PixelFlex2: A Comprehensive, Automatic, Casually-Aligned Multi-Projector Display

Andrew Raij*, Gennette Gill*, Aditi Majumder†, Herman Towles* and Henry Fuchs*

*Department of Computer Science
University of North Carolina at Chapel Hill

†Department of Information and Computer Science
University of California, Irvine

Abstract

We introduce PixelFlex2, our newest scalable wall-sized, multi-projector display system. For it, we had to solve most of the difficult problems left open by its predecessor, PixelFlex, a proof-of-concept demonstration driven by a large, multi-headed SGI graphics system. PixelFlex2 retains the achievements of PixelFlex (high-performance through single-pass rendering, single-pixel accuracy for geometric blending with only casual placement of projectors), while adding a) higher performance and scalability with a Linux PC-cluster, b) application support with either the distributed-rendering framework of Chromium or a performance-oriented, parallel-process framework supported by a proprietary API, c) improved geometric calibration by using a corner finder for feature detection, and d) photometric calibration with a single conventional camera using high dynamic range imaging techniques rather than an expensive photometer.

1. Introduction

Large area multi-projector displays are becoming increasingly popular for various applications, changing the way we interact with our computing environment. Over the last five years, researchers have made great strides investigating the major obstacles to building a tiled display, including geometric registration, photometric uniformity and a scalable application interface.

The community has addressed the geometric registration task using a variety of approaches [15, 16, 17, 22, 17, 7]. We believe automatic and robust calibration is an important step towards making multi-projector displays easy to deploy and use in a variety of configurations. To achieve this goal, it is critical that more analysis of the sources of geometric error be done. In this paper, we discuss our findings related to feature extraction and the impact on evaluating the homography required to pre-warp tiled images by comparing the accuracy of two methods. The methods compared are evaluating the centroid of 2D Gaussian features versus finding the corners of binary checkerboard patterns.

There also exists much work on correcting the color variation across multiple projectors using an expensive point light sensing device like a radiometer or a photometer [20, 19, 1, 21]. However, such sensors do not always have computer interfaces, making automatic calibration difficult. Furthermore, point sensors may be limiting in their characterization of spatially varying intensities that we find visually annoying in large-format tiled projector displays. There also exists several works on smoothing the transition at overlap regions using feathering techniques [17, 9]. However, since the blending functions are often derived from inaccurate assessments of the projector's intensity transfer function (ITF) or they do not take into account the spatial photometric variation and black offset issues, they cannot always make the tiled array look seamless for a wide range of source imagery.

Recent research by [13, 12] uses a radiometer and a camera to measure and correct the spatial photometric variation in multi-projector displays. Here, the point light sensing device is used only to measure each projector's ITF, while the camera is used to quantify the spatial variation. In PixelFlex2, we adopt the same correction method to compute both an attenuation mask and black offset mask. However, we present a new method for using high dynamic range imaging techniques with a black and white camera to accurately measure the ITF of each projector and produce the black and white luminance surface definitions required to generate both masks. The accuracy of our method is compared directly with a calibrated spectroradiometer from PhotoResearch. Thus, we are now able to achieve photometric calibration completely automatically using an inexpensive, digital black and white camera - the same camera PixelFlex2 uses for geometric calibration.

For the application developer, the distributed rendering framework provided by the Chromium project [8] now managed under SourceForge.com provides a very nice architecture for transparently running OpenGL applications. We are actively running Chromium 1.2 on PixelFlex2 under Linux with 100Mbps ethernet or Myrinet. As a performance enhancing alternative, we have also developed PxFxLib, a

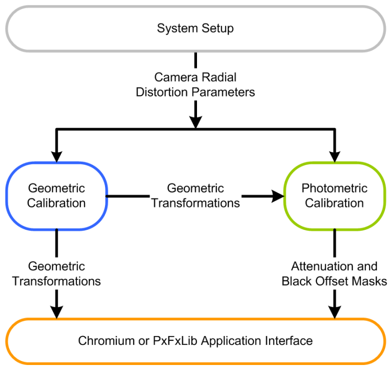
System Setup
Camera Radial Distortion Parameters
Geometric Calibration
Geometric Transformations
Photometric Calibration
Geometric Transformations
Attenuation and Black Offset Masks
Chromium or PxFxLib Application Interface

Figure 1: PixelFlex2 Functional Block Diagram

parallel-rendering framework. Operation in this mode requires some modification of the application code to run on PixelFlex2.

To the best of our knowledge, PixelFlex2 represents the most comprehensively integrated, casually-aligned tiled display that features automatic geometric and photometric calibration using a single digital camera and a flexible framework under which to design and run OpenGL applications.

The remainder of this paper is organized to mirror the PixelFlex2 functional block diagram shown in Figure 1. Section 2 provides a system setup overview, Sections 3 and 4 review our geometric and photometric calibration methods including our new contributions, and Section 5 compares the Chromium and our PxFxLib application frameworks. The final section provides conclusions and outlines future work.

2. System Setup

In addition to physically arranging the projectors and camera, system setup includes the one-time evaluation of their radial distortion characteristics and the placement of fiducials on the display plane.

2.1. Projector Setup

Like its predecessor, PixelFlex2 is a front projection system in which the projectors are casually aligned to maintain a modest 10% to 20% overlap. The current implementation assumes no optical distortion in the projected imagery other than keystoning caused by off-axis projection. To satisfy this constraint, the display surface is planar and the projector zoom optics are set using the method described by Yang, et al. [22] to minimize radial distortion.

2.2. Display Coordinate System

Four fiducials are placed at the corners of the display plane to define a rectangular quad enclosing the projected imagery. Care is taken in their placement as they are used to establish scale (display aspect ratio) and a true sense of display vertical and horizontal within the room. These fiducials are later used during geometric calibration as feature points in computing the homography from display to camera.

2.3. Calibration Camera Setup

We use a single black and white camera (Sony SX900 1394 camera with 1280×960 resolution) for all calibration. The camera is set up to view all projected imagery and the four fiducials. Using Bouget's Camera Calibration Toolbox [2], we evaluate the radial distortion characteristics of the camera and use these coefficients to undistort all subsequent images captured with the calibration camera.

3. Geometric Calibration

The mathematics for rendering geometrically seamless imagery across a tiled set of projectors is well established [15, 1, 7, 16, 17, 22]. It requires knowledge of the function that maps points in the projector to points in the display space. Assuming no optical distortion by the projectors and a planar display surface, then this mapping function is a simple 3×3 homography H_{dp} that can be established with as few as four point correspondences. Furthermore, if we are rendering with a traditional 3D computer graphics pipeline, H_{dp} can be expanded to a 4×4 matrix and pre-concatenated to the view projection matrix. This will correctly pre-warp the imagery in each projector in one rendering pass at no additional computational expense [15].

3.1. Homography Evaluation

For clarity we will briefly review how we compute the display to projector homography H_{dp} for each projector in our system. To determine H_{dp}, we first compute the homography from the display to the camera, H_{dc}, and then compute the homographies from the camera to each projector, H_{cp}. H_{dp} is then defined as the concatenation of H_{cp} and H_{dc}.

$$H_{dp} = H_{cp} H_{dc} \tag{1}$$

H_{dc} is found exactly as in [22]. Correspondences are made between the four fiducials placed around the display plane and their image in the camera, and an exact homography is computed. At the moment, the correspondences are made manually by the user but it would be fairly trivial to do this automatically.

While there is only one display to camera homography, there is a unique camera to projector homography H_{cp} for each projector. The process is similar - establish correspondences in the projector and the camera by projecting features that are automatically found in the camera image, and calculate the corresponding homography. Unlike the process for computing H_{dc}, more than four point correspondences are made between projector and camera so the homography is computed such that the linear least-squares error is minimized. Hence, the accuracy of the homography is influenced by the accuracy and consistency in determining the feature points in the camera image. The next section discusses the geometric calibration improvements we achieved by moving from detecting the centroid of 2D Gaussian features to finding the corners of projected checkerboard patterns.

3.2. Feature Detection - Centroids vs. Corners

In finding projector-camera correspondences, we and others [22, 18] have projected circular features with a 2D Gaussian intensity distribution and processed the camera image to find the centroid of each feature. This approach can yield acceptable geometric calibration, but is prone to error if analyzed closely as shown in Figure 2. We suspect that a major source of error in determining the centroid of Gaussian features is caused by off-axis projection transforming circular features into elliptical features. In the limit, if one could do perfect integration of the photon energy, the correct centroid would be determined, but projective distortion of the camera rectilinear sampling pattern leads to some bias in our centroid computation. Furthermore, correct results assume that the intensity transfer function of every projector pixel is identical. Locally this is nearly true, but projectors do not produce a perfectly flat field output, and this gradient can be severe away from the center of projection. The centroid detection also requires enough camera dynamic range and the correct exposure so as not to clamp the Gaussian feature at white or black.

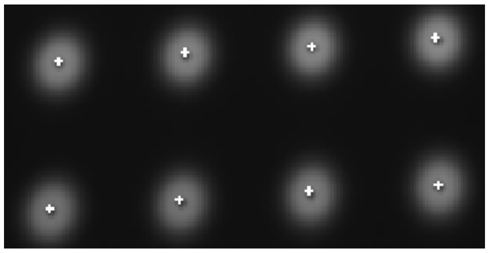
Figure 2: The detected centroids of the gaussian features are marked with white cross hairs. The dark shadow visible behind most crosshairs are the actual locations of the centroids.

Rather than attempt a more sophisticated approach to Gaussian-feature centroid estimation as in [18], we have moved to a feature matching method that is based on finding corners in a checkerboard image. We now use a 5×7 checkerboard and find its corners using a slightly more robust version of the OpenCV [14] checkerboard corner finding function. We purposely use more rows than columns so the detected corners can be correctly ordered spatially into rows and columns. Finally, we project a white circular feature on the lower left hand corner of the checkerboard to make the ordering of projector and camera features independent of projector orientation.

3.3. Feature Extraction Results

Although we are using fewer features, we have found that this corner finding method produces higher quality results than the centroid finding approach we previously used. The binary checkerboard and image processing makes it significantly easier to choose an appropriate threshold value, as well as camera exposure that is robust for a larger range of projector configurations.

Using the same projector and camera configuration, we geometrically calibrated our system with our Gaussian centroid-finder and our new checkerboard corner-finder. As a means of comparing the quality of these two feature detection methods (and by extension, the quality of our homographies), we calculate the projector-to-camera transfer error for each correspondence, which is defined as follows:

$$t = dist(H_{pc}\, p,\ c) \tag{2}$$

where p is a known point in projector space, c is the detected location of p in camera space, H_{pc} is the homography from projector to camera space, $dist$ is a Euclidean distance function and t is the transfer error for the correspondence pair $(p,\ c)$.

Table 1 compares the two feature detection methods, showing the mean and covariance of the transfer error of all correspondences for each of the eight projectors. All measurements are in camera pixels. The mean transfer error is reduced by approximately 50% and the covariance of the transfer error is almost an order of magnitude smaller using corner-finder feature extraction compared to centroid detection of Gaussian features. In particular, the significant decrease in the covariance of the transfer error implies that the quality of the feature detection across each projector is much more consistent (i.e., the detected points statistically fit the computed homography much better) with the

	Centroid		Corner-Finder	
Proj	Mean	Covariance	Mean	Covariance
1	0.1981	0.0115	0.0815	0.0009
2	0.1948	0.0080	0.0802	0.0015
3	0.1773	0.0084	0.0740	0.0011
4	0.1582	0.0078	0.0835	0.0023
5	0.2215	0.0135	0.1162	0.0030
6	0.1969	0.0083	0.1088	0.0020
7	0.2066	0.0104	0.1161	0.0021
8	0.2242	0.0102	0.1015	0.0021

Table 1: Comparison of Transfer Error for Centroid and Corner-Finder Feature Detectors

corner-finder calibration method. Furthermore, we have visually determined that the maximum pixel error of geometric registration is no greater than one pixel and is less in most places. Figure 3 is a high-resolution closeup of a four projector overlap area with the display rendering a diagonal grid. This image shows the high degree of multi-projector registration accuracy we have achieved with our improved feature detection method.

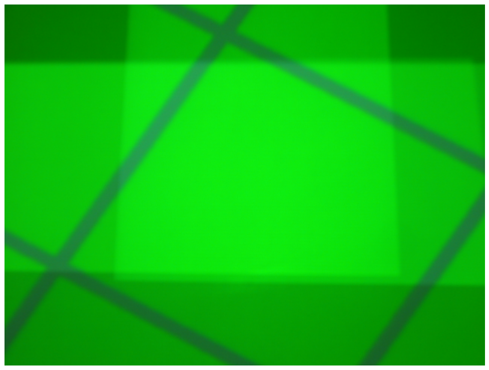
Figure 3: A closeup of a four projector overlap region showing good geometric match of a diagonal grid pattern.

4. Photometric Calibration

With tiled displays the light of overlapping projected images is additive and must be attenuated in these regions to achieve photometric seamlessness. Previously, we have attempted to correct for this with methods requiring explicit knowledge of the location of projector edges and overlap regions. The photometric solution we present here is entirely image based, eliminating the need to explicitly find the projector edges.

The method described by Majumder in [12] is to compute a per projector attenuation mask that is multiplied by the image before it is sent to the projector. This largely solves the photometric blending issues for bright imagery, but there is some black offset present in LCD and DLP projectors that cannot be corrected with attenuation. One solution is to also compute a black offset compensation mask that is added to the projected imagery so the black level of any point in the display is identical. To make the attenuation and black offset masks independent of the input pixel value, it is necessary for the intensity transfer function (ITF) of each projector to be linear. But, projectors in general have a non-linear ITF. The solution is to evaluate the ITF of each projector and to insert a reciprocal function in the pixel pipeline which linearizes the overall projector transfer response [12]. In summary, the goal of the photometric calibration task is to find the ITF of each projector, and create the attenuation and black offset compensation masks.

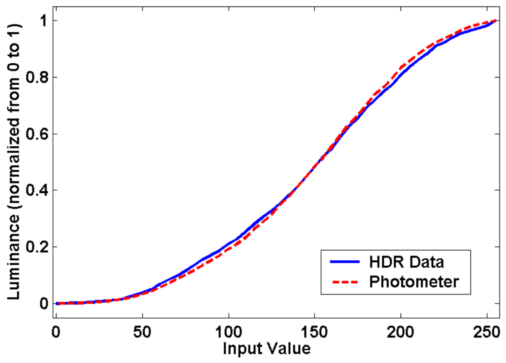

Figure 4: Projector ITF based on HDR measurements with our camera, as well as that measured by a calibrated spectraradiometer. Note the close match.

4.1. Intensity Transfer Function Evaluation

Previously, [22, 12] used a spectraradiometer to accurately measure the intensity transfer function (ITF) of the projectors. These devices can be prohibitively expensive as well as hard to use because they require manual alignment for each projector measured. To overcome these limitations, we have adapted the high dynamic range (HDR) imaging method developed by Debevec and Malik [5] to measure the ITF of the projector using the same black and white camera that we use for geometric calibration.

First we measure the ITF of the entire display - not just a single projector. This is done by simultaneously displaying on each projector the same input intensity. One HDR image is evaluated for every input intensity from black (0) to white (255) yielding 256 HDR images, each of which represents a discrete measurement of the ITF across the whole display. Every HDR image is created from 15 exposures ranging from $\frac{1}{6000}$ to 2 seconds in approximately 1-stop increments. To reduce the noise in these measurements, we take five images per exposure setting and average them.

Majumder [11] has shown that the shape of the ITF of a given projector is spatially invariant, and we have verified the same for our Proxima 6850 projectors. Hence, we can characterize the ITF at every pixel in a projector with a single function. We use the camera-projector homographies H_{cp} to automatically determine regions in the HDR images where each projector is not overlapped. Then the centroid of each non-overlapped region of each projector is automatically computed and the ITFs of a small set of pixels about each centroid are averaged to determine the ITF of each respective projector. This ITF is used to compute the linearizing lookup table as described in [12]. Figure 4 compares the response obtained using high dynamic range imaging and a calibrated spectraradiometer from PhotoResearch. The average difference between the HDR-derived response and the spectraradiometer-measured response is approximately 1% of the total luminance range of the projector.

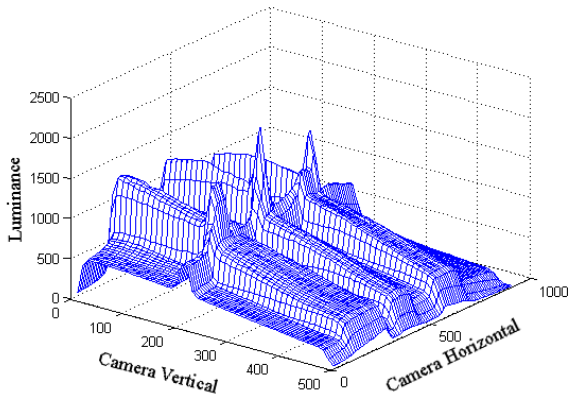

Figure 5: The white luminance surface generated from an HDR image for a 4×2 projector array. Luminance variations attributable to the overlap regions, spatial non-uniformity of individual projectors, and vertical keystoning are all visible.

While all projector ITFs are evaluated simultaneously, the current process of image capture and HDR processing takes approximately two (2) hours on a 550MHz PC. Fortunately, the normalized ITF of a projector changes negligibly over time [11] so the projector ITFs are only re-evaluated periodically or whenever projector brightness or contrast controls are changed, rather than every time the display is re-calibrated. Furthermore, we are confident that the time to evaluate ITFs can be significantly reduced with future software optimizations and a faster PC.

4.2. Display Luminance Surface Evaluation

To create the attenuation and black offset compensation masks, we first evaluate the minimum and maximum luminance of the entire display. Majumder [12] evaluates luminance surfaces by taking an image of each projector not overlapped by any others and then adding all the images together using geometric information. Using a camera in conventional ways makes this step necessary since the full range of luminance is difficult to capture at any one exposure setting. However, HDR images can easily capture the entire range of luminance of the display. One HDR image is taken to capture the white luminance surface and another is taken to capture the black luminance surface. Figure 5 shows an example of a white luminance surface for our display.

To decrease the sensitivity to geometric mapping errors near projector edges, we attenuate the overlapping edges of each projector's image with a cosine-shaped function. This is akin to using the physical aperture mask of [10]. This attenuation creates C0 and C1 continuity at the projector edges by eliminating step transitions in the attenuation mask that would otherwise be computed at projector boundaries. In order to avoid any unnecessary global decrease in luminance, we ensure that the edge attenuation is applied only in overlap regions. Figure 6 shows the entire display with edge attenuation applied to a white image sent to each projector. Since the attenuation mask is computed using HDR images with edge attenuation applied, it is imperative that this same edge attenuation be applied in the operational system. This is forcefully accomplished by modifying the attenuation mask computed for each projector (described in Section 4.3) to also include an identical edge attenuation. Majumder [12] previously applied edge attenuation after the images were captured, but by applying the attenuation in the projected images used for the white surface luminance measurement, it is possible to achieve perfect geometric registration of the edge mask for each projector.

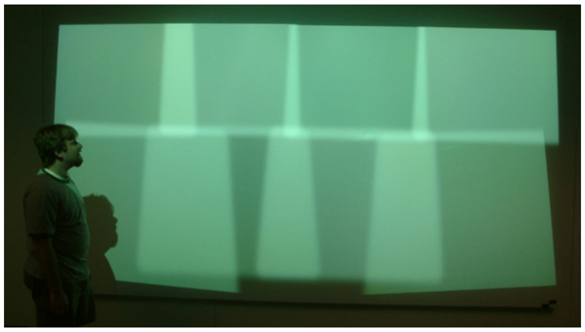

Figure 6: An image of eight overlapping projectors with edge attenuation.

4.3. Generating Masks for Photometric Correction

If the projector ITF is linear, the basic process of photometric correction can be given with one simple equation:

$$\alpha x + \beta \tag{3}$$

where *x* is the input color received from the graphics application, α is the coefficient stored in the attenuation mask and β is the coefficient stored in the black offset compensation mask.

To calculate α and β, we need to know the actual black response, the actual white response, the desired white response, and the desired black response at every pixel in every projector. We set the desired white to be the minimum white response of the whole display w_{min}, and the desired black to be the maximum black response of the whole display b_{max}, ensuring that every pixel can achieve the desired response. This will create a display that has a globally uniform response for every input [13].

For a given pixel in a projector, a graph can be drawn as shown in Figure 7. The line from b_{max} to w_{min} represents the desired response for the pixel. The line representing the measured response starts at the black response (br) and ends at the white response (wr). The relationship between these

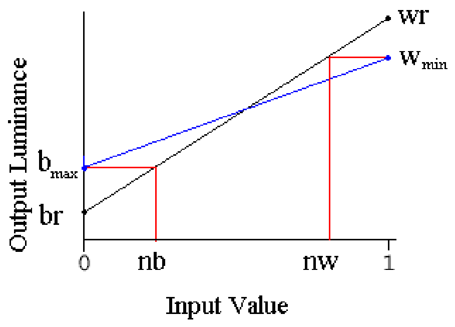

Figure 7: A graph of measured response as compared to desired response

two functions determines the new white (nw) and new black (nb) that achieve the desired response. It is easy to see that

$$nw = \frac{w_{min} - br}{wr - br}, \quad nb = \frac{b_{max} - br}{wr - br} \tag{4}$$

The correct alpha and beta masks must define a line from nw to nb. Alpha is the slope ($nw - nb$) and beta is the y intercept (nb).

$$\alpha = \frac{w_{min} - b_{max}}{wr - br}, \quad \beta = \frac{b_{max} - br}{wr - br} \tag{5}$$

w_{min} and b_{max} are evaluated by searching the white and black luminance surfaces discussed in Section 4.2. The actual region of interest of these luminance surfaces in camera space is defined by the convex hull of all projected images, which can be determined using the camera-projector homographies H_{cp}. We then apply the equations for α and β to compute attenuation and black offset values for every pixel of each projector. This process requires a resampling of the camera-space white and black luminance surfaces in the image space of each projector. As a final step, the same edge attenuation mask used when capturing the luminance surfaces is applied to the masks. The entire process of capturing the black and white luminance surfaces and processing them to produce the α and β masks for each projector takes approximately fifteen minutes but we believe it can be optimized to take much less time. The process should be repeated every time the geometric shape of the display is changed (i.e. a projector is moved) or when projector lamp age significantly changes the maximum or minimum luminance a given projector can display. Figure 8 shows the same image displayed on PixelFlex2 before and after photometric calibration.

5. Application Interface

We provide two application rendering frameworks for running OpenGL applications on PixelFlex2. The first uses Chromium in a sort-first, distributed rendering configuration. Many OpenGL applications are binary compatible

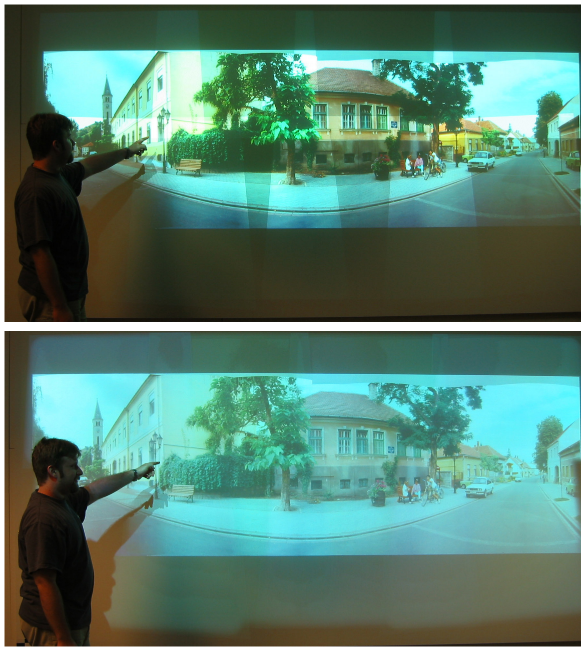

Figure 8: The same image before (top) and after (bottom) photometric correction

with Chromium and do not need to be modified and recompiled to run. This greatly simplifies the task of getting applications to run on a tiled display system.

The second application rendering framework is called PxFxLib, a simple library we designed for parallel-process rendering. Under PxFxLib, each render node runs the application process independently. The display appears seamless due to centralized swap buffer synchronization, as well as the proper distribution of homographies, ITF linearization tables, and attenuation and black offset masks to each application process. Unlike Chromium, PxFxLib requires the modification of the application source, and it can be a complex issue to synchronize all aspects of certain applications (e.g. maintaining a common viewpoint for all rendered views can be problematic in a flight simulator if the physics simulation is also running on every node and taking unsynchronized pilot input from only one node). However, for many applications this is not a problem, and the modifications necessary to use PxFxLib are minimal.

5.1. Implementation Details

We do not discuss the Chromium architecture here, but refer the reader to [8, 4] for details. Instead we focus on our PxFxLib implementation. PxFxLib provides a modified OpenGL application with network access to a swapserver. The modified application is started on each of the render nodes and uses a PxFxLib object to connect to the swapserver and receive the display-to-projector homography it needs to produce its portion of the geometrically seamless image. The application is also modified to apply the homography just before the view projection matrix. After all rendering for a frame is completed, the application process uses the PxFxLib object to inform the swapserver that it is ready to swap buffers. When the swapserver receives swap requests from all the application processes, it simultaneously grants them the right to swap buffers. This simple synchronization scheme is sufficient for most animated applications.

If photometric correction is to be performed, it takes place after all the rendering is complete, just before the application swaps buffers. First the image to be sent to each projector is multiplied per-pixel by the attenuation mask. The black offset mask is added to the result. As previously discussed, both of these steps are designed to work on a linear display device. To achieve an overall linear projector response in real-time, the graphics hardware color lookup table is used to re-map pixel intensities by the reciprocal of the projector's ITF. Figure 9 illustrates the photometric pipeline process. The implementation is based on that presented in [1] with the addition of a black offset compensation mask.

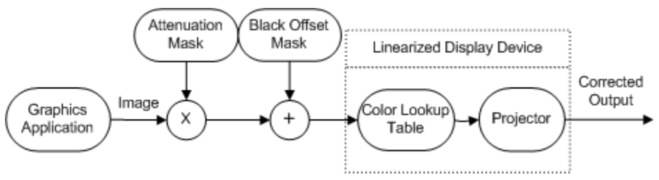

Figure 9: PixelFlex2 photometric correction pipeline.

5.2. Performance

Table 2 shows the performance in raw frames per second of two applications running on PixelFlex2. Since our photometric correction work is not yet integrated into Chromium, all PxFxLib application numbers are for rendering with photometric correction disabled so a comparison can be made with Chromium. The first application is Atlantis, a simple marine life simulation. The second application is FlightGear, a more sophisticated open source flight simulator. Figure 10 shows images of these two applications running on PixelFlex2.

As Table 2 shows, for these two applications PxFxLib outperforms Chromium, even when Chromium is using a Myrinet high-speed network. This is because Chromium is

Application	Chromium	PxFxLib
Atlantis	43*	204†
FlightGear	2†	50†

Table 2: Performance Numbers in Frames per Second for Applications Running on PixelFlex2. Our render nodes consist of 8 Dual AMD Athlon 1800s with 2 gigabytes of RAM and Nvidia Geforce3 Graphics Cards. (* Myrinet gm, † 100mb/s TCP/IP)

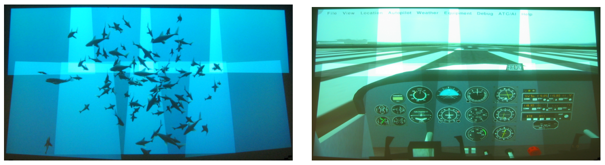

Figure 10: Images ($11' \times 5.5'$) of Atlantis (left) and FlightGear (right) running on PixelFlex2 without photometric correction.

bottlenecked by the amount of immediate-mode geometry that must be sent over the network to the rendering nodes for each frame. PxFxLib does not suffer from this limitation because the identical application runs on each render node with the only communication being the frame swap synchronization. Each parallel application process is responsible for its own rendering and no geometry is sent over the network.

We have experienced excellent Chromium performance running CEI's Ensight, a commercial modelling and visualization product. Although we do not have raw performance statistics to present, we have found that Ensight performs well on Chromium even when a user is interacting with a large model. This is because Ensight makes good use of display lists, and Chromium caches display lists on each render node so that there is no need to resend the geometry at every frame.

6. Conclusions and Future Work

We believe the work presented here is a significant step towards bringing high-resolution multi-projector displays out of the laboratory and into the office. Figure 11 shows ten-year old Miriam Fuchs setting up six projectors and a camera, geometrically calibrating them, and watching a movie. While this demonstrates the flexibility and ease-of-use of PixelFlex2, many important issues remain, including 1) compensating for chrominance differences between projectors and improving seamlessness of blacks 2) improving overall display contrast by computing a perceptually (rather than globally) uniform attenuation mask, 3) improving the performance of Chromium or similar frameworks that provide binary compatible display support, and 4) calibrating continuously and non-intrusively during system use.

We would also like to improve our application interface so that non-OpenGL applications can render to our display wall. The Distributed Multihead X (DMX) project [6] is focusing on this problem and we are interested in integrating DMX into our system.

In conclusion, with continued research we can imagine a future where today's megapixel office display is replaced with an ultra-high resolution, wall-sized display built from small inexpensive projectors. Such devices may enable higher productivity than today's laptops or desktop PCs for both individual work and for group activities.

Figure 11: Ten year old Miriam Fuchs setting up and using a 6-projector PixelFlex2 configuration.

Acknowledgments

This research has been supported by the US Department of Energy's ASCI VIEWS program (B519834) and DARPA's DARWARS Program (ONR N00014-03-1-0589).

References

[1] Justin Binns, Gennette Gill, Mark Hereld, David Jones, Ivan Judson, Ti Leggett, Aditi Majumder, Matthew McCroy, Michael E. Papka and Rick Stevens, "Applying Geometry and Color Correction to Tiled Display Walls", *IEEE Visualization*, 2002.

[2] Jean-Yves Bouguet, "Camera Calibration Toolbox for Matlab", URL: *http://www.vision.caltech.edu/bouguetj/*

[3] C. J. Chen and Mike Johnson, "Fundamentals of scalable high resolution seamlessly tiled projection system", *Proceedings of SPIE Projection Displays VII*, 2001.

[4] "Chromium Documentation Version 1.2 (April 4, 2003)", URL: *http://chromium.sourceforge.net/*

[5] Paul E. Debevec and Jitendra Malik, "Recovering High Dynamic Range Radiance Maps from Photographs", *ACM SIGGRAPH*, August 1997.

[6] "Distributed Multihead X", URL: *http://dmx.sourceforge.net/*

[7] Mark Hereld, Ivan R. Judson, and Rick Stevens. "Dottytoto: A Measurement Engine for Aligning Multi-Projector Display Systems", *Argonne National Laboratory preprint ANL/MCSP958-0502*, 2002.

[8] Greg Humphreys, Mike Houston, Yi-Ren Ng, Randall Frank, Sean Ahern, Peter Kirchner and James T. Klosowski, "Chromium: A Stream Processing Framework for Interactive Graphics on Clusters", *ACM SIGGRAPH*, July 2002.

[9] Kai Li, H. Chen, Y. Chen, D. W. Clark, P. Cook, S. Damianakis, G. Essl, A. Finkelstein, T. Funkhouser, A. Klein, Z. Liu, E. Praun, R. Samanta, B. Shedd, J. P. Singh, G. Tzanetakis and J. Zheng, "Early Experiences and Challenges in Building and Using A Scalable Display Wall System", *IEEE Computer Graphics and Applications*, vol 20(4), pp 671-680, 2000.

[10] Kai Li and Yuqun Chen, "Optical Blending for Multi-Projector Display Wall System", *The 12th Lasers and Electro-Optics Society 1999 Annual Meeting*, November 1999.

[11] Aditi Majumder, "Properties of Color Variation Across a Multi-Projector Display", *SID Eurodisplay*, 2002.

[12] Aditi Majumder, Rick Stevens, "LAM : Luminance Attenuation Map for Photometric Uniformity Across a Projection Based Displays", *ACM Virtual Reality and Software Technology*, 2002.

[13] Aditi Majumder, "A Practical Framework To Achieve Perceptually Seamless Multi-Projector Displays", *Ph. D. Thesis*, Department of Computer Science, UNC-Chapel Hill, 2003.

[14] "The Open Computer Vision Library", URL: *http://sourceforge.net/projects/opencvlibrary/*

[15] Ramesh Raskar, "Immersive Planar Display using Roughly Aligned Projectors", *IEEE Virtual Reality*, MARCH 2000.

[16] Ramesh Raskar, Michael S. Brown, Ruigang Yang, Wei-Chao Chen, Herman Towles, Brent Seales, and Henry Fuchs, "Multi Projector Displays Using Camera Based Registration", *IEEE Visualization*, 1999.

[17] Ramesh Raskar, Greg Welch, Matt Cutts, Adam Lake, Lev Stesin, and H. Fuchs. "The Office of the Future: A Unified Approach to Image based Modeling and Spatially Immersive Display" *ACM SIGGRAPH*, 1998.

[18] Matt Steele and Christopher Jaynes, "Parametric Subpixel Matchpoint Recovery with Uncertainty Estimation: A Statistical Approach," *IEEE Workshop on Statistical Analysis in Computer Vision*, June 2003.

[19] Maureen C. Stone, "Color Balancing Experimental Projection Displays", *9th IS&T/SID Color Imaging Conference*, 2001.

[20] Maureen C. Stone. "Color and Brightness Appearance Issues in Tiled Displays", *IEEE Computer Graphics and Applications*, 2001.

[21] Grant Wallace, Han Chen, and Kai Li, "Color Gamut Matching for Tiled Display Walls", *Immersive Projection Technology Symposium (IPT)*, May 2003.

[22] Ruigang Yang, David Gotz, Justin Hensley, Herman Towles and Michael S. Brown, "PixelFlex: A Reconfigurable Multi-Projector Display System," *IEEE Visualization 2001*, October 2001.