A Practical and Flexible Tiled Display System

Michael S. Brown
Department of Computer Science
Hong Kong University of Science and Technology
Clearwater Bay, Kowloon
Hong Kong, China

W. Brent Seales
Department of Computer Science
University of Kentucky
Lexington, KY 40503
U.S.A.

Figure 1: A PC-cluster tiled display using three casually positioned projectors running an unmodified OpenGL application (*Atlantis*). The display geometry is changed by tilting one of the projectors on its side. Camera-based geometric registration has the entire display back in seamless operation *in less than one minute.*

Abstract

Computer graphics and high-resolution digital imagery are becoming increasingly pervasive in communities not traditionally associated with graphics. Commodity graphics cards and digital cameras, along with powerful modeling software, allow organizations such as libraries, museums, and small businesses to produce expressive and realistic computer generated imagery, far surpassing the capabilities of current desktop monitors. What remains elusive to these groups is access to affordable and easy to use large format display technology.

We present a practical system for deploying flexible projector-based tiled displays. Our framework integrates two key components: (1) self-calibrating display geometry with real-time geometric correction and (2) PC-based distributed rendering that supports an established graphics API. Our system's display geometry is easy to configure and reconfigure, accommodates casually tiled projectors and arbitrary display surfaces, and can be operational in a matter of minutes. In addition, the underlying distributed rendering architecture (WireGL) is transparent to existing OpenGL applications, requiring no custom APIs or re-compilation of existing OpenGL executables. In short, we present a practical and flexible low-cost tiled display system that is simple to deploy and easy to operate.

Keywords: Graphics Systems, Distributed Graphics, Rendering Systems, Virtual Reality, Projector Mosaic, Applications

1 Introduction

Surpassing the "through-a-window" interaction of desktop monitors, large format displays are firmly established as the premiere way to visualize computer generated imagery. These displays provide wide field-of-view imagery suitable for group viewing and offer more pixels and screen real-estate than their desktop counterparts. A variety of large scale displays have emerged and are now commercially available [7, 8, 9, 10]. Most of these systems rely on a projector-based tiled design using a number of light projectors operating together to form a single logical display.

While large format displays provide impressive visuals, the high cost and technical expertise needed to own and operate these displays has restricted their usage to only a few large institutions and well-funded universities. At one time, this was a reasonable phenomenon. Large format visualization was coupled with high performance computing and only a small number of institutions produced datasets that required such expensive and complex display systems.

In the last few years, however, the need for large format visualization has begun to transcend high performance computing with high-resolution, large scale computer graphics data being created and used outside the scientific and engineering communities. Cultural heritage preservation efforts provide a striking example of this trend, with projects that are producing billion polygon models [4, 15], extensive models of virtual archeological sites [2, 3], and extremely high-resolution digital imagery of antiquities [11]. Non-expert users can now easily create high-resolution imagery with commodity digital cameras and image mosaicing software. 3D scanning hardware and sophisticated modeling software make high-resolution 3D models easier and faster to produce. This emerging class of users is creating computer generated imagery that warrants large format visualization. However, traditional large format displays remain beyond most organizations' budgets and expertise.

What is needed is a cost-effective approach for building large format displays that are easy to assemble, easy to deploy in existing spaces, and easy to operate and maintain. Such technology would enable a more expressive and vivid means of graphics visualization for the new class of digital media purveyors inside and outside the scientific and engineering community.

Our Contribution

We present a system that addresses two key problems with large format displays: (1) projector registration and (2) efficient PC-based image generation. Our system is targeted at organizations that have limited budgets, personnel, and available space, but are in need of large format display technology. In particular, we have designed a system for deploying modestly sized (4-6 projector),

but highly flexible, tiled displays composed entirely of commodity-based components. To accomplish this, we have incorporated robust camera-based geometric registration and real-time corrective warping with the WireGL [14] distributed PC-based graphics system for clusters. This self-calibrating display removes all but the simplest tasks, reducing traditional restrictions on projector alignment and display surface shape. By combining PC-based cluster rendering with highly flexible geometric registration, we provide a practical, relatively inexpensive tiled display system suitable for use by groups such as museums, libraries, schools, small businesses, and exhibitors.

2 Background and Related Work

The drawbacks of current *commercial* large scale displays stem primarily from the following two design practices:

[Expensive Rendering Engines] Large scale displays typically rely on multi-processor, multi-piped, expensive rendering engines (such as the SGI Onyx2 family) to generate per projector output. Such machines are costly and require trained users for administration and maintenance. In addition, these machines often require specialized application programming interfaces (API) and/or customized software to display multi-piped imagery; existing applications are typically not supported, or require substantial modifications.

[Precise Geometry] Another major drawback is the reliance on *precise* display geometry. Projectors and display surfaces must be in exact, pre-defined alignment to produce seamless imagery. This restriction requires special purpose mounting and display infrastructure and a great deal of time and effort to deploy and maintain. This also highly constrains the physical space where the display system can be deployed, usually forcing the space to become dedicated to housing the system as a permanent fixture.

Two distinct research directions have emerged to address the above problems. One area focuses on using networked PC clusters with off-the-shelf accelerated graphics cards in place of expensive multi-processor rendering engines. The second area investigates ways to lessen the burden of geometric registration by automatically correcting for geometric distortion arising from unaligned projectors and/or arbitrary display surfaces. We discuss research advances made in these two areas, referring to them as **PC-Cluster Displays** and **Geometric Correction**, respectively. Our goal is to effectively merge these two research areas.

2.1 PC-Cluster Displays

Princeton's Scalable Display Wall (Li et al [17]) was one of the first large scale display implementations to exploit PC clusters with accelerated graphics cards. The design assumes the display surface is planar (a wall), and under that assumption, demonstrates the viability of the PC-clusters approach by providing a variety of significant user- and system-level solutions [16, 23] for tiled display. Other efforts followed this direction [12, 27], leveraging accelerated graphics cards in PCs in lieu of expensive monolithic rendering.

Recently, Humphreys et al [13, 14] provided a generalized PC-based distributed graphics framework called WireGL[1]. WireGL is designed specifically to support the OpenGL API, a well accepted and established graphics API. The WireGL framework increases rendering performance by distributing rendering tasks among a cluster of PCs, each rendering to a logical "tile". These distributed tiles can be efficiently reassembled to create an output image for one or more output devices. WireGL is particularly well adapted for tiled displays, where each projector constitutes a logical "tile" in the WireGL framework.

Although PC clusters lower rendering costs, their usage in large scale displays typically requires precise projector alignment and generally assumes a planar display surface. This rigid alignment is a tedious task requiring very accurate spatial positioning. Often, mechanical platforms are used per projector to provide the necessary precision. To complicate matters, alignment must be frequently repeated due to vibrations in the shelving and projector pixel drift [12]. Thus, PC clusters make displays more affordable, but *not easier* to deploy and maintain.

2.2 Geometric Correction

Work by Raskar [20] and Surati [26] presented a solution to create a seamless display from a set of casually aligned projectors projecting onto a surface. This technique used a single camera to establish the relative geometry of individual projectors' framebuffers with the illuminated display surface. Projected features were observed and registered in the common coordinate frame of the camera's image plane, and an image-based *warp* for each projector's framebuffer could be computed. This post-render warp corrects projectors' framebuffers such that when displayed, the mosaiced imagery creates a seamless (geometrically correct) image on the display surface.

Other geometric registration approaches have emerged [6, 19, 22, 21]. The underlying theme of these efforts is to remove the requirement of precisely aligned projectors by performing an image-based corrective warp before display. This idea has recently been incorporated into projector hardware. 3D Perceptions *CompactView* projectors [1] are capable of real-time image warping, a feature specifically integrated to provide geometric registration for multi-projector display environments.

While the above research and commercial hardware introduced techniques for correcting geometric distortions, they do not address image generation for the display.

2.3 Related Work

There has been work to combine geometric registration and image generation. Princeton's Scalable Display has recently incorporated a planar homography correction into their PC-cluster based display [5]. This work restricts display on planar surfaces and is designed for a dedicated large scale display wall. Yang et al [30] presented *PixelFlex*, a configurable display wall that uses projectors with computer controlled mirrors to mechanically change the projectors' configuration. While this is a flexible display, it relies on an multi-piped SGI *RealityMonster* rendering engine to generate imagery and requires special purpose mounting hardware for the projectors. In addition, *PixelFlex* required modifications and re-compilation of existing applications before used on the display.

The display system presented in this paper is different from previous approaches in several key ways. *First*, we provide a much more flexible PC-based solution that does not require a planar display surface. We can accommodate any reasonably continuous surface. This allows our display to be deployed in a wider variety of locations and by a wider audience of users. *Second*, other systems such as [17, 30] have been designed to be "'display walls" used in dedicated environments. This allows their work to focus on very large scale arrays, with deployment ranging from 8 to 32 projectors. Our system is designed for a different class of users, with more modest resources, but more flexible needs. We provide a flexible (even mobile) display that can be readily changed and reconfigured, and is suitable to be deployed in temporary venues. *Third*, utilizing the WireGL architecture enables our display to support the widely accepted OpenGL graphics API; existing and new

[1] The latest version of this system is now called *Chromium*.

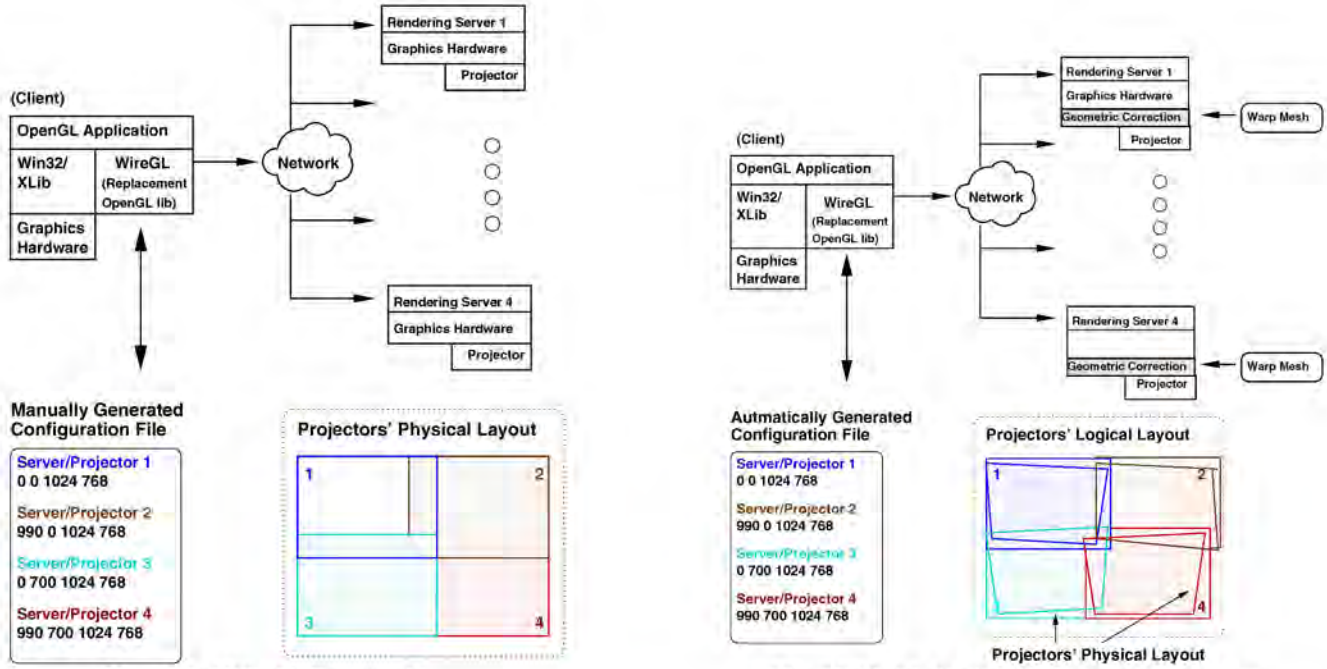

Figure 2: (Right) Unmodified WireGL Architecture: A user-generated configuration file describes the physical layout of the projectors. When an existing OpenGL executable is started, the WireGL replacement library parses the configuration file. Subsequent run-time OpenGL commands are intercepted and sent to the appropriate rendering server based on physical layout. (Left) Modified WireGL Architecture: The client loads in the new configuration file generated by the camera-based registration procedure. When the client connects to the rendering servers, the servers load the appropriate geometric correction information also created by the registration software. Real-time corrective warping is performed on the graphics card before being displayed. *(Diagram adapted from [13])*

OpenGL applications can be run *without* any modification or even re-compilation.

2.4 Flexible Display System

To provide a flexible display, we have successfully incorporated self-calibrating camera-based geometric registration and real-time corrective warping into the WireGL distributed rendering architecture. The WireGL framework provides the underlying OpenGL API support, while our camera-based registration and WireGL modifications allow the display system to be easily deployed and modified (see figure 1).

Our display system consists of two components. First, a camera-based geometric registration procedure determines the necessary per projector geometric warps to create a seamless display imagery. Second, modifications are made to the WireGL graphics pipeline to include a real-time corrective warp. This corrective warp uses the data obtained from the geometric registration procedure. These two components are detailed in the following section, along with results and a brief discussion.

3 Design and Implementation

We first describe the WireGL framework and its normal mode of operation. We then discuss our geometric registration procedure, followed by the details of the integration of geometric correction into the WireGL framework.

3.1 Tiled Display Using WireGL

WireGL is an *open-source* distributed graphics system [28] which provides the image generation foundation of our display system. The idea behind WireGL is that individual local framebuffers on a cluster of PCs can be combined to generate a larger virtual framebuffer. We briefly describe the WireGL framework here, for further details, please see [14]. Figure 2 (left) shows a diagram of the WireGL framework as applied to tiled displays. Rendering servers[2] run on commodity PCs with accelerated graphics cards. Each rendering server will render imagery to a unique *tile* in the virtual framebuffer. Each tile will correspond to the output of a projector. A configuration file specifies the names of the rendering servers, along with each rendering servers' tile's physical layout (startx, starty, width, height) in the virtual display (figure 2(left-bottom)). This user-specified spatial layout corresponds to the physical alignment of the projectors.

A replacement OpenGL library (OpenGL32.lib on Windows and *libGL.so* on Unix platforms) is used by WireGL to support existing applications. This replacement OpenGL library can be conceptualized as the *client* to the rendering servers. When an existing OpenGL application is executed, run-time OpenGL API calls are intercepted, marshalled, and sent to the appropriate rendering server by the client.

Logical tiles in the WireGL system must be *perfectly* rectangular. This means the *physical* alignment of the projectors must project onto the display surface in a perfect rectilinear fashion. This constraint makes deploying a WireGL display tedious. In addition, it restricts display to planar surfaces. In the following sections, we de-

[2]Humphreys refers to rendering servers as *pipeservers* [14].

scribe how we modified this framework to support a more flexible projector alignment.

3.2 Geometric Registration Procedure

To remove the rectilinear projector constraint imposed by WireGL, we a use camera-based registration technique to compute the necessary corrective warp per projector. Our geometric registration technique is similar to that described by Raskar [20]. A single uncalibrated camera observes the tiled arrangement. The tiled display geometry is registered into the common coordinate frame of the camera's image plane, where a virtual framebuffer can be assigned, and the contribution of each projector to this virtual framebuffer can be calculated. Ideally, the camera observes samples such that for each projector, P_i, we obtain a mapping of the projector's pixel $P_i(x, y)$, to its observed position in the camera's image plane, $C(u, v)$. In practice, this is not possible because the camera resolution is much lower than the resolution of the projectors. Instead, the camera subsamples equally-spaced fiducials (circles) projected from each of the projectors. These fiducials form a tessellated mesh in a projector's framebuffer *and* in the camera image plane. This mesh will guide the necessary warping for geometric correction (described in section 3.3.4). Figure 3 shows the observed imagery.

We make no assumptions about the tiled projectors' orientations and allow projectors to be positioned upside down or on their side. To reliably identify the projected features, we use a structured-light binary-encoding scheme which requires $log(n)$ patterns to be projected and observed by the camera per projector, where n is the number of features (see [29] for details). Since the geometric registration procedure is performed by a single OpenGL application that also controls the camera, projected imagery and camera capture are synchronized. The geometric registration application itself runs via the WireGL framework through a dummy configuration file, which specifies a disjoint spatial configuration with no projector overlap.

Figure 3 shows how projected fiducials from each projector appear in the camera along with the composited fiducials and their corresponding mesh. Section 3.3.4 details how this data is used to perform the geometric correction for the displayed imagery.

We note here that the quality of the display's registration depends on how accurately the camera can detect the projected fiducials. The higher resolution the camera has the more accurately it can identify the projected fiducials. Unfortunately, high-resolution video cameras are still very expensive and generally require special-purpose frame-grabbers, and as a result, such cameras are out of the budget of our targeted audience. Instead, it is more realistic to assume that a standard NTSC/PAL commodity video camera with relatively low resolution will be used. To accommodate these lower resolution devices, we adjust the size of our projected fiducials. Empirically, we found that 100 fiducials, each 20 pixels in *diameter*, provided excellent results. We use the centroid of these projected fiducials as the sampled $P_i(x, y)$. This results in a 10-pixel border in the projector's framebuffer that is not used. We justify this loss of usable pixels in order to use a low-cost camera; however, high-resolution cameras can help reduce this loss.

3.3 Integrating Geometric Correction

3.3.1 Tiling the Registered Projectors

Using WireGL's notions of tiles, we choose a rectangular coordinate frame to describe the display's virtual framebuffer. We use the bounding box of all fiducials from all projectors as observed by the camera (shown in figure 3). Projected fiducials are normalized to this coordinate frame. This may result in a logical display that contains pixels that are non-addressable. This design decision and alternatives are discussed more in section 5.

The registration application generates the necessary configuration file. WireGL requires that the spatial layout of each projector be specified in pixel locations that correspond to the logical display. This assumes a uniform pixel size among the individual projectors. To convert our normalized coordinate system into a pixel coordinate system, we choose scale factors in the X and Y direction that accommodate the projectors with the largest width and height in the normalized coordinate system. The scale factors generate a tile the size of the projector's physical framebuffer. This logical tile is guaranteed to accommodate every projector in the virtual display.

After the scale has been set, we surround each projector with a 2D tile. A tile may be larger than needed for a projector. This results from WireGL's assumption of equally-sized projector pixels. Correcting this problem is currently being addressed by on-going implementations.

In addition to generating the configuration file, a file for each projector is created, encoding the warp information as follows. Each projected fiducial, $P_i(x, y)$, is a vertex V, in the triangulated mesh. This triangulated mesh (warp mesh) is saved, along with the following information for each vertex, V; its position in the projector's framebuffers (*V.projector_X, V.projector_Y*), and its location in its logical tile as (*V.tile_X, V.tile_Y*). This information is shown in figure 4.

3.3.2 Modified WireGL Architecture

Figure 2 shows a diagram of the modified WireGL architecture. Rendering servers have an added "warping" routine. The warping process post-warps the rendered tile. The modified WireGL can operate with and without warping. A flag in the configuration files specifies whether or not warping will be performed. If warping is specified, the appropriate warp meshes are loaded when the client application makes its initial connection to the rendering servers. This information contains the $V_{projector}(x, y) \rightarrow V_{tile}(x, y)$ correspondence encoded as a tessellated mesh as described in the previous section.

3.3.3 Intensity Blending

In addition to warping the projected image, intensity blending for projector overlapped regions can be incorporated. Intensity blending attenuates projected imagery that would otherwise cause noticeable bright areas on the display surface in areas of multiple projector illumination. From the geometric registration data, we can compute *alphamasks* for arbitrarily overlapped regions using a technique presented in [19], which is based on an image feathering algorithm for image mosaics [25]. The framebuffer-sized alphamasks specify scale factors from [0-1] that attenuate the pixel values of the corrected image accordingly. Figure 5 shows an example.

3.3.4 Performing the Warp

We assume that the OpenGL application is double-buffered [3]. The geometric warp is performed when the rendering servers receive the *glSwapBuffer()* command from the client. Geometric correction is accomplished by texture-mapping the rendered "tile" to the necessary projected image. The rendered tile, which resides in the framebuffer, is copied to texture memory. The tile is then warped back to the framebuffer using the warp mesh. Recall that this mesh encodes equally-spaced vertices in the projector's framebuffer that have corresponding positions in the *tile*. The following describes the warp in OpenGL pseudo code:

[3]For brevity we do not discuss the details for OpenGL API, please refer to [24] for more information.

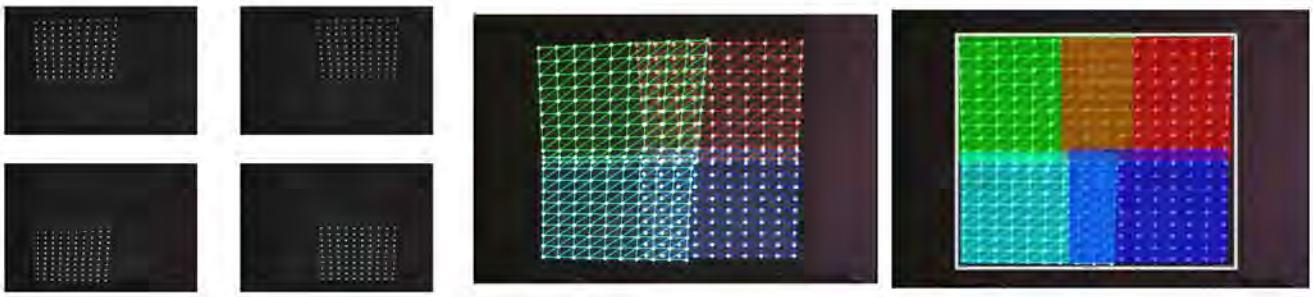

Figure 3: Geometric registration procedure for four projectors. Fiducials from individual projectors are identified in the camera image plane. The fiducials within a given projector form a triangulated mesh. WireGL "tiles" are assigned to each projector.

```
/* GEOMETRIC WARP PROCEDURE*/
/* Copy framebuffer(Tile) to texture memory */
glBindTexture(_WarpTexture_);
glCopyTexSubImage2D( . . . );

/* PERFORM NON-LINEAR WARP */
/* T are the triangles in the tessellated mesh   */
glBegin(GL_TRIANGLES);

 for each triangle T

  for each vertex V in triangle T
   glTexCoord2D( V.tile_X, V.tile_Y );
   glVertex( V.projector_X, V.projector_Y );
  end

 end

glEnd();

/* Intensity Blending */
glBindTexture(_AlphaMask_);
glAlphaSettings( . . . );
render_texture_screen_quad();

/* display corrected image */
swap_buffers();
```

In the above code, variables *V.projector_X* and *V.projector_Y* are the equally-spaced samples in the projected framebuffer. Variables *V.tile_X* and *V.tile_Y* describe their corresponding positions in the logical tile. Figure 4 provides a diagram of the warping. After the warp has been performed, an alphamask is used to attenuate the intensity values for image blending.

Using the geometric correction technique described above, our displayed imagery requires two passes. The first pass renders the desired image while the second pass post-warps the desired image in a piece-wise fashion before actual display. Unlike a global affine or projective transform of the entire framebuffer (which can be performed in a single pass in the graphics pipeline), the piecewise warping approach can approximate non-linear image transformations. This is useful in correcting display distortion arising from non-planar display surfaces and/or projector radial distortion. Any reasonable continuous display surface can be accommodated by this 2-pass algorithm. Disjoint surfaces or surfaces with erratic shape will cause visual artifacts in the geometric correction; however, it is reasonable to assume that such surfaces will typically not be used for display. Increasing the number of projected fiducials (i.e. increasing projector samples in the camera image plane) may be necessary to accommodate highly curved surfaces.

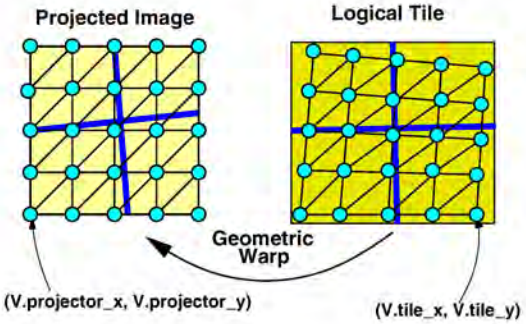

Figure 4: Diagram of warping procedure. Projected fiducials are registered to the logical tile. The tile is rendered by standard WireGL operation. The tile is then piecewise warped to its "corrected" location in the projected image.

4 Results

We have implemented the design outlined in the previous section and demonstrated its effectiveness on a variety of arrangements. Our implementation has two steps. First, the geometric registration application is executed to determine the necessary *warp* information and to create the WireGL configuration for the projector layout. Next, the modified WireGL rendering servers are started and wait for a client connection.

We have deployed two systems; these differ only in the PC's configuration controlling the projectors. The first system was composed of four NT-based Dell Dimension 4100 with low-end Nvidia GeForce2 MX graphics cards with 32MB on-board memory serving as rendering servers. The second used four Windows2000-based Dell Optiplex with high-end Nvidia GeForce-3 graphics cards with 64MB on-board memory. A variety of projectors were used for different configurations, including Sharp, Epson, and Toshiba brands all with XGA (1024x768) resolution. The client machine was a Dell Dimension XPS T650r which also had a Matrox framegrabber for the geometric registration procedure. A TRX900 Sony MiniDV video camera was used in the registration process. This commodity camera has no perceivable radial distortion. The PC-clusters were networked with a 100Mb ethernet switch. For many experiments, the systems were placed on a mobile cart. This allowed us to move the system freely to other rooms, where we could quickly construct a new tiled display. The registration procedure takes roughly *eight seconds* for each projector in the tiled display.

The display imagery shown in the results are generated by a variety of existing OpenGL applications easily found on the web. We also show custom OpenGL applications that we developed, such

Figure 5: Intensity blending using alphamasks. One mask per projector is generated. This mask attenuates overlapping framebuffers. A 2x2 projector array with and without intensity blending is shown.

as a 2D image viewer. The displayed imagery is being created by standard OpenGL executables and not special purpose applications. These applications do not require re-compilation – the display is transparent to their operation. We are even able to run demanding OpenGL applications, such as Quake III.

Figure 6(a) demonstrates the entire system deployed in a targeted environment. An existing room is shown with the projectors, PCs, and camera; besides an ethernet switch, no other hardware was needed. Due to keystoning, this front projector arrangement would be very difficult to align without the assistance of special mounting hardware. Figure 6(c) shows the geometric registration results. The non-linear warping provides seamless geometric alignment for tiled projectors. Figure 6(d) shows the system in use.

Figure 7 demonstrates the display on a non-planar surface. This continuously curved surface is typical of a rotunda found in buildings such as libraries and museums. Geometric correction on this type of surface could not be performed using simple planar homographies and is an advantage of our piecewise warping approach.

Figure 1 on the title page shows the quick, reconfigurable nature of the display. A 1×3 array is running the OpenGL *Atlantis* demo. The position of one of the projectors is substantially changed. Re-registration is performed, and the display is back in operation in less than one minute. This highly flexible design allows even novice users to maintain and operate their own tiled display.

Figure 8 shows a projector being removed from the display. The display is re-registered using only two projectors and is back in operation in a matter of minutes. When the projector is returned, it can easily be integrated back into the display configuration.

Last, figure 9 shows a three-projector array running Quake III Arena. The Quake executable was not modified; however, all OpenGL extensions were turned off. Some of the pixels have been clipped due to the non-rectangular shape of the display. This design decision to allow clipping in order to maximize pixel usage is discussed in section 5.

4.1 Performance and Overhead

4.1.1 Registration Speed and Accuracy

As previously stated, our geometric registration takes only eight seconds per projector. A 2×2 projector array can be in complete operation within *one minute* after the registration procedure has started. This includes the time to calculate and output the new warp files for each rendering server.

Using the geometric correction described in section 3.2, we can generate a visually seamless registration between overlapping projectors. The non-linear warp accommodates projector and display surface distortion. Our experience from the countless number of times we have performed the registration procedure is that we have no problem obtaining sub-pixel registration. Registration accuracies measured on the display surface are typically within a millimeter, while average per-pixel coverage is also around a millimeter. When we do encounter slight mis-registrations, these are most often in very localized areas. Even these mis-registrations are virtually undetectable from a typical viewing distance of a few feet from the display surface.

While we have not performed formal experiments on human subjects' abilities to see geometric mis-registration in our display, we do note that we have demonstrated this display in operation in a variety of arrangements to dozens of visitors, faculty, and students. We have rarely received comments on noticeable mis-registration. In most cases, we have to explain at length that the projectors are not aligned and that the alignment is only obtained by corrective warping.

4.1.2 Display's Frame-Rate

The real-time geometric correction (warping) procedure does require additional per-frame processing. However, since this operation is only related to the time necessary to warp the framebuffer, and not the content being displayed, it is a constant overhead. This overhead time is related to the performance of the graphics card.

We have deployed *two* display systems using the framework described in this paper. For our first system, we used low-end Nvidia GeForce2 MX Cards with 32MB on-board memory. For this system, running with four projectors, we obtained a performance of 16 frames per second(fps) running the Atlantis demo (shown on the first page). This is opposed to 45 fps without geometric warping.

In our second system, we used high-end Nvidia GeForce3 cards with 64MB on-board memory. For this system, with the same configuration and application, we obtained 30 fps with geometric warping and 60 fps without geometric warping. The only significant change between the two systems is the graphics cards used. We believe future cards will provide even better performance.

4.1.3 Resource Overhead

In addition to performance, the geometric correction requires memory overhead on the graphics card. Using a 1024×768 framebuffer, 5MB of on-board texture memory is reserved to perform the warping (4MB) and blending (1MB) buffers. Graphics cards typically have 32-64MB of memory and we have yet to experience any problems from the utilization of 5MB for the geometric warping procedure.

5 Discussion and Future Work

[Rectangular Display] Our design maximizes the usage of pixels by using a logical bounding box for the display's virtual coordinate system. This can result in clipped regions. Instead, we could have easily found the largest *bounded* box and made the display rectangular. While this is more suitable for most applications which assume a rectangular framebuffer, it forces a concept which does not necessarily apply to arbitrarily shaped displays.

[Scalability] The use of a single camera limits the scalability of our display. Although we are targeting an audience that we believe will only budget a few projectors, scalability to several projectors is desirable. The WireGL graphics system demonstrates acceptable performance for up to 32 rendering servers. 32 projectors would easily be out of the view of a single camera. Recent research by Chen [5] shows that several cameras can be used to register a very large display wall. While this work is restricted to planar surfaces, we are encouraged by the results and are pursuing a similar approach for arbitrary surfaces.

[Performance Trade-off] Geometric correction does require a performance price. Our current system is suitable for interactive usage with 30 fps; however, higher frame rates are always preferable. A promising option is to have the geometric correction performed directly on the projectors. 3D Perceptions CompactView projectors [1] already offer this feature. Such technologies could easily be incorporated into our display framework.

[Color Balance] Projectors can exhibit a wide range of noticeable color differences even between the same model of projector. As a result, color-balancing between projectors remains a problem for all tiled-display implementations. Research by Majumder [18] has shown promising results; however, this preliminary work required expensive equipment (spectroradiometer) and is not yet suitable for real-time correction in PC graphics hardware. While the color-balance problem is tolerated, it is an important area for future work.

6 Conclusion

We have designed and demonstrated a practical system for the flexible deployment and operation of projector-based tiled displays. By incorporating camera-based geometric registration and real-time geometric warping with the WireGL distributed graphics framework, we provide a self-calibrating tiled display system that is easily configured and reconfigured and can accommodate a variety of user environments and arrangements. The display is composed entirely of off-the-shelf commodity components and supports existing OpenGL applications. The goal of this system is to provide affordable and easy to operate large format displays. We hope this display system, and subsequent derivations, will enable organizations such as libraries, museums, and small businesses with limited budgets, personnel, and available space a means for more expressive and vivid graphics visualization.

References

[1] CompactView X10. 3D Percpetion AS. *Solbraveien 41, P.O. Box 455, N-1373 Asker, Norway*, http://www.3d-perception.com/.

[2] J. A. Barcelo, M. Forte, and D. H. Sanders, editors. *Virtual Reality in Archaeology*. ArcheoPress, Oxford, April 2000.

[3] R. Beacham and H. Denard. The pompey project: Digital research and virtual reconstruction on rome's first theatre. In *Association of Computing and the Humanities (ACH/ALLC) Conference*, New York University, New York, June 13-17 2001.

[4] F. Bernardini, J. Mittleman, and H. Rushmeier. Case Study: Scanning Michelangelo's Florentine Pietà. In *SIGGRAPH 99 Course 8*, Los Angeles, August 1999.

[5] H. Chen, R. Sukthankar, G. Wallace, and T-J. Cham. Calibrating scalable multi-projector displays using camera homography trees. In *Computer Vision and Pattern Recognition (technical sketch)*, Kauai, Hawaii, Dec 9-14 2001.

[6] Y. Chen, H. Chen, D. W. Clark, Z. Liu, G. Wallace, and K. Li. Automatic alignment of high-resolution multi-projector displays using an un-calibrated camera. In *IEEE Visualization 2000*, Salt Lake City, UT, October 8-13 2000.

[7] C. Cruz-Neira, D. J. Sandin, and T. A. DeFanti. Surround-screen projection-based virtual reality: The design and implementation of the CAVE. In *Computer Graphics (Proc. SIGGRAPH)*, pages 135–142, 1993.

[8] M. Czernuszenko, D. Pape, D. Sandin, T. DeFanti, G. Dawe, and M. Brown. The immersadesk and infinitywall projection-based virtual reality displays. In *Computer Graphics (Proc. SIGGRAPH)*, volume 31, pages 46–49, May 1997.

[9] Elumens. Elumens corporation. *1100 Crescent Green, Suite 211, Cary, NC 27511, USA*, http://www.elumens.com.

[10] Fakespace. Fakespace systems. *Kichener, Ontario, Canada N2G 4J6*, http://www.fakespacesystems.com.

[11] H. M. Gladney, F. Mintzer, F. Schiattarella, J. Bescos, and M. Treu. Digital access to antiquities. *Communications of the ACM*, 41(4):49–57, April 1998.

[12] M. Hereld, I. Judson, and R. Stevens. Introduction to building projection-based tiled displays. *IEEE Computer Graphics and Applications*, 20(4):22–28, 2000.

[13] G. Humphreys, I. Buck, M. Eldridge, and P. Hanrahan. Distributed rendering for scalable displays. In *IEEE Supercomputing 2000*, Dallas, TX, Nov. 4-10 2000.

[14] G. Humphreys, M. Eldridge, B. Ian, G. Stoll, M. Everett, and P. Hanrahan. Wiregl: a scalable graphics system for clusters. In *SIGGRAPH 2001, Computer Graphics Annual Conference Series*, Los Angeles, CA, 2001.

[15] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk. The digital Michelangelo project: 3D scanning of large statues. In *Computer Graphics (Proc. SIGGRAPH)*, pages 131–144, 2000.

[16] K. Li and Y. Chen. Optical blending for multi-projector display wall system. In *Proceedings of the 12 th Lasers and Electro-Optics Society 1999 Annual Meeting*, November 1999.

[17] K. Li and et al. Early experiences and challenges in building and using a scalable display wall system. *IEEE Computer Graphics and Applications*, 20(4), 2000.

[18] A. Majumder, Z. He, H. Towles, and G. Welch. Color calibration of projectors for large tiled displays. In *IEEE Visualization 2000*, Salt Lake City, UT, USA, October 2000.

[19] R. Raskar, M. S. Brown, R. Yang, W. Chen, G. Welch, H. Towles, B. Seales, and H. Fuchs. Multi-projector displays using camera-based registration. In *IEEE Visualization*, pages 161–168, San Francisco, October 1999.

[20] R. Raskar, M. Cutts, G. Welch, and W. Stuerzlinger. Efficient image generation for multiprojector and multisurface display. In *Eurographics Workshop in Vienna, Austria*, pages 139–144, Vienna, Austria, June 29 - July 1 1998.

[21] R. Raskar, J. van Barr, and J.-X. Chai. A low-cost projector mosaic with fast registration. In *ACCV 2002*.

[22] R. Raskar, G. Welch, M. Cutts, A. Lake, L. Stesin, and H. Fuchs. The office of the future: A unified approach to image-based modeling and spatially immersive displays. In *Computer Graphics (Proc. SIGGRAPH)*, pages 179–188, 1998.

[23] R. Samanta, J. Zheng, T. Funkhouser, K. Li, and J. P. Singh. Load balancing for multi-projector rendering systems. In *SIGGRAPH/Eurographics Workshop on Graphics Hardware*, August 1999.

[24] M. Segal and K. Akeley. The OpenGL graphics system: A specification. Technical report, Mountain View, CA,USA, 1993.

[25] H.-Y. Shum and R. Szeliski. Panoramic image mosaics. Technical Report 23, Microsoft Research, 1993.

[26] R. Surati. *Scalable Self-Calibration Display Technology for Seamless Large-Scale Displays*. PhD thesis, Department of Computer Science, Massachusetts Institute of Technology, January 1999.

[27] B. Wei, C. Silva, E. Koutsofious, S. Kirshnan, and S. North. Visualization research with large displays. *IEEE Computer Graphics and Applications*, 20(4):50–54, 2000.

[28] WireGL. http://graphics.stanford.edu/software/wiregl. *(c) 2001 The Board of Trustees of the Leland Stanford Junior University*, 2001.

[29] R. Yang, M.S. Brown, W. B. Seales, and H. Fuchs. Geometrically correct imagery for teleconferencing. In *ACM Multimedia 99*, Orlando, FL, Nov 1st 1999.

[30] R. Yang, D. Gotz, H. Towles, and M.S. Brown. Pixelflex: A dynamically configurable display system. In *IEEE Visualization 01*, 2001.

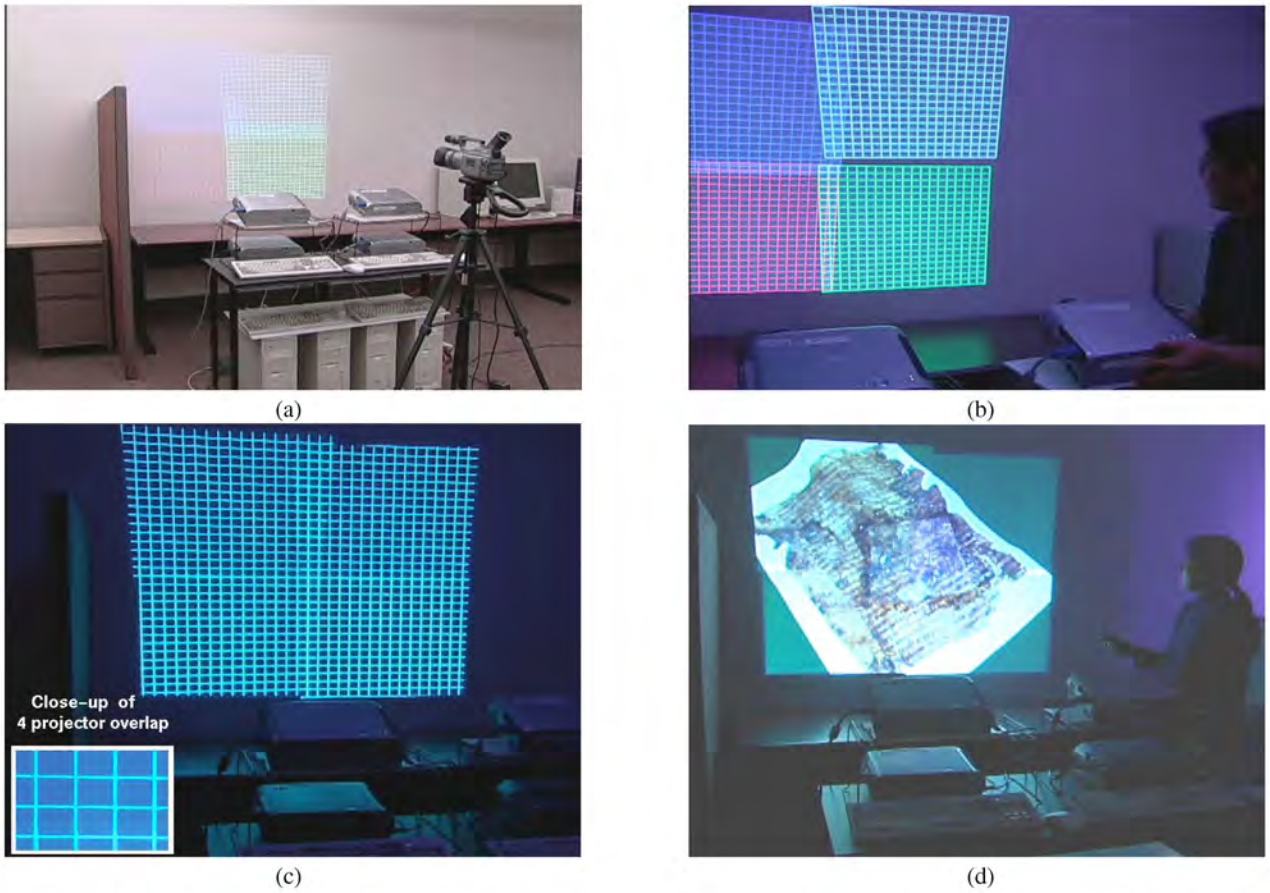

Figure 6: (a) Deployment in an existing room, typical of targeted audience. (b) Front-projected keystoning would make physical alignment virtually impossible. (c) Registration (with close-up inset). (d) Display in use.

Figure 7: (a) Projectors displaying on a non-planar curved surface. This surface is representative of a large rotunda found in a library or museum. (b) Our display can easily accommodate such surfaces generating seamless imagery via a non-linear warp. (c) An example from a different angle.

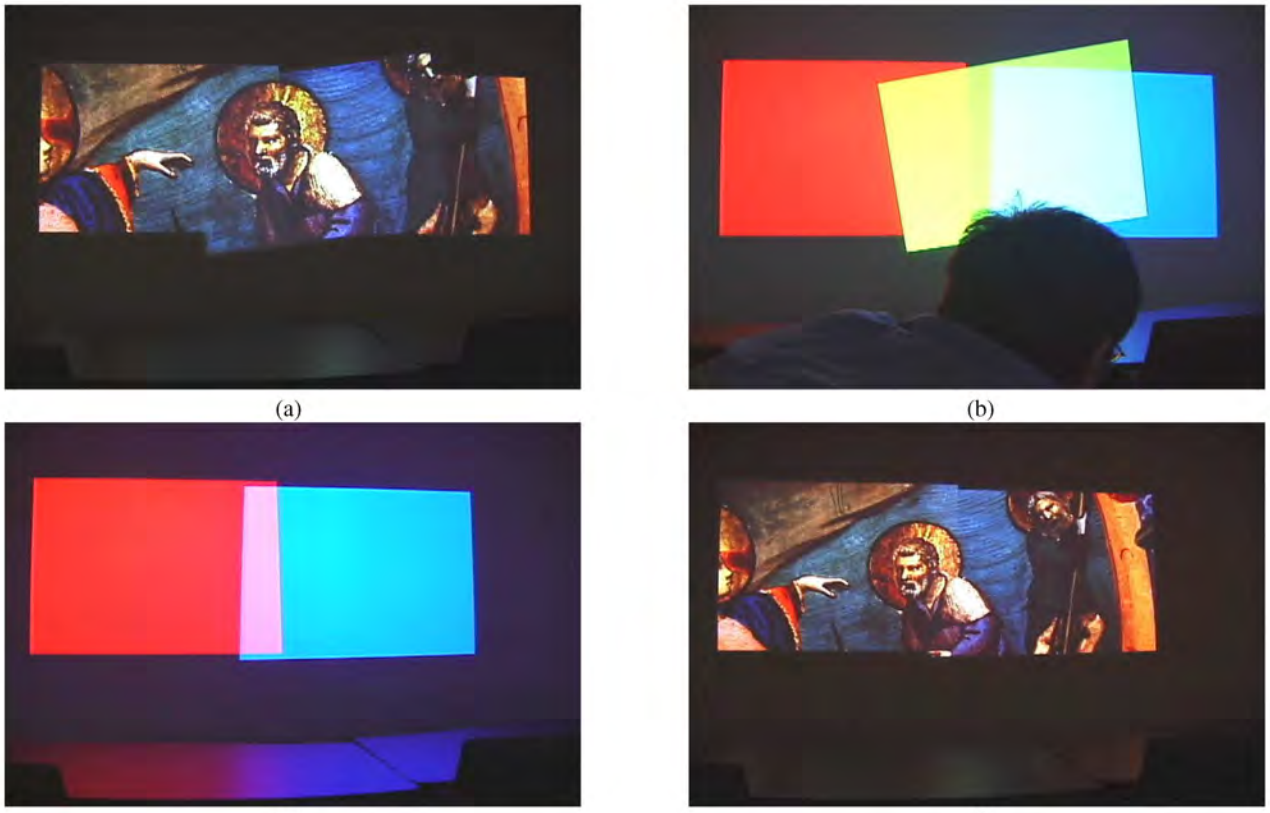

Figure 8: A 1×3 display is changed to a 1×2 display in a matter of minutes. This flexible nature of the display easily accommodates such changes.

Figure 9: (a) Three casually aligned projectors. (b) and (c) the display running Quake III Arena Demo.