The Design and Implementation of PixelFlex: A Reconfigurable Multi-Projector Display System

David Gotz
Department of Computer Science
University of North Carolina at Chapel Hill

Abstract

In this technical report, we present a detailed overview of *PixelFlex*, a multi-projector display system that combines multiple roughly aligned projectors into a unified high-resolution display. We describe the current prototype, the automated calibration process, and two rendering algorithms that support interactive applications. The first rendering algorithm is a one-pass technique that assumes little or no optical projector distortions. It corrects for the linear shear distortions (keystoning) that result from casual projector alignment and non-orthonormal projection. The second rendering algorithm is a two-pass technique that corrects for both linear projection distortions and non-linear distortions introduced by non-planar display surfaces and projector lens distortion.

1 Introduction

Over the last few years, computers have become more powerful by almost all measurements. Processor speeds have increased, hard drive sizes have gone up, and memory has become cheaper. However, typical computer displays have not increased in resolution at the same speed. To address this issue, researchers have explored the possibility of combining multiple standard resolution projectors into a single system to increase the total display resolution. Such multi-projector display systems are currently under development at a number of research labs. Examples of these systems are the CAVE [1], the Scalable Display Wall [5], the InfoMural [3], and a number of other video wall and dome products [9, 10, 6].

However, each of the display systems listed above requires precise geometric alignment of each projector. At the University of North Carolina at Chapel Hill, researchers have built a prototype of a unique display system known as *PixelFlex* [11]. This system is capable of rendering blended imagery across multiple, roughly aligned projectors at interactive frame rates.

2 PixelFlex

The *PixelFlex* display system combines multiple projectors to form a single unified display device. Unlike many other multi-projector systems, each projector can be casually aligned with arbitrary overlap regions between projectors. Figure 2 shows such an configuration. To account for the arbitrary configuration, *PixelFlex* uses a computer controlled camera to perform closed-loop calibration. During this calibration stage, the system registers each of the projectors into a common coordinate frame. The result of this calibration stage is then used during the rendering process.

Figure 1: A PixelFlex user interacts with an X Windows desktop.

Figure 2: Eight projectors arranged with arbitrary overlap.

PixelFlex provides a far more flexible display device than most other multi-projector systems. It is capable of providing variable pixel density, switching between multiple saved configurations, and a performing relatively quick calibration of new configurations.

This report will provide a detailed description of each of four major components that make up *PixelFlex*:

- **Hardware Prototype**: The specific hardware that makes up the prototype and how it is arranged in our lab.
- **System Control Panel**: The software system that allows users to control the hardware as well as run the calibration process and display system.
- **Calibration Process**: The software responsible for calibrating *PixelFlex*. The result of this process is used while rendering.

Figure 3: The PixelFlex projector array.

- **Rendering Algorithms**: Two algorithms designed to render for *PixelFlex*. The two algorithms demonstrate the tradeoffs between speed and accuracy.

Following the description of these four components, this report will present a brief conclusion.

3 Hardware Prototype

We have installed a prototype of *PixelFlex* in a conference room. We built the prototype from standard off-the-shelf equipment. It uses eight projector/pan-tilt unit (PTU) rigs, each containing a Proxima DP6850 LCD projector. These projectors can display at up to 1024×768 resolution and are rated at 1500 ANSI lumens. The Proxima projectors allow computer control of zoom, focus, color balance, brightness and contrast. These controls are accessed via a serial (RS-232) communication port.

In front of each projector is a Directed Perception pan/tilt unit (PTU). The PTUs are also computer controlled via RS-232 ports. The PTU provides two degrees of freedom by allowing the computer to set both the pan angle and tilt angle. We have attached a front surface mirror to each PTU. By controlling the pan and tilt parameters, the system can point the mirror in any direction. The mirror is used to aim the projector's image onto the display surface. The mirror must be a front-surface mirror to prevent inter-reflections that soften the focus of the displayed image. A photograph of a projector/PTU rig can be seen in the inset of Figure 3.

The projectors are arranged for front-projection. This means that they are placed on the same side of the display surface as the viewer. As a result, the conference room did not need any special structural modifications. We simply attached the eight projector/PTU rigs to the ceiling in a four-by-two grid. While not necessary, we opted to make the prototype a little less obtrusive by raising a few ceiling tiles by approximately two feet. This resulted in a small alcove in the ceiling in which we placed the array of projectors. See Figure 3 for a photograph of the projector array.

In order to control the sixteen RS-232 devices, we use a Digi International DigiPort 32 serial port server that provides IP based access to 32 serial ports. The DigiPort 32 server is accessed over an Ethernet-based network from a Windows 2000-based personal computer (PC). The PC also contains a Matrox Meteor II frame grabber used to capture images from a standard NTSC camera.

In addition to the PC, which serves as the control center for the system, we currently use an SGI Reality Monster machine for all rendering tasks. We use two independent IR2 graphics pipes where each pipe provides four channels for rendering.

This prototype allows us to explore many of the issues related to multi-projector displays. It allows us to use existing surfaces (such as the conference room walls) for the display surface. It also allows us to research both geometric and photometric calibration issues.

4 System Control Panel

The control panel serves as the nerve center of *PixelFlex*. The control panel software resides on the Windows 2000 PC and controls the projectors and PTUs. It also allows users to save preset configurations for later use. In addition, it allows users to launch the various calibration programs.

The control panel lets users control a number of projector settings. Zoom, focus, brightness, contrast, red and green color balance, power, picture mute, and input can all be selected via standard radio buttons and scroll numbers. The same interface tools allow users to change pan and tilt angles for the mirrors. Figure 4 shows a screen shot from the control panel graphical user interface.

After a user adjusts the system into a new configuration, the settings can be saved to disk. The user can then restore the system to the saved settings by selecting the saved configuration from a menu and loading the values from disk. This allows the user to store multiple configurations and quickly switch between them.

The control panel also allows users to launch two geometric calibration programs. The first program is the *affine matrix calculation* procedure. The second is the *structured light calibration* application. These two programs will be discussed in more detail in Section 5.

5 Calibration Process

There are two major portions of the calibration process. The first part, *geometric registration*, calculates a mapping to a global coordinate system for each projector. The second part, *photometric calibration*, measures some optical characteristics of each projector.

5.1 Geometric Registration

The first step in calibrating the system is geometric registration. This portion of the system uses a computer-controlled camera to view each projector's display area and calculate a common coordinate frame for the unified display. The first task is to calibrate the camera. Next, we determine a global coordinate system and a mapping from camera space to this global space. Then, the camera is used to observe structured light patterns emitted by each projector. After building a mesh that describes the relationships between each projector and the global coordinate system, the system processes the mesh to compute a number of properties such as overlap regions and the effective display area.

5.1.1 Camera Calibration

An ideal lens would not distort light as it passed through the lens surface. Unfortunately, no lens is ideal. When making accurate measurements with a camera, it is important

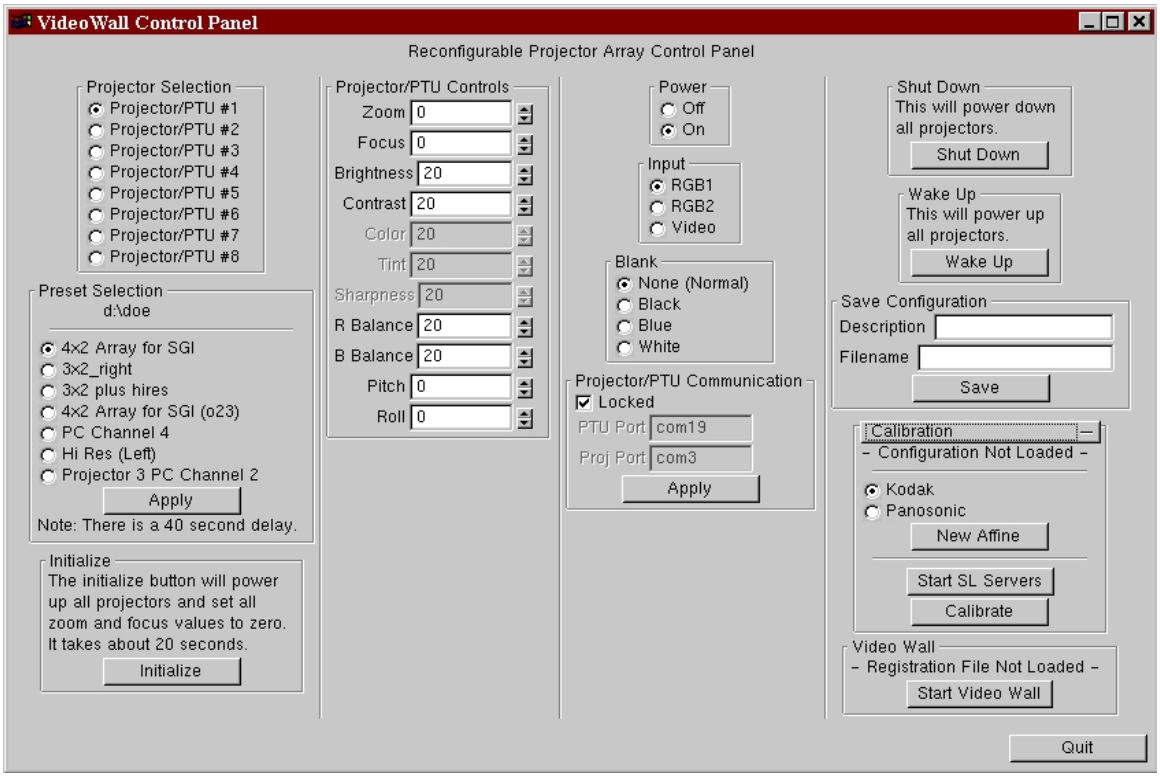

Figure 4: A screen shot from the system control panel.

to correct for any lens distortions. Typical camera calibration procedures approximate lens distortion by characterizing two main types of distortion: (1) *radial distortion* and (2) *tangential distortion*.

Radial distortion refers to the warping of light as a function of the distance from the optical center of the lens. Radial distortion is often referred to as "barrel distortion" or "pin-cushion distortion." We use three parameters to characterize the radial distortion.

In actuality, the radial distortion is not centered directly upon the camera's optical center. Tangential distortion refers to the location of the radial center point with respect to the optical center. Two parameters are used to characterize the tangential distortion.

In order to calibrate our digital camera, we have written an application that uses a calibration routine developed at Intel's Image Lab as part of OpenCV [4]. The application calculates the five distortion parameters by examining pictures of a checkerboard. While the unknowns can be computed by analyzing a single picture, the results may be inaccurate due to noise and limited numerical precision. Better values for the parameters can be obtained by using multiple images and performing an optimization on the results. We typically use approximately 15 photographs of the checkerboard pattern. In addition to the distortion parameters, the calibration routine also determines all relevant intrinsic camera parameters such as focal length and optical center.

5.1.2 Mapping Camera Space to Global Space

Once the intrinsic parameters are known, the system must determine the camera's extrinsic parameters. These parameters describe the location and orientation of the camera with respect to the display surface. Because the display surface

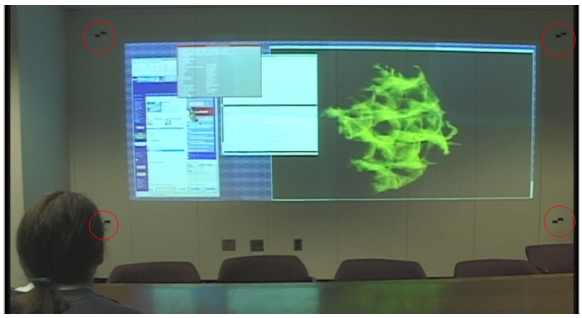

Figure 5: The four fiducials used to define the global coordinate system are circled in red.

in this prototype is restricted to lying in a plane, a simple 3x3 matrix, C, can be used to encapsulate the extrinsic parameters. Matrix C is known as a *collineation matrix* and is defined up to scale. This means that there are just eight unknowns that must be solved.

In order to solve for these unknowns, we must have sixteen values to place into a linear system that defines the collineation matrix. The sixteen values refer to eight 2-dimensional coordinates. Four of these coordinates come from known points on the plane that contains the display surface: (x'_i, y'_i). These four points have been physically measured before the calibration process begins and are marked on the display surface with four fiducials within the field of view of the camera. Figure 5 shows our display surface and the four fiducials.

To find the remaining four 2-dimensional coordinates, a photograph of the fiducials is taken with the camera. This

image is then undistorted to remove both radial and tangential distortion. The camera pixel coordinates for each of the four fiducials (x_i, y_i) are recorded and used to calculate both F (Equation 1) and W (Equation 3). We then solve the linear system shown in Equation 4 for a. The solution for a determines the collineation matrix C as shown in Equation 5.

$$F = \begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 \\ x_2 & y_2 & 2 & 0 & 0 & 0 & -x'_2x_2 & -x'_2y_2 \\ 0 & 0 & 0 & x_2 & y_2 & 2 & -y'_2x_2 & -y'_2y_2 \\ x_3 & y_3 & 3 & 0 & 0 & 0 & -x'_3x_3 & -x'_3y_3 \\ 0 & 0 & 0 & x_3 & y_3 & 3 & -y'_3x_3 & -y'_3y_3 \\ x_4 & y_4 & 4 & 0 & 0 & 0 & -x'_4x_4 & -x'_4y_4 \\ 0 & 0 & 0 & x_4 & y_4 & 4 & -y'_4x_4 & -y'_4y_4 \end{bmatrix} \quad (1)$$

$$a = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{2,1} & a_{2,2} & a_{2,3} & a_{3,1} & a_{3,2} \end{bmatrix}^T \quad (2)$$

$$W = \begin{bmatrix} x'_1 & y'_1 & x'_2 & y'_2 & x'_3 & y'_3 & x'_4 & y'_4 \end{bmatrix}^T \quad (3)$$

$$F \cdot a = W \quad (4)$$

$$C = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & 1 \end{bmatrix} \quad (5)$$

$$\begin{bmatrix} x_{display} * w \\ y_{display} * w \\ w \end{bmatrix} = C \begin{bmatrix} x_{camera} \\ y_{camera} \\ 1 \end{bmatrix} \quad (6)$$

The collineation matrix C is used to transform 2D camera pixel coordinates into 2D display surface coordinates as shown in Equation 6. The display-surface coordinates, defined by the four fiducials, serve as the common coordinate frame for the *PixelFlex* system.

5.1.3 Structured Light Registration

Once the camera has been fully calibrated, the system begins the structured light registration algorithm. This part of the calibration process defines a mapping for each projector's pixel space to the global coordinate system. This mapping accounts for projector lens distortion.

The structured light algorithm works by illuminating a grid of features on each projector. In our system, we are using a ten by ten grid of *Gaussian blobs* as our features. The blobs are circles whose intensity is defined by a Gaussian distribution. They are fully illuminated in the center and fall of to black as the radius increases. This blob structure allows the algorithm to determine the blob center's location with sub-pixel accuracy. This is important because the camera has limited resolution and any error in finding the center of the blobs will result in geometric discontinuities along the projector boundaries in any rendered imagery. Figure 6 shows the intensity distribution associated with a Gaussian blob.

Each projector is calibrated independently. First, the system illuminates all 100 features in the grid and the camera captures an image. Next, the system corrects for radial and tangential distortion in the captured image. Once the distortion has been removed, the system applies a number of

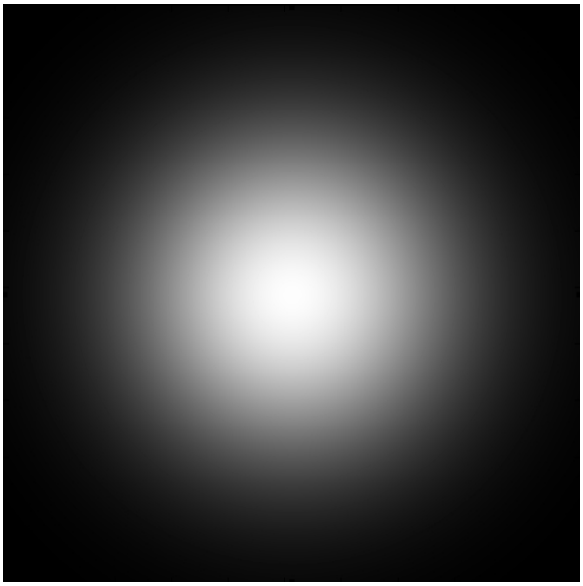

Figure 6: A Gaussian blob.

standard computer vision techniques to determine the center of each blob in image pixel coordinates. This is done to sub-pixel accuracy by calculating a weighted center based on the spatial intensity distribution of the blobs. For each blob, the center location is transformed to the common coordinate system by applying the collineation matrix.

At this point, the system has located 100 features in camera space and has transformed these locations into the global coordinate system. However, to complete the mapping between pixel space and global space, the system needs to know the projector's pixel coordinate for each blob. To do this, the system uses a binary coded structured light algorithm to identify each blob feature. Each blob is given a unique identification number from 1 to 100. These numbers can be represented with seven bits. The display system first illuminates all features whose least significant bit is 1. Features whose least significant bit is 0 are left dark. An image is taken with the camera, corrected for distortion, and analyzed to determine which features were illuminated. The same process is repeated six more times, once for each bit needed to represent the identification numbers.

After analyzing all seven binary coded images, the system knows the identification number for each blob. This identification number allows the system to look up the projector's pixel coordinate for the blob and match this coordinate with the corresponding display surface coordinate.

This process is repeated for every projector in the system. At the completion of this process, the system has 100 samples per projector in the mapping from projector pixels to the global display coordinate system. To compute the entire mapping, the system can interpolate between the measured samples. However, the system does not perform this interpolation because it would result in a huge mapping function with one sample per pixel. Instead, the system builds a much smaller mesh that encapsulates the registration data. This mesh allows us to utilize high-speed graphics hardware to perform the interpolation at runtime.

Because the system knows the location of the each sample in projector pixel space, a mesh can be calculated for each projector by finding the Delaunay triangulation of the

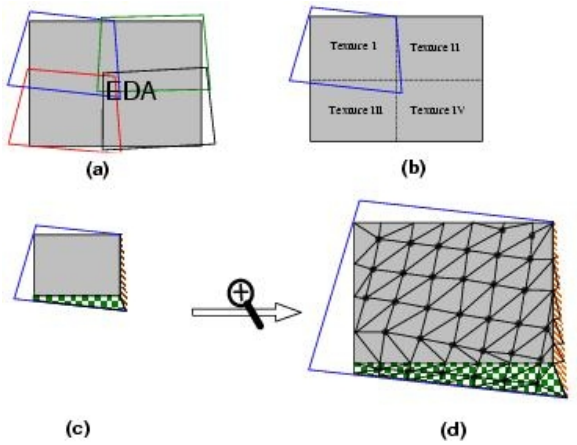

Figure 7: (a) the Effective Display Area (EDA); (b) the EDA is divided into four texture patches; (c) A projector contains three texture patches; (d) Re-triangulation with normalized texture coordinate.

sample points. We triangulate the data because computer graphics hardware provides extremely fast linear interpolation for triangles. While linear interpolation is not the most accurate method of interpolation, we have found through experimentation that a sampling density of 100 samples per projector provides an accurate piecewise-linear approximation with few visual artifacts in the final *PixelFlex* image surface. These meshes are then sent to the mesh processing algorithm.

5.1.4 Processing the Registration Mesh

The structured light registration process generates a subsampled mapping between projector pixels and world coordinates. We must process this initial mapping into the form needed by the two-pass rendering algorithm. This section describes the mesh processing stage while Section 6 discusses the rendering algorithms themselves.

We would like to use the registration data as texture map coordinates into the frame buffer. However, due to a hardware limit on maximum texture size, we have to break down the complete mapping into smaller texture patches, as shown in Figure 7.

Because the projectors are casually aligned, the outer boundary of the PixelFlex display is not guaranteed to be rectangular. Because displays are typically rectangular, we determine a rectangular area from our registration data by computing the maximum inscribed area on the display surface. To calculate this area, we merge the projection areas of all projectors to form a single unified display area. Starting with the centroid of this union, we grow a rectangular area until all four of its sides touch the boundary.

The maximum inscribed rectangular area defines the Effective Display Area (EDA) on the projection screen, shown in Figure 7 (a).

Based on the size of the EDA and the average pixel density, we divide the EDA into small texture patches (shown in Figure 7 (b)), each no bigger than 1024 x 1024 to utilize our system's texture mapping hardware. Other systems may have different maximum texture sizes. We need to further adjust the texture coordinates and triangulation in each patch since texture coordinates of a given graphical primitive must reference the same texture. For each combination

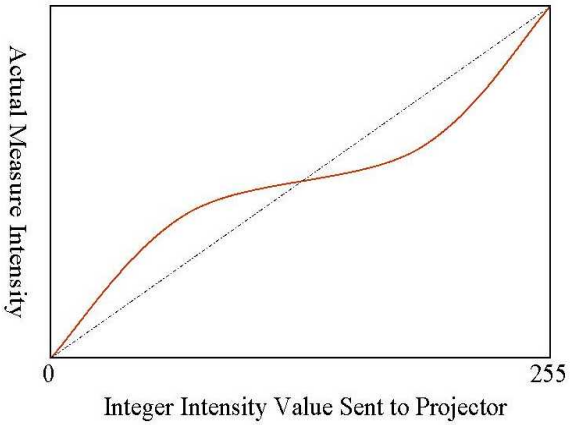

Figure 8: The shape of a typical gamma curve is outlined in red. The black line depicts the desired linear response.

of texture patch (T) and projector(P)'s projection area, we perform a boolean AND operation between them. The intersection is projector space P's contribution to texture T. A projector may contain multiple texture patches; and a texture patch may be used in several projectors, as shown in Figure 7 (c).

The intersections are defined by points on the boundary. To compensate for various distortions, we fill the interior with original feature points from the registration process. Finally, we feed the points in the intersection to a Delaunay triangulation [2] program to create a 2D mesh in the projector's screen space, shown in Figure 7 (d). The texture coordinates are translated and normalized between 0 and 1 within each patch.

5.2 Photometric Calibration

Photometric calibration is an integral portion of *PixelFlex*. The somewhat arbitrary placement of projectors in the system creates a number of overlap regions which, if not accounted for, will create obvious seams in the final display even if the geometric registration is perfect.

The first step in photometric calibration is determining an alpha mask for each projector. An alpha mask is used to attenuate brightness on a pixel-by-pixel basis. For each area on the display surface, the total alpha value for all overlapping pixels must equal one. As a result, alpha values for pixels that do not overlap are set to one. For all other pixels, the alpha value is distributed across all overlapping pixels. The alpha value for overlapping pixels is calculated by looking at the number of overlapping pixels, the pixel's distance from the edge of the projector's boundary, and the pixel density for that projector. The distance from the edge metric is included to allow smooth transitions between projectors. Pixel density is used because projectors with a higher pixel density are capable of displaying more accurate imagery and they should be given priority. The alpha value $A_m(u, v)$ for projector m's pixel (u, v) is computed using the equation:

$$A_m(u,v) = \frac{a_m(m,u,v)p_m^n}{\sum_i a_i(m,u,v)p_i^n} \quad (7)$$

where p_i is the pixel density for projector i and n is an attenuation factor supplied by the user.

The alpha map method of blending intensity across overlap regions assumes that projectors exhibit a linear intensity

response. Unfortunately, this is usually not the case. Each projector has a unique *gamma curve* that describes how the projector maps the value contained in a video signal into an intensity of light. This gamma curve is normally expressed as a function of an integral value between zero and 255. Figure 8 shows a common shape for a gamma curve. One result of the gamma curve is that the actual luminance at intensity value A is not necessarily half the luminance at intensity value $(2 \times A)$. This non-linearity breaks the assumptions made in the alpha map blending algorithm. To solve this problem, we must linearize the gamma curve.

For each projector, we use a spectral radiometer to take measurements of the intensity value for all 256 discreet input values in the video signal. Th data from these measurements provides a mapping between specified intensity and the actual light intensity emitted by the projector. We then calculate an inverse lookup table (LUT) that allows us to quickly linearize the gamma response. We use this LUT to map an idealized linear value into the actual value used by the projector. The result is that an intensity value of $(2 \times A)$ now has twice the luminance of an intensity value of A. Each projector has a unique LUT that is used during the rendering process to correct for non-linear projector response. This LUT is called the *gamma correction LUT*.

6 Rendering Algorithms

The rendering portion of the *PixelFlex* system is the front-end portion of the system that is invoked by the users of the system. We have implemented two rendering applications: (1) a *two-pass X Windows desktop* and (2) a *one-pass 3D viewer*.

The two-pass algorithm corrects for all distortions (keystoning, projector lens distortion, etc.) present in the system with a piece-wise linear approximation. However, we have found through our experiments that when operating near the middle of a projector's zoom range, non-linear distortions such as radial distortion are minimal. As a result, a one-pass technique that corrects only for linear distortions such as keystoning can be used when the projectors are set in the minimal distortion range and the display surface is planar.

6.1 Two-Pass X Windows Desktop

The two-pass X Windows Desktop application is based on VNC [8], an open source application that allows users to run a remote X server and send the desktop across a network to their local machines. We used the source code for VNC as a starting point for our renderer and added *PixelFlex*-specific code as needed.

When started, the first task the rendering module performs is to load the data collected during the calibration portions of the system. The LUT data from the photometric calibration is loaded into the hardware lookup tables found on our SGI machine.

Once the LUTs are loaded, the system is ready to begin rendering. A high-resolution desktop is rendered to an off-screen buffer. This allows a standard X Windows server and window manager to be used. The X Windows server receives all window commands in the normal fashion and the results are rendered into a typical rectangular frame buffer. The only differences are that this frame buffer has an extremely high resolution and it is not assigned to any display device. Instead it is a virtual buffer that resides in main memory.

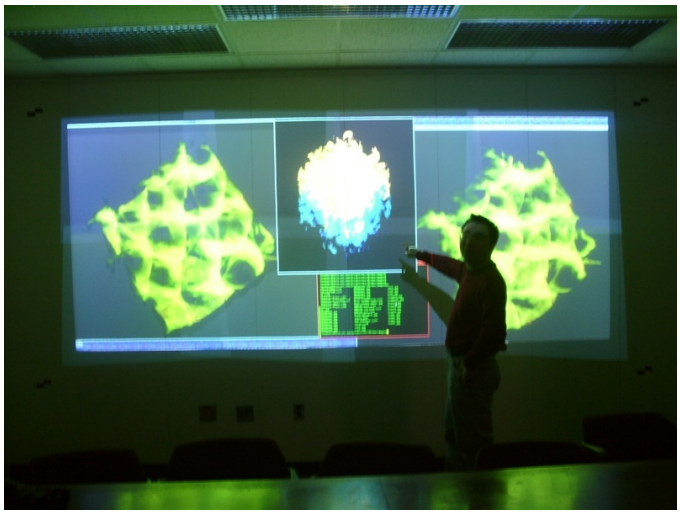

Figure 9: A PixelFlex user examines visualization data.

Figure 10: Two users look at the microprint on a twenty dollar bill.

The next step is to divide the virtual buffer into smaller textures that fit within the maximum limits defined by the OpenGL library. We call these *texture tiles*. In our current prototype, these are 1024×1024 texture tiles. These tiles are then sent to the specialized rendering procedure.

In this portion of the rendering module, each projector is rendered individually. First, the texture tiles are brought in from the virtual frame buffer. Next, the triangle mesh is rendered, one texture tile portion at a time. These triangles are texture mapped using the appropriate texture tile. The texture mapping performs a piecewise linear warp of the virtual frame buffer to the global display coordinate system.

The next step is to apply the alpha mask calculated in the mesh processing stage. Each projector has a unique alpha mask. The alpha mask contains a constant between 0 and 1 for each pixel in the frame buffer. These constants are multiplied by the color information. This results in smooth photometric transitions between projectors in the overlap regions.

As the SGI rendering pipeline sends out the data to the

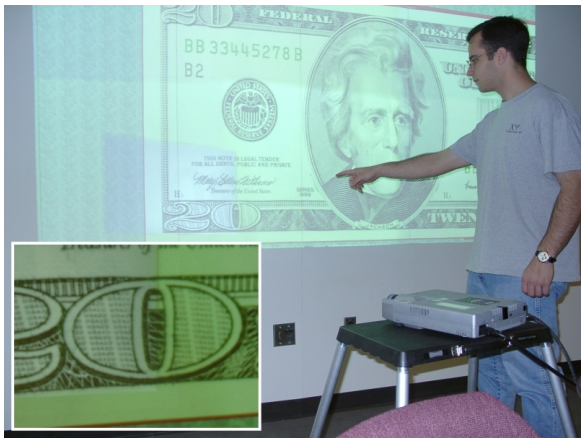

Figure 11: A high resolution inset is used to allow more accurate rendering of the microprint on a twenty dollar bill.

projectors, it applies the hardware LUTs to the frame buffer. Because of the LUT, each pixel is sent to the projector after it has undergone gamma correction. The result is a fully seamless high-resolution display. Figures 9 and 10 show photographs of the system in action.

Due to the unique flexibility allowed by the *PixelFlex* design, the display system is capable of providing high resolution insets. This is accomplished by directing the images from two projector at the same area on the display surface. One projector should be zoomed such that it has a higher pixel density than the other projector. Figure 11 demonstrates such a configuration.

6.2 One-Pass 3D Viewer

The one-pass 3D viewer is an OpenGL-based application that takes over the entire display and is capable of displaying 3D models with only one pass through the rendering pipeline. The 3D viewer uses the rendering technique described by Raskar [7].

The algorithm requires four points per projector in order to compute the collineation matrices for each projector. During the structured light registration stage of calibration, the system records 100 features arranged in a ten by ten grid. The 3D viewer uses the four corner features from the grid to compute the collineation matrices.

The matrices are then appended to the appropriate OpenGL matrix stack for each projector and rendering proceeds as normal. Blending is accomplished by rendering a polygon at the near clipping plane using the alpha mask computed during the photometric calibration stage.

This one-pass rendering method achieves geometric registration at no added cost by folding the linear geometric correction warp into the matrix stack. As a result, the complex process of registration mesh generation and texture tile management is eliminated. Furthermore, the one-pass technique does not introduce the texturing system artifacts of the two-pass rendering process. Figures 12 and 13 show the 3D viewer application.

7 Conclusion

In this technical report, we have presented the design and implementation details for *PixelFlex*, a reconfigurable multi-projector display system. We have provided details about our current eight-projector prototype and the infrastructure needed by the prototype.

We have also detailed the calibration procedure and described how the system uses a computer controlled camera to automate the process. Finally, we have presented two rendering algorithms. One technique is a one-pass algorithm that corrects linear distortions and works well for planar display surfaces and near-linear projector optics. The second is a two-pass technique capable of correcting for both linear and non-linear distortions, as well as rendering on non-planar display surfaces.

Together, these features make *PixelFlex* a unique display system capable of easy reconfiguration, automatic calibration, and real-time interactive rendering.

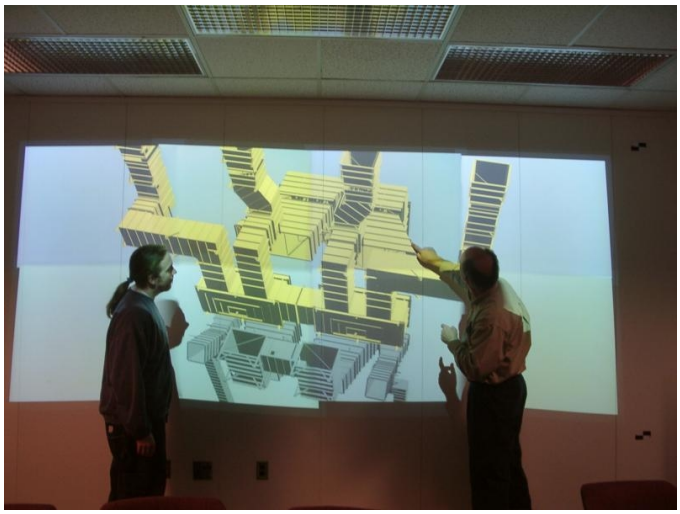

Figure 12: Two users examine parts of a power plant model.

Figure 13: PixelFlex in a *stacked* configuration where four of the projectors almost entirely overlap the other four projectors. This demonstrates that the algorithm is capable of supporting stereo display configurations in the future. The inset shows the rough projector alignment.

8 Acknowledgments

This research is funded by the Department of Energy ASCI VIEWS program under contract B504967 with support from Philip Heermann of Sandia National Labs. I would also like to thank my colleagues in the *Office of the Future* research

group at the University of North Carolina at Chapel Hill: Justin Hensley, Ruigang Yang, Aditi Majumder, and Herman Towles. I would also like to thank Michael Brown from the University of Kentucky.

References

[1] Carolina Cruz-Neira, Daniel J. Sandin, and Thomas A. DeFanti. Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE. *Computer Graphics*, 27(Annual Conference Series):135–142, 1993.

[2] B. Delaunay. Sur la sphere vide. izv. akak. nauuk sssr. *Otdelenie Matematicheskii i Estestvennyka Nauk*, 7:793 – 800, 1934.

[3] G. Humphreys and P. Hanrahan. A distributed graphics system for large tiled displays. In *IEEE Visualization 1999*, San Francisco, October 1999. http://citeseer.nj.nec.com/241717.html.

[4] Intel. OpenCV: Open Source Computer Vision Library. http://www.intel.com/research/mrl/research/opencv/.

[5] K. Li, H. Chen, Y. Chen, D. W. Clark, P. Cook, S. Damianakis, G. Essl, A. Finkelstein, T. Funkhouser, T. Housel, A. Klein, Z. Liu, E. Praun, R. Samanta, B. Shedd, P. J. Singh, G. Tzanetakis, and J. Zheng. Early experiences and challenges in building and using a scalable display wall system. *IEEE Computer Graphics and Applications*, 20(29–37):671–680, 2000.

[6] The University of Minnesota. Power wall. http://www.lcse.umn.edu/research/powerwall/ powerwall.html.

[7] Ramesh Raskar. Immersive Planar Display using Roughly Aligned Projectors. In *IEEE VR 2000*, New Brunswich, NJ, USA, March 2000.

[8] T. Richardson, Q. Stafford-Fraser, K. R. Wood, and A. Hopper. Virtual Network Computing. *IEEE Internet Computing*, 2(1):33–38, 1998. http://www.uk.research.att.com/vnc/.

[9] Trimension Systems. http://www.trimension-inc.com.

[10] Panoram Technologies. http://www.panoramtech.com.

[11] Ruigang Yang, David Gotz, Justin Hensley, Herman Towles, and Michael S. Brown. PixelFlex: A Reconfigurable Multi-Projector Display System. In *Submitted to IEEE Visualization 2001*, San Diego, CA, USA, October 2001.