as acquired with a panoramic lens. (b) The same image in latitude-longitude representation of a hemisphere. (c) An output spherical image.**

these distortions for all objects on a chosen surface; as long as this surface crudely approximates the scene structure, the amount of distortions decreases. Essentially, normalization provides a compromise between model-based and image-based rendering: we render based on a partial set of input rays, but approximate a perspective view using a coarse model of the scene. This is discussed in Section 3.

The distortion analysis can be used for image augmentation. Since IBR scenes are static, it is often desirable to add virtual objects to the scene. When doing so, the objects must be rendered by the same projection model as the image-based background, in order for them to blend into the scene. In Section 4 we show how to modify the geometry of the augmented objects to follow the distortions of the X-Slits projection. This allows us to use perspective rendering tools to augment the images, and still obtain consistent and compelling scenes.

The supplementary material includes a walkthrough video sequence demonstrating an image-based virtual environment with an augmented object.

2. Omnidirectional X-Slits Mosaicing

We adopt here the circular X-Slits camera model, in which one of the slits is a circle in the $X-Z$ plane, and the second is a linear slit in the Y direction. In this case, each scene point defines a plane with the linear slit, which intersects the circular slit at *two* points, and thus each scene point has two rays. Of the two intersections with the circular slit, we choose the one that is closer to the scene point, and the corresponding ray is defined to be the unique ray through the scene point (this is the "outgoing" ray, emanating out of the circular slit). To complete the definition of the circular X-Slits camera model, we choose the image surface to be the sphere at infinity, meaning that the correspondence between a ray and a point in the image is defined only by the azimuth and elevation angles. Such an image is *onmidirectional*, as rays in all directions are imaged.

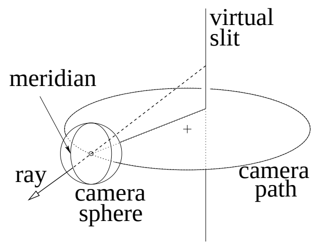
The input camera is a central camera, thus all the rays captured by the camera pass through a single point (its center of projection). Unlike regular perspective cameras which sample these rays on a planar rectangle (the image plane), our input central cameras sample the rays on a hemisphere at the center of projection of each camera. Generating a new omnidirectional X-Slits view of the scene consists of generating a spherical view from a chosen slit location, i.e., an image of the rays passing through a virtual slit as they intersect a sphere centered at a point on the slit.

The path of the input camera is assumed to be a circle of radius 1 in the $X-Z$ plane. The *virtual slit* is a vertical line passing through the point $\mathbf{e} = (x_e, 0, z_e)$, which is defined to be the location of the virtual "eye". With this definition, moving around the scene is a matter of moving the virtual slit and generating the X-Slits image corresponding to each new position. The new view at each slit location is sampled on the *output sphere*, centered around the virtual eye $\mathbf{e}$. By definition, each input pixel corresponds to an *input ray*, that passes through one of the *input camera* positions on the camera path.

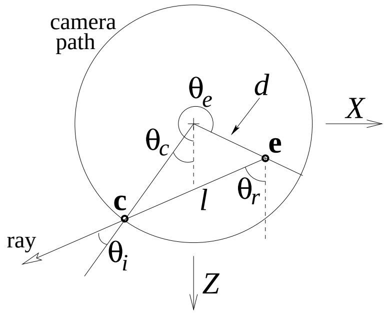
Omnidirectional X-Slits rendering is done, as in the linear case, by means of mosaicing. Strips are taken from each input image, and stitched together into a new image. The strips are selected according to the location of the virtual eye, and the result is a X-Slits image that looks as if it were taken from that location.

Specifically, given a virtual slit location, we need to determine which rays we should sample from which input camera. Each input camera center $\mathbf{c}$ defines a plane Π_c with the virtual slit; all the rays on this plane pass through the virtual slit and a point on the circular slit (the camera center $\mathbf{c}$). Thus this plane includes all the rays that are sampled from camera $\mathbf{c}$ (no other ray through $\mathbf{c}$ will be used at the current virtual slit location). The intersection of Π_c with the input camera hemisphere is a meridian (see Fig. 2a). Thus, each input camera contributes a strip of rays that lies on an input meridian.

The sampled strip is pasted as a strip on the output sphere. Since, by definition, Π_c passes through the cen-



(a)



(b)

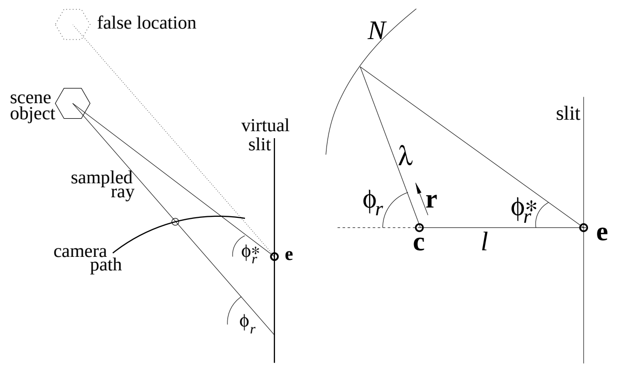
Figure 2: **Omnidirectional X-Slits mosaicing. (a) Overview: the rays passing through the virtual slit and the input camera center form a plane of rays, which intersects the input hemisphere in a meridian. (b) Top view: determining which strip of pixels is sampled from which input image.**

ter of the output sphere $\mathbf{e}$, its intersection with the output sphere is also a meridian. Note, however, that the sampled rays intersect the camera center $\mathbf{c}$, and *not* the center of the output sphere $\mathbf{e}$.

It follows from the discussion above that meridians from the input cameras are pasted as meridians in the output cameras. Hence, it would be beneficial to use an image representation that is based on latitude and longitude (Fig. 1b,c). In this representation, the coordinates of a pixel are its longitude and latitude on the sphere, so each meridian on the sphere is a column in the image. Generating omnidirectional X-Slits images becomes a matter of mosaicing vertical strips, as in linear X-Slits.

We now wish to determine which strip is sampled from which input camera. Given a vertical slit passing through $\mathbf{e}$, let us define the polar coordinates $d = \sqrt{x_e^2 + z_e^2}$, $\theta_e = \arctan \frac{x_e}{z_e}$. As can be seen from Fig. 2b (and triangle geometry), for every θ_r, the strip to paste at the θ_r meridian of the output sphere should be taken from the θ_i meridian of the input camera at θ_c, where

$$\begin{aligned} \theta_i &= \arcsin(d \sin(\theta_r - \theta_e)) \\ \theta_c &= \theta_r - \theta_i \end{aligned} \tag{1}$$



(a) (b)

Figure 3: **(a) Distortion: the object is seen at a different elevation angle from the input camera and from the virtual eye. (b) Normalization: cancelling out distortion by correcting the elevation angle of the input ray, using a normalization surface.**

3. Distortion

We shall now analyze the nature of X-Slits distortions, caused by the absence of a single center of projection. Specifically, we will compare between X-Slits images corresponding to the model described above, and the regular perspective image corresponding to an omnidirectional camera centered at $\mathbf{e}$.

Recall that the plane of rays Π_c determined by (1) is the same as if it were perspective projection, but the rays within the plane do not intersect in $\mathbf{e}$, but rather in the input camera centers, which are different for each plane Π_c. A scene point $\mathbf{p}$ that is seen at some elevation angle ϕ_r by the input camera, would be seen at a different elevation angle ϕ_r^* by the virtual eye $\mathbf{e}$ (see Fig. 3a). Without correcting the elevation angle of each ray, $\mathbf{p}$ would appear shifted vertically at a false location (Fig. 4a).

In order to cancel out this distortion, we need to determine the correct elevation angle for each input ray, estimating how the scene point would have been seen from $\mathbf{e}$. This would produce a correct perspective view of the scene, but it requires a dense and accurate knowledge of the 3D structure of the scene.

When accurate depth information is not available or hard to obtain, we can still produce appealing images by using a coarse estimation of depth. In general, we define a *normalization surface*, which crudely approximates the scene structure, and use it to reproject the rays before pasting them into the mosaic. This allows us to generate images that look compelling, without relying on an elusive depth map. The normalization procedure is described next.

3.1. Normalization

In general, normalization is done by intersecting each sampled ray with the normalization surface, and reprojecting this intersection through the virtual camera center $\mathbf{e}$. Given an input ray of azimuth θ_r and elevation ϕ_r, the input camera's position on the circle (denoted θ_c) is determined by (1), see Fig. 2. The sampled ray is defined as $\mathbf{c} + \lambda\mathbf{r}$, where $\mathbf{c} = (-\sin\theta_c, 0, \cos\theta_c)^T$ denotes the input camera location and $\mathbf{r} = (-\sin\theta_r\cos\phi_r, \sin\phi_r, \cos\theta_r\cos\phi_r)^T$ denotes the ray direction (see Fig. 3b). Given a normalization surface expressed implicitly as $N(\mathbf{p}) = 0$, the intersection is at $\min\{\lambda | N(\mathbf{c} + \lambda\mathbf{r}) = 0, \lambda > 0\}$.

To begin with, let us consider the case of a *spherical* normalization surface. Thus, $N(x, y, z) = x^2 + y^2 + z^2 - R^2$, where R is the radius of the normalization sphere. Substituting $\mathbf{c} + \lambda\mathbf{r}$ in N produces

$$\lambda^2\mathbf{r}^T\mathbf{r} + 2\lambda\mathbf{r}^T\mathbf{c} + \mathbf{c}^T\mathbf{c} - R^2 = 0 \tag{2}$$

Solving for $\lambda > 0$ yields the intersection of the ray with N at $\lambda = -k + \sqrt{k^2 + R^2 - 1}$, where $k = \cos\phi_r\cos\theta_i$. From Fig. 3b one can see that the ray should be reprojected through the virtual camera center at an elevation angle of

$$\phi_r^* = \arctan\frac{\lambda\sin\phi_r}{\lambda\cos\phi_r + l} \tag{3}$$

where $l = \cos\theta_i + d\cos(\theta_r - \theta_e - \pi)$ is the distance between the slit and the input camera (see Fig. 2b). Note that unnormalized omnidirectional X-Slits images correspond to $R \to \infty$.

Normalization onto a sphere is appropriate for scenes that lie at a relatively constant distance from the viewer, e.g., a room viewed from its center. If the room is elongated, however, the sphere provides a poor approximation of the scene's structure (Fig. 4b). In this case, normalizing onto an ellipsoid may be more appropriate.

For an ellipsoid, let us redefine $\mathbf{c}$ and $\mathbf{r}$ as homogeneous coordinates in $\mathcal{P}^3$ with the forth coordinate set to 1 and 0, respectively, and let $N(\mathbf{p}) = \mathbf{p}^T\mathbf{Q}\mathbf{p}$ where $\mathbf{Q}$ is the 4×4 matrix that describes the ellipsoid (or any quadric, for that matter). The intersection is then at $\lambda = -k + \sqrt{k^2 - m}$ where

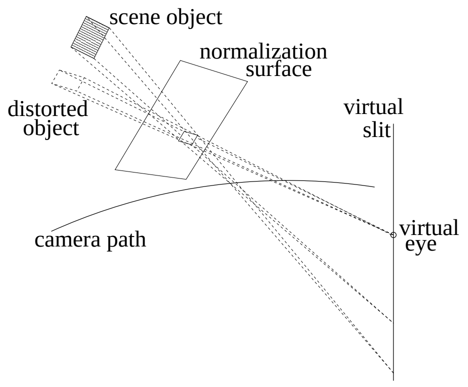
$$k = \frac{\mathbf{c}^T\mathbf{Q}\mathbf{r}}{\mathbf{r}^T\mathbf{Q}\mathbf{r}} \qquad m = \frac{\mathbf{c}^T\mathbf{Q}\mathbf{c}}{\mathbf{r}^T\mathbf{Q}\mathbf{r}} \tag{4}$$

and the ray is reprojected according to (3). Fig. 4c provides an illustrative example, and some comparisons between the different normalization methods.

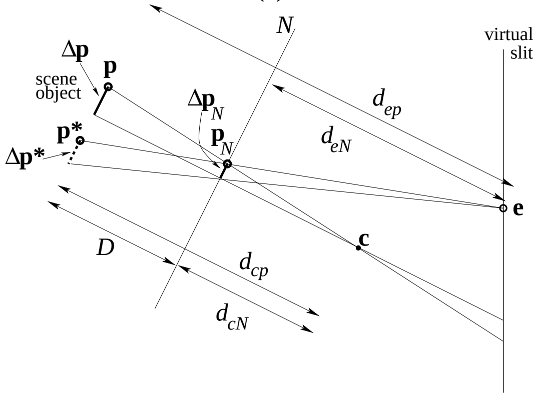
In general, one can use an estimated sparse depth map to construct a general normalization surface, and use ray tracing techniques to reproject the rays.

3.2. Measuring distortions

Since the normalization surface gives only a crude approximation of the scene structure, it is not likely to eliminate all distortions, and in some cases it may even introduce new distortions.



(a)



(b)

Figure 5: **Distortion under normalization. (a) Overview: when the normalization surface is incorrect, reprojection makes the object appear shifted vertically. (b) Side view: the aspect ratio distortion as it is related to the distances between the input camera, the virtual eye, the normalization surface and the object.**

We measure distortions in X-Slits images by the change in aspect ratio. In perspective projection, aspect ratio is preserved, and any rectangle in the scene that is parallel to the image plane would be projected into a rectangle with the same proportions between width and height. In X-Slits projection, this is usually not the case.

Normalization corrects this problem for objects that are on the normalization surface, since they are projected as if they were perspective. As we will show below, the farther an object is from the normalization surface, the more distorted its aspect ratio would be.

More formally, consider a point $\mathbf{p}$ on an object in the scene. We would like to estimate the aspect ratio distortion

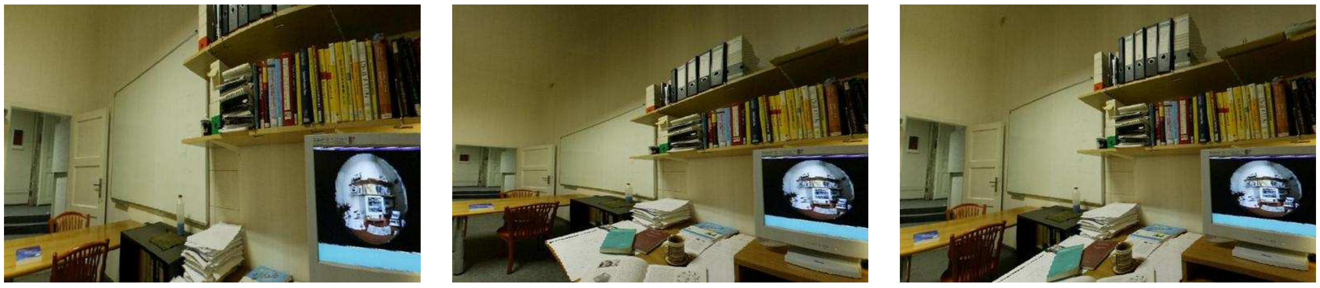

(a) (b) (c)

Figure 4: **Normalization – a view of the synthesized scene without normalization (a), and with normalization using a sphere (b) and an ellipsoid (c). Note how the aspect ratio is different for objects at different depths in the spherical case. See video sequences in the supplementary material.**

in a neighborhood around $\mathbf{p}$ in a normalized X-Slits image, when $\mathbf{p}$ is not on the normalization surface.

The point $\mathbf{p}$ and its neighborhood is captured by rays passing through the input cameras. These rays intersect the normalization surface N at $\mathbf{p}_N$, and are reprojected during normalization (Fig. 5a). This normalization is correct only for a point $\mathbf{p}$ on N; otherwise the object appears to be in a false location.

As we shall see, aspect ratio distortion is not necessarily constant, so we will estimate the *local* aspect ratio distortion at $\mathbf{p}$. Assume that the normalization surface around $\mathbf{p}_N$ is parallel to the object surface around $\mathbf{p}$. Denote the plane that contains the virtual slit and the ray through $\mathbf{p}$ by Π_p (Fig. 5b shows a view of Π_p). If the length on Π_p of the patch around $\mathbf{p}$ is $\Delta\mathbf{p}$, and the length of the patch as it is projected on N is $\Delta\mathbf{p}_N$, then it follows from triangle similarity that

$$\frac{\Delta\mathbf{p}}{\Delta\mathbf{p}_N} = \frac{d_{cp}}{d_{cN}} \tag{5}$$

where d_{cp} is the distance on Π_p between $\mathbf{c}$ and the plane tangent to the object at $\mathbf{p}$, and d_{cN} is the distance between $\mathbf{c}$ and the plane tangent to N at $\mathbf{p}_N$. For the same reason, if $\Delta\mathbf{p}^*$ is the length of the false object, then

$$\rho \equiv \frac{\Delta\mathbf{p}^*}{\Delta\mathbf{p}} = \frac{d_{ep}}{d_{cp}} \cdot \frac{d_{cN}}{d_{eN}} \tag{6}$$

Since the X-Slits projection is perspective in the horizontal direction and only introduces distortions in the vertical direction, and since normalization also deals only with the vertical direction, there is no change in the horizontal direction in the way the patch around $\mathbf{p}$ is projected. Therefore, the ratio ρ is the aspect ratio distortion that point $\mathbf{p}$ undergoes when projected with a X-Slits projection normalized by surface N (note the resemblance to the result in [15]).

Denoting the distance between the normalization surface and $\mathbf{p}$ as D (hence $d_{eN} = d_{ep} + D$ and $d_{cN} = d_{cp} + D$), when the the virtual eye is *behind* the camera path (i.e., $d_{ep} > d_{cp}$), the aspect ratio grows with D: when D is positive – the object will appear taller, and vice versa (with no change when $D = 0$). This is reversed when the virtual eye is in front of the camera path.

When the normalization surface is not parallel to the scene object and there is a difference in elevation angle between them, it can be shown that the aspect ratio becomes

$$\rho = \frac{\Delta\mathbf{p}^*}{\Delta\mathbf{p}} = \frac{d_{ep}}{d_{cp}} \cdot \frac{d_{cN}}{d_{eN}} \cdot \frac{\sin\alpha_{pi}}{\sin\alpha_{po}} \cdot \frac{\sin(\alpha_{po}-\alpha_{pN})}{\sin(\alpha_{pi}-\alpha_{pN})} \tag{7}$$

where α_{pi}, α_{po} are the angles between the object plane and the input and reprojected rays, respectively, and α_{pN} is the elevation angle difference between the object plane and the normalization surface.

3.3 Discussion

As the distance between the scene and the normalization surface decreases, so does the aspect ratio distortion. Thus, in order to achieve correct aspect ratio, we need to approximate the scene as well as possible. However, this may lead to other, often worse distortions.

Specifically, Equation (6) states that the aspect ratio is a function of the distances between the normalization surface, the scene surface, the virtual eye and the input camera. Changes in these distances across the image cause changes in aspect ratio. Therefore, strong changes in the distance to the normalization surface may cause strong changes in aspect ratio. If these changes correspond to changes in the distance to the *scene*, then they cancel each other. However, it is usually hard to obtain a depth map that fits the scene structure accurately, especially where depth changes abruptly (e.g., at depth edges); in these areas in particular, abrupt changes in the normalization surface may cause strong noticeable distortions. It is therefore often preferable to use a coarse smooth approximation of the depth map, with only moderate changes.

4. Augmented Reality

Generating realistic views of a precaptured scene in real-time is useful for virtual reality. A user's head motion may be tracked and the appropriate views of the scene can be generated and displayed at a reasonable rate. However, the rendered scene is static, and it may be desirable to add virtual objects to the scene, which would be rendered and superimposed on the X-Slits image. We shall discuss only the geometric issues of augmented reality with the X-Slits projection (a survey of augmented reality can be found in [2]).

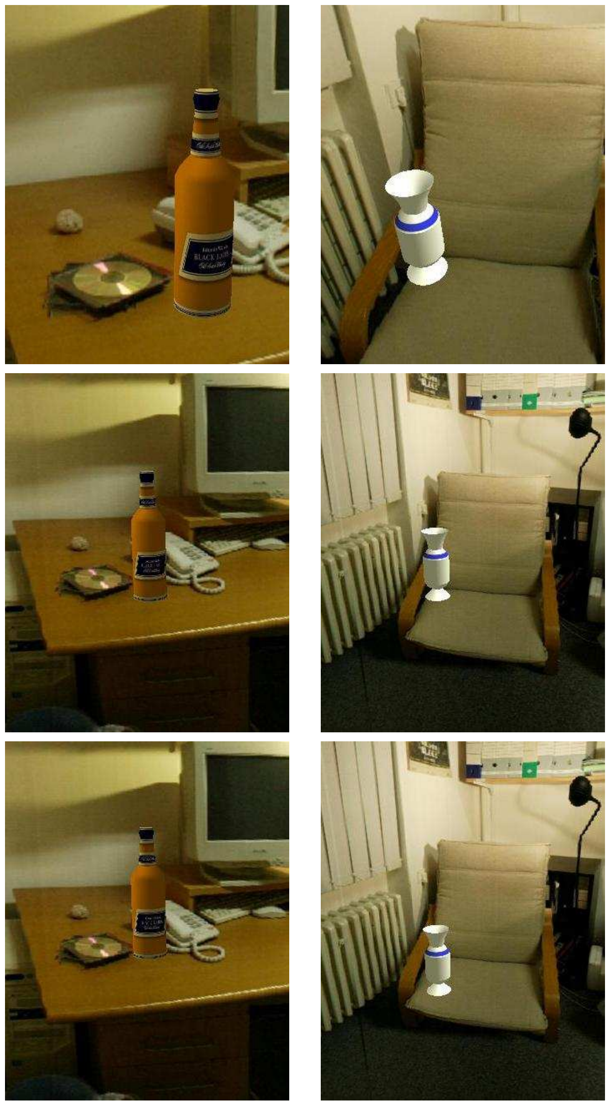


Figure 6: **Augmented Reality. (top) The image-based views of the scene with augmented objects. (middle) The same scene from different viewpoints. (bottom) Without vertical correction, the objects' locations are not correct. See also the video sequences in the supplementary material.**

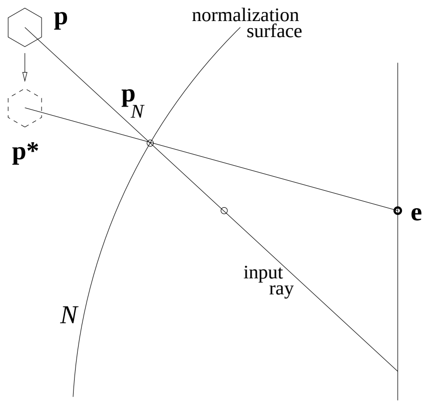


Figure 7: **Augmented Reality. Scene point $\mathbf{p}$ is reprojected into the virtual eye through point $\mathbf{p}_N$ on the normalization surface, so if a virtual object is augmented at $\mathbf{p}$, it should be shifted vertically to appear correct.**

Since the rendered scene is a X-Slits image, the added object must also be projected according to the same projection model in order to appear consistent. When adding an object in a certain place in the scene, it should appear as if it was there when viewed from different positions, and this can only be accomplished if the objects are projected using the same projection model.

Most computer graphic renderers generate perspective images. In order to use such engines for the rendering of X-Slits images, we must first transform the augmented objects in a manner similar to the reprojection discussed above, so that when projected using the regular perspective projection, they would appear correct in the X-Slits image. As shown above, the distortions and normalization associated with the X-Slits projections are in the vertical direction alone, so transforming the augmented objects is just a matter of vertical shifting.

More specifically, suppose an object is augmented at point $\mathbf{p}$, and the ray through $\mathbf{p}$ intersects the normalization surface at $\mathbf{p}_N$. In order for the object to look as if it was at $\mathbf{p}$, it must be on the reprojected ray through $\mathbf{p}_N$ (see Fig. 7). Shifting the object's location vertically prior to imaging, so that it is on this ray, would have this effect.

Given a point (x, y, z) where we wish to add an object, and given a slit at $X = x_e, Z = z_e$, the azimuth of the object is $\theta_r = \arctan\frac{-\Delta x}{\Delta z}$ where $\Delta x = x - x_e$, $\Delta z = z - z_e$, and the elevation angle relative to the input camera is

$$\phi_r \quad = \quad \arctan\frac{y}{L-l} \tag{8}$$

l above is the distance between the slit and the input camera as in (3), and $L = \sqrt{\Delta x^2 + \Delta z^2}$ is the horizontal distance between the slit and the object. It now follows that the distorted location of the augmented object is $y^* = L \cdot \tan\phi_r^*$, where ϕ_r^* is given in (3). If we shift the object's position vertically to this height, it can be projected normally and appear as if it were in the X-Slits image of the scene at the desired location (Fig.6).

Since the distortion is variable, this transformation should be applied separately to every point on the augmented object. In practice, it is usually sufficient to move the object according to just one point, e.g., the point where it is supposed to touch the scene.

5. Implementation Issues

Using the latitude-longitude representation of spherical images as discussed in Section 2, the rendering of omnidirectional X-Slits images is just a matter of sampling columns from images and pasting them in the output image, like linear X-Slits rendering. Normalization requires vertical transformation of each pixel, which may be a costly calculation for a realtime application. However, when the input camera path is small in relation to the normalization sphere, ϕ_r^* is nearly linear in ϕ_r, and a (much faster) linear transformation is sufficient.

The top and bottom pixels in each input column correspond to elevation angles $\phi_r = \pm\pi$. Substituting these values in (3), we get the normalized elevation angles of $\phi_r^* = \pm \arctan \frac{\lambda}{l}$, so normalization can be done approximately by scaling each column according to this formula.

Displaying the omnidirectional X-Slits images with a display device involves projecting the spherical image on a plane. Graphic engines that handle perspective projection are abundant, so it is useful to take advantage of such a system, by mapping the rendered image on a sphere centered about the virtual eye. Furthermore, if the sphere is approximated as a mesh, the normalization described in Section 3 can be done on each vertex of the mesh quite efficiently, instead of on each pixel of the X-Slits image.

The same graphic engine can be used for augmented objects. Provided their position is corrected according to (8), they can simply be rendered along with the mesh.

6. Conclusion

We have shown how to use the circular Crossed-Slits projection for generating realistic new omnidirectional views of a scene. Unlike other image-based methods, X-Slits image generation can be done with a relatively small amount of image data. Given an input sequence of images taken by a panoramic camera rotating off-axis, one can generate omnidirectional views of the scene from different viewpoints by means of simple mosaicing. We have shown how data should be represented in order to perform this task efficiently.

Crossed-Slits images are inherently distorted, since they are not perspective. We have analyzed these distortions and described a method for eliminating them by using a coarse approximation of the scene structure. The result is a method that combines image-based rendering by ray sampling with approximating a perspective view using a coarse 3D model.

Augmenting objects into the image-based rendered scene requires the objects to obey the same geometric model as the background. We have shown how the location of the augmented objects can be shifted, so that they would appear veridical when rendered by a perspective engine, especially when viewed from different viewpoints.

References

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

[2] R. T. Azuma. A Survey of Augmented Reality. In *Presence: Teleoperators and Virtual Environments 6*, 4:355–385.

[3] H. Bakstein, T. Pajdla. Rendering Novel Views from a Set of Omnidirectional Mosaic Images. In *Workshop on Omnidirectional Vision and Camera Networks 2003*, June 2003.

[4] C. Buehler, M. Bosse, S. Gortler, M. Cohen, L. McMillan. Unstructured lumigraph rendering. In *Proc. of ACM SIGGRAPH'01*, pp. 425–432, 2001.

[5] D. Feldman, A. Zomet. Generating Mosaics with Minimum Distortions. In *Proc. of IVR*, Washington DC, July 2004.

[6] S. Gortler, R. Grzeszczuk, R. Szeliski, M. Cohen. The lumigraph. In *Proc. of ACM SIGGRAPH'96*, pp. 43–54, 1996.

[7] S. Kang. A survey of image-based rendering techniques. In *Videometric VI*, 3641:2–16, Jan 1999.

[8] M. Levoy, P. Hanrahan. Light field rendering. In *Proc. of ACM SIGGRAPH'96*, pp. 31–42, 1996.

[9] L. McMillan, G. Bishop. Plenoptic modeling: An image-based rendering system. In *Proc. of ACM SIGGRAPH'95*, pp. 39–46, Los Angeles, California, August 1995.

[10] T. Pajdla. Stereo with Oblique Cameras. *IJCV*, 47(1):161-170, Kluwer May 2002.

[11] S. M. Seitz. The Space of All Stereo Images In *Proc. of ICCV'01*, pp. I:26–33, Vancouver, Canada, July 2001.

[12] H. -Y. Shum, L. He. Rendering with concentric mosaics. In *Proc. of SIGGRAPH'99*, pp. 299–306, August 1999.

[13] H. -Y. Shum, R. Szeliski. Stereo reconstruction from multi-perspective panoramas. In *Proc. of ICCV'99*, 1:14–21, 1999.

[14] T. Takahashi, H. Kawasaki, K. Ikeuchi, M. Sakauchi. Arbitrary view position and direction rendering for large-scale scenes. In *Proc. of CVPR*, pp. 296–303, Hilton Head, SC, June 2000.

[15] A. Zomet, D. Feldman, S. Peleg, D. Weinshall. Mosaicing New Views: The Crossed-Slits Projection. *TPAMI*, 25(6):741–754, 2003.