Immersive Planar Display using Roughly Aligned Projectors

Ramesh Raskar

The Office of the Future Group, P.I. Henry Fuchs
University of North Carolina at Chapel Hill
http://www.cs.unc.edu/~raskar/Planar/

Abstract

When a projector is oblique with respect to a planar display surface, it creates keystoning and the projected image is distorted. We present a rendering technique to display perspectively correct images for a moving user. This allows using roughly aligned projectors and eliminates the need for frequent electro-mechanical adjustments.

The rendering process has no additional cost and can be implemented with traditional graphics hardware. We compute the collineation induced due to the display plane during preprocessing. The main idea of the paper is to use this collineation to render and warp the images of 3D scenes in a single pass via approximation of the depth buffer. We also describe how this method can be extended to display systems with multiple overlapping projectors. This technique can be easily used in CAVE, Immersive Workbenches and PowerWalls.

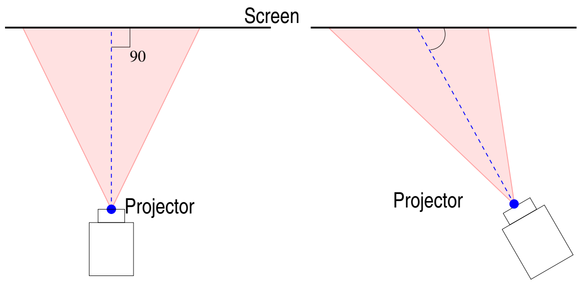

Figure 1. (a) Traditional projectors are orthogonal and create rectangular images (b) Oblique projectors create keystoned images.

1 Introduction

Projectors are typically mounted in such a way that their optical axis is perpendicular to the planar display surface. Such configurations are also used in immersive environments to render perspectively correct imagery for a head-tracked moving user. They include CAVE [4], PowerWall [10] ($m \times n$ array of projectors) or ImmersiveDesk (back-lit and tilted desktop workbenches) [5][11][1]. By design, typical display systems try to maintain the image plane parallel to the plane of the display surface. However, this leads to the need of constant electro-mechanical alignment and calibration of the projectors, screens and the supporting structure.

When the projector optical axis is not perpendicular to the display screen, the resultant image is keystoned and appears distorted (Fig 1). We will call this type of projection as *oblique projection* and the traditional projection as *orthogonal projection*. When a projector is roughly positioned, it is generally oblique. Even if the projector is orthogonally positioned in large display systems, after a period of time it can become oblique due to mechanical or thermal variations. We address the problem of rendering perspectively correct images with oblique projectors. Our goal is to avoid frequent mechanical adjustments and instead, compensate for the image distortion using the graphics pipeline. Some techniques already exist to pre-warp the images to avoid visible distortion. However, our technique achieves the results without additional cost of rendering or affecting the visual quality. We use the collineation between the points on the display screen and the projector pixels, induced due to the planarity of the screen. By using the collineation during rendering we show that oblique projectors can be used to easily create immersive displays. We use traditional graphics pipeline with a modified projection matrix and an approximation of the depth-buffer.

1.1 Previous Approach

In most cases, the problem of obliqueness is avoided simply by design. The orthogonal projectors create well-defined rectangular images. In some projectors the projected cone appears to be off-axis, but the display screen is still perpendicular to the optical axis. In workbenches

[5], mirrors are used, but the resultant image is designed to appear rectangular (corresponding to rectangular framebuffers). Great efforts are taken to ensure that the projector image plane is parallel to the display screen and that the corner pixels are matched to pre-defined locations on the screen. This leads to expensive maintenance and frequent adjustments of the projectors, screens and the supporting structure. The structure itself is usually very large and rigid. In CRT projectors, some distortions are corrected by adjusting via a simple electromagnetic warp in the driving circuit. However, it is not applicable to newer, compact digital projectors.

When the projectors are oblique, a popular technique implemented in software is to use a two-pass rendering method to pre-warp the projected image. In the first pass, one computes the image of the 3D virtual scene from the viewpoint of the head-tracked user. The result is stored in texture memory and in the second pass the image is warped using texture mapping. This warped image when displayed by the projector appears perspectively correct to the user. Such two-pass rendering techniques have been used for planar surfaces [15] as well as irregular display surfaces [6][13][14][12]. Some digital projectors also provide limited keystone correction facility by warping images before they are projected.

The second pass of rendering increases the computation cost and in the case of immersive displays may also increase the rendering latency. In addition, texture mapping of limited resolution image of the result of the first pass leads to resampling artifacts such as aliasing and jaggies. Due the loss in the visual quality, such two-pass techniques are avoided in most of the popular setups and currently used only in experimental systems. The two-pass techniques may become useful when the projectors with very high resolution are available.

1.2 Single-pass Approach

In our approach, we use a single-pass technique to achieve rendering and warping. There are three logical steps. We first compute the image of the virtual 3D scene assuming the projector is orthogonal. Then we warp this image to compensate for the obliqueness of the projector using the collineation between display plane and projector's image plane. The depth values are also affected due to the warping, and hence they are transformed so that they can be used for visibility computations. However, all the three steps are implemented using a single 4×4 projection matrix as in the traditional rendering using graphics hardware. Hence, the rendering is performed without any additional cost and at the resolution of the framebuffer.

Section 2 describes the details of creating a projection matrix that can achieve rendering and warping. Section 3 addresses the problem of non-linearity introduced in the depth buffer values. In section 4, we describe how the techniques can be used for display systems with multiple projectors creating images on a shared planar surface. Details of our prototype implementation are in section 5. In this paper, we will focus on front projection display system, but the techniques are applicable to rear projection systems such as CAVE or Immersive Workbenches as well.

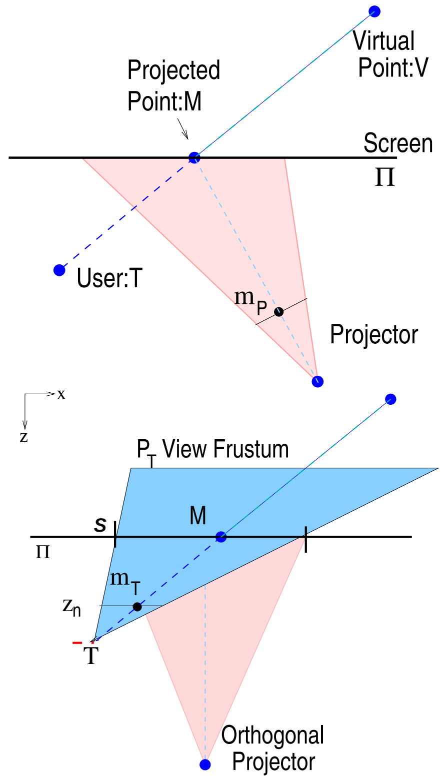

Figure 2. (a) Oblique projectors can be used to create perspectively correct imagery of virtual 3D scenes. (b) A simple off-axis projection matrix P_T is sufficient for orthogonal projectors.

2 Oblique Projection

Consider rendering images of virtual 3D objects for a moving user using an oblique projector. As shown in Fig 2(a), let us say the user is at T. For planar display surfaces, the virtual 3D point V on the object must be displayed at M. Since the projector pixel m_P illuminates the point M on the screen, the corresponding rendering process should map point V to pixel m_P. This can be achieved in two steps. First compute the image of V from the user location T, which we denote by m_T (Fig 2(b)). Then find the mapping between m_T and m_P (Fig 3). A simple observation is

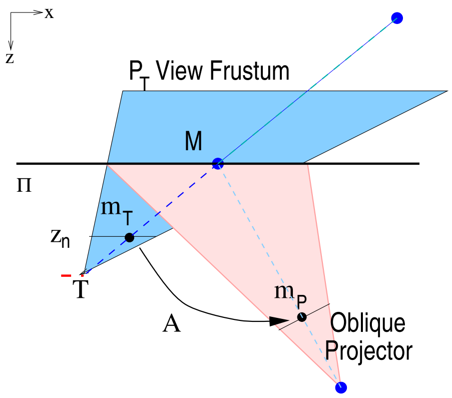

Figure 3. The modified projection matrix achieves off-axis projection P_T followed by a collineation $A_{4\times4}$.

that the two images of a common virtual point are related by a collineation induced by the plane of the screen. The collineation is well known to be a 3×3 matrix defined up to scale, which we denote by A. This observation allows us to create a new projection matrix as a product of a traditional off-axis projection matrix,P_T (from the user's viewpoint) and a matrix, $A_{4\times4}$ (from the 3×3 collineation matrix).

2.1 Orthogonal Projection

The first step of creating image m_T for V for a given user location is same as the process of displaying images assuming orthogonal projectors (Fig 2(b)). The method is currently used in many immersive display systems. Here we will describe a more general technique that will be extended in the next subsection. We also need to deal with possibly non-rectangular projected image.

Without a loss of generality, let us assume that the display plane, Π, is defined by $z = 0$. There are various ways to create the projection matrix P_T and the corresponding collineation matrix $A_{4\times4}$. We will use a method that updates P_T as the user moves but the collineation matrix remains constant.

Let us consider the definition of P_T for the user at T. We create an (world coordinate) axis aligned rectangle S on Π bounding the keystoned quadrilateral illuminated by the projector. Define a view frustum by first creating a pyramid with $T = [T_x, T_y, T_z]$ and the four corners of S. If the depth of near and far plane is z_n and z_f, respectively, then truncate the pyramid with planes, $z = T_z - z_n$ and , $z = T_z - z_f$. The process of creating the view frustum is similar to the one used in graphics APIs (such as OpenGL's *glFrustum* setup) and hence can be easily implemented. The projection matrix for this view frustum is, $P_T = Frustum(T, S, z_n, z_f)Translate(-T)$ and is updated as the user moves.

If the projector was orthogonal, S could be the same as the rectangular area illuminated by the projector. However, even if the area is not rectangular (because, say, the shape of framebuffer chosen for projection is not rectangular), the projection matrix P_T can be used to render correct images. This technique, for example, can be used for immersive planar displays with traditional orthogonal projectors.

2.2 Collineation

The image calculated assuming orthogonal projector can be corrected to compensate for the obliqueness using collineation. The collineation between image created with P_T and the image to be rendered in projector's framebuffer is induced due to the plane of the screen. The collineation matrix, $A_{3\times3}$ is well-known to be defined up to scale and maps pixel coordinates from one image to a second image [7][3]. We need to compute the 8 unknown parameters of $A_{3\times3}$ that relates the two images of V: m_T due to P_T, and its image in projector, m_P.

$$
\begin{aligned}
m_P &\cong A_{3\times3} m_T \\
\begin{bmatrix} m_{Px} \\ m_{Py} \\ 1 \end{bmatrix} &\cong \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & 1 \end{bmatrix} \begin{bmatrix} m_{Tx} \\ m_{Ty} \\ 1 \end{bmatrix}
\end{aligned} \quad (1)
$$

Note that the choice of view frustum for P_T makes this collineation independent of the user location and hence remains constant. (The symbol $\cong$ denotes equality up to scale). If the 3D positions of points on Π illuminated by four or more pixels of the projector are known, the 8 parameters of the collineation matrix, $A = [a_{11}, a_{12}, a_{13}; a_{21}, a_{22}, a_{23}; a_{31}, a_{32}, 1]$, can be easily calculated. Since the collineation matrix remains constant, it can be computed off-line. The Implementation section describes a simple method to simultaneously calculate the required collineation matrix and transformation between tracker and world coordinate system.

2.3 Single-pass Rendering

We would ideally like to compute the pixel coordinates m_P directly from the 3D virtual point V. This can be achieved by creating a single matrix from $A_{3\times3}$ and P_T. As a traditional projection matrix, P_T transforms 3D homogeneous coordinates into 3D normalized homogeneous coordinates (4 element vectors). Typically, one can obtain the pixel coordinates after the perspective division. We create a new matrix, $A_{4\times4}$ to transform these normalized 3D coordinates to the projector pixel coordinates, but trying to keep the depth values intact.

$$A_{4\times4} = \begin{bmatrix} a_{11} & a_{12} & 0 & a_{13} \\ a_{21} & a_{22} & 0 & a_{23} \\ 0 & 0 & 1 & 0 \\ a_{31} & a_{32} & 0 & 1 \end{bmatrix} \qquad (2)$$

The complete projection matrix is $P'_T = (A_{4\times4}P_T)$. (It is easy to verify $[m_{Px}, m_{Py}, m_{Pz}, 1]^T \cong A_{4\times4}P_T[V,1]^T$.) This achieves the desired rendering and warping using a single projection matrix without additional cost. The image in the framebuffer is generated in a single pass and hence there are no resampling artifacts. Finally, when the image is projected on any surface coplanar with Π, the displayed virtual object appears perspectively correct.

3 Visibility using Depth Buffer

Although the naive approach described above creates correct images of virtual 3D points, it is important to note that the traditional depth-buffer cannot be effectively used for visibility and clipping.

3.1 Problems with depth buffer

The depth values of virtual points between near and far plane due to P_T are mapped to $[-1, 1]$. Let us say, $[m_{Tx}, m_{Ty}, m_{Tz}, m_{Tw}]^T = P_T[V, 1]^T$ and $m_{Tz}/m_{Tw} \in [-1, 1]$. After collineation, the new depth value, is actually

$$m_{Pz} = \frac{m_{Tz}}{(a_{31}m_{Tx} + a_{32}m_{Ty} + m_{Tw})} \qquad (3)$$

The modified depth value (i) may not be in $[-1, 1]$ resulting in undesirable clipping of the geometry with near and far plane of the view frustum and (ii) is a function of pixel coordinates, changes quadratically and hence cannot be linearly interpolated during scan conversion for visibility computation (Fig 4(b)). During traditional perspective projection rendering, homogeneous coordinates allow hyperbolic interpolation of depth value so that the task for computing depth values at each pixel during scan conversion involves a simple linear interpolation. However, we cannot achieve two successive hyperbolic interpolations. In other words, we must first compute an image with P_T ("divide by w"), and then warp the resultant image. Unfortunately, this leads to the two-pass rendering method which we are trying to avoid: first render the image and load it in texture memory and then achieve warping using texture mapping.

3.2 Approximation of depth buffer

We can achieve the rendering and warping in a single pass, however, using an approximation of the depth buffer.

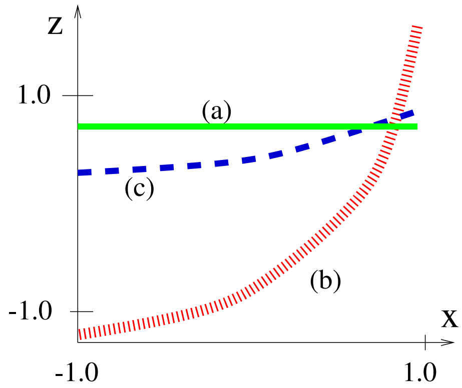

Figure 4. The plot shows depth buffer values along a scan line for points along constant depth. (a) Using P_T (b) After collineation, $A_{4\times4}P_T$, the depth values range beyond $[-1, 1]$ and do not change linearly (c) With an approximation of depth-buffer, $A'_{4\times4}P_T$, traditional graphics pipeline can be used to render perspectively correct images for a tracked moving user.

Note that, because the rectangle S is chosen to be larger than displayed imagery, m_{Tx}/m_{Tw} and $m_{Ty}/m_{Tw} \in [-1, 1]$ for points displayed inside S. Hence,

$$\frac{(1 - |a_{31}| - |a_{32}|)m_{Tz}}{(a_{31}m_{Tx} + a_{32}m_{Ty} + m_{Tw})} \in [-1, 1] \qquad (4)$$

This will avoid undesired clipping with near and far plane. Further, by construction of P_T, the angle between projector's optical axis and the normal of the planar surface is same as the angle between the optical axis and normal of retinal plane of view frustum for P_T. Thus, if this angle is small (i.e. $|a_{31}|$ and $|a_{32}| \ll 1$), the depth values are modified but the changes are monotonic and almost linear across the framebuffer as shown in Fig 4(c). The modified projection matrix is $A'_{4\times4}P_T$, where

$$A'_{4\times4} = \begin{bmatrix} a_{11} & a_{12} & 0 & a_{13} \\ a_{21} & a_{22} & 0 & a_{23} \\ 0 & 0 & 1 - |a_{31}| - |a_{32}| & 0 \\ a_{31} & a_{32} & 0 & 1 \end{bmatrix} \qquad (5)$$

4 Multiple Projectors

The same technique can also be extended to register multiple overlapping projectors to create larger displays on a wall. Some popular systems using $m \times n$ array of projectors are PowerWall [10] and Information Mural [8]. The major challenge is matching the adjacent images so that the displayed image appears seamless. Again, this is typically achieved with rigid structure which is difficult to maintain and needs frequent alignment. We can instead use roughly aligned projectors or even relatively oblique projectors.

Consider two overlapping projectors creating seamless images of virtual 3D scene on a shared planar surface. Let us say the projection matrices for the two projectors are P_1 and P_2. We exploit the collineation between the images in the two projector framebuffers. If the 3×3 collineation matrix mapping pixel coordinates from projector 1 to those in projector 2 is A_{21}, then the projection matrix P_2 can be replaced with $P_2' = A_{21-4\times4}P_1$.

Note that, as we have seen earlier, although P_2' will create correct images of 3D virtual points, we cannot use the traditional depth buffer for visibility and are forced to use approximated depth values. If P_1 itself is a result of oblique projection, so that $P_1 = A'_{4\times4}P_T$, the corresponding collineations $A_{3\times3}$, and A_{21} are used together. In this case, the axis-aligned rectangle S bounds the area illuminated by both the projectors. Let us say $A_{21T} = A_{21}A_{3\times3}$. We first create the corresponding 4×4 matrix $A'_{21T-4\times4}$ (similar to equation 5). We then replace P_2 with $P_2' = A_{21T-4\times4}P_T$ to achieve correct rendering, warping and depth buffer transformation. In practice, it is easier to compute the collineation A_{21} than calibrating the projector and computing the projection matrix P_2.

It is also necessary to achieve intensity blending of overlapping projector. Due to the exact mapping defined by A_{21}, it is very easy to calculate the actual quadrilateral (or triangle in some cases) of overlap on screen as well as in projector image space. The intensities of pixels lying in this quadrilateral in both projector are weighted to achieve intensity roll-off [9][2][16] and necessary blending. The technique for intensity blending in case of two-pass rendering are described with more details in [14][15] and are applicable here with minor modifications. On the other hand, corners of projected regions can be truncated to create large and rectangular imagery with multiple projectors. The region in projector framebuffer is masked off by rendering a black polygon at near plane.

5 Implementation

We have implemented a system with three overlapping projectors displaying images on a 12x8 ft planar screen. The user is tracked with an optical tracking system. The origin in world coordinate system (WC) is defined to be near the center of projection of the first projector. The collineation between desired image for the user and the first projector's framebuffer is calculated using approximately 8 points. The mapping between overlapping projectors' pixels (again described by a collineation) are actually computed by observing individual pixels of each projector with a distortion-free uncalibrated camera. The camera field of view covers projection of all three projectors and collineation between each projector and the camera are computed. Note that, we do not need to determine explicit correspondences between pixels of overlapping projectors. For example, if A_{c1} is the collineation between first projector and the camera, and A_{c2} is the relationship between second projector and the camera, then A_{21} is simply $A_{c2}^{-1}A_{c1}$.

During preprocessing, one needs to compute the extent of projector illumination in WC, transformation between the tracker coordinate system (TC) and WC, and the collineation $A_{3\times3}$. There are many ways of computing the collineation and TC-WC transform. For example, one can have pre-defined locations on screen with known 3D coordinates (e.g. corners of the cube in CAVE, or ruled surface on a workbench). In this case, it is relatively easy to compute which projector pixels illuminate these markers and then compute the unknown parameters.

Usually pre-defined markers are not available, for example, when one wants to simply aim a projector at a screen and render head-tracked images. Many techniques exist to compute the desired transformation. We will describe a technique that computes the collineation and transformation simultaneously. Assume the screen defines the WC, the display plane $\Pi \equiv z = 0$, plane normal (towards the user) specifies the positive z-axis with the origin (preferably) near the center of projected quadrilateral. We take tracker readings of locations on the screen illuminated by turning on one projector pixel at a time. We will compute the transformation between the tracker coordinate system and WC (with origin on screen). Assign one of the location illuminated by a projected pixel as origin of WC, O, and note its tracker reading (effectively giving the translation between origins in both coordinate systems). To find the relative rotation, find the best-fit plane in tracker coordinates for the screen, giving the normal to the screen denoted by vector $r3$. Assign one illuminated pixel in approximate horizontal direction, M_x, as a point on WC x-axis. Find the vector $M_x^{TC} - O^{TC}$ in tracker coordinates, normalize it to get vector $r1$. The rotation is $R = [r1;\ r3 \times r1;\ r3]$. The transformation is given by 4×4 matrix $[R\ (-R * O^{TC});\ 0\,0\,0\,1]$. Transform the 3D coordinates of points on screen illuminated by projector pixels and finally compute the collineation $A_{3\times3}$. The collineation allows computation of extents of projector illumination in WC and hence the axis-aligned bounding rectangle S.

For multiple overlapping projectors, we use the camera to find collineation between the projected images and also use the same information for computing weighting functions for intensity blending. More details are available at *http://www.cs.unc.edu/~raskar/Planar/*.

5.1 Summary of Techniques

Here are the sequence of steps suggested for using an oblique projector to create immersive display.

During pre-processing :

- Turn on four or more projector pixels, $m_{P1}...m_{Pn}$, one at a time and find the tracker reading of the locations on screen illuminated by these pixels, $M_1...M_n$.
- Find transformation between tracker and world coordinate system as described above. We will denote transformed points, which now lie on the plane $\Pi \equiv z = 0$, by M_i^Π.
- Find collineation, $A_{\Pi P}$, between M_i^Π and m_{Pi}. Using $A_{\Pi P}^{-1}$, compute 3D points on Π illuminated by corners pixels of the projector. Then find axis aligned rectangle, S, that bounds the four corners of the illuminated quadrilateral (the min and max of x and y coordinates).
- For a random user location T (e.g. $[0, 0, 1000mm]$), compute P_T for a frustum created with S. Find normalized image coordinates $m_{Ti} \cong P_T M_i^\Pi$. Compute the collineation $A_{3\times3}$, between m_{Ti} and projector pixels m_{Pi}. Create $A'_{4\times4}$ using equation (5).

During run-time at every frame:

- Update T for the new user location and compute P_T (using graphics API function such as *glFrustum*).
- Use $A'_{4\times4}P_T$ as the new projection matrix during rendering.

As one can see, the only human intervention involved is finding 3D tracker readings M_i for surface points illuminated by four of more projector pixels m_{Pi}. Illuminate projector pixels that are distributed in framebuffer to improve the accuracy of solution for $A_{3\times3}$ using least-squared error method.

6 Applications

The new techniques can be easily used in current immersive display systems. For example, in immersive workbenches, it is easy to take tracker readings at a few points (illuminated by the projector), compute the collineation offline and then use the modified projection matrix for rest of the session. There is no need to worry about exact alignment of the projector or possible shift in tracker/projector coordinate transformations.

In CAVE, one can roughly align projectors so that they actually create images larger than usable size of the screen. Since the usable screen sections are already marked (we can assume that the markers are stable as compared to the configuration of projectors and mirror), one only needs to find the projector pixels that illuminate those markers. The corners of the cube also allow geometric continuity across the screens that meet at right angles. The part of the imagery being projected outside the usable screen area can be masked during rendering.

Finally, this technique is useful for quickly creating a simple immersive display by setting up a screen, a projector and a tracking system. For stereo displays, the off-axis projection matrix P_T is different for the left and the right eye, but the collineation matrix $A_{3\times3}$ remains constant. As shown in the video, the same technique can also be used to create registered, seamless images with multiple overlapping projectors.

6.1 Issues

The techniques are valid only when the assumed pinhole projection model (dual of the pin-hole camera model) is valid. Projectors typically have a limited depth of field and hence when they are oblique all pixels may not be in focus. Thus, the imagery in focus for only a limited range of angles between the screen plane normal and the projector optical axis. The intensities across the screen are also not uniform when projector image plane is not parallel to the screen. However, this can be easily compensated by changing the intensity weights during rendering (using, for example alpha blending). The transformation of depth-buffer values essentially reduces the usable range of depth values and hence the depth resolution. Further we have not eliminated the non-linearity of the depth values. Although it is not a problem when the projector is almost orthogonal, at very large angles, one can see popping of hidden polygons that are just behind polygons covering a large area in image space. To avoid popping, one can subdivide large polygons. In practice, these side-effects are very rare and as seen in the video, available at the project webpage, the displayed images are visually equivalent to those generated by orthogonal projection systems.

7 Conclusion

We have presented a technique that allows rendering correct images even when the projectors are only roughly positioned. Techniques for pre-warping the images using a two-pass rendering are already available. However, the two major advantages of our technique are that it does not increase the cost of rendering and does not introduce resampling artifacts such as aliasing. The technique of warping images of a plane using collineation are well known, but we extend this to rendering correct images of 3D scenes via approximation of the depth buffer. The possibility of rendering images with oblique projectors that are visually equivalent can eliminate the need of cumbersome electro-mechanical adjustment of projectors, screens and supporting structure.

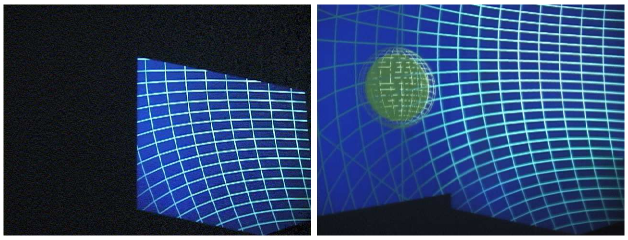

Figure 5. (a) Example of image displayed by a highly oblique projector which is keystoned and but corrected using collineation. (b) Its contribution to image displayed by three overlapped projectors after intensity blending.

Acknowledgments: I would like to thank Gary Bishop for useful discussion that led to this paper. I would also like to thank my colleagues in the Office of the Future group at UNC: Mike Brown, Ruigang Yang, Wei-Chao Chen, Herman Towles, Greg Welch, Brent Seales and Henry Fuchs, for helpful discussions and prototype implementation.

References

[1] M. Agrawala, A. Beers, B. Froehlich, P. Hanrahan, I. MacDowall, and M. Bolas. The Two-User Responsive Workbench: Support for Collaboration Through Individual Views of a Shared Space. In *SIGGRAPH 97 Conference Proceedings*, Aug. 1997.

[2] P. J. Burt and E. H. Adelson. A Multiresolution Spline with Applications to Image Mosaic. *ACM Trans. on Graphics*, (2):217–236, 1983.

[3] A. Criminisi, I. Reid, and A. Zisserman. Duality, Rigidity and Planar Parallax. *Lecture Notes in Computer Science*, 1407:846–??, 1998.

[4] C. Cruz-Neira, D. Sandin, and T. DeFanti. Surround-screen Projection-based Virtual Reality: The Design and Implementation of the CAVE. In *SIGGRAPH 93 Conference Proceedings*, volume 27, pages 135–142, Aug. 1993.

[5] M. Czernuszenko, D. Pape, D. Sandin, T. DeFanti, L. Dawe, and M. Brown. The ImmersaDesk and InfinityWall Projection-Based Virtual Reality Displays. In *Computer Graphics*, May 1997.

[6] J. Dorsey, F. Sillion, and D. Greenberg. Design and Simulation of Opera Lighting and Projection Effects. In *SIGGRAPH 95 Conference Proceedings*, pages 41–50, Aug. 1991.

[7] O. Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. MIT Press, Cambridge, Massachusetts, 1993.

[8] G. Humphreys and P. Hanrahan. A Distributed Graphics System for Large Tiled Displays. In *IEEE Visualization*, San Francisco, CA, Oct. 1999.

[9] P. Lyon. Edge-blending Multiple Projection Displays On A Dome Surface To Form Continuous Wide Angle Fields-of-View. In *Proceedings of 7th I/ITEC*, pages 203–209, 1985.

[10] PowerWall. http://www.lcse.umn.edu/research/ powerwall/powerwall.html.

[11] Pyramid Systems. http://www.pyramidsystems.com/.

[12] R. Raskar, M. Brown, Y. Ruigang, W. Chen, G. Welch, H. Towles, B. Seales, and H. Fuchs. Multiprojector Displays using Camera-based Registration. In *IEEE Visualization*, San Francisco, CA, Oct. 1999.

[13] 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. In *SIGGRAPH 98 Conference Proceedings*, July 1998.

[14] R. Raskar, G. Welch, and H. Fuchs. Seamless Projection Overlaps Using Image Warping and Intensity Blending. In *Fourth International Conference on Virtual Systems and Multimedia*, Gifu, Japan, Nov. 1998.

[15] R. Surati. *Scalable Self-Calibrating Display Technology for Seamless Large-Scale Displays*. PhD thesis, Massachusetts Institute of Technology, 1999.

[16] R. Szeliski. Video Mosaics for Virtual Environments. *IEEE Computer Graphics and Applications*, 16(2):22–30, Mar. 1996.