International Journal of Image and Graphics
Vol. 8, No. 2 (2008) 243–261

World Scientific
www.worldscientific.com

DESIGN AND IMPLEMENTATION OF A FOVEAL PROJECTION DISPLAY

BENJAMIN A. AHLBORN*, OLIVER KREYLOS†
SOHAIL SHAFII‡, BERND HAMANN§
and OLIVER G. STAADT¶

Department of Computer Science, University of California
One Shields Ave, Davis, CA 95616, USA
**Ahlborn@hotmail.com*
†*okreylos@ucdavis.edu*
‡*ssshafii@ucdavis.edu*
§*bhamann@ucdavis.edu*
¶*ogstaadt@ucdavis.edu*

Received 5 October 2007
Revised 25 January 2008
Accepted 15 February 2008

We introduce a system that adds a foveal inset to large-scale projection displays. The effective resolution of the foveal inset projection is higher than the original display resolution, allowing the user to see more details and finer features in large data sets. The foveal inset is generated by projecting a high-resolution image onto a mirror mounted on a panCtilt unit that is controlled by the user with a laser pointer. Our implementation is based on Chromium and supports many OpenGL applications without modifications.We present experimental results using high-resolution image data from medical imaging and aerial photography.

Keywords: Tiled display; multi-resolution display; foveal display; interaction; calibration.

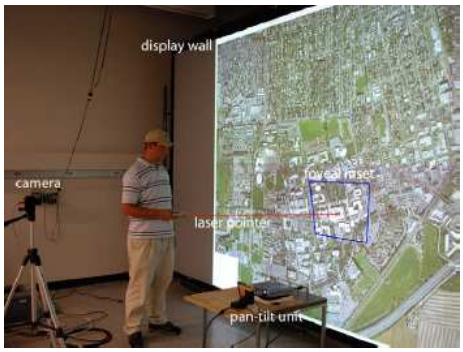
1. Introduction

Large display environments have become increasingly important over the past decade and are used frequently for displaying high-resolution data resulting from imaging applications and simulations. The size and complexity of such data sets increases steadily, and the resolution of single-projector displays is no longer sufficient to reveal details without zooming in and, thus, losing important context information. One possible solution to this problem is the use of tiled displays that use multiple projectors to increase the total resolution of the system. Even though high-quality projectors are now available at reasonable cost, increasing the number of tiles by adding more rows and columns increases the cost of the system significantly. For example, adding one row and one column to a 4×3-tile display increases the number of projectors (and rendering nodes) from 12 to 20.

It can be argued that system resolution should be increased homogeneously across the display area. For example, it is often not necessary to increase the resolution in the periphery of the display by the same amount as in the center of the display. One way of exploiting this is to have a higher-resolution region in the center of the lower-resolution display. However, a fixed-location inset constrains user interaction. The human visual system overcomes the problem of a static foveal region in the retina with saccades, rapid movement of the eye between fixation points.

Display environments where a high-resolution projection is overlaid on a lower resolution display to provide higher detail in a particular area are called *foveal displays*.[3] The area of higher resolution is the *foveal inset*. We have developed a positional foveal inset mechanism for a tiled display. This is a novel method for interacting with large displays. A high-resolution projector and a mirror mounted on a panCtilt unit (PTU) are used to move the foveal inset on a tiled display. This provides a method for examination of areas of interest in very high detail without the expense of adding more tiles to the display. It also allows the system to keep pace with advancements in projector technology. Instead of upgrading a large number of projectors, only a single foveal inset projector needs to be replaced.

When the foveal inset is projected onto the display screen, it appears skewed due to the oblique projection used to project onto the display plane. For the foveal inset image to appear aligned with the rest of the display, a homography matrix must be computed to map the foveal inset image plane to the display image plane. The foveal inset projection must be pre-warped using the homography matrix. Additionally, the area in the main display where the foveal inset lies must be removed to avoid image blurring caused by the overlapping tiled display and foveal inset, see Fig. 1.



(a) The user directs the projection of the foveal inset using a laser pointer. The size of the projected inset is significantly smaller than the tiles of the rear-projected display wall, thus providing a higher resolution.

(b) Magnified view of a high-resolution slice of a cryosection of a monkey brain. The boundary between the high-resolution foveal inset (bottom) and the lower-resolution display wall (top) is clearly visible.

Fig. 1. Foveal inset. Note that the pixel dimensions of the foveal inset projector and the display wall projector are identical. (Aerial photographs courtesy of the City of Davis, CA. Monkey brain data set courtesy of E.G. Jones, UCD Center for Neuroscience.)

2. Previous Work

Pixelflex is a reconfigurable tiled display system developed at the University of North Carolina, Chapel Hill.[7,18,24] This system uses a set of projectors with computer controlled pan, tilt, zoom, and focus settings; and a camera to provide a reconfigurable system that can be set to new configurations in a matter of minutes. The system uses the camera to generate homographies and blending functions to provide a mural display in a variety of configurations. These configurations can be saved and reloaded, but the system does not provide run-time interactive support.

The Escritoire is a multi-projector display presented by Ashdown and Robinson.[3] This system uses two digital projectors and two mirrors to create a virtual desktop environment. One projector covers a large area of the desk, while the other provides a high-resolution area for viewing items in detail. Items can be moved in and out of the high-resolution area using pens held in each hand. While this system is a foveal display, it does not provide run-time reconfiguration of the foveal inset location.

Raskar *et al.* developed a geometrically aware, self-calibrating projector called an iLamp.[19] These are small portable projectors with a camera, and a network interface. These devices can be used to augment reality, or multiple iLamps can be arranged to create an ad-hoc tiled display. Another portable reconfigurable tiled display system is described by Brown and Seales.[4] This system was intended to be portable and easily reconfigurable for use in multiple locations. The system is transparent to OpenGL applications because it uses the WireGL[9] software layer for distributed rendering.

Pinhanez presents a projection system that uses a pan-tilt mounted mirror to allow a number of different projection surfaces to be used by a user.[15,16] A fixed number of display surfaces are predetermined and calibrated prior to program execution. These surfaces are used to present desktop-like projections and to augment physical objects. Gesture recognition techniques for interacting with these displays were presented by Kjeldsen *et al.*[8] Pingali *et al.* described a system for automatically selecting a display surface based on where the user is located, providing a user following display.[14] Kjeldsen *et al.* also described a system for dynamic, reconfigurable interfaces that reposition and change interface widgets on the fly.[11] This work was mainly geared toward using different surfaces to display application data and augment physical objects. Our work differs in that we wish to augment and integrate into applications running on tiled displays for higher resolution.

Sanneblad and Holmquist presented a new type of display interaction called ubiquitous graphics.[20] Their system augments tablet PCs, hand-held devices, and other portable devices capable of high-end graphics with position trackers. The system described supports multiple users using a variety of different display devices. These devices are used as peephole model which allows the user to hold them up to the screen and view the corresponding area in more detail.

The use of a pan-tilt unit and high-resolution projector to create a positional foveal inset on a tiled display has been previously investigated.[23] The previous

work presents a mathematical model for representing the projection onto a mirror and reflection onto the tiled display. It was demonstrated that this transformation is represented as a 2D homography between the tiled display image plane and the projector image plane. In this implementation it was attempted to calculate homographies on the fly, using the pan and tilt angles as input to forward compute the homography. This has the advantage that the foveal inset can be displayed in any position within the range of the PTU. It does, however, require calculation of the intrinsic projector parameters and precise mounting of the mirror about the center of rotation of the pan and tilt axes. It was found that this method was numerically sensitive to the precision of the intrinsic projector parameters. Part of our work is based on Ref. 23. We attempt to calculate the homographies in advance for a set number of positions, however, rather than forward compute them. This is an expanded and revised description of our previous work in Refs. 1 and 21.

3. Background

3.1. *Pinhole camera model*

A pinhole camera is a device that allows light through a single point in a plane, in order to produce an image on a parallel viewing plane. The pinhole camera model is a geometric abstraction of this simple device. In its simplest form, the pinhole camera model is based on an ideal perspective projection through a focal point onto the image plane. The focal length, f, is the distance from the focal point to the closest point on the image plane.

Let (x, y, z) represent a point in 3D space and (u, v) represent a point on the image plane. The relation between these points can be described by the homogeneous projection matrix

$$\begin{bmatrix} us \\ vs \\ s \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}.$$

This simple model can be extended to incorporate pixel skew, α, pixel size, $p_x \times p_y$, and center of projection, (cx,cy). Let f_x and f_y be the ratios f/p_x and f/p_y and let $g = \tan\alpha \bullet f_y$. The resulting projection matrix is

$$\begin{bmatrix} us \\ vs \\ s \end{bmatrix} = \begin{bmatrix} f_x & \gamma & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}.$$

This model has the advantage that all transformations between pixel and object coordinates can be calculated using a single matrix multiplication. Additionally, it is a sufficiently accurate model of digital camera devices. It does not, however, take into account camera lens distortion.

3.2. *Camera lens distortion*

The pinhole camera model assumes a linear relationship in the transformation from realworld object coordinates to camera coordinates. This assumptions does not hold for real cameras, which use lenses that introduce nonlinear distortion. OpenCV[5] describes a series of equations for modeling lens distortion:

$$\begin{aligned} x_u &= x_d + x_d(k_1 r^2 + k_2 r^4) \\ &= +(2p_1 x_d y_d + p_2(r^2 + 2x_d^2) \\ y_u &= y_d + y_d(k_1 r^2 + k_2 r^4) \\ &= +(2p_2 x_d y_d + p_1(r^2 + 2y_d^2) \\ r^2 &= x_d^2 + y_d^2 \end{aligned} \quad .$$

Here (x_u, y_u) is the point in the ideal pinhole camera model, and (x_d, y_d) is the observed point that exhibits camera lens distortion. This model uses four coefficients to represent lens distortion. Coefficients k_1 and k_2 represent tangential lens distortion, and r_1 and r_2 represent radial distortion. If these coefficients are known, the above equation can be employed to find x_u and y_u.

3.3. *Chromium*

Chromium is a system for interactively rendering OpenGL graphics applications on clusters of workstations.[9,10] Chromium works by intercepting OpenGL library calls and replacing them with Chromium-implemented library calls. This allows Chromium to distribute OpenGL function calls across clusters, or just to modify the arguments/functionality of a specific OpenGL function. The basic building block of Chromium is a Stream Processing Unit (SPU). An SPU resides on a single cluster node. Chromium creates a rendering pipeline by connecting SPUs either on the same clusters node or on different cluster nodes. When multiple SPUs reside on the same machine, an SPUs child is the next SPU in the pipeline that resides on that machine. If the next SPU in the pipeline resides on another machine, it is a server of the previous SPU. A single SPU may have several servers. One application of this is rendering on tiled displays. An SPU passes a call along in the pipeline to its child by using a pointer to the child SPUs function dispatch table. An SPU transmits functions to a server by sending packed data over a network connection. Many different SPUs are provided with Chromium, for performing tasks like tiled rendering, motion blur effects, and image warping. SPUs can be combined in a variety of ways over any number of networked computers to perform different tasks.

In addition to using the SPUs that are provided with Chromium, custom SPUs can be developed and used. Chromium contains a mechanism for SPU inheritance, that can be used when creating a custom SPU. An SPU that is inherited from another SPU is the inheriting SPUs super SPU. This allows the new SPU to only implement part of the OpenGL API, and use the super SPUs implementation for

the remainder. The super SPU functions can be called directly using a pointer to the super SPU dispatch table, similar to the way a child SPU is used.

4. System Configuration

4.1. *Hardware setup*

Our tiled display wall consists of six tiles arranged in a 3×2 grid. Each tile is $6' \times 4\frac{1}{2}'$ for a total size of $18' \times 9'$. A tile is displayed using two Sanyo PLC-XT16 projectors to support stereographic imaging.[a] The projectors are run by a cluster of Linux machines with 2 GHz AMD Opteron processors and 1 MB of memory. The head node of the cluster is a Linux machine with dual 2 GHz AMD Opteron processors and 8 GB of memory. We are using Point Grey Flea[17] cameras for calibration and interaction. These cameras are capable of capturing 1024×768 pixel color images at 30 fps.We are using a Directed Perception Pan-Tilt Unit[6] PTU-C46 to control the inset position. The PTU has position resolution of 184 arc-seconds. Our unit is configured to move at 1000 positions per second.We have mounted a mirror to the PTU using a gimbal adapter. This setup is shown in Fig. 2. The PTU is connected to the head node in the cluster as illustrated in Fig. 3. The projector used to project

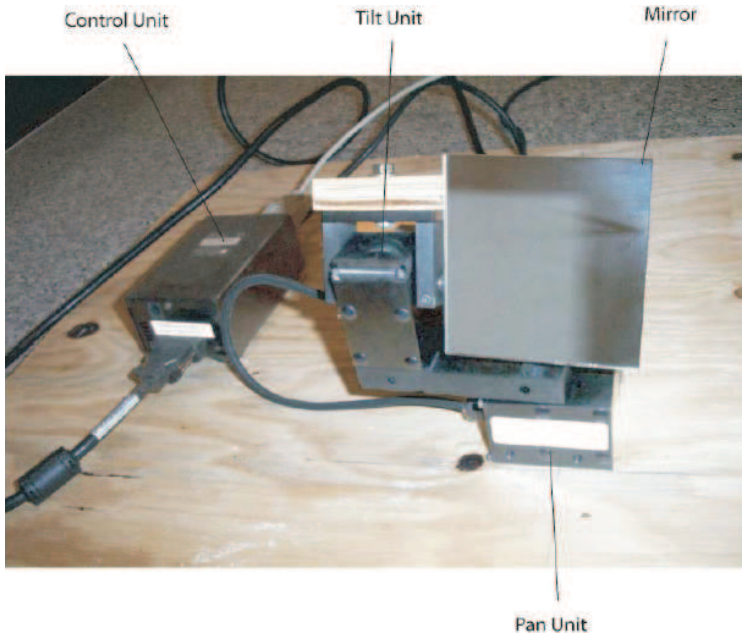


Fig. 2. The panCtilt unit (PTU) with mounted mirror and control unit.

[a]Our foveal inset system currently does not use the stereographic capabilities or the tiled display.

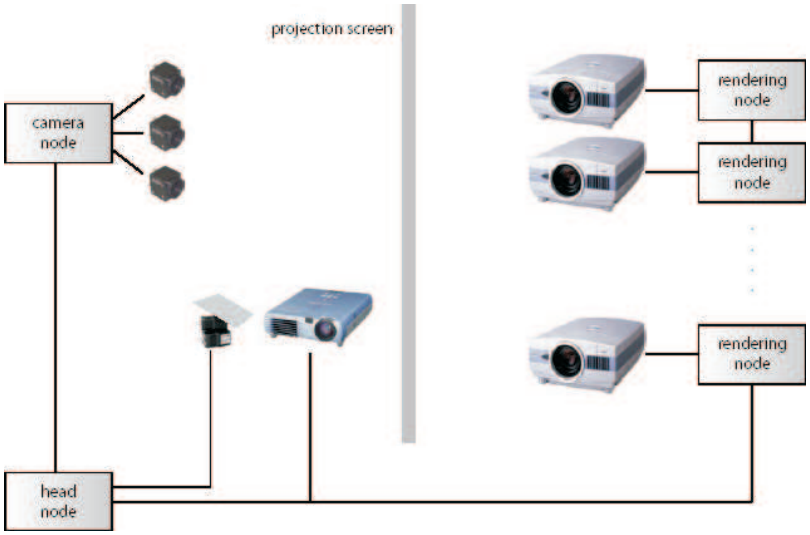


Fig. 3. System configuration.

Table 1. Technical specification for three projectors.

Manufacturer	Sanyo	Mitsubishi	Christie
Model	PLC-XT16	XD50U	LU77
Horizontal Resolution	1024	1024	1600
Vertical Resolution	768	768	1200
Lens	1:1	1.2:1	7.0:1
Min. Throw Dist. (ft)	12.4	6	18.3
Min. Image Width (in)	72	32	32
Min. Image Height (in)	54	24	24
Eff. Resolution (dpi)	14	32	50

Note: We use six Sanyo PLC-XT16 projectors for the rear-projected display wall and one Mitsubishi XD50U for foveal inset projection. The Christie LU77 could be used as an alternative. It provides higher resolution and increased throw distance at the same size of the projected inset (*Note that the values listed for the Sanyo PLC-XT16 reflect our display wall configuration and not the minimal values).

the foveal inset is a Mitsubishi XD50U.[13] Information regrading the configuration and specifications of the projectors used in our system are summarized in Table 1.

4.2. *Software design*

The position of the foveal inset is specified using a hand-held laser pointer. The control of the foveal inset position is implemented to interface with a laser pointer interaction system.[2] The foveal inset controller receives information regarding the laser pointer position from the tracking application via a network socket. The foveal

inset is then positioned about this location on the display by adjusting the pan and tilt angles of the PTU. This allows the user to run-time specify the position of the foveal inset on the display, making it possible for areas of interest to be displayed in higher resolution than the rest of the display.

5. Calibration

Our system performs a coordinate mapping from the foveal inset image plane to the display image plane. Mapping a 2D point in homogeneous coordinates on a plane to another plane can be achieved using a 3×3 homogeneous matrix.[23] The mirror which reflects the foveal inset is in a number of different positions as the PTU moves, effectively changing the image plane of the inset. For this reason, a different homography is required for each PTU position. Due to the PTUs high resolution, pre-computing these homographies for each possible pan-tilt angle pair is impractical. Instead, our system calibrates for a configured subset of the possible positions. This allows the range of foveal inset positions to be configured in such a way that all desired areas of the display are covered and minimizes the amount of calibration time and system memory needed to use the system.

The calibration of the foveal inset is based on the method by Sukthankar *et al.*[22] They presented a method for calculating 2D homographies using a set of point correspondences in two planes and used this homography to pre-warp a projected image in a presentation environment. They recognized that a point (X, Y) in one plane is related to a point (x, y) in another plane by the equation

$$(x, y) = \left(\frac{p_1 X + p_2 Y + p_3}{p_7 X + p_8 Y + p_9}, \frac{p_4 X + p_5 Y + p_6}{p_7 X + p_8 Y + p_9} \right).$$

This can also be expressed in matrix form as

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{pmatrix} p_1 & p_2 & p_2 \\ p_4 & p_5 & p_6 \\ p_7 & p_8 & p_9 \end{pmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}. \tag{1}$$

Let $p = (p_1 \cdots p_9)^T$, and P the 3×3 homography matrix with the same elements as p. Sukthankar *et al.*[22] defined the following $2n \times 9$ matrix:

$$A = \begin{pmatrix} X_1 & Y_1 & 1 & 0 & 0 & 0 & -X_1 x_1 & -Y1 y_1 & -x_1 \\ 0 & 0 & 0 & X_1 & Y_1 & 1 & -X_1 x_1 & -Y1 y_1 & -y_1 \\ X_2 & Y_2 & 2 & 0 & 0 & 0 & -X_2 x_2 & -Y_2 y_2 & -x_2 \\ 0 & 0 & 0 & X_2 & Y_2 & 2 & -X_2 x_2 & -Y_2 y_2 & -y_2 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ X_n & Y_n & n & 0 & 0 & 0 & -X_n x_n & -Y_n y_n & -x_n \\ 0 & 0 & 0 & X_n & Y_n & n & -X_n x_n & -Y_n y_n & -y_n \end{pmatrix}.$$

It follows from Eq. (1) that given n point correspondences, (X_i, Y_i), (x_i, y_i), the homography matrix P can be calculated by minimizing the product $|A_p|$. The optimal p is the eigenvector corresponding to the smallest eigenvalue of $A^T A$. This vector can easily be found using the singular value decomposition of the matrix $A^T A$.

Our calibration procedure makes use of a set of calibrated cameras. This process determines the lens distortion coefficients generates homographies between the display image plane and the camera image plane. The first step in the calibration is to display a series of lines on the display. For every line, each camera captures images of the display. All images are processed to extract the pixels that compose the line. Distortion from the camera lens will make the line in an image appear curved. If the distortion parameters are known, the pixels can be corrected and the line will appear straight. We use this fact to determine the lens parameters for each camera using a non-linear optimization. The optimization is based on how accurate a linear fit can be applied to a set of lines after being corrected by a set of lens parameters, see Fig. 4.

Once an optimal set of parameters is found the lines are straightened. The intersections of the lines in the display image plane and in each camera image plane are used to generate a set of point correspondences between the display and each camera. This set of point correspondences is used to generate a homography matrix between the display image plane and each camera image plane. The result of this step is a set of cameras facing the display, which are capable of mapping points in their image space to points in the displays image plane.

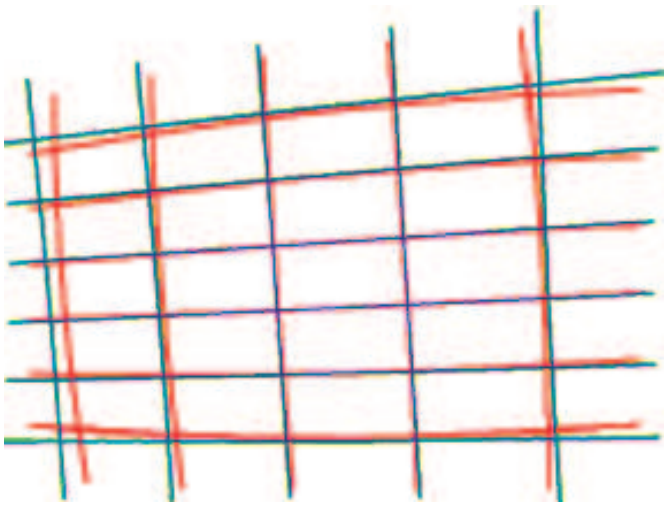


Fig. 4. Original point correspondences at the intersection of lines captured by a camera during calibration are shown in dark gray. Lines in green result from undistorting the camera image. Fitted lines for corrected points are depicted in light gray.

The foveal inset calibration is implemented in a similar manner. For each panCtilt position a series of lines are displayed within the inset. Each camera captures images of the inset while a line is being displayed. As in the case for the camera calibration, the lines are detected and corrected. The corrected points are then projected into the display image plane, using the calibrated camera homographies. The lines from all cameras are combined into a single set of lines in the display image plane. The intersection of these lines and the intersection of the lines in the inset image plane form a set of point correspondences between the two planes. This set of point correspondences is used to calculate homographies which map between the inset and the display. This process is done for all desired PTU positions. The generated homographies and their corresponding pan-tilt positions are written to a calibration file, which is loaded during the start-up phase by the foveal inset controller.

6. Interaction

We use a laser pointer to interact with our system.[2] The center points of each calibrated foveal inset position are used to determine which of the positions is most appropriate for the selected location. When the client is started and has read the calibration data, the center of each foveal inset is projected into the display image plane and inserted into a nearestneighbor search structure When the inset controller needs to position the foveal inset, it uses this search structure to find the closest center point to the laser pointer position. The PTU position is then set to the corresponding pan-tilt angles and the corresponding homography is used to pre-warp the foveal inset image.

The controller receives messages indicating both the laser pointer position and when the laser pointer is no longer visible. When a positional message is received, the coordinates are internally recorded. When the laser pointer is no longer visible, i.e., has been turned off, the client moves the PTU and uses the pre-warp matrix to the vertex coordinates.

7. Integration

The modification of the OpenGL pipeline needed to render the inset in a rectified manner requires multiplying the projection matrix by the pre-warp homography. This can be seen by examining how the projection matrix alters vertex coordinates that are being passed down the rendering pipeline. If C is the current projection matrix and v is the vertex being rendered (after applying the modelview matrix), the standard OpenGL pipeline would transform this point to v' as

$$v' = Cv.$$

We multiply the pre-warp homography, H, to the projection matrix, resulting in

$$v' = HCv.$$

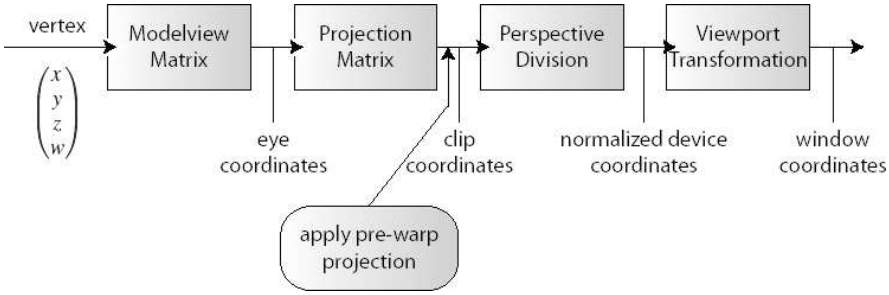


Fig. 5. Modification of OpenGL pipeline to render pre-warped inset.

This multiplication has the effect of transforming the vertex coordinates by the pre-warp homography after the vertex has been projected into the image plane. This modification to the OpenGL pipeline is depicted in Fig. 5. Our current implementation only handles OpenGL applications which project orthographically to the $[-1, 1] \times [1, 1]$ square.

This is because our pre-warp matrix is generated from foveal inset NDC to display NDC. An additional matrix transformation, however, could be inserted between the projection matrix and homography to allow the vertex coordinates to be scaled into the $[-1, 1]\times[-1, 1]$ square prior to applying the pre-warp homography.

In addition to handling pre-warped rendering of the foveal inset, our system must also disable rendering of the region that the inset occupies in the tiled display. This is accomplished by rendering a black quadrilateral over the foveal inset region. To determine where the quadrilateral lies, the corners of the foveal inset region are transformed into the display image plane. Since the calibration is done from NDC to NDC, these points are always the corners of the $[-1, 1] \times [-1, 1]$ square. Once the quadrilateral coordinates are known, the quadrilateral is rendered over the inset region.

In order to modify the OpenGL pipeline without modifying application code, we have implemented the inset controller as a Chromium SPU.[10] This SPU is a combination of the Chromium RenderSPU and PassthroughSPU. Our SPU inherits from the RenderSPU, and also implements PassthroughSPU functionality. This allows it to render the inset locally and pass rendering information down the pipeline. The SPU is shown in Fig. 6.

Our SPU intercepts all OpenGL commands. Most are passed on to its super SPU and its child SPU without modification. Some are used to perform internal bookkeeping for the inset rendering window/context, i.e., *glCreateContext, glXMakeCurrent*, etc. In our current implementation, the pre-warp matrix is multiplied onto the OpenGL projection matrix when our SPU intercepts a *glBegin* call, and removed when it intercepts a *glEnd* call. This approach ensures that all primitives drawn using vertex primitives are transformed properly, as OpenGL forbids any changes to its matrices inside a *glBegin/glEnd* pair. This approach was initially chosen due

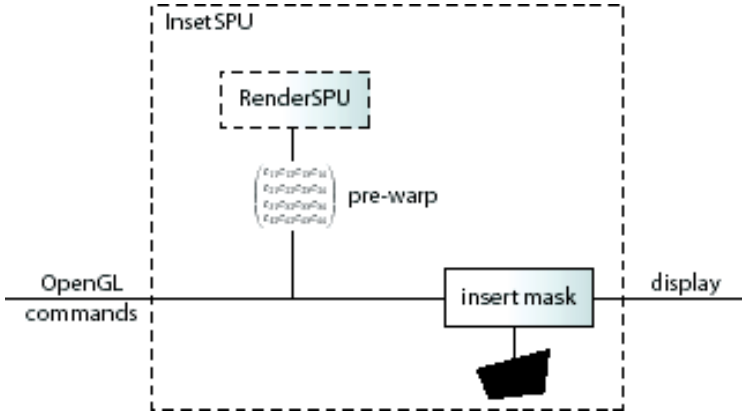


Fig. 6. InsetSPU used for rendering foveal inset with Chromium.

to its simplicity, but it is highly inefficient. Multiplying an arbitrary matrix onto an OpenGL matrix using *glMultMatrix* is fairly costly, and it is currently done once for each glBegin/glEnd pair. In the near future, we will change our approach to intercepting all OpenGL matrix commands, to insert our prewarp matrix only when an application changes the OpenGL projection matrix. Similarly, in our current implementation the region of the display overlayed by the inset is blanked by rendering a black quadrilateral whenever a *glSwapBuffers* call is intercepted by our SPU. This is fairly efficient, as the quadrilateral is only rendered once per frame, but can lead to unpredictable error behavior since the OpenGL state machine has to be moved to a well-known state to ensure that the quadrilateral is rendered properly in black. Texture mapping, fog, etc. all have to be disabled. Since it is impossible to anticipate all future state elements that could interfere with quadrilateral rendering, we will soon change our implementation to use a safer means to blank out the inset region, for example using the stencil buffer.

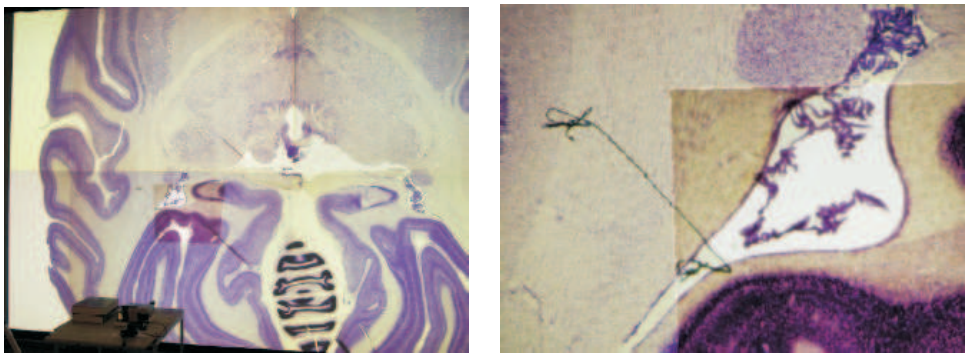
8. Experimental Results

To evaluate the impact of higher-resolution foveal insets on applications visualizing highresolution data, we used a prototype image viewer application. This application, shown in Fig. 8, allows a user to interactively pan and zoom very large image files using out-of-core rendering methods. The application uses a quadtree-based multiresolution representation which is created in a pre-processing step, and uses OpenGL to render an image as a set of texture-mapped square tiles at several different levels of resolution. The first example image shown in the figures is a stained slice from a cryosection of a monkey brain. The image has a resolution of 5000×5800 pixels, and the multiresolution representation occupies 113 MB on disk. The second example image is generated from aerial photography, with a resolution of about 22000×16500 pixels and an on-disk size of 1.38 GB. The third example is

an image of a metric scale. It has a resolution of 8800 × 6400 pixels and a file size of 216 MB.

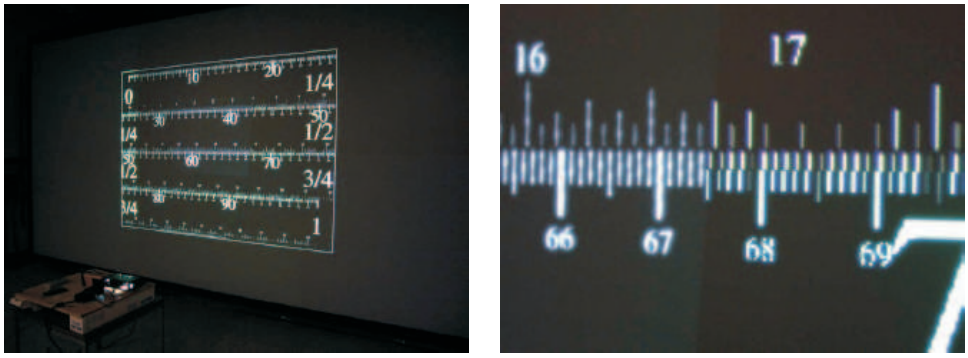
Since the foveal inset provides a locally increased resolution, and the appropriate levelof- detail for image rendering is chosen based on pixel size, the application automatically renders the image at a higher resolution in the area covered by the inset (see Fig. 7).

The metric-scale image in Fig. 8 illustrates how aliasing artifacts of the lower-resolution rear-projected image affect the display. High-resolution tick marks are visible in the foveal inset on the left, but not in the rear projection on the right, see Fig. 8(b). The use of the foveal display is shown in Fig. 1(b) and Fig. 7. The close-up views clearly show the increased detail in the foveal inset and the accuracy of our system calibration. Figure 9 shows that the front-projected foveal inset image seamlessly matches the rear-projected display image. The projection of the inset is shown in Fig. 9(a) and the masked rear-projection in Fig. 9(b). Figure 9(c) depicts the final image as a combination of the two. Note that photometric seamlessness could be improved using a method such as presented by Majumder and Stevens.[12] When the user directs the foveal inset toward the periphery of the display wall, the inset becomes increasingly distorted. The degree of distortion depends on different factors, including projector lens and size of the display wall versus throw distance of the foveal inset projector. The projector used in our experiments has a relatively short throw distance and must be placed close to the display wall to obtain a small inset (see also Table 1). This effect is visible in Fig. 10. However, although the inset is distorted significantly, the close-up view in Fig. 10(b) confirms the accuracy of our calibration. Our systems also supports situations where the inset is only projected partially onto the display screen (see Fig. 11). In our configuration, the effective resolution of the inset projected onto a corner of the display drops below



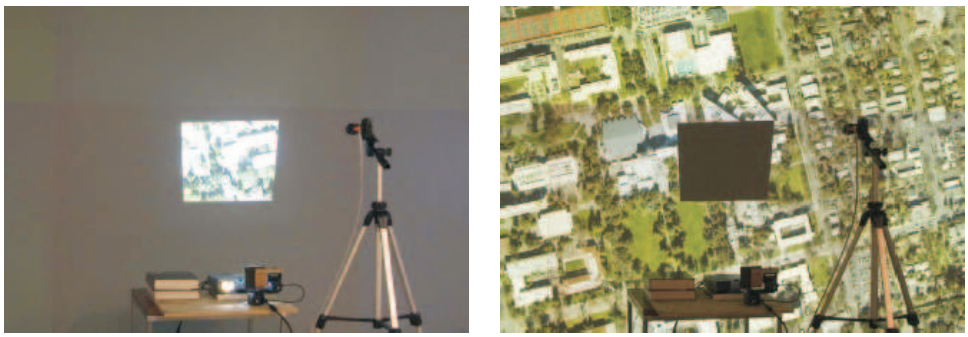
(a) (b)

Fig. 7. (a) Visualization of high-resolution slice of cryosection of monkey brain. (b) Magnified view of the boundary between the inset and the display wall. The foveal inset provides the user with a higher-resolution view of the selected area.

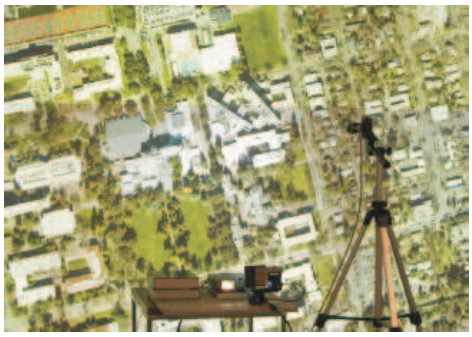


(a) View of the metric-scale image. (b) Close up of the scale.

Fig. 8. The inset covers the left side of the image. The right side (display wall projection) exhibits severe aliasing artifacts caused by its limited resolution. Some of the tick marks that are visible in the inset vanish completely on the right side. Also, due to the lower resolution of the display wall, the small digits are difficult to read.



(a) (b)

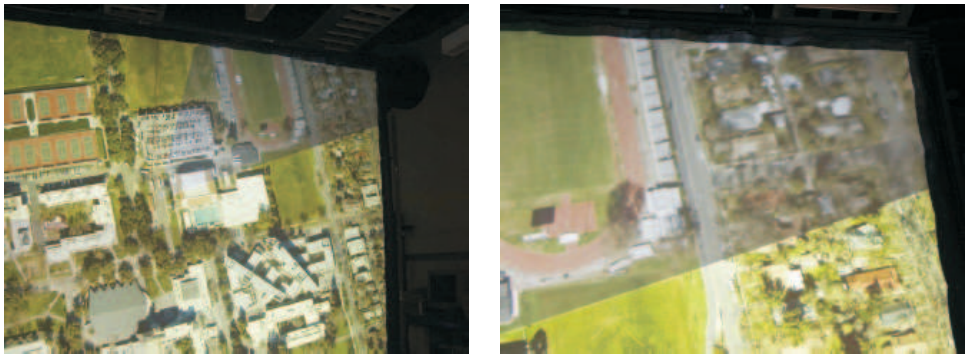


(c)

Fig. 9. (a) Projection of foveal inset only. (b) Projection of main application onto display wall with foveal inset area masked. (c) Final view combining rear projection and foveal-inset projection.

(a) (b)

Fig. 10. (a) If the throw distance of the foveal inset projector is too short, insets in the periphery of the display wall are distorted significantly. (b) Close-up view demonstrates that our calibration ensures a high-quality projection of the inset.



(a) (b)

Fig. 11. (a) The inset is moved to the top-right corner of the display wall and is partially cut off. (b) Even though the calibration produces satisfactory results, the close-up view reveals that the effective resolution of the foveal inset projection has dropped below the resolution of the display wall. A projector with longer throw distance can be used to solve this problem.

the natural resolution of the display wall as shown in Fig. 11(b). An apparent solution to this problem is to use projectorClens combination with a longer throw distance for the foveal inset (e.g., a Christie LU77 listed in Table 1), which would result in a less distorted projection of the inset. Although we use high-resolution 2D images in our examples, our systems supports OpenGL applications (within the limitations of Chromium). 3D applications that can benefit from the use of a foveal inset include, for example, high-resolution volume visualization and flow visualization, which typically exhibit very fine details.

9. Discussion

Many important design issues have arisen in the development of this system. These are primarily concerned with equipment selection and placement. The calibration of the system relies on being able to accurately generate homography matrices between pixels in different devices. A variety of issues with the cameras, projectors, and environment can cause variations in the ability to do this.

9.1. *Camera setup*

The camera setup is one major area where small changes can impose drastic differences in calibration accuracy. The ability of the camera to accurately capture the calibration lines on the display is the foundation of the calibration procedure. It is our experience that this is best done by decreasing the gain and increasing the shutter speed as mush as possible. This allows the lines to appear clearly with a minimal amount of noise.

Another important issue when setting up the system is the relationship between the cameras, the tiled display, and the foveal inset. A camera that is placed closer to the display will need a wider field if view (i.e., shorter focal length) to capture the same area as a camera placed further away. The shorter the focal length of the lens, the greater is spherical lens distortion. While our system rectifies the captured frames, less distortion will lead to a more accurate calibration. On the other hand, the camera must be positioned in a way that allows it to clearly see the calibration lines displayed in the foveal inset. If placed too far away, or focusing on too large an area, the images may not be captured in enough detail for an accurate calibration. In an ideal situation, one camera pixel would correspond to one pixel in the inset projector.

9.2. *Projector setup*

The projectorClens combination used for displaying the foveal inset is another important aspect of our design. This choice will directly affect its positioning relative to the screen, and the system range. A projector that has a short throw distance must be placed very close to the screen in order for the inset to remain a reasonable size. The projector we used in our experiments will project an 32 in $\times$ 24 in image at a throw distance of 60. Placing the projector this close to the screen reduces the range of pan-tilt positions over which the system is effective. As the mirror reflects the foveal inset at larger angles, the throw distance for the projected image increases. The closer the projector is placed to the screen, the more this change in throw distance is magnified. As the throw distance for the projector changes, the further out of focus the projected image will become. At extreme angles, it is often impossible to keep a single image in focus. This is a result of the throw distance changing drastically over the inset image. This decline in image sharpness decreases calibration accuracy.

There is also a problem with the effective resolution of a heavily skewed inset. Insets which are projected at large oblique angles onto the display will appear larger than those projected more directly onto the display. The greater the amount of skew, the more stretched the projected image becomes. This stretching reduces the pixel density on the screen. When an inset is stretched too much, its effective resolution is no longer greater than that of the display wall itself.

An example of a calibration for a heavily skewed inset position can be seen in Figs. 10 and 11. In Fig. 10, it can be seen that our calibration has produced good results, however, not quite as accurate as with a less skewed tile. This is particularly evident at the left side of the inset, where a small gap of a couple pixels can be see can. This is caused by the focus and intensity variations caused by the oblique projection of the inset. Figure 11 is an inset position in which the entire inset is not on the display wall. This inset exhibits focus and intensity issues worse than those in Fig. 10. The calibration for this inset produces acceptable results, however, the effective resolution of the inset tile has become less than that of the display wall, due to extreme image distortion.

An ideal projector for this system would be one that has a long throw distance, capable of projecting a small image from a large distance. Being able to place the projector further from the screen allows the entire inset to be kept in focus, in a larger range of positions. It also allows for more consistent image intensity for the foveal inset. These changes caused by the decreased variance in throw distance are far smaller than when the projector is placed very close to the screen. This type of projector would allow for more accurate calibration over a larger range of pan-tilt positions. The technical specifications for a Christie LU77 projector are provided in Table 1 as an example of a projector that would be well suited for this purpose.

9.3. *Display screen material*

We use a soft screen in our system with a low gain of 1.0. Our calibration procedure assumes that the screen has a perfectly flat surface. However, since the screen is not made from a rigid material, the screen exhibits a minor sag, which causes areas of the display surface to be non-planar. We have not found this to be an issue with the calibration of our prototype, however, a display which exhibits this effect on a larger scale may have problems with calibration accuracy.

Another critical issue is related to small movement of the screen during calibration caused, for example, by air flow. In our experience, movement of fractions of an inch in either direction can translate to a shift of three to five pixels in the inset. To minimize this effect, we ensure that air condition vents are not directed at the screen and that nobody is in close proximity to the screen during calibration.

9.4. *Chromium integration*

The performance of our current software implementation could further be improved. Instead of intercepting *glBegin* and *glEnd* function calls in Chromium to multiply

the warp matrix outside of *glBegin/glEnd* pairs, we could intercept all OpenGL matrix function calls. We could then insert the warp matrix whenever the OpenGL projection matrix changes, which is very rare for most programs. This would reduce significantly the number of *glMultMatrix* and improve rendering efficiency.

10. Future Work

We plan to develop an extension of this work which supports stereographic applications to take advantage of the stereo projection capabilities of our tiled display wall. We plan to support stereo calibration and integration into the toolkit which currently provides support for stereographic rendering on the tiled display. In addition to providing stereographic viewing, this will allow one to control the foveal inset using techniques commonly used in virtual reality applications.

Furthermore, we plan to further improve the accuracy of the system calibration and the overall performance. Since the PTU supports high-velocity movement, we would like to investigate the possibility of controlling the position of the foveal inset based on the information obtained from a head tracker.

Acknowledgments

This work was supported by the National Science Foundation under contracts ACI 9624034 (CAREER Award) and ACI 0222909, through the LSSDSV program under contract ACI 9982251, and a large ITR grant. We gratefully acknowledge the support of the W. M. Keck Foundation provided to the UC Davis Center for Active Visualization in the Earth Sciences (KeckCAVES). We thank ATI for providing the graphic boards.We would further like to thank Justin Walker for developing an early prototype of the system and Sohail Shafee for helping with the experiments. We thank the members of the Visualization and Computer Graphics Research Group at IDAV at the University of California, Davis.

References

1. B. A. Ahlborn, *Augmenting Large Displays for Enhanced Interaction*, Master's thesis, University of California, Davis, 2005.
2. B. A. Ahlborn, D. Thompson, O. Kreylos, B. Hamann and O. G. Staadt, "A practical system for laser pointer interaction on tiled displays," *Proceedings of ACM Virtual Reality Software and Technology 2005* (ACM Press, ACM, 2005), pp. 106–109.
3. M. Ashdown and P. Robinson, "Escritoire: A personal projected display," *IEEE MultiMedia*, **12**(1), 34–42 (2005).
4. M. S. Brown and W. B. Seales, "A practical and flexible tiled display system," *PG '02: Proceedings of the 10th Pacific Conference on Computer Graphics and Applications*, IEEE Computer Society (Washington, DC, USA, 2002), p. 194.
5. I. Corporation (2000), Open Source Computer Vision Library Reference Manual, http://sourceforge.net/projects/opencvlibrary/.
6. Directed Perception, Inc. (December 2005), http://www.dperception.com.
7. D. Gotz, *The Design and Implementation of Pixelflex: A Reconfigurable Multi-Projector Display System*, **18** (2001).

8. J. Hartman and T. Levas, "Interacting with steerable projected displays," *FGR '02: Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition*, IEEE Computer Society (Washington, DC, USA, 2002), p. 402.
9. G. Humphreys, M. Eldridge, I. Buck, G. Stoll, M. Everett and P. Hanrahan, "Wiregl: A scalable graphics system for clusters," *SIGGRAPH '01: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (ACM Press, New York, USA, 2001), pp. 129–140.
10. G. Humphreys, M. Houston, R. Ng, R. Frank, S. Ahern, P. D. Kirchner and J. T. Klosowski, "Chromium: A stream-processing framework for interactive rendering on clusters," *SIGGRAPH '02: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques* (ACM Press, New York, USA, 2002), pp. 693–702.
11. R. Kjeldsen, A. Levas, and C. Pinhanez,"Dynamically reconfigurable vision-based user interfaces," *ICVS '03: Proceedings of the 3rd International Conference on Vision Systems* (Graz, Austria, 2003).
12. A. Majumder and R. Stevens, "Perceptual photometric seamlessness in projection-based tiled displays," *ACM Trans. Graph.* **24**(1), 118–139 (2005).
13. Mitsubishi Electric Corp. (December 2005), http://global.mitsubishielectric.com.
14. G. Pingali, C. Pinhanez, T. Levas, R. Kjeldsen and M. Podlaseck, "User-following displays," *ICME '02: Proceedings of the IEEE International Conference on Multimedia and Expo*, IEEE Computer Society (Lausanne, Switzerland, 2002).
15. C. Pinhanez, "Using a steerable projector and a camera to transform surfaces into interactive displays," *CHI '01: CHI '01 Extended Abstracts on Human Factors in Computing Systems* (ACM Press, New York, USA, 2001), pp. 369–370.
16. C. S. Pinhanez, "The everywhere displays projector: A device to create ubiquitous graphical interfaces," *UbiComp '01: Proceedings of the 3rd International Conference on Ubiquitous Computing* (Springer-Verlag, London, UK, 2001), pp. 315–331.
17. Point Grey Research (December 2005), Inc. http://www.ptgrey.com.
18. A. Raij, G. Gill, A. Majumder, H. Towles and H. Fuchs, Pixelflex2: A Comprehensive, Automatic, Casually-Aligned Multi-Projector Display.
19. R. Raskar, J. van Baar, P. Beardsley, T. Willwacher, S. Rao and C. Forlines,"ilamps: geometrically aware and self-configuring projectors," *ACM Trans. Graph* (2003).
20. J. Sanneblad and L. E. Holmquist, "Ubiquitous graphics. Emerging technologies," *ACM SIGGRAPH* (2005).
21. O. G. Staadt, B. A. Ahlborn, O. Kreylos and B. Hamann,"A foveal inset for large display environments," *VRCIA '06: Proceedings of the 2006 ACM International Conference on Virtual Reality Continuum and Its Applications* (ACM Press, New York, USA, 2006), pp. 281–288.
22. R. Sukthankar, R. Stockton and M. Mullin, "Smarter presentations: Exploiting homography in camera-projector systems," *Proceedings of the International Conference on Computer Vision* (2001).
23. J. Walker, *Pan Tilt Unit and Tiled Displays*, Master's thesis, University of California, Davis, 2004.
24. R. Yang, D. Gotz, J. Hensley, H. Towles and M. Brown, Pixelflex: A Reconfigurable Multiprojector Display System (2001).

April 12, 2008 15:30 WSPC/164-IJIG 00307