Oblique Projector Rendering on Planar Surfaces for a Tracked User

Ramesh Raskar
University of North Carolina at Chapel Hill

Projectors are typically mounted so 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, PowerWall ($m \times n$ array of projectors) or ImmersiveDesk (back-lit and tilted desktop workbenches). 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 display systems.

We will show that it is possible to allow oblique projection on planar display surfaces and still render correct imagery of 3D scenes without additional computational cost. We describe here how oblique projection can eliminate need of frequent alignment with simple calibration. We use traditional graphics pipeline with a modified projection matrix and an approximation of the depth-buffer.

Oblique Projection

Consider rendering the virtual point V for the user at T using an oblique projector as shown in Fig **??**.a. For planar display surfaces, the images m_T and m_P of the virtual point V can be computed by first finding projection of V onto the display surface, M. A simple observation is that the two images of common virtual point are related by a collineation, which is well known to be a 3×3 matrix defined upto scale. This observation allows us to create a new projection matrix during rendering for the projector 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).

Without a loss of generality, lets assume that the display plane, Π, is defined by $z = 0$. There are various ways to create P_T and $A_{4\times4}$. We will use a method that updates P_T as the user moves but the collineation matrix remains constant. We create an axis aligned rectangle S on Π bounding the key-stoned quadrilateral illuminated by the projector. Define a view frustum by first creating a pyramid with T and the four corners of S and the truncating it with a near plane, $z = T_z - z_n$, and a far plane, $z = T_z - z_f$. This is similar to OpenGL's *glFrustum* setup. 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.

Next we calculate the collineation between images of V: m_T due to P_T, and its image in projector, m_P. Note that the choice of view frustum for P_T makes this collineation independent of the user location and hence remains constant. 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. We create a new matrix, $A_{4\times4}$ to transform the pixel coordinates but trying to keep the depth values intact,

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

The complete projection matrix is $A_{4\times4}P_T$ and renders perspectively correct images.

Depth Buffer Approximation

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. The depth values of virtual points between near and far plane due to P_T are mapped to $[-1, 1]$. Lets 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 $m_{Tz}/(a_{31}m_{Tx} + a_{32}m_{Ty} + m_{Tw})$ which (i) may not be in $[-1, 1]$ resulting in undesirable clipping and (ii) is a function of pixel coordinates, changes quadratically and hence cannot be linearly interpolated during scan conversion for visibility computation (Fig **??**.c). In general, we cannot achieve two hyperbolic interpolations for the depth values with a single 4×4 matrix. In other words, we must first compute an image with P_T ("divide by w"), and then warp the resultant image. This requires a

two-pass rendering method: first render the image and load it in texture memory and then achieve warping using texture mapping. However, we can achieve the rendering and warping in a single pass using an approximation of the depth buffer. Note that m_{Tx}/m_{Tw} and $m_{Ty}/m_{Tw} \in [-1, 1]$ for points rendered inside the rectangle S. Hence $(1 - |a_{31}| - |a_{32}|)m_{Tz}/(a_{31}m_{Tx} + a_{32}m_{Ty} + m_{Tw})$ is guaranteed to be in $[-1, 1]$. 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 retinal plane of 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 **??**.c.

$$A'_{4\times4} = \begin{bmatrix} a_{11} & a_{12} & 0 & a_{13} \\ a_{21} & a_{22} & 0 & a_{23} \\ 0 & 0 & 1 - |a_{31}| - |a_{32}| & 0 \\ a_{31} & a_{32} & 0 & 1 \end{bmatrix} \tag{2}$$

Applications

The modified projection matrix can be easily calculated by measuring the tracker-sensor at the four corners of the illuminated quadrilateral. The effect of quadratic changes in depth values is minimized when the projector is almost perpendicular. The modified method has no additional rendering cost and we have easily implemented it using traditional graphics pipeline. For example in CAVE or in Immersive Workbenches, special effort is taken to map projectors pixels to the pre-defined corners. Using the technique described, a rough positioning followed by a simple calibration is sufficient to render correct images in a coordinate system registered with the tracker. As shown in the accompanying video, we have implemented the system to render perspectively correct images of 3D scenes for a tracked user. We also demonstrate how the technique can be extended to register multiple overlapping projectors to create larger displays on a wall. More details are available at http://www.cs.unc.edu/~raskar/Oblique/.

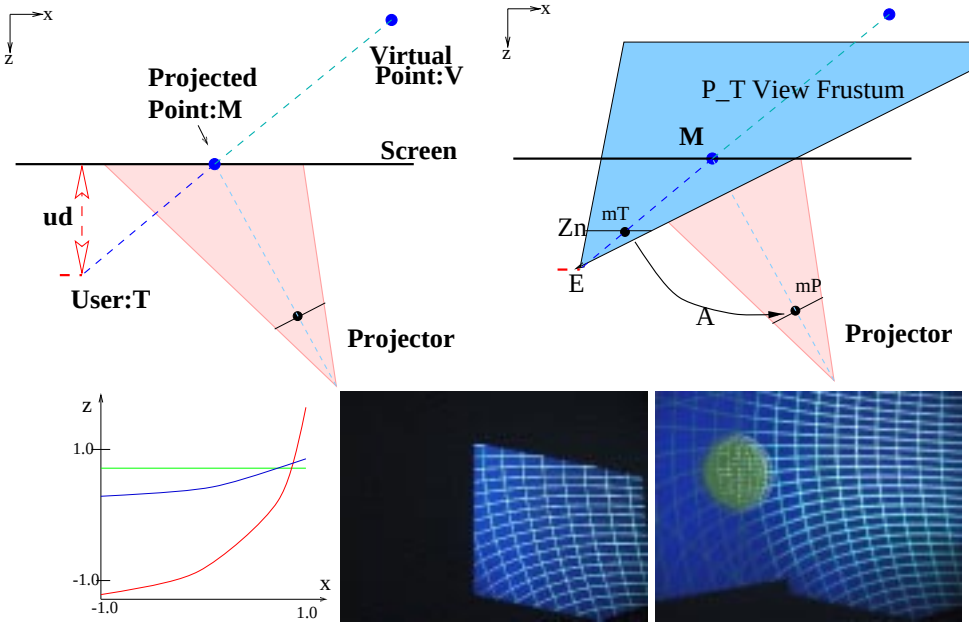

Figure 1: (a) Oblique projectors create key-stoned imagery. (b) The modified projection matrix achieves off-axis projection P_T followed by a collineation $A_{4\times4}$. (c) The plot shows depth buffer values along a scan line for points along constant depth. Using P_T (green). After collineation (red) the depth values range beyond $[-1, 1]$ and do not change linearly. With an approximation of depth-buffer (blue) traditional graphics pipeline can be used to render perspectively correct images for a tracked moving user. (d) An oblique projector (e) and its contribution to overlapped projectors.

Acknowledgments: I would like to thank Gary Bishop and 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.