Scalable Alignment of Large-Format Multi-Projector Displays Using Camera Homography Trees

Han Chen [*]
Computer Science
Princeton University

Rahul Sukthankar [†]
HP Labs (CRL) and
The Robotics Institute, CMU

Grant Wallace [*]
Computer Science
Princeton University

Kai Li [*]
Computer Science
Princeton University

Abstract

This paper presents a vision-based geometric alignment system for aligning the projectors in an arbitrarily large display wall. Existing algorithms typically rely on a single camera view and degrade in accuracy[1] as the display resolution exceeds the camera resolution by several orders of magnitude. Naïve approaches to integrating multiple zoomed camera views fail since small errors in aligning adjacent views propagate quickly over the display surface to create glaring discontinuities. Our algorithm builds and refines a camera homography[2] tree to automatically register any number of uncalibrated camera images; the resulting system is both faster and significantly more accurate than competing approaches, reliably achieving alignment errors of 0.55 pixels on a 24-projector display in under 9 minutes. Detailed experiments compare our system to two recent display wall alignment algorithms, both on our 18 Megapixel display wall and in simulation. These results indicate that our approach achieves sub-pixel accuracy even on displays with hundreds of projectors.

CR Categories and Subject Descriptors: I.3.3 [Computer Graphics]: Picture/Image Generation – Digitizing and scanning, Display algorithms, Viewing algorithms; I.4.1 [Image Processing and Computer Vision]: Digitization and Image Capture – Imaging geometry, Camera calibration; B.4.2 [Input/Output and Data Communications]: Input/Output Devices – Image display.

Additional Keywords: large-format tiled projection display, display wall, camera-projector systems, camera-based registration and calibration, automatic alignment, scalability, simulation, evaluation.

1. INTRODUCTION

Large format high-resolution display devices are becoming increasingly important for scientific visualization, industrial design and entertainment applications. A popular approach to building such displays is the *projector array* [7][4], where several commercially-available projectors are tiled to create a seamless, high-resolution display surface, (see Figure 1).

The projectors in the array require precise geometric alignment to prevent artifacts in the final display. This is typically done by manually aligning the projectors. Unfortunately, the process is labor-intensive and time-consuming; furthermore, the display wall requires frequent re-alignment since the projectors shift slightly due to vibration and thermal flexing in the mounts. Given the increasing demand for large format display walls, alignment solutions must scale well to multi-projector arrays of arbitrary size.

[*] {chenhan, gwallace, li}@cs.princeton.edu
[†] rahuls@cs.cmu.edu

[1] Throughout the paper "accuracy" is intended to mean "local registration accuracy" unless otherwise stated.

[2] We use homography synonymously with collineation: a planar transformation which maintains collinear points.

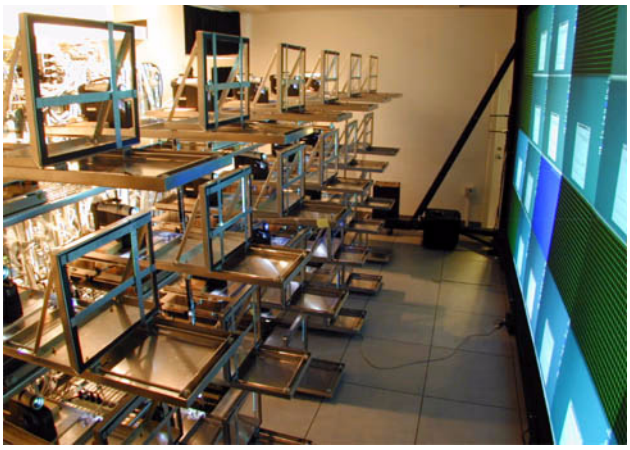


Figure 1. The Princeton Scalable Display Wall is an 18'×8' display with an effective resolution of 6000×3000 pixels. This behind-the-scene photograph shows the array of 24 microportable projectors.

Recent advances in commodity graphics hardware have made it possible to pre-warp imagery in real time to correct misalignment in a tiled display wall. This enables software based solutions for multi-projector display alignment without compromising image quality. Several ideas for camera-based automation of projector array alignment have recently been proposed. Surati [11] 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. While this approach is adequate for a 2×2 array of projectors, it scales poorly for larger displays since creating and accurately mounting an absolute measurement grid onto the display surface is infeasible. The PixelFlex system [13] 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 wall. These methods assume that the entire display surface is small enough to be completely visible in one camera's field of view. As display walls become larger, capturing a single camera image of the entire display surface becomes increasingly impractical: a single pixel in the camera maps to unacceptably large areas on the display surface once the display wall grows beyond a certain size.

This motivates approaches that can integrate information about the projector geometry from a set of camera images, each of which observe only a small portion of the display surface. Raskar *et al.* [8] employ two calibrated cameras in conjunction with projected patterns to recover the 3-D model of a possibly non-planar projection surface. The need to calibrate camera views to the system makes this approach difficult to scale. Chen *et al.* [3] use a pan-tilt-zoom camera to observe the individual overlap regions in the projector array. Information about local discontinuities, e.g. point-matches and line-matches across the seam, is acquired using an iterative process, and a large global optimization problem is constructed using this data. Simulated annealing is used to find the pre-warps that minimize discontinuity errors. The primary advan-

tage of their algorithm (referred to as SimAnneal in the remainder of this paper) is that, in principle, it scales well to large display walls since the uncalibrated camera can easily scan the overlap regions. However, simulated annealing converges slowly, particularly in high-dimensional parameter spaces. Furthermore, unless the initial manual alignment between projectors is good, the optimization algorithm can converge to an incorrect solution.

This paper presents a scalable approach to display wall alignment that is both more accurate and faster than existing approaches. It is motivated by the single-projector keystone correction system described in [10], adapted to employ images taken from multiple, uncalibrated camera views. Our approach efficiently scales to projector arrays of arbitrary size without sacrificing alignment accuracy. The experiments described in this paper were performed on an 18'×8', 24-projector display with an effective resolution of 6000×3000 pixels (see Figure 1).

The remainder of this paper is organized as follows. Section 2 details the camera homography tree algorithm for automatic display wall alignment. Section 3 describes our evaluation methodology, presents our approach for automatic measurement of alignment errors, and presents a simulator (validated on real data) that enables scalability experiments on very large format display walls. Section 4 presents experimental results investigating the accuracy, scalability and running time of our algorithm and comparing the approach to two recent systems, SimAnneal [3] and PixelFlex [13]. Section 5 summarizes the paper's contributions.

2. DISPLAY WALL ALIGNMENT

Our display wall setup consists of 24 projectors, a cluster of workstations and a pan-tilt-zoom video camera. The Compaq MP-1800 microportable XGA-resolution DLP projectors are arranged in a 6 wide by 4 high array behind a rear-projection screen, as shown in Figure 1. The projectors are mounted so that their display regions cover an 18'×8' region, generating an effective resolution of approximately 18 Megapixels. The projectors are controlled by a cluster of commodity PCs connected over a high-speed network. The pan-tilt-zoom NTSC-resolution camera is mounted on the ceiling in front of the projection screen.

To make our discussion general, we define the following notations. The term *wall*-(H,V) denotes a display wall array with $H\times V$ projectors, arranged in a rectangular grid with H projectors horizontally and V projectors vertically. Each projector is assumed to have a resolution of 1024×768; thus a *wall*-(6,4) has an approximate resolution of 6000×3000. The term, *cam*-$N\times N$, denotes a set of camera poses and zoom lens setting, where an $N\times N$ subset of the projector array is completely visible in each camera image. There are $n_v=max(H-N+1,1)\times max(V-N+1,1)$ distinct camera views available as input. For instance, a *cam*-3×3 viewing a *wall*-(6,4) observes a 3×3 set of projectors in each image; one could pan the camera to four horizontal and two vertical positions to obtain 8 different views of the display surface, each of which consists of a unique subset of projectors. Note that these views can be generated either from a single pan-tilt-zoom camera or from n_v fixed cameras. Our algorithm works with either scenario. We further define the term *cam-all* to represent a single, wide-angle camera view that can see the entire display area at once, i.e. *cam-all* = cam-$M\times M$, where $M=max(H,V)$. For example, the *cam-all* for a *wall*-(6,4) is *cam*-6×6.

2.1 Perspective Correction with 2-D Homographies

We assume that: the positions, orientations and optical parameters of the cameras and projectors are unknown; camera and projector optics can be modeled by perspective transforms; the projection surface is flat. Thus, the various transforms between screen, cameras, and projectors can all be modeled as 2-D planar homographies:

$$\begin{pmatrix} xw \\ yw \\ w \end{pmatrix} = \begin{pmatrix} h_1 h_2 h_3 \\ h_4 h_5 h_6 \\ h_7 h_8 h_9 \end{pmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}$$

where (x,y) and (X,Y) are corresponding points in two frames of reference, and $\boldsymbol{h}=(h_1,\ldots,h_9)^T$ (constrained by $|\boldsymbol{h}|=1$) 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 [10]. It is summarized below.

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

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

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

Our system employs this technique to compute two types of homographies: camera-to-camera and projector-to-camera. Each is described in greater detail below, and illustrated in Figure 2.

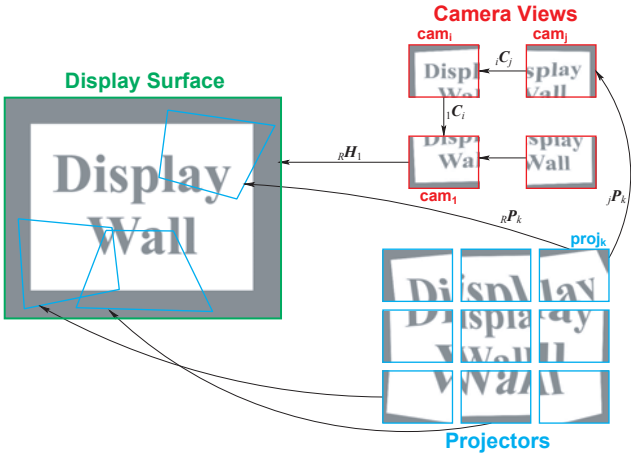


Figure 2. This diagram shows the relationship between the various homographies described in the text. Our system's goal is to recover the homography (${}_R\boldsymbol{P}_k$) mapping each projector k, to the global reference frame. Although ${}_R\boldsymbol{P}_k$ cannot directly be observed, it can be derived by composing ${}_j\boldsymbol{P}_k$, the homography between that projector and some camera view, and the chain of homographies connecting that view to the root of the homography tree. The geometric distortion of projected images is then corrected using a pre-warp of ${}_R\boldsymbol{P}_k^{-1}$.

First, camera-to-camera homographies capture the relationship between different camera views of the display surface. Although each view typically observes only a few projectors, the system combines these views to generate a reference frame for the entire display surface (see Section 2.3). Conceptually, this is equivalent to automatically building a panoramic mosaic from a set of photographs. One cannot directly compute a homography between two camera views that do not overlap since they share no point corre-

spondences. Therefore, our system builds a tree of homography relationships between adjacent views that spans the complete set of views; the mapping from any given view to the panoramic reference frame is determined by compounding the homographies along the path to the reference view at the root of the tree:

$$ {}_R\boldsymbol{H}_j = {}_R\boldsymbol{H}_1 \times_1 \boldsymbol{C}_i \times \ldots \times_i \boldsymbol{C}_j $$

where ${}_R\boldsymbol{H}_j$ is the homography mapping points from view j to the global reference frame, ${}_s\boldsymbol{C}_t$ are homographies connecting adjacent camera views and ${}_R\boldsymbol{H}_1$ maps the root camera view to the global reference frame.[3]

Second, the projector-to-camera homographies transform each projector's area of projection into some camera's coordinate system. These homographies are determined as follows. Each projector k displays calibration slides with highly-visible features, whose locations are known in projector coordinates. By observing the locations of these features in the camera image j, we can determine the relevant projector-to-camera homography ${}_j\boldsymbol{P}_k$. Since we know the mapping between any camera j and the reference frame, this enables us to compute ${}_R\boldsymbol{P}_k$, the transform from projector k to the reference frame:

$$ {}_R\boldsymbol{P}_k = {}_R\boldsymbol{H}_1 \times_1 \boldsymbol{C}_i \times \ldots \times_i \boldsymbol{C}_j \times_j \boldsymbol{P}_k $$

Note that ${}_R\boldsymbol{P}_k$ expresses the geometric distortion induced by the projector's oblique placement. This distortion is removed by prewarping each projector k's output by ${}_R\boldsymbol{P}_k^{-1}$.

2.2 Sub-pixel Accuracy Feature Detection

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 distortion model from [5]:

$$ x' = x + x[k_1 r^2 + k_2 r^4 + k_3 r^6] + [2p_1 xy + p_2(r^2 + 2x^2)] $$
$$ y' = y + y[k_1 r^2 + k_2 r^4 + k_3 r^6] + [2p_2 xy + p_1(r^2 + 2y^2)] $$

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. We have also developed a method to automatically correct the distortion along with feature extraction.

The feature detection component of our system displays a sequence of calibration slides on the projectors that are visible in each camera view. The goal is to reliably identify point features in adjacent camera views that correspond to the same location on the physical screen. 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. The intersections between these line sets can be determined with greater accuracy than standard corner detection. The details are described below.

For each camera view, horizontal lines are first displayed on a projector. Figure 3a shows an example of the captured image. Next, vertical lines are displayed on the same projector, as shown in Figure 3b. Note that these lines are "horizontal" and "vertical" only in the projector's own coordinate system. They may not be horizontal or vertical on the display surface nor in the camera image since the projector and camera orientations may be oblique. Each image is then processed as follows. 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 as blue dots in Figure 3c and 3d. The output of this procedure is a set of position estimates with floating-point precision along each visible line.

In the second step, the system determines the line equations that best fit the observed data in each camera image. Unfortunately, the observed lines are not precisely straight due to camera lens distortion. This is not easily corrected with offline calibration methods, because the camera distortion parameters change with zoom settings. This motivates the development of a novel online calibration method that combines distortion correction with line fitting. We use the sum of deviations of all points from the fitted line as the energy function. A non-linear optimizer is used to optimize the five-parameter distortion model to minimize the energy.

In the third step, the horizontal and vertical lines are intersected. This creates a set of accurate, stable point features for each camera view. A typical implementation employs calibration slides with five vertical and four horizontal lines, resulting in 20 point features per projector, as shown in Figure 3e. When the *cam*-2×2 configuration is used, features from four projectors are visible in each camera view, as shown in Figure 3f.

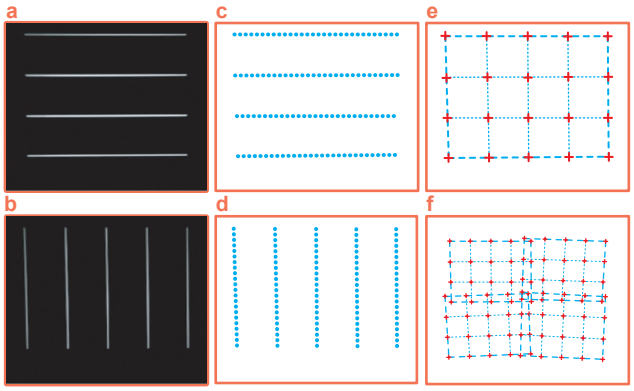


Figure 3. An example of the image processing and feature extraction procedure of our system.

These features are now used to compute the projector-to-camera and camera-to-camera homographies shown in Figure 2. Computing projector-to-camera homographies is straightforward since the locations of the 20 features are known *a priori* in the projector's coordinate frame. The camera-to-camera homographies are determined using all of the features that are visible in overlapping camera views. For instance, when the *cam*-2×2 configuration is used, two common projectors are visible in adjacent camera views; this means that 40 feature points are available for the camera-to-camera homography calculation. When two camera views share no common projector, the homography relating them must be obtained indirectly, using a chain of known homographies, as described in the following section.

2.3 Camera Homography Trees

To accurately register camera views, we introduce the concept of a *Camera Homography Tree*, and an algorithm for optimizing it.

We call $G(V,E)$ a Camera Homography Graph (CHG), where each vertex in V represents a camera view and an edge in E corresponds to a directly-computable homography between two views, i.e. an

[3] The transform ${}_R\boldsymbol{H}_1$ ensures that the global frame axes are aligned with the display surface rather than the root camera view; ${}_R\boldsymbol{H}_1$ is computed by observing four known features on the display surface from any view.

edge connects two vertices only if they share at least one common projector. For a rectangular display wall, a CHG usually looks like a lattice. Figure 4 shows a wall-(6,4) with cam-2×2; 15 views are available, forming a 5×3 lattice. Usually, a horizontal or vertical edge represents two strongly overlapping views, resulting in a better estimate of its homography; whereas a diagonal edge represents two weakly overlapping views, and thus a poorer estimate.

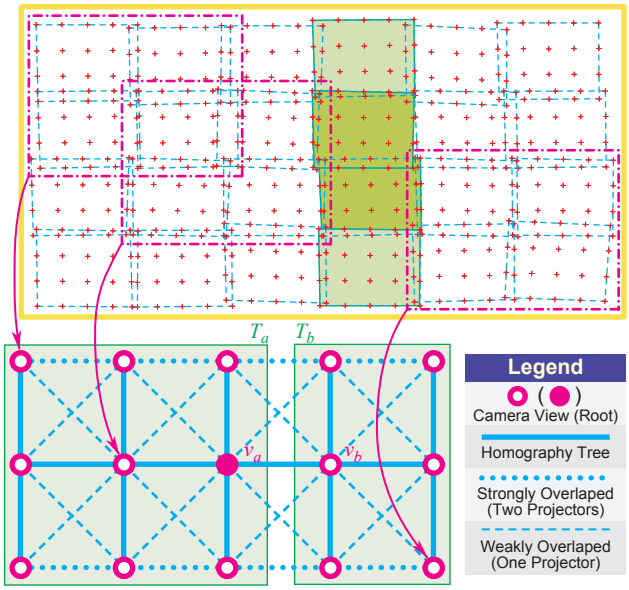


Figure 4. The Construction and Optimization of a Camera Homography Tree for *Wall*-(6,4) with *Cam*-2×2 Configuration.

As discussed in Section 2.1, as long as a CHG is connected, the homography between any two vertices can be computed by compounding a chain of homographies along a path connecting these two vertices. Ideally, the compound of homographies along any closed loop in an CHG should be identity; we call this graph a *Consistent CHG*. However, due to imperfection in the optics and limited resolution of the imaging device, this is usually not true; the result is an *Inconsistent CHG*. When multiple paths exist between two vertices, the calculated homography between these two vertices may depend on the choice of path. Clearly this needs to be remedied if we want to accurately register all camera views. Similar problems exist for 2-D image mosaicing. Shum *et al.* [9] and Kang *et al.* [6] have proposed methods for global registration. Their algorithms work on continuous-tone images, and have to extract features automatically. As mentioned before, we can detect features reliably with sub-pixel accuracy; this enables us to develop a novel algorithm that registers camera views precisely.

By definition, a tree is a connected graph without loops. Therefore, if a CHG is a tree, it is always consistent. Given a CHG $G(V,E)$, a *Camera Homography Tree* (CHT) is simply a spanning tree $T(G,E_T)$ of G, where $E_T \subseteq E$. In T, every pair of camera views is connected by a unique path. Although a CHT is consistent, it tends to be inaccurate when used directly – error in a single edge affects all homography paths containing that edge; also, being a subset of the original graph, only a portion of the feature correspondence information is utilized. We describe our method of constructing a initial CHT and optimizing the homographies along its edges to best represent the original CHG.

The goal of constructing the initial CHT is to minimize the path length from any vertex to the root, and also to minimize the path length between any adjacent camera views. To satisfy these criteria, we pick a vertex near the center of a CHG as the root, or reference view. A fishbone-shape tree is then constructed, with its "spine" aligned with the long side of the lattice, as shown in Figure 4. Each edge is initialized with the homography directly computed from common features visible in both camera views, as described in Section 2.1.

In the optimization stage, we iteratively refine the edges of a CHT to better represent the original CHG. In each iteration, the edges are updated in a bottom-up order. Each edge $e=(v_a,v_b) \in E_T$ forms a cut set of T – when removed, T becomes a forest of two trees: $T_a(G_a,E_a)$ and $T_b(G_b,E_b)$, where $v_a \in G_a$, $v_b \in G_b$, $G_a=G-G_b$, $E_a=E_T \cap (G_a \times G_a)$, and $E_b=E_T \cap (G_b \times G_b)$. An example of this is outlined in Figure 4. The initial homography along the edge (v_a,v_b) is computed with features from the fourth projectors in the second and third rows, as shown in a darker shade. To refine this homography, we treat T_a and T_b as two CHT's and map features in each tree to the views of v_a and v_b. This gives us more common features between v_a and v_b, so we can compute a better homography for e. In this example, features from the entire fourth column of projectors, i.e. projectors with both dark and light shades in Figure 4, contribute to the refined homography. This process is continued until the variance of multiple samples of each point feature in the root view is below a threshold. We found that stable homography estimates are obtained after a small number of iterations. Details are available in [2].

This algorithm enables us to create an effective virtual camera with very high resolution from multiple uncalibrated low resolution views. With these techniques, our system achieves scalable sub-pixel alignment on very large display walls, as described in Section 4.

3. EVALUATION METHODOLOGY

This section first proposes metrics for evaluating display wall alignment systems. It then introduces an automated vision-based system for measuring them. Finally, it details a simulator for evaluating the scalability of our algorithm on arbitrarily large display walls.

3.1 Metrics for Display Wall Alignment Systems

We use three metrics for evaluating the performance of a display wall alignment system: local alignment error, global alignment error, and running time.

Local alignment error quantifies the registration between adjacent projectors. Qualitatively, misalignment creates artifacts such as discontinuities and double-images in the overlap region (see Figure 5a). Quantitatively, the error can be characterized by the displacement between a point shown on one projector and the same point displayed by an adjacent projector. An appropriate measurement unit for this error is the average size of a pixel projected on the display wall. This unit is invariant to the physical dimensions of the display wall.

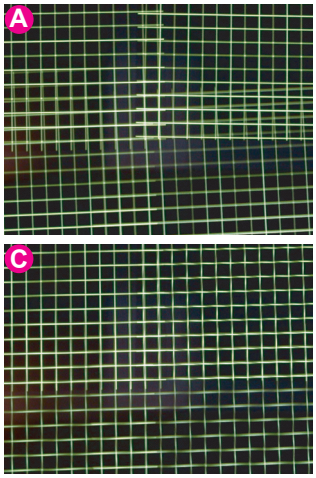
Let $\boldsymbol{H}_k={}_R\boldsymbol{P}_k^{-1}$ be the homography that maps point $\boldsymbol{p}=(x,y,1)^T$ from the display surface into projector k's reference frame. Let $\hat{\boldsymbol{H}}_k^{-1}$ be the alignment system's estimate for the inverse mapping. Due to alignment errors, $\hat{\boldsymbol{H}}_k^{-1}\boldsymbol{H}_k \neq \boldsymbol{I}$. In other words, when projector k attempts to illuminate point $\boldsymbol{p}$, it actually illuminates the point $\boldsymbol{p}_k=\hat{\boldsymbol{H}}_k^{-1}\boldsymbol{H}_k\boldsymbol{p}$. Let $\boldsymbol{\Omega}$ be the set of all features, and $\boldsymbol{\Phi}$ be the set of all projectors. We define local error to be:

$$E_l = \sum_{\forall \boldsymbol{p} \in \boldsymbol{\Omega}} \sum_{\forall (i,j) \in \boldsymbol{\Phi} \times \boldsymbol{\Phi}} I(i,\boldsymbol{p}) \cdot I(j,\boldsymbol{p}) \cdot \left\| \boldsymbol{p}_i - \boldsymbol{p}_j \right\|^2$$

where $I(i,\boldsymbol{p})$ is an indicator variable. $I(.) = 1$ if $\boldsymbol{p}$ falls within the display frustum of projector i and $I(.) = 0$ otherwise. This formula-

tion of the local error does not require knowledge of absolute points on the display wall, $\boldsymbol{p}$. It is sufficient to examine pairs of $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ and measure the relative distance between them. In the experiments described below, we obtain local error by displaying a grid pattern and measuring the projected discrepancy between grid points which are displayed by projectors in overlap regions.

Some alignment algorithms, such as SimAnneal explicitly observe point- and line-mismatches in the overlap regions and attempt to optimize pre-warp parameters to minimize this error. However, most algorithms, including ours, simply aim to register each projector to the global reference frame as accurately as possible, trusting that an accurate global registration will lead to small local errors.



a) The uncalibrated setup;
b) The result after single-view calibration: the average local error is 1.95 pixels;
c) Sub-pixel accuracy achieved using homography tree algorithm *cam*-2×2: the average local error is 0.55 pixel.

Figure 5. Zoomed views of alignment errors on a *wall*-(6,4).

Global alignment error is a metric for measuring the overall registration of a display wall. A projector array with excellent local alignment may still exhibit alignment errors for two reasons: (1) the projected image may be globally warped so that its edges are not parallel to the sides of the display surface; (2) small errors in local alignment can accumulate as homographies are chained, resulting in non-linear distortions in the projected image. We define global alignment error to be the displacement between pixels in the projected image and their desired locations, as measured in the reference frame:

$$E_g = \sum_{\forall \boldsymbol{p} \in \Omega} \sum_{\forall k \in \Phi} I(k,\boldsymbol{p}) \cdot \left\| \boldsymbol{p} - \boldsymbol{p}_k \right\|^2$$

This global error metric requires knowledge of the absolute locations of points on the display surface, thus making accurate measurements of global alignment quite difficult. We approximate the global error by measuring the nonlinearity of a regularly spaced grid patter. Fortunately, the human visual system is tolerant of slight global misalignments while being very sensitive to local discontinuities. In practice, once the projected display is roughly aligned to the global coordinate frame, local errors dominate the user's perception of the display wall.

To be of practical value, a alignment algorithm must be fast as well as accurate. There are two components of running time: the time taken to acquire images, and the time needed for computation, including image processing and calculating homographies. We present timing results comparing our system to existing approaches.

3.2 Automatic Measurement of Alignment Errors

Manual measurement of local and global alignment errors is a tedious, time-consuming process. Fortunately, we can employ the same camera hardware used for calibrating the display wall in evaluating its local alignment accuracy.

To measure local alignment error, each projector displays a set of calibration patterns. Unlike the patterns used during the calibration phase, these patterns are aligned (to the best of the system's ability) to the global reference frame. The camera captures detailed images of the seam regions where projection areas overlap. It records the displacement between a point displayed by one projector and the same global point displayed by other projectors at the seam. The average displacement over all seam regions on the display surface gives an estimate of local alignment error (in pixels).

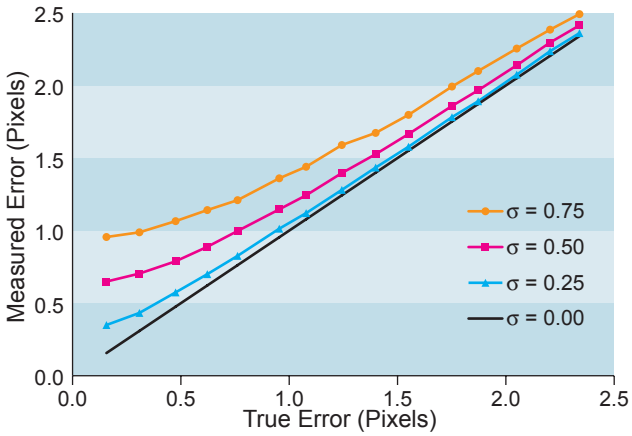


Figure 6. This graph plots the ground truth error (obtained from the simulator) against the observed error, as determined by our automated measurement system. σ is the standard deviation of the noise model for feature detection. From this graph, we see that our automated measurement system slightly (but consistently) overestimates the error of the display wall alignment systems.

In principle, one must be cautious about using the same camera hardware and similar image processing algorithms, both for aligning the display wall and measuring its alignment accuracy. For this reason, we simulated the automatic measurement system in the display wall simulator (see Section 3.3). In this series of tests, we assumed that the noise in feature detection could be modeled using a zero-mean Gaussian distribution, $N(0,\sigma)$. Figure 6 plots the actual local error (ground truth available to the simulator) against the measured local error, for a range of noise models. A noise-free measurement process would obviously generate a straight line $y=x$. We note that all of the curves lie above the $y=x$ line; this means that our automated measurement system is biased, and the bias is a consistent overestimate. Our experimental data indicates that our automated measurement system has $\sigma=0.5$ pixels in both x- and y-direction for each feature detected. Since we use four points for each of the 38 seams on *wall*-(6,4), the standard deviation on average local error estimates is $\tilde{\sigma}=0.5/(\sqrt{4\times38})=0.04$ and the 97% confidence interval is $\pm3\tilde{\sigma}=\pm0.12$ pixel.

3.3 Display Wall Simulator

To supplement our experiments on the 24-projector display wall we implemented a display wall simulator in Matlab. The primary function of the simulator is to investigate whether our camera homography tree algorithm scales well to very large format display walls, both in terms of alignment accuracy and running time. Additionally, the simulator may enable us to determine the components of a display wall system that are most likely to impact alignment accuracy. We broadly classify sources of alignment errors into four categories, each parametrized with one coefficient.

• **Projector optics**: We can expect errors to increase when the projectors' optics cannot be accurately modeled using a perspective projection model. We use the same five-parameter distortion model described in section 2.2 to simulate the projector optics. The *Projector Distortion Factor*, p, is coupled to both radial and tangential distortions, i.e. we define:

$$(k_1,k_2,k_3,p_1,p_2)=p\times(1.0,1.0,0.2,0.02,0.005)$$

The projectors in our system exhibit little distortion; we estimate $p=0.02$, which corresponds to an average warp of 0.32 pixels at the projected image's edge. In simulations we use p from 0.01 to 0.04.

• **Camera optics**: Although our cameras exhibit significant distortion, especially in the widest-zoom setting, the effective distortion is greatly reduced after either offline camera calibration or our automatic distortion correction technique. However, these methods can not fully rectify the images, we model the residual distortion with a *Camera Distortion Factor, c*:

$$(k_1,k_2,k_3,p_1,p_2)=c\times(1.0,1.0,0.2,0.02,0.005)$$

We estimate $c=0.05$ for our camera, which corresponds to an average warp of 0.23 pixels along the edge of the camera image. In the simulations, we use c from 0.00 to 0.05.

• **Image processing**: The simulator uses an abstract model for image processing. We assume that the system locates line features in the camera image, and that the position estimate for these features is corrupted with a zero-mean Gaussian noise. We define *Image Noise Factor, n* to be from 0.0 to 1.5, where $n=1.0$ corresponds to ± 0.5 pixel error, or $N(0,0.5)$.

• **Non-planar display surface**: Our alignment system assumes that all transforms can be modeled using 2-D planar homographies – an assumption that relies on a planar display surface. The simulator models the shape of the display screen as a Gaussian surface parametrized on a single variable, the *Screen Curvature Factor*, s, i.e. we define the screen as a surface:

$$f(x,y)=s\times 200\times\Psi(x,W_s)\times\Psi(y,H_s)$$

$$\Psi(x,a)=(e^{-(4x/a)^2}-e^{-4})/(1-e^{-4})$$

where W_s and H_s are the width and height of the screen, and all units are in mm. In the experiments, we use s from 0.0 to 0.2, where $s=0.1$ corresponds to a 20 mm central bulge, which equals to the real measurement in our 18'×8' rear-projection screen.

We present some of the simulation results in Section 4.3.2.

4. RESULTS

This section presents results from several series of experiments. Section 4.1 compares our approach to two existing display wall alignment algorithms. Section 4.2 examines how local error improves as the number of camera views is increased. Section 4.3 confirms that the camera homography tree algorithm remains accurate as the number of projectors in a display wall increases. These experiments on the 24-projector wall are further supported by simulation runs on very large format displays. Section 4.4 compares our algorithm's running time with existing approaches.

4.1 Comparisons with Existing Techniques

Two recent systems, Chen *et al.*'s SimAnneal [3] and Yang *et al.*'s PixelFlex [13] were selected as suitable benchmarks for display wall alignment. We were able to obtain an implementation of the former for evaluation purposes, and were able to re-implement relevant portions of the latter algorithm from published details. The PixelFlex algorithm utilizes only a single camera view covering the entire 18'×8' rear-projection screen in our setup.[4] The SimAnneal algorithm requires detailed views of inter-projector seams, and these images were collected automatically using a pan-tilt-zoom camera. The same camera control software (with different parameters) was used in our system to collect the *cam-N×N* views as input to the camera homography tree algorithm. Before presenting results, we briefly describe the image processing aspects for each algorithm.

SimAnneal uses a sequence of point and line feature correspondences wherever two or more projectors share a seam. The displacement between corresponding features displayed by different projectors provides an error metric that is minimized using simulated annealing. The implementation of SimAnneal we obtained assumes for its initial conditions that the homography between adjacent projectors can be adequately approximated by a simple translation and scale. This assumption is valid only when projectors are initially well-aligned; our uncalibrated setup, as seen in Figure 5a, exhibits significant error from rotation and perspective effects. As a result, SimAnneal performs poorly in our experiments, rarely achieving less than 12 pixels of local error after 500K iterations. For this reason, we report results from [3], where SimAnneal was evaluated on a much smaller *wall*-(4,2) display. We expect that with proper initialization, this value would represent a closer, but still optimistic estimate of SimAnneal's potential accuracy on a *wall*-(6,4).

In the PixelFlex system, each projector displays an array of circles with a Gaussian intensity distribution, and the centroids of the observed Gaussians in the camera image provide a sub-pixel-accurate estimate of the feature locations [13]. Our implementation of their algorithm required straightforward modifications to the *cam-all* version of our system: the main difference is the use of Gaussian rather than line features.

Table 1 summarizes the results of our experiments. In the single-view (*cam-all*) configuration with no homography trees, our system's accuracy is comparable to the existing systems. We notice that *cam-all* performs slightly better than PixelFlex. The reason is that, under perspective projection, lines are invariant but Gaussians undergo asymmetric distortion – possibly inducing a bias in PixelFlex's estimate of feature location. When our system is able to take advantage of multiple camera views, e.g. in the *cam*-2×2 configuration, the result improves dramatically – the average local error is reduced by half, as shown in Figure 5c.

Table 1. Alignment results of various algorithms on *wall-(6,4)*.

System	Number of Camera Views	Local Error Avg (Max)	Global Error Avg
SimAnneal	152	1.35 (N/A)	N/A
PixelFlex	1	1.73 (3.9)	1.5
Cam-all	1	1.19 (4.1)	1.3
Cam-2×2	15	0.55 (2.3)	1.8

4.2 Improving Accuracy with More Camera Views

This experiment investigates the trade-offs between effective camera resolution and homography chain length. Higher effective camera resolution, achieved using a tighter zoom, enables the system to locate features in the calibration slides with greater accuracy. On the other hand, the smaller field-of-view implies that more camera shots are needed to span the complete display wall. Figure 7 demonstrates that the camera homography tree algorithm does improve local accuracy. The single view approach, *cam-all*, exhibits slightly more than 1 pixel error on *wall*-(6,4). Additional camera views improve accuracy, enabling us to achieve sub-pixel error

[4] This camera configuration was identical to the *cam-all* configuration in our algorithm.

rates on a 6000×3000 display using only a 640×480 resolution camera. The best result is achieved by *cam*-2×2, which exhibits less than half the error of the best single camera view algorithm.

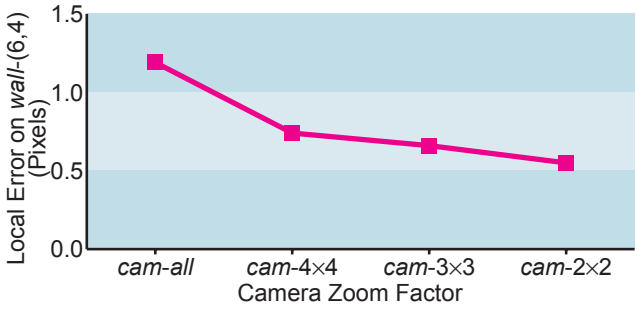


Figure 7. This graph shows how the camera homography tree algorithm improves average local errors. A single camera view (*cam-all*), even with super-resolved feature detection, is unable to achieve sub-pixel alignment accuracy on a *wall*-(6,4). These experiments were performed on our 24-projector display wall.

4.3 Scalability

The previous experiment showed that the camera homography tree algorithm significantly improves accuracy on our current display wall hardware. The following two experiments investigate whether the multi-view algorithm continues to outperform single-view alignment techniques across different scales of displays. The first shows that the observed behavior is also true for smaller displays and the second indicates that our algorithm scales to very large format displays consisting of hundreds of projectors.

4.3.1 Scalability Results on 24-projector Display Wall

Figure 8 investigates the camera homography tree algorithm's behavior on display walls of different sizes. These experiments were performed on the 24-projector wall, using several rectangular sub-arrays of projectors ranging from *wall*-(2,2) to the complete display. The results confirm that the multiple view approach to display wall alignment scales well with projector wall size.

We note that local error for *cam*-2×2 is higher than expected in the *wall*-(4,3) scenario. We believe that this may be due to screen curvature; initial simulation results support our hypothesis. We are conducting additional experiments to examine the issue.

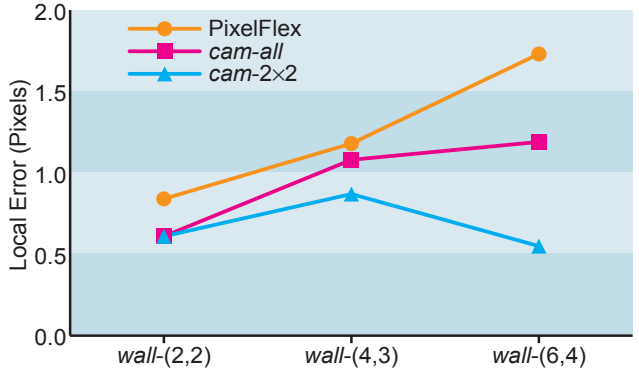


Figure 8. This graph shows how the average local error for different configurations of the camera homography tree algorithm scale with display wall size. Note that we achieve sub-pixel error rates even on the largest display wall. These experiments were performed on our 24-projector display wall.

4.3.2 Scalability Results on Display Wall Simulator

To evaluate our algorithm's performance on very large displays, we run the simulator on the following display walls: *wall*-(*H*,*V*), where $(H,V) \in \{(2,2), (3,2), (4,3), (6,4), (9,6), (12,8), (18,12), (24,16)\}$. Figure 9 shows the expected local error versus the total number of projectors in a display. Each curve in the graph corresponds to a choice of *cam*-*N*×*N*, where $N \in \{2,3,4,6,9,12,18,24\}$. There is no benefit to using a wider field-of-view camera than necessary on a given display wall. Therefore, each *cam*-*N*×*N* curve starts at the largest display wall that can be completely seen within a single view. By definition, the curve *cam-all* simply connects the starting data point of each *cam*-*N*×*N* curve. Simulation parameters were selected to be similar to the current physical setup: $p=0.02$, $s=0.1$, $c=0.05$, $n=1.0$ and each data point was generated by averaging the results of five runs.

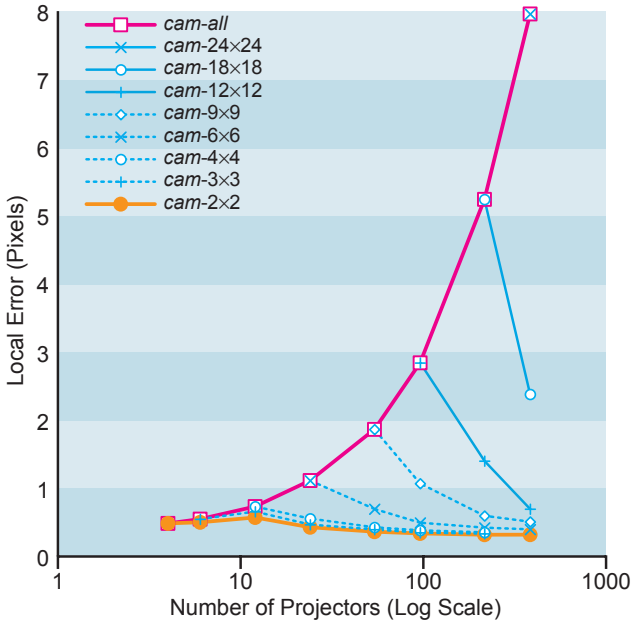


Figure 9. Semi-log plot of local alignment error for simulated display walls of various sizes. The horizontal axis gives the number of projectors comprising the display, i.e. h×v for *Wall*-(h,v). It shows that single-view alignment algorithms fail to achieve satisfactory accuracy, whereas the camera homography tree algorithm scales well.

As Figure 9 shows, the simulation data on smaller display walls is consistent the experimental evidence presented before. This graph also shows simulations extended to very large scale displays. We make the following observations about Figure 9. First, the alignment error for a single-view algorithm (*cam-all*) grows almost linearly as projectors are added. Second, the curves for the tightly-zoomed camera configurations (e.g. cam-2×2) are almost flat. This validates our earlier claim that the camera homography tree algorithm scales well with display wall size. Third, note that for a particular *cam-n*×*n* curve, the local error decreases as the number of projectors increases. This is because, on a larger display, each projector appears in more camera views. Our homography tree algorithm is able to utilize multiple views of point features to better refine the homographies. This results in improved accuracy. Finally, one can derive significant benefits from the camera homography tree algorithm even by chaining together a small number of views. For instance, on *wall*-(24,16), going from *cam*-24×24 (i.e. *cam-all*) to *cam*-18×18 cuts the local alignment error from almost 8 pixels to about 2.5 pixels.

4.4 Running Time

As mentioned earlier, there are two major components of the running time: the time required to collect the necessary images; and the time needed to process these images and calculate the homographies used to align the projectors. Table 2 presents the timing results. Our system is implemented in Matlab 6.0 with a moderate amount of optimization. Since our implementation of the PixelFlex algorithm uses the same code base, its running time is almost identical to *cam-all*. Therefore, it is not listed. It is clear that our system is fast: we can align *wall*-(6,4) in under 9 minutes.

Table 2. Comparison of running time (minutes) between SimAnneal and our system. *Data* column lists the time taken to collect images; *Comp* column is the computation time needed to calculate homographies from the data. SimAnneal was not evaluated on *wall*-(4,3); timing information reported for *wall*-(4,2) in [3] was used. Our system is faster by more than an order of magnitude on every display walls.

Display Size	SimAnneal			Our System		
	Setup (# steps)	Data (min)	Comp (min)	Setup	Data (min)	Comp (min)
wall-(2,2)	10k	10.0	15.2	*cam-all*	0.13	0.17
wall-(4,3)	20k	33.0	34.5	*cam-all*	0.53	0.25
				cam-2×2	2.00	0.92
wall-(6,4)	50k	90.0	95.5	*cam-all*	1.06	0.42
				cam-4×4	2.52	0.92
				cam-3×3	4.30	1.67
				cam-2×2	5.90	2.50

5. CONCLUSION

Multi-projector display systems have become increasingly popular for a wide variety of applications. As display walls incorporate more projectors, it becomes necessary to develop automated approaches to help with designing, building and maintaining these systems. One key aspect to scaling tiled displays in both size and number is having a good automated alignment system. This paper describes a practical vision-based system for automatically calibrating such large format multi-projector displays. It incorporates an automated sub-pixel accuracy feature detection algorithm for tiled displays that simultaneously calibrates intrinsic camera parameters. It also includes an automatic vision-based system for measuring display wall alignment accuracy. A comprehensive series of experimental tests on a 24-projector wall demonstrate that our camera homography tree algorithm significantly improves local alignment accuracy by incorporating information from multiple, uncalibrated camera views. Our algorithm's accuracy exceeds that of existing solutions, and unlike those approaches, scales better (in both accuracy and speed) as projectors are added to the display system.

The system also includes a simulator tool. It helps system designers examine the impact of design decisions on the expected accuracy of a display wall. We have run many simulation tests, and the results (validated on real data) indicate that our approach is practical even for very large format displays. These simulation results would help a designer determine the quality and number of cameras needed, and the time necessary for aligning a display wall.

Our system is now in regular use at the Princeton Scalable Display Wall. The size of this display has warranted a new class of scalable alignment solutions, such as the one described here. We anticipate such solutions will increasingly be required by future displays.

6. ACKNOWLEDGEMENTS

The Princeton Scalable Display Wall project is supported in part by Department of Energy grant DE-FC02-99ER25387, by NSF Infastructure Grant EIA-0101247, by NSF Next Generation Software Grant ANI-9906704, by NCSA Grant ACI-9619019 (through NSF), by Intel Research Council, and by Intel Technology 2000 equipment grant. Thanks to Tat-Jen Cham, Gita Sukthankar, and Mark Ashdown for valuable feedback on the paper. Also, thanks for all the comments by the anonymous reviewers.

References

[1] D.Brown. Lens Distortion for Close-Range Photogrammetry. *Photometric Engineering*, 37(8), 1971.

[2] H.Chen, R.Sukthankar, G.Wallace, and T.-J.Cham. Accurate Calculation of Camera Homography Trees for Calibration of Scalable Multi-Projector Displays. Technical Report TR-639-01, Princeton University, September 2001.

[3] Y.Chen, D.Clark, A.Finkelstein, T.Housel, and K.Li. Automatic Alignment of High-Resolution Multi-Projector Display Using an Uncalibrated Camera. In *Proceedings of IEEE Visualization*, 2000.

[4] T.Funkhouser and K.Li. Large Format Displays. *Computer Graphics and Applications*, 20(4), 2000. (Guest editor introduction to special issue).

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

[6] E.Kang, I.Cohen and G.Medioni. A Graph-Based Global Registration for 2D Mosaics. In *Proceedings of International Conference on Pattern Recognition*, 2000.

[7] K.Li, H.Chen, Y.Chen, D.Clark, P.Cook, S.Daminakis, G.Essl, A.Finkelstein, T.Funkhouser, A.Klein, Z.Liu, E.Praun, R.Samanta, B.Shedd, J.Singh, G.Tzanetakis, and J.Zheng. Building and Using a Scalable Display Wall System. *Computer Graphics and Applications*, 20(4), 2000.

[8] R.Raskar, M.Brown, R.Yang, W.Chen, G.Welch, H.Towles, B.Seales, and H.Fuchs. Multi-Projector Displays using Camera-Based Registration. In *Proceedings of IEEE Visualization*, 1999.

[9] H.Shum and R.Szeliski. Panoramic Image Mosaics. Technical Report MSR-TR-97-23, Microsoft Research, 1997.

[10]R.Sukthankar, R.Stockton, and M.Mullin. Smarter Presentations: Exploiting Homography in Camera-Projector Systems. In *Proceedings of International Conference on Computer Vision*, 2001.

[11]R.Surati. *A Scalable Self-Calibrating Technology for Seamless Large-Scale Displays*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1999.

[12]R.Tsai. A Versatile Camera Calibration Technique for High Accuracy 3D Machine Vision Metrology using Off-the-shelf TV Cameras and Lenses. *IEEE Journal of Robotics and Automation*, RA-3(4), 1987.

[13]R.Yang, D.Gotz, J.Hensley, H.Towles, and M.Brown. PixelFlex: A Reconfigurable Multi-Projector Display System. In *Proceedings of IEEE Visualization*, 2001.