桌面环境下拼贴显示的自动对准*

Grant Wallace[1+], Han Chen[2], Kai Li[1]

[1](Department of Computer Science, Princeton University, USA)

[2](IBM TJ Watson Research Center, USA)

Automatic Alignment of Tiled Displays for a Desktop Environment

Grant Wallace[1+], Han Chen[2], Kai Li[1]

[1](Department of Computer Science, Princeton University, USA)

[2](IBM TJ Watson Research Center, USA)

+ Corresponding author: E-mail: {gwallace,li}@cs.princeton.edu, chenhan@us.ibm.com

Received 2004-08-30; Accepted 2004-09-06

Wallace G, Chen H, Li K. Automatic alignment of tiled displays for a desktop environment. *Journal of Software*, 2004,15(12):1776~1786.

http://www.jos.org.cn/1000-9825/15/1776.htm

Abstract: Tiling an array of projectors has become a practical way to construct a high resolution display system. Unfortunately, such high-resolution display systems have limited use because they require specially developed parallel visualization programs that run on a custom-designed parallel machine or a PC cluster. This paper presents an automatic alignment mechanism for arbitrarily tiled displays running a desktop environment so that users can run ordinary applications developed for desktop PCs. The system consists of three primary procedures: detecting projector misalignment, calculating corrective transformations, and real-time warping for the desktop environment. This allows users to run any 2D, 3D or video applications without modifications or special hardware support. Our experiments indicate that the system is able to achieve sub-pixel accuracy and achieve real-time warping with minimum system performance degradation.

Key words: automatic alignment; tiled display; high-resolution display; desktop environment; real-time warping

摘 要: 通过拼贴一组投影仪来构建高分辨率显示系统已成为现在一个较实际的办法.但是,这样的高分辨率显示系统用途有限,因为它们需要在定制的并行机或个人计算机集群上运行一些专门开发的并行形象化程序

* The Princeton Scalable Display Wall Project is supported in part by Department of Energy Grant DEFC0201ER25456, by NSF Infrastructure Grant No.EIA0101247, by NCSA Grant No.ACI9619019 (through NSF), by Intel Research Council, and by Intel Technology 2000 Equipment Grant. Han Chen is supported in part by a Gordon Wu Fellowship.

Grant Wallace, American, male, M.S. Computer Science (1999 Rutgers University), Research Staff (since 2000 at Princeton University). Research interests include large-scale display systems, collaborative software environments, operating systems. **Han Chen** is a research staff member in IBM T. J. Watson Research Center. His research interests include distributed computing systems, scalable display system, and multimedia. He received his Ph.D. degree in 2003 and his M.A. degree in 1999 from Princeton University. He received his B.S. degree from Tsinghua University of Beijing, China in 1997. **Kai Li**, Charles Fitzmorris Professor at Princeton University (since 1986). Research interests include computer architecture, operating systems, parallel systems, scalable display systems, and scalable storage systems. ACM Fellow in 1998.

才能实现.针对个人计算机桌面环境下的任意拼贴显示给出了一种自动对准机制,使得桌面用户可以在其上运行普通的桌面应用软件.该系统包括 3 个步骤:检测投影仪对齐失准,计算出纠正所需之变换,对桌面环境进行实时的变形.这样就允许用户在运行任何 2D,3D 或视频程序时无须作任何修改,也无须使用专门的硬件支持.实验结果表明,系统能够获得亚像素级的精度,并且能在系统性能衰减最小的情况下达到实时变形.

关键词: 自动对准;拼贴显示;高分辨率显示;桌面环境;实时变换

中图法分类号: TP391 **文献标识码**: A

1 Introduction

Large-Format high-resolution displays are increasingly useful in a variety of application environments including control rooms, CAD design, education, and business. For collaborative displays to be effective, it is important that they are easy to use and provide enough resolution and size to be readily viewable by the entire group.

An effective way to build a high-resolution display system is to tile an array of projectors together and drive it with a high-performance graphics machine or a cluster of PCs. This approach has been used to tile tens of projectors together to build wall size display systems that deliver tens of million pixels per frame for large-scale data visualization applications[1]. While such systems have been useful for many large-scale scientific and collaborative applications, its application domain has been limited for two main reasons: cost and ease of use. High-Performance graphics machines are very expensive. Using a PC cluster to drive tiled displays can reduce the cost substantially, but it is still quite cumbersome to develop parallel visualization applications for a PC cluster. Ideally, users would like to drive a tiled display with a commodity PC system. This would be the most economical approach and could run any existing desktop application without modification. In addition, this approach presents users with an intuitive and familiar user interface.

Fig.1 A two projector, automatically aligned, tiled Windows desktop

The challenge is to develop a system to align tiled displays precisely and run desktop applications seamlessly and efficiently. An important aspect when tiling together projectors is to achieve precise geometric alignment. Even small amounts of misalignment will lead to gaps and double images which make the display unacceptable. Manual alignment is possible but tends to be time consuming and inaccurate. Automated approaches can be fast and accurate[2~4], but must be applied in real-time to the displayed imageries. Applying the alignment requires that a

projective warp be applied to each projector's output. Warping the imagery is typically done on a per-application basis by introducing an additional rendering stage. In order to run all applications in the desktop environment seamlessly, one needs to develop a system that can warp the imagery for the entire desktop efficiently and transparently without accessing any source codes or rebuilding binary executables for any applications.

The remainder of this paper presents a system we developed called DeskAlign which automatically aligns and warps the windows desktop of a tiled display driven by a single PC. Some previous work will be discussed in Section 2. Section 3 will talk about design choices in creating such a system. Section 4 will describe our system's implementation and section 5 will present some evaluations and experiences with using DeskAlign. Section 6 will present our conclusions.

2 Related Work

Early display systems built with tiled projectors require manual alignment which is both time consuming and inaccurate[1]. Advances in graphics hardware have made it possible to warp imagery in real-time to correct misalignments for a large-scale tiled display system. Several techniques for camera-based automatic alignment of tiled projectors have been proposed. The common technique is to use a camera to detect projector feature points and derive transformations that can be used to warp the projected pixels, thereby delivering seamless imagery on the projected surfaces. Surati[5] builds lookup tables that map pixels from each projector to points on the display surface; this is done by physically attaching a calibration grid (printed by a high-precision plotter) onto the surface. PixelFlex[3] uses a single, wide field of view camera in conjunction with structured light patterns from each projector to determine camera projector homographies, enabling automatic alignment of a reconfigurable display system.

Using a single camera image of the entire display surface becomes difficult as the size of the display system increases. This motivates approaches to integrate information about the projector geometry from a set of camera images, each of which observes only a portion of the display surface. Raskar *et al* proposed an approach to use two calibrated cameras in conjunction with projected patterns to recover the 3D model of a non-planar projection surface[4]. A similar technique was recently proposed to build into a projector[6]. In order to scale automatic alignment for a large number of tiled projectors, the Princeton scalable display wall used an uncalibrated pan-tilt-zoom camera to detect the relative misalignments of a large number of tiled projectors and used a simulated annealing algorithm to solve a global optimization problem to derive the transformations for imagery warping[7]. An improved technique was later proposed to build and refine a camera homography tree to automatically register any number of uncalibrated camera images[2], this can achieve sub-pixel alignment accuracy for a display system built with tens of tiled projectors.

Previous approaches to running desktop environments on tiled displays have focused on PC cluster architectures. The common architecture uses a proxy machine. The proxy looks like a single display to the applications, but it then divides the display content and redistributes it to the tile nodes. A Windows implementation of such a system is the Virtual Display Driver (VDD)[1]. VDD creates a virtual Windows Desktop of arbitrary resolution. When applications running on that computer make GDI drawing calls, the calls are intercepted, scaled and sent to the appropriate nodes of the tiled display as shown in Fig.2. Distributed Multiheaded X (DMX)[8] is a similar proxy for X Window environments. An X-server runs on one PC and accepts display commands and then redistributes them to the cluster nodes (Fig.3). Both VDD and DMX operate with 2D drawing primitives in order to reduce the network bandwidth which would be required when sending pixel information. These 2D based proxies suffer two main drawbacks. First, they are operating system version dependent and can rely on unpublished interfaces; this makes them difficult to implement and maintain. Secondly, the proxy can become a performance

bottleneck in redistributing the graphics calls.

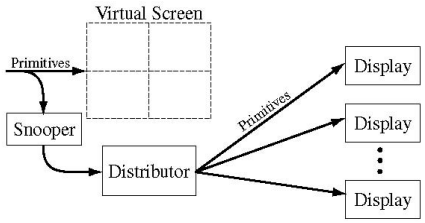

Fig.2 Virtual display driver for tiled display

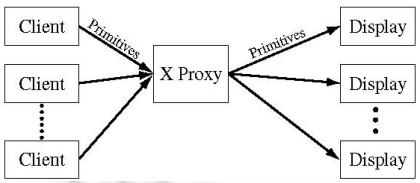

Fig.3 Distributed multiheaded X

Another approach is to distribute pixels instead of 2D and 3D primitives. An example of this approach is an adaptation of Virtual Network Computing (VNC)[9]. VNC allows a user to connect to a remote computer and view/interact with the desktop environment at the pixel level. It requires the remote computer to run VNC Server. VNC Server transfers pixels, which are compressed using simple algorithms, to the client computer. The special VNC software to tile displays is called VNCWall. The VNCWall server is able to handle requests for multiple rectangular subsections of the display. This allows each node in the display to connect to the server and ask for a different subsection thus creating a tiled desktop. In order to use this approach to run a tiled desktop environment from a single PC, it needs to use a loopback mode allowing both client and server to run on the same machine (Fig.4). However, this approach is inefficient; it cannot provide real-time window refresh and dragging nor smooth cursor movements. Furthermore, this approach does not perform the necessary imagery warping for automatic alignment of tiled displays.

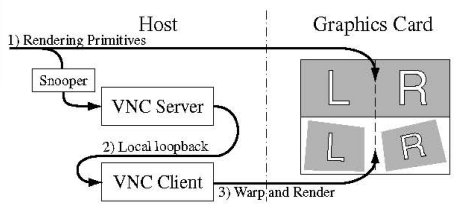

Fig.4 VNC loopback mode on a single PC

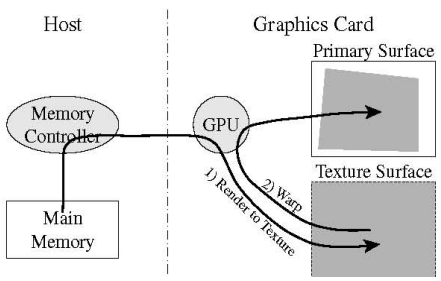

Fig.5 Rendering architecture of NVKeystone

3 Design Choices

When creating an automatically aligned tiled desktop, there are three main steps: determining projector misalignment (deriving transformations), applying projector transformations, and distributing the desktop content to the display tiles. Each step has several design choices.

The first step, deriving the transformations for tiled displays, is loosely coupled with the next two steps. As mentioned in section 2 there are several existing techniques for determining projector misalignment. The design choice mostly depends on the scale and resolution of the screen configuration. One thing to note is that the method of detecting misalignment is independent of the other design choices.

The second and the third steps, the method of applying the transformations and the method of distributing the desktop content to tiled displays, are tightly coupled. If the system responsible for tiling the desktop has information about the positions of the projectors, then it can adjust the amount of content it transfers for each projector. For instance, if a projector covering a small area of screen is surrounded by projectors covering larger areas, then the tiling system can send a smaller section of the desktop to the one, while sending more to the others. This allows the physical positioning of the projectors to be coarse and still produce a good final result. On the other hand, if the tiling system has no knowledge of the projector positions (other than which grid area it occupies), then it sends the same resolution to each display and the post rendering transformations must make the size of the projectors' output match that of the smallest. This has the disadvantage of wasting projector resolution or requiring a more precise physical placement of the projectors.

In order for the tiling system to be aware of the projector alignment, either the desktop system or its proxy must be able to handle this information. Current desktops do not have the capability to handle detailed projector position information. This type of integration would require the use of a proxy such as VDD, VNCWall, DMX, or possibly future versions of Windows Terminal Services*. Proxies can add considerable overhead as data must be shipped to the proxy and then redistributed to the display nodes. Even if everything is on a single PC it still requires copying the data around as opposed to just sending drawing calls to the graphics card.

If the tiling system is unaware of projector alignment, as is the case with the existing systems' multi-monitor support, then we must apply alignment transformations after the content has been rendered on the graphics card. These transformations can be done in one of three places: the graphics card, the projector, or specialized pixel engines sitting between the graphics card and projector. Although some projectors have the ability to perform projective transformations, they are very expensive. There are video-switch pixel engines that can perform transformations but they are also expensive solutions. Among these alternatives, the most cost effective approach is to perform the transformations on the graphics card.

An ideal way to perform transformations on a graphics card is to let a program specify required post-rendering transformations via a natively supplied API. Unfortunately, the current commercial solution, the NVIDIA NVKeystone extension (Fig.5), is limited to perform warping of one display per machine which conflicts with our goal of using one PC to drive multiple projectors. Another limitation is that it is designed for manual adjustment; there is no way to pass transformation information via an API.

In the absence of graphic card support for post-rendering transformations, we propose a two-pass rendering approach by adding an imagery warping stage to the end of the rendering pipeline. Adding another rendering stage is difficult on PC platforms. In order to perform both rendering passes on the same graphics card, with a single graphics pipeline, one needs to have the ability to warp the rendered pixels in the frame buffer and have the control over when the buffer swapping occurs. Unfortunately, such control requires integration into the operating system. An alternative approach is to use a second graphics pipeline for the transformations[10]. This approach can leverage the quad headed graphics cards which have 4 graphics pipelines. To perform two pass rendering we group two pipelines together for each display. This approach can deliver good performance at a relatively low cost.

* Current versions of Windows Terminal Server only allow one client connection at a time and have no sub-region support.

4 DeskAlign System

We have designed and implemented a system called DeskAlign which allows the Windows desktop and Windows applications to run on tiled projectors transparently and seamlessly. DeskAlign implements an automatic alignment mechanism to align tiled projectors that are physically misaligned. It uses a camera to determine the projector positions and calculates an appropriate perspective transformation for each projector. The display transformations are then applied on a multi-headed graphics card running on a single PC. These transformations warp the output imagery of each projector such that the final display appears aligned.

The system is comprised of three components. 1) Detecting projector feature points 2) Determining corrective perspective transformations from the feature points and 3) Applying the corrective transformations on the graphics card using two pass rendering. We have chosen to use the Camera Homography Tree (CHT) alignment algorithm[2] for components 1 and 2. Component 3 implements the transformation in a similar way to Ref.[11].

4.1 Detecting projector positions

The first step in automatically aligning a tiled display is determining the relative positions of the projectors. Simple image processing techniques can typically locate features to the nearest pixel in an input image. However, since a single pixel in our camera images covers several projected pixels on the display surface, our application demands more sophisticated methods. Also, commodity video camera lenses usually exhibit noticeable distortions, making simple perspective camera models insufficient. We use the following five-parameter model to correct for lens distortion.

$$x'=x+x[k_1r^2+k_2r^4+k_3r^6]+[2p_1xy+p_2(r^2+2x^2)]$$

$$y'=y+y[k_1r^2+k_2r^4+k_3r^6]+[2p_2xy+p_1(r^2+2y^2)]$$

where $r^2=x^2+y^2$, and (k_1,k_2,k_3) are the radial distortion coefficients, and (p_1,p_2) the tangential distortion coefficients. These distortion parameters can be obtained via standard offline calibration procedures[12].

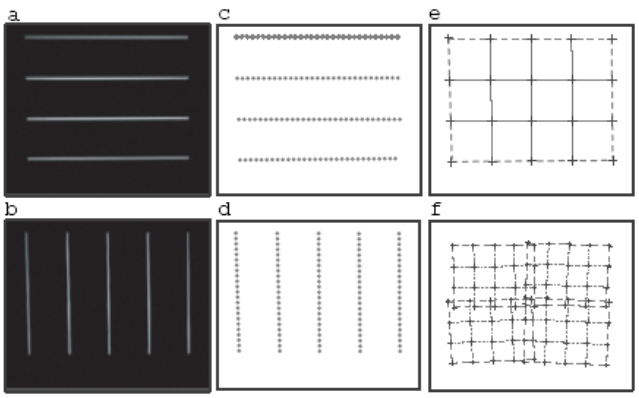

Fig.6 An example of the image processing and feature extraction procedure of our system

The feature detection component of our system displays a sequence of calibration slides on the projectors. The standard approach would be to project a single known pattern, such as a checkerboard, from each projector and use the checkerboard's corners as features. We improve upon this by projecting pairs of patterns: a set of horizontal lines followed by a set of vertical lines (Figs.6(a) and (b)). The intersections between these line sets can be determined with greater accuracy than standard corner detection. To process the images we fit a quadratic function

to the intensity values inside every 9×1 and 1×9 window in the image. A strong peak of the function under a window indicates that a line crosses through the window, and this provides a sub-pixel accuracy estimate of the line's local position, shown by dots in Figs.6(c) and (d). The output of this procedure is a set of position estimates with floating-point precision along each visible line. These feature point positions are then adjusted for camera lens distortion using the model described above and line equations are fit to the observed data. The horizontal and vertical lines are intersected for creating a set of accurate, stable point features for each projector within a camera view.

A GUI application called 'DeskDetect' was developed to handle the data collection phase (Fig.7). It gathers configuration information from the user, and captures feature point images of the projectors. DeskDetect coordinates the use of a camera with the tiled display. It sends commands to draw horizontal and vertical lines to the tiled display and takes images of these features.

4.2 Calculating corrective transformations

Once we've detected the feature points of the projectors, we need to calculate perspective 2D homographies, one per projector, that when applied will make the tiled display appear aligned. In the initial state of our system, we do not know the positions, orientations and optical parameters of the projectors or camera. But we assume that the camera and projector optics can be modeled by perspective transforms and that the projection surface is flat. Thus, the various transforms between camera, screen, and projectors can all be modeled as 2D planar homographies:

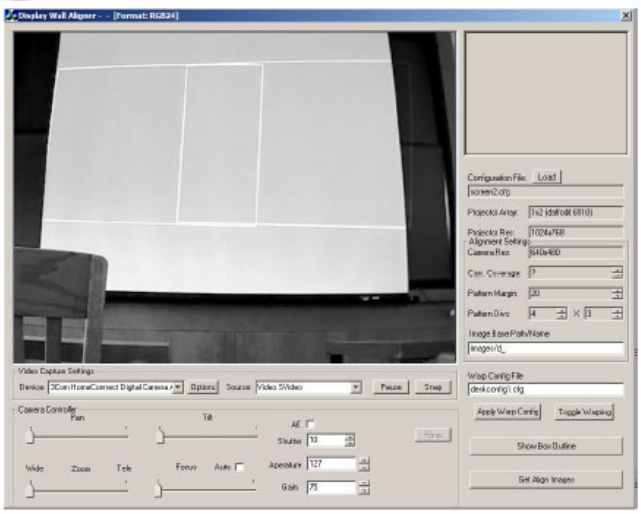

Fig.7 DeskDetect GUI for collecting projector alignment information

$$\begin{pmatrix} xw \\ yw \\ w \end{pmatrix} = \begin{pmatrix} h1 & h2 & h3 \\ h4 & h5 & h6 \\ h7 & h8 & h9 \end{pmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix},$$

where $(x,y)$and (X,Y) are corresponding points in two frames of reference, and $\vec{h}=(h_1...h_9)^T$ (constrained by $|\vec{h}|=1$) are the parameters specifying the homography. These parameters can be determined from as few as four point correspondences using standard methods. We employ the closed form solution described in Refs.[13,14]. It is summarized below.

Given n feature point matches, $\{(x_i,y_i),(X_i,Y_i)\}$,i=1,...,n. Let

$$A=\begin{pmatrix} X_1 & Y_1 & 1 & 0 & 0 & 0 & -X_1x_1 & -Y_1x_1 & -x_1 \\ 0 & 0 & 0 & X_1 & Y_1 & 1 & -X_1y_1 & -Y_1y_1 & -y_1 \\ X_2 & Y_2 & 1 & 0 & 0 & 0 & -X_2x_2 & -Y_2x_2 & -x_2 \\ 0 & 0 & 0 & X_2 & Y_2 & 1 & -X_2y_2 & -Y_2y_2 & -y_2 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ X_n & Y_n & 1 & 0 & 0 & 0 & -X_nx_n & -Y_nx_n & -x_n \\ 0 & 0 & 0 & X_n & Y_n & 1 & -X_ny_n & -Y_ny_n & -y_n \end{pmatrix}$$

and compute the eigenvalues of A^TA. $\vec{h}$ is given by the eigenvector corresponding to the smallest eigenvalue.

Our system employs this technique to compute projector-to-camera homographies. The projector-to-camera homographies transform each projector's area of projection into the camera's coordinate system. These homographies are determined as follows. Each projector P*k* displays calibration slides with highly-visible features, whose locations are known in projector coordinates (as described in section 4.1). By observing the locations of these features in the camera image, we can determine the relevant projector-to-camera homography *cHk*. We determine the camera-to-screen mapping *sHc* by having the user click on the 4 corners of the screen in the camera image. This enables us to compute *sHk*, the homography mapping projector *Pk* to the screen:

$$sHk=sHc\times cHk.$$

Note that *sHk* expresses the geometric distortion induced by the projector's oblique placement. This distortion is removed by prewarping each projector *Pk*'s output by sH_k^{-1} (Fig.8).

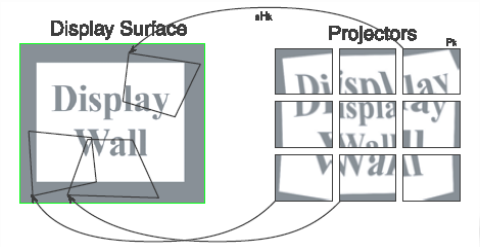

Fig.8 The display is aligned by applying a perspective transformation for each projector

4.3 Applying alignment transformations

The perspective transformations described in the pervious section must be applied to the output pixels in real time for the tiled display to appear aligned. As described in section 3, we chose to implement our alignment transformations on the graphics card. This eliminates the need to modify proprietary operating system code and allows us to avoid specialized hardware pixel engines or projectors.

In order to add a warping stage to the pixels rendered from the entire desktop environment without modifying proprietary operating systems, we have chosen to use two graphics pipelines for each display. The DeskAlign software will grab the pixels from the frame buffer of the first graphics pipeline and push the pixels to the second pipeline to warp according to the transformations. The frame buffer of the second pipeline will then hold the resulting pixels. The DeskAlign software controls the frame buffer swapping of the second pipeline and drives the tiled projectors (Fig.9). This approach requires multi-headed graphics cards and uses half of them to drive the projectors.

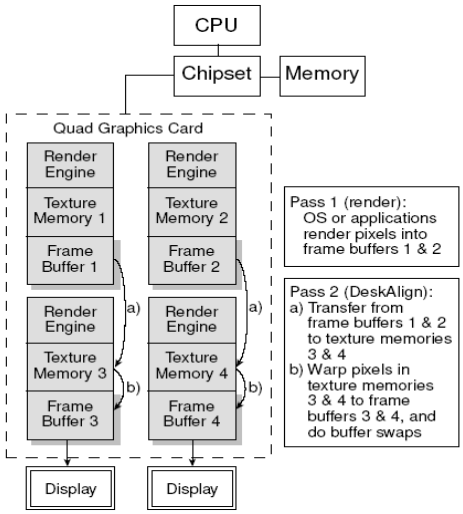

Fig.9 Two pass rendering is used to apply the perspective transformation

We implemented the DeskAlign system for the Windows desktop environment. Initially, DeskAlign opens a fullscreen application window on all of the graphics outputs connected to projectors. It then periodically copies the pixels from the frame buffer of the first graphics pipeline to the texture memory of the second graphics pipeline and then utilizes the texture mapping hardware to warp the pixels into the connected frame buffers that drive the projectors. DeskAlign leverages the DirectX software to perform the move and warping between the two graphics pipelines.

The main potential for a performance bottleneck is the frame buffer copy. Our experience shows that as long as the pixel copy occurs entirely within the graphics card, performance will be good. But if the copy has to transfer back-and-forth through main memory, performance will be unacceptable. DirectX can perform on-card pixel copies to memory locations within the same address space. It is generally possible to configure quadheaded graphics cards to use a shared memory address space. For instance Nvidia quad cards can be set into “span” mode which pairs graphics outputs together.

5 Evaluation and Experiences

Our evaluation goal is to see how well the DeskAlign system performs in the Windows desktop environment. We are interested in two performance goals: How easy is the DeskAlign system to setup and at what frame rates the system can drive the tiled displays in the presence of warping for automatic alignment.

We choose to measure performance in an underpowered environment. Our test platform is a PC with a 866 MHz Pentium III processor and 256 MB memory, a PNY NVIDIA Quadro4 400 NVS graphics card and two Compaq MP1800 projectors.

The system requires a relatively short set up time of about 10-15 minutes. Once all hardware and software have been installed in the PC, the setup involves hooking up the projectors and a camera, capturing some alignment images, running the alignment algorithm to generate the projector transformations and then sending the configuration information to the DeskAlign system. The alignment steps take under five minutes, and most of the setup time is in hooking up the projectors, booting the computer and configuring the NVIDIA driver. A final manual adjustment in software can also be made to get the resulting display as large and square to the screen as possible. This is done by dragging the corners of the display generated by DeskAlign to the desired positions, similar to a

technique used in Ref.[13]. The resulting display is aligned with subpixel accuracy when using an inexpensive webcam.

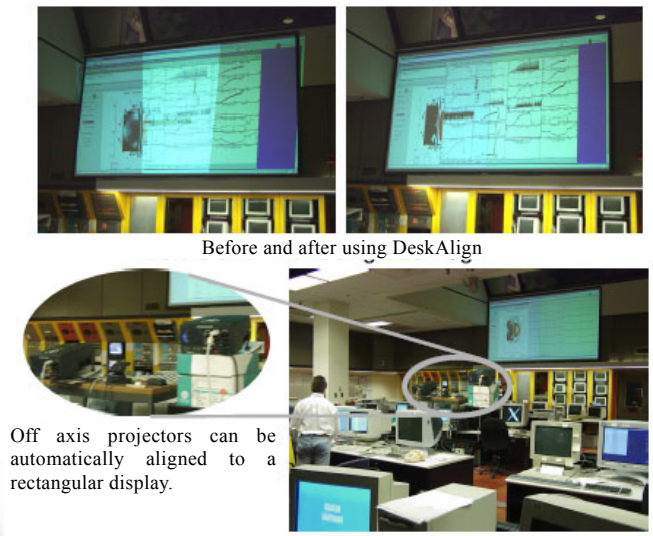

Fig.10 Automatically aligning a two projector display

In order for a PC to run pixel-intensive applications well, it must be able to deliver real-time frame rates. Our measurements show that when DeskAlign is configured to redraw the warped frame buffer 30 times/sec, the system uses about 10% of the CPU, most of which is due to graphics pipeline hardware contention. The CPU overhead minimally affects the usability of the system, and a content refresh of 30Hz is more than sufficient. The result is an automatically aligned desktop display which can be driven from off-axis projectors to ease setup or minimize shadows (Fig.10).

The system is able to run all applications and view their outputs in the Windows desktop environment. Pixel overlays are the only special case. Hardware overlays are of ten used for the mouse cursor and for video applications. Overlay pixels are never rendered to the frame buffer, but rather are combined when the video signal is sent out. These types of pixel overlays will not be captured or rendered in DeskAlign, but they can be disabled within the Windows system. When hardware overlay is disabled, the DeskAlign system can run all desktop applications transparently.

6 Conclusion

This paper presents a software system that creates an automatically aligned tiled display from a single PC. We have designed and implemented a prototype system called DeskAlign. The system uses the Camera Homography Tree approach to detect and generate corrective transformations that can eliminate projector misalignments. We have shown that this can achieve sub-pixel accuracy with a low-end uncalibrated camera and that the process takes only a few minutes.

We have also shown that coupling two graphics pipelines on the same graphics card is an effective approach to achieve warping in the pixel rendering pipeline. The DeskAlign system can run Windows desktop applications transparently. It only consumes only about 10% of the CPU cycles on an underpowered PC platform while delivering pixels at the frame rate of 30 frames per second.

Acknowledgement The Princeton Scalable Display Wall project is supported in part by Department of Energy grant DEFC0201ER25456, by NSF Infrastructure Grant EIA0101247, by NCSA Grant ACI9619019 (through NSF), by Intel Research Council, and by Intel Technology 2000 equipment grant. Han Chen is supported in part by a Gordon Wu Fellowship.

References:

[1] Li K, *et al*. Early experiences and challenges in building and using a scalable display wall system. IEEE Computer Graphics and Applications, 2000,20(4):671~680.

[2] Chen H, Sukthankar R, Wallace G, Li K. Scalable alignment of large format multiprojector displays using camera homography trees. In: Proc. of the IEEE Visualization. 2002.

[3] Yang R, Gotz D, Hensley J, Towles H, Brown M. Pixelflex: A reconfigurable multiprojector display system. In: Proc. of the IEEE Visualization. 2001.

[4] Raskar R, Brown M, Yang R, Chen W, Welch G, Towles H, Seales B, Fuchs H. Multiprojector displays using camerabased registration. In: Proc. of the IEEE Visualization. 1999.

[5] Surati R. A scalable SelfCalibrating technology for seamless LargeScale displays [Ph.D. Thesis]. Department of Electrical Engineering and Computer Sceince, Massachussetts Institute of Technology, 1999.

[6] Raskar R, van Baar J, Beardsley P, Willwacher T, Rao S, Forlines C. iLamps: Geometrically aware and SelfConfiguring projectors. In: Proc. of the ACM SIGGRAPH. 2003.

[7] Chen Y, Clark D, Finkelstein A, Housel T, Li K. Automatic alignment of highresolution multiprojector display using an uncalibrated camera. In: Proc. of the IEEE Visualization. 2000.

[8] Martin K, Dawes D, Faith R. Distributed multihead X design. http://dmx.sourceforge.net/dmx.html

[9] Richardson T, StaffordFraser Q, Wood K, Hopper A. Virtual network computing. IEEE Internet Computing, 1998,2(1):33~38.

[10] Pingali G, Pinhanez C, Levas A, Kjeldsen R, Podlaseck M, Chen H, Sukaviriya N. Steerable interfaces for pervasive computing spaces. In: IEEE Int'l Conf. on Pervasive Computing and Communications PerCom 2003. 2003.

[11] Chen Y, Chen H, Clark D, Liu Z, Wallace G, Li K. Software Environments for Clusterbased Display Systems. 2001.

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

[13] Sukthankar R, Stockton R, Mullin M. Smarter presentations: Exploiting homography in CameraProjector Systems. In: Proc. of the Int'l Conf. on Computer Vision. 2001.

[14] Heckbert P. Fundamentals of texture mapping and image warping [MS. Thesis]. UCB/CSD 89/516, CS Division, U.C. Berkeley, 1989.

[15] Funkhouser T, Li K. Large format displays. IEEE Computer Graphics and Applications, 2000,20(4).