A Distributed Cooperative Framework for Continuous Multi-Projector Pose Estimation

Tyler Johnson, Greg Welch, Henry Fuchs, Eric La Force and Herman Towles*

University of North Carolina at Chapel Hill

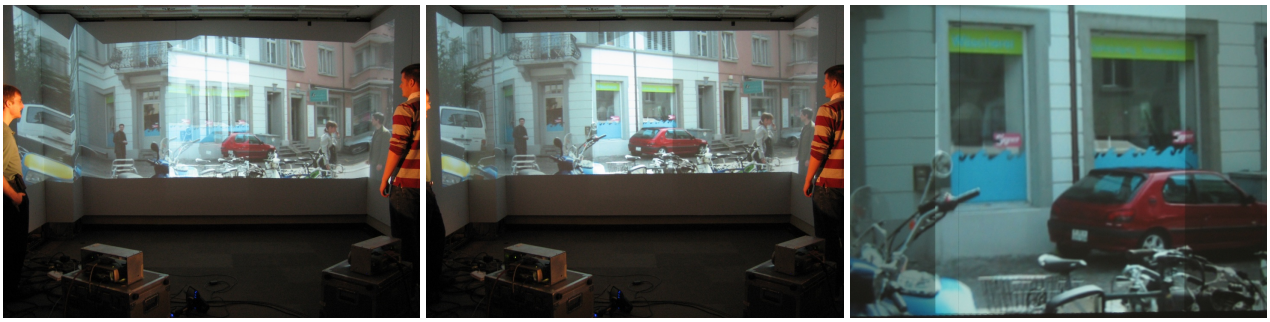

Figure 1: A two-projector display after both projectors have been moved (left) and after roughly ten seconds of operation (middle,right).

ABSTRACT

We present a novel calibration framework for multi-projector displays that achieves continuous geometric calibration by estimating and refining the poses of all projectors in an ongoing fashion during actual display use. Our framework provides scalability by operating as a distributed system of "intelligent" projector units: projectors augmented with rigidly-mounted cameras, and paired with dedicated computers. Each unit interacts asynchronously with its peers, leveraging their combined computational power to cooperatively estimate the poses of all of the projectors. In cases where the projection surface is static, our system is able to continuously refine all of the projector poses, even when they change simultaneously.

Keywords: Projector displays, continuous calibration.

Index Terms: I.3.3 [Computer Graphics]: Picture/Image Generation—Display Algorithms; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual Reality;

1 INTRODUCTION

Projection-based displays have long been used in the creation of large, immersive environments for virtual reality (VR), simulation, and training. These displays have become increasingly useful due to advancements in automatic calibration that allow the images of multiple projectors to be registered together accurately on complex display surfaces, while simultaneously compensating for the surface shape. While these techniques have been shown to be accurate and robust, the geometric aspects of the calibration are typically only computed *prior* to display use. However even for "fixed" configurations of projectors, physical perturbations, electrical changes, and even gravity can cause the apparent or actual projector poses to change over time, decreasing display quality.

Our goal is a system that continually and automatically adjusts to even the slightest change in projector poses, while the system is in use. Such continuous pose estimation should increase the robustness of projection-based displays, while also offering new flexibility. For example, the most suitable positioning of projectors may vary between applications due to certain field-of-view or spatial resolution requirements. In these situations it might be desirable to deliberately reposition the projectors without having to interrupt display use to perform a re- calibration of the entire system.

In this paper, we present a novel technique allowing the poses of multiple projectors to be estimated continuously during actual display use. Our framework is designed around "intelligent" projector units (IPUs): a projector augmented with two rigidly-mounted cameras, and paired with a dedicated computer. (See Figure 2.) The IPUs operate in a distributed fashion, each cooperating with its neighbors to continuously estimate all of the poses. In cases where the projection surface is static, our system continuously refines all of the poses, even when they change (IPUs move) simultaneously.

2 RELATED WORK

Our work is related to a number of approaches that have been developed to deal with various aspects of continuous calibration in projection-based displays. Yang and Welch [18] describe how application imagery projected at display time can be used to automatically estimate the shape of the display surface and account for changes in its shape over time using a Kalman filter-based framework. Cotting et al. [4] describe how the shape of the display surface can be estimated over time by embedding imperceptible calibration patterns into projected imagery.

Raij and Pollefeys [14] and Raskar et al. [15] describe techniques to automatically calibrate clusters of projectors displaying onto planar display surfaces. PixelFlex [17, 13] provided a tiled display system for planar and near-planar surfaces that allowed each projectors image to be easily and quickly repositioned to create new display configurations that could be calibrated within minutes. Bhasker et al. [2] describes a distributed approach to automatic calibration of multi-projector tiled displays that is truly scalable.

Our work is most closely related to techniques that use camera image measurements to recalibrate projectors at display time without interrupting the projection of user imagery. In [5], Cotting et al. describe how their imperceptible pattern embedding approach

*tmjohns, welch, fuchs, elaforc, herman@cs.unc.edu

can be used to recalibrate projectors in a multi-projector display that have been moved. Johnson and Fuchs [8] provide a method for continuously estimating the pose of a single projector using only application image features, while Zhou et al. [19] describe how application image features can be used to continuously calibrate projectors in a multi-projector display. In [20], Zollman et al. present a hybrid technique that can compensate for small changes in display configuration using optical flow, and will resort to active structured light projection when the optical flow becomes unreliable.

3 System Design and Goals

In this section, we describe the philosophy behind the design of our system, and describe how the use of "intelligent" projector units (projectors augmented with two rigidly-mounted cameras, and paired with dedicated computers) supports our design goals.

3.1 Rapid Projector Recalibration During Display Use

The primary goal of our overall work is to support rapid set-up and reconfiguration of multi-projector displays. In this paper, we focus on the goal of achieving rapid projector recalibration during actual display use by continuously estimating the poses of all projectors.

In order to estimate changes in projector pose over time, the use of an auxiliary measurement device is required. We have chosen to use cameras for this purpose due to the inherent duality between cameras and projectors—both use lenses to direct light, but one for the purpose of sensing and the other for the purpose of display. The utility of such combinations of projectors and cameras is described in a variety of previous work [2, 15, 4, 19].

We have designed our system to be flexible enough to recover the poses of all projectors even in the case that all projectors have been moved. We believe this functionality is necessary because even in displays where projectors are not intentionally moved, the poses of all projectors are likely to change slightly over time. Previous work [19, 5] solves this problem only partially by using projector-camera pairs with known calibration as references in re-estimating the poses of other projector-camera pairs that have been moved. This approach fails when all projector-cameras pairs have been moved, and it is not clear how errors may accumulate over time as devices are moved, recalibrated, and then used to recalibrate other devices.

In order to achieve the goal of allowing any and all projectors to be moved simultaneously, we introduce the concept of using known display surface geometry as an additional reference in estimating projector pose. This allows us to eliminate the distinction between calibrated and uncalibrated projectors and instead refine the calibration of all projectors continuously over time. In our current system we assume that the geometry of the entire display surface is known apriori and does not change, but plan to relax these constraints in future work.

Another important goal of our design is to allow continuous projector pose estimation to take place without affecting the imagery projected by the user. While techniques for embedding imperceptible patterns [4, 5, 6] into projected imagery ensure a steady supply of features that can be used as camera image measurements, these techniques are currently limited in the types of projectors that can be used, i.e. DLP or stereo, and the embedding process requires that some amount of image quality be sacrificed. For this reason, we have chosen to use the projected imagery itself as a source of camera image features that can be used to estimate projector pose during display use.

3.2 Distributed Cooperative Operation

In order to be useful in large displays with many projectors, continuous calibration approaches must be scalable. As described in [2] a distributed calibration methodology has far greater potential for scalability and fault tolerance than more traditional centralized approaches where all calibration data is aggregated and processed on a single machine.

Our design allows for scalability by pairing computation with each projector to create a number of self-contained, intelligent units that act in a cooperative fashion, leveraging their combined computational power to estimate the pose of each projector through the exchange of information over a local network.

3.3 Intelligent Projector Units

Intelligent Projector Units (IPUs) are the basic building blocks of our display. An IPU, as seen in Figure 2, consists of a projector augmented with rigidly-mounted cameras and computation that includes network capability. While the computation component currently consists of a separate PC, we have future plans to fully integrate the computation with the rest of the unit.

Figure 2: An Intelligent Projector Unit.

4 Distributed Computational Framework

In this section, we describe our distributed computational framework for continuous calibration of projector pose in multi-projector displays.

4.1 Assumptions

1. The internal calibration of each IPU is fixed and known. (The internal calibration consists of the intrinsic parameters of the projector and both cameras, such as focal lengths and principal points, as well as the relative positions and orientations of all three devices.)

2. The geometry of the display surface is static and known.

3. Projectors remain mostly stationary, however they may drift over time or occasionally be moved by the user.

4.2 General Approach

In the distributed computational framework we develop here, each IPU is tasked with the responsibility of estimating its own pose. In general, the pose of a camera or projector has six degrees of freedom that correspond to its position and orientation. The three position parameters x, y, z represent the device's center-of-projection, while the three rotational parameters ψ, θ, ϕ represent the orientation of its principal axis.

When the internal calibration of an IPU is known (assumption 1 in Section 4.1) knowledge of the pose of any *one* of the optical devices (projector or camera) that are part of the IPU is sufficient

to completely constrain the pose of all three devices. Taking advantage of this property, we arbitrarily choose one of the two cameras of an IPU to be its *primary* camera, whose pose will be continuously estimated, and designate the other its *secondary* camera. We define the pose of an IPU to be equivalent to the pose of its primary camera. The pose estimate of the IPU's primary camera can then be transformed into a pose estimate of its projector, which is needed to warp its projected imagery to register it to the other projectors and compensate for the shape of the display surface.

In our framework, each IPU estimates the pose of its primary camera using image (feature) correspondences between cameras. An image correspondence between two cameras consists of a pixel location in the image of each device that both correspond to the same 3D point in the scene. In general, image correspondences between cameras are a function of the intrinsic and extrinsic calibration of both devices and the geometry of the scene that is observed.

4.2.1 Local and Remote Correspondences

In continuously estimating the pose of its primary camera, each IPU makes use of two types of image correspondences. The first type consists of correspondences between its primary and secondary cameras. We refer to these as *local correspondences* since each IPU can obtain these correspondences independently of the other IPUs. The second type of correspondence used in our system is between an IPU's primary camera and the primary cameras of other IPUs. We refer to these as *remote correspondences*.

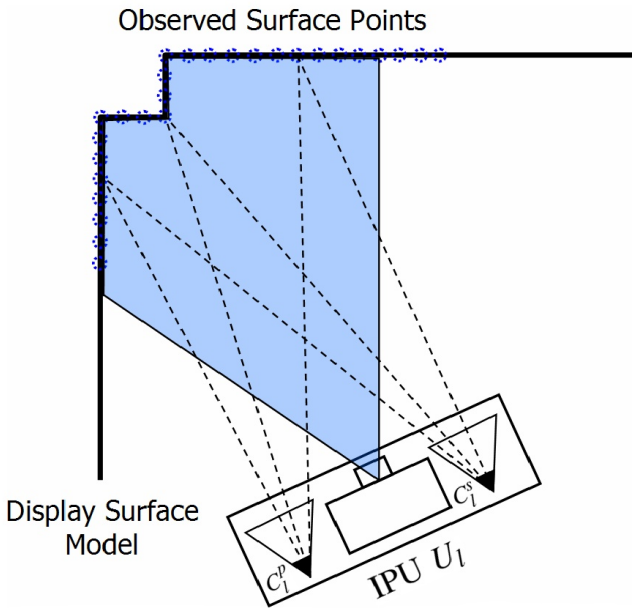

Figure 3: The pose of an IPU U_l is constrained by image correspondences between its primary and secondary cameras C_l^p and C_l^s by forcing the structure of the display surface that is currently observed (blue points) to coincide with the known display surface model.

Both local and remote correspondences produce constraints on an IPU's pose. As seen in Figure 3, local correspondences constrain the structure of the display surface that is currently observed by an IPU since its primary and secondary cameras are calibrated as a stereo camera pair. The pose of the IPU is then constrained by requiring that the currently observed surface geometry (shown as blue points) match the known model of the display surface.

Remote correspondences also provide constraints on the pose of an IPU. In this case, correspondences are measured between the primary camera C_l^p of an IPU U_l and the primary camera C_r^p of another IPU U_r. To the extent that the pose of C_r^p is known, it can act a reference in computing the pose of C_l^p as illustrated in Figure 4. Using the estimated pose of C_r^p, each correspondence can be back-projected into a ray that, when intersected with the known display surface model, produces a point on the surface. The resulting set of surface points and their measured positions in C_l^p's image can be used to estimate the pose of C_l^p to the extent that the pose estimate of C_r^p is accurate.

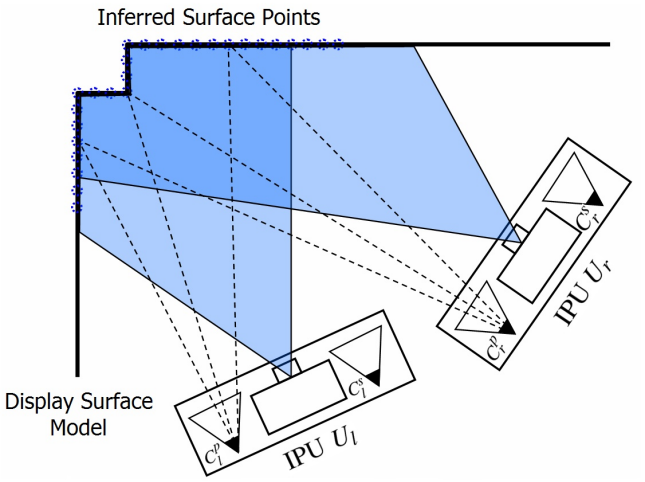

Figure 4: The pose estimate of IPU U_r is used as a reference in estimating the pose of IPU U_l via image correspondences between their primary cameras.

4.3 Kalman filter-Based Estimation

While various geometric algorithms could be used to estimate the pose of an IPU using local and remote correspondences, we choose to use a Kalman filter [9] for this purpose. There are several advantages to this approach. First, temporal filtering allows the effects of measurement noise on pose estimates to be mitigated. Without temporal filtering, measurement noise can cause small variations in the estimated pose over time, ultimately resulting in small changes in the projected imagery that are quite distracting to the viewer. Second, with the Kalman filter it is straightforward to account for uncertainty in the pose estimates of other IPUs for which remote correspondences have been measured. This is due to the fact that in addition to estimating the state of the process, the Kalman filter also estimates the state error covariance—an indication of uncertainty in the state estimate due to the failure of measurements to fully constrain a solution.

4.3.1 Filter Operation

In our system, each IPU maintains a Kalman filter that processes the local and remote correspondences obtained at each time-step and produces a filtered pose estimate. Because perspective projection is non-linear, we use an *extended* Kalman filter.

In what follows, superscripts p and s are used to differentiate primary and secondary cameras, while subscripts l and r are used to denote *local* and *remote*. Since all IPUs operate identically, consider an arbitrary IPU U_l with primary and secondary cameras C_l^p and C_l^s. Let x_l and P_l be the pose and error covariance estimates of C_l^p and let local correspondences be denoted as $z_l^p \Leftrightarrow z_l^s$, where z_l^p is measured in C_l^p and z_l^s is measured in C_l^s. Additionally, let $U_{r_1}, U_{r_2}, ...U_{r_n}$ be the set of IPUs with primary cameras $C_{r_1}^p, C_{r_2}^p, ..., C_{r_n}^p$, for which remote correspondences have been measured, and let their respective pose estimates and error covariances

be $x_{r_1}, x_{r_2}, ..., x_{r_n}$ and $P_{r_1}, P_{r_2}, ..., P_{r_n}$. Finally, we will denote remote correspondences as $z^p_{l,r_i} \Leftrightarrow z^p_{r_i,l}$, where z^p_{l,r_i} is a set of feature measurements in local primary camera C^p_l that correspond to a set of feature measurements $z^p_{r_i,l}$ in remote primary camera $C^p_{r_i}$.

The state vector $\widehat{X}_k$ that is estimated by our Kalman filter at each timestep k aggregates the pose of C^p_l and the poses of the $C^p_{r_1}, C^p_{r_2}, ..., C^p_{r_n}$

$$\widehat{X}_k = \begin{bmatrix} x_l \\ x_{r_1} \\ x_{r_2} \\ \vdots \\ x_{r_n} \end{bmatrix}. \quad (1)$$

Formulating the state vector in this way allows the filter to take into account uncertainty in the poses of the $C^p_{r_1}, C^p_{r_2}, ..., C^p_{r_n}$ since their individual error covariances appear in the overall error covariance $\widehat{P}_k$ estimated by the filter

$$\widehat{P}_k = \begin{bmatrix} P_l & P_{r_1,l} & P_{r_2,l} & \dots & P_{r_n,l} \\ P_{l,r_1} & P_{r_1} & 0 & \dots & 0 \\ P_{l,r_2} & 0 & P_{r_2} & & \vdots \\ \vdots & \vdots & & \ddots & 0 \\ P_{l,r_n} & 0 & \dots & 0 & P_{r_n} \end{bmatrix}. \quad (2)$$

The individual error covariances of C^p_l and the $C^p_{r_1}, C^p_{r_2}, ..., C^p_{r_n}$ lie on the main diagonal while the covariances relating C^p_l to the $C^p_{r_1}, C^p_{r_2}, ..., C^p_{r_n}$ lie in the first row and column. The 0s everywhere else result from the property that at IPU U_l, there are no measurements relating the poses of the $C^p_{r_1}, C^p_{r_2}, ..., C^p_{r_n}$ to one another.

A Kalman filter acts as a predictor-corrector. At each time step the filter employs a *time update* and *measurement update*. These are described in the following sections.

Time Update

The time update phase is responsible for propagating the filter state $\widehat{X}_{k-1}$ and error covariance $\widehat{P}_{k-1}$ forward in time from the previous step, to produce *apriori* state and error covariance estimates $\widehat{X}^-_k$ and $\widehat{P}^-_k$ at time k. Because we assume the IPUs are primarily stationary (assumption 3 in Section 4.1) we employ "constant" motion models, with the following corresponding time update equations:

$$\begin{aligned} \widehat{X}^-_k &= \widehat{X}_{k-1} \\ \widehat{P}^-_k &= \widehat{P}_{k-1} + \widehat{Q}_k. \end{aligned}$$

The term $\widehat{Q}_k$ added to $\widehat{P}_{k-1}$ models random variations *between* filter updates. We allow a different process noise matrix for each of the $x_l, x_{r_1}, x_{r_2}, ..., x_{r_n}$, but assume no correlation between them. In our framework, each IPU estimates its own process noise covariance Q_l using the technique described in [11].

$$\widehat{Q}_k = \begin{bmatrix} Q_l & 0 & 0 & \dots & 0 \\ 0 & Q_{r_1} & 0 & \dots & 0 \\ 0 & 0 & Q_{r_2} & & \vdots \\ \vdots & \vdots & & \ddots & 0 \\ 0 & 0 & \dots & 0 & Q_{r_n} \end{bmatrix}. \quad (3)$$

Measurement Update

In the measurement update phase, the measurements $\widehat{Z}_k$ at time k are used in conjunction with a set of measurement predictions $\widetilde{Z}_k$ to correct the apriori state $\widehat{X}^-_k$ and error covariance $\widehat{P}^-_k$ estimates into *aposteriori* state $\widehat{X}_k$ and error covariance estimates $\widehat{P}_k$. The measurement update equations also require a measurement noise covariance matrix R and a jacobian matrix $\widehat{H}_k$ that indicates the sensitivity of the measurements to changes in the state parameters.

The measurement update equations that are used are

$$\begin{aligned} K_k &= \widehat{P}^-_k \widehat{H}^T_k \left(\widehat{H}_k \widehat{P}^-_k \widehat{H}^T_k + R \right)^{-1} \\ \widehat{X}_k &= \widehat{X}^-_k + K_k \left(\widehat{Z}_k - \widetilde{Z}_k \right) \\ \widehat{P}_k &= \left(I - K_k \widehat{H}_k \right) \widehat{P}^-_k. \end{aligned}$$

Using the jacobian $\widehat{H}_k$, the measurement covariance R, and the apriori error covariance $\widehat{P}^-_k$, the Kalman gain K_k is computed. The Kalman gain is used to weight the measurement residual between $\widehat{Z}_k$ and $\widetilde{Z}_k$ to produce a correction to the apriori state estimate.

In our filter, the measurement vector $\widehat{Z}_k$ aggregates all measurements in C^p_l into a single measurement vector

$$\widehat{Z}_k = \begin{bmatrix} z^p_l \\ z^p_{l,r_1} \\ z^p_{l,r_2} \\ \vdots \\ z^p_{l,r_n} \end{bmatrix}. \quad (4)$$

The measurement prediction $\widetilde{Z}_k$ is generated by a mathematical function that predicts the measured values based on the current apriori state estimate. In the case of local correspondences where z^p_l corresponds to z^s_l, we use the function h_l to produce a prediction $\widetilde{z}^p_l$ of z^p_l using the following parameters

$$\widetilde{z}^p_l = h_l \left(x_l; z^s_l, \kappa^p_l, \kappa^s_l, \Delta_I, S \right), \quad (5)$$

where x_l is the pose of C^p_l, κ^p_l and κ^s_l are the intrinsics of C^p_l and C^s_l, Δ_l is the coordinate transformation between C^p_l and C^s_l, and S is the shape of the surface.

The function h_l performs the following operation. The pose x_l of C^p_l is used in addition to κ^p_l, κ^s_l, and Δ_l to produce projection matrices for C^p_l and C^s_l. Each of the z^s_l is back-projected into a ray using the projection matrix of C^s_l, and these rays are intersected with the surface S to produce a set of surface points. The projection matrix of C^p_l is then applied to each of these surface points to produce $\widetilde{z}^p_l$.

In the case of remote correspondences, where z^p_{l,r_i} corresponds to $z^p_{r_i,l}$ for remote IPU U_{r_i}, we use the measurement function h_{r_i} to produce $\widetilde{z}^p_{l,r_i}$, the prediction of z^p_{l,r_i},

$$\widetilde{z}^p_{l,r_i} = h_{r_i} \left(x_l, x_{r_i}; z^p_{r_i,l}, \kappa^p_l, \kappa^p_{r_i}, S \right), \quad (6)$$

where x_l is the pose of C^p_l and x_{r_i} is the pose of $C^p_{r_i}$, κ^p_l and $\kappa^p_{r_i}$ are the intrinsics of C^p_l and $C^p_{r_i}$ and S is the shape of the surface.

The operation of h_{r_i} is analogous to that of h_l. The poses x_l and x_{r_i} of C^p_l and $C^p_{r_i}$ are used in conjunction with κ^p_l and $\kappa^p_{r_i}$ to produce projection matrices for C^p_l and $C^p_{r_i}$. Each of the $z_{r_i,l}$ is back-projected into a ray using the projection matrix of $C^p_{r_i}$, and each of these rays is intersected with the surface S to produce a set of

surface points. The projection matrix of C_l^p is then applied to each of these surface points to produce $\widetilde{z}_{l,r_i}^p$.

The vector $\widetilde{Z}_k$ is then

$$\widetilde{Z}_k = \begin{bmatrix} h_l\left(x_l^-; z_l^s, \ldots\right) \\ h_{r_1}\left(x_l^-, x_{r_1}^-; z_{r_1,l}^p, \ldots\right) \\ h_{r_2}\left(x_l^-, x_{r_2}^-; z_{r_2,l}^p, \ldots\right) \\ \vdots \\ h_{r_n}\left(x_l^-, x_{r_n}^-; z_{r_n,l}^p, \ldots\right) \end{bmatrix}. \quad (7)$$

The jacobian matrix $\widehat{H}_k$ indicates the sensitivity of the measurements to changes in the state parameters. Since $\widehat{Z}_k$ is split between local and remote correspondences, so too is $\widehat{H}_k$,

$$\widehat{H}_k = \begin{bmatrix} \frac{\partial h_l(x_l^-; z_l^s, \ldots)}{\partial \widehat{X}} \\ \frac{\partial h_{r_1}(x_l^-, x_{r_1}^-; z_{r_1,l}^p, \ldots)}{\partial \widehat{X}} \\ \frac{\partial h_{r_2}(x_l^-, x_{r_2}^-; z_{r_2,l}^p, \ldots)}{\partial \widehat{X}} \\ \vdots \\ \frac{\partial h_{r_n}(x_l^-, x_{r_n}^-; z_{r_n,l}^p, \ldots)}{\partial \widehat{X}} \end{bmatrix}, \quad (8)$$

Due to the dependence of h_l on only C_l^p's pose and the dependence of h_{r_i} on only the poses of C_l^p and $C_{r_i}^p$, $\widehat{H}_k$ has the following block structure

$$\widehat{H}_k = \begin{bmatrix} H_l & 0 & 0 & \ldots & 0 \\ H_{l,r_1} & H_{r_1,l} & 0 & \ldots & 0 \\ H_{l,r_2} & 0 & H_{r_2,l} & & \vdots \\ \vdots & \vdots & & \ddots & 0 \\ H_{l,r_n} & 0 & \ldots & 0 & H_{r_n,l} \end{bmatrix}. \quad (9)$$

where

$$H_l = \frac{\partial h_l(x_l^-; z_l^s \ldots)}{\partial x_l} \quad (10)$$

$$H_{l,r_i} = \frac{\partial h_{r_i}(x_l^-, x_{r_i}^-; z_{r_i,l}^p, \ldots)}{\partial x_l} \quad (11)$$

$$H_{r_i,l} = \frac{\partial h_{r_i}(x_l^-, x_{r_i}^-; z_{r_i,l}^p, \ldots)}{\partial x_{r_i}}. \quad (12)$$

The final piece of the measurement update equations to discuss is the measurement noise covariance matrix R. We assume that measurement noise is constant over time and independent across measurements, but that the noise level in local and remote correspondences may differ. The matrix R is then a diagonal matrix where the diagonal entries corresponding to local correspondences have value r_l and diagonal entries corresponding to remote correspondences have value r_r.

5 Implementation

In this section we describe the distributed system that realizes the computational framework described in Section 4.

5.1 Pre-Calibration

Before system operation, the internal calibration of each IPU and the geometry of the display surface must be measured in addition to obtaining an initial estimate of each IPU's pose.

Our process for estimating the internal calibration of each IPU consists of first calibrating the IPU's stereo camera pair using the Matlab Camera Calibration Toolbox. Once the cameras have been calibrated, the projector calibration is estimated by projecting structured light patterns onto a non-planar surface and capturing the projected patterns with the IPU's cameras. The resulting images are then decoded to produce three-way image correspondences between the cameras and the projector. The correspondences between the cameras are then triangulated into 3D points to produce a set of 3D-2D correspondences in the projector that is then used to calibrate the projector using the DLT algorithm [1].

Once the internal calibration of each IPU has been estimated, they are arranged to form a display, and the display surface geometry and initial pose of each IPU is estimated. In this pre-calibration process, we require that the camera field-of-view of each IPU overlaps with the camera field-of-view of at least one other IPU. The IPUs then take turns projecting encoded structured light patterns while the cameras of all IPUs capture images. Decoding of these structured light patterns allows precise inter- and intra-IPU image correspondences to be obtained.

The intra-IPU correspondences are used to reconstruct a point-cloud representation of the display surface from the perspective of each IPU. The inter-IPU correspondences are then used to stitch these individual reconstructions together, resulting in a point-cloud representation of the display surface and an estimate of the pose of each IPU together in a common coordinate system.

To construct a polygonal model from this point-cloud representation, we use the RANSAC-based plane-fitting algorithm described in Quirk [12] that is robust against noise and outlying points resulting from false stereo matching. This algorithm extracts planes from the point cloud representation of the display surface and intersects them to produce a polygonal model of the surface.

5.2 Distributed Architecture

In order to continuously estimate its pose, each IPU must collect local and remote correspondences and process them using its Kalman filter. While local correspondences can be collected at each IPU without the need to communicate with other IPUs, a mechanism of obtaining remote correspondences is required. The system we have developed accomplishes this through inter-IPU communication of camera images over a local network.

We have developed a request/response architecture that allows IPUs to operate asynchronously by requesting primary camera images captured at a specified time from other IPUs. In order to synchronize the timestamps between cameras on different IPUs, we use a camera synchronization box from Point Grey Research.

To facilitate the ability of IPUs to respond to requests for camera images captured at a specified time, each IPU maintains a history of recently captured camera images in what we call a *camera buffer*. Each IPU maintains two camera buffers, one each for its primary and secondary cameras. A special camera buffer thread is dedicated to updating the camera buffer by replacing the oldest image in the buffer with a new image from the camera whenever one is available. In this way, a recent history of camera images is available at each IPU that can be searched based on timestamp when an image request is received from another IPU.

5.2.1 Collection and Processing of Local Correspondences

The following process occurs asynchronously at each IPU to collect a set of local correspondences. First, the latest image is requested from the camera buffer of the primary camera, call this image I_l^p. Once this image has been obtained, the corresponding image in time I_l^s from the camera buffer of the secondary camera is requested.

The next step is to obtain correspondences between I_l^p and I_l^s. We do this by first detecting a set of features in I_l^p using the OpenCV implementation of Shi and Tomasi's "Good Features to

Track" [16]. Correspondences for these features are then found in I_l^s using OpenCV's implementation of KLT tracking [10, 3]. Finally, each correspondence is checked against the epi-polar constraint between the primary and secondary cameras to yield the $z_l^p \Leftrightarrow z_l^s$.

Each IPU is provided with local access to its own fixed internal calibration as well as the display surface model. In conjunction with the $z_l^p \Leftrightarrow z_l^s$, this provides all necessary information to compute H_l from Equation 10 as well as $\widetilde{z}_l^p$ using Equation 5. We estimate the jacobian H_l numerically using forward differences.

5.2.2 Collection and Processing of Remote Correspondences

Remote correspondences are collected by requesting camera images from other IPUs that were captured by their primary cameras at the same time as I_l^p. When such a request is processed by another IPU U_{r_i}, its response includes not only the requested camera image $I_{r_i}^p$, but also its intrinsics $\kappa_{r_i}^p$, current pose estimate x_{r_i}, and apriori error covariance $(P_{r_i} + Q_{r_i})$.

Once the requested image $I_{r_i}^p$ has been received from another IPU, correspondences between I_l^p and $I_{r_i}^p$ are measured. We do this using an approach that first finds a set of correspondences between $I_{r_i}^p$ and $\widetilde{I}_{r_i}^p$, a prediction of $I_{r_i}^p$ generated on graphics hardware using the current estimated calibration of C_l^p and $C_{r_i}^p$, and then transforms these into a set of correspondences between I_l^p and $I_{r_i}^p$. This approach greatly improves feature matching success for algorithms like KLT when there are large perspective distortions between the two views, as is likely the case for camera images from different IPUs. More details on this technique can be found in [8].

Once the correspondences $z_{l,r_i}^p \Leftrightarrow z_{r_i,l}^p$, between I_l^p and $I_{r_i}^p$ have been measured, all information necessary to compute H_{l,r_i}, $H_{r_i,l}$, and $\widetilde{z}_{l,r_i}^p$ from Equations 11, 12, and 6 is available. We compute H_{l,r_i} using the closed-form solution in [7] and estimate $H_{r_i,l}$ numerically using forward differences.

5.2.3 Continuous Operation

Algorithms 2 and 3 summarize our implementation for collecting and processing local and remote correspondences and Algorithm 1 illustrates how we have organized these processes into a continuous calibration loop that is executed by each IPU at display time. This loop is executed independently of the rendering in a separate calibration thread. This calibration thread is implemented to run concurrently with the rendering thread, but without causing rendering performance to drop below a certain framerate.

In our current implementation, each IPU broadcasts its image requests to all IPUs in the display and processes each of their responses as they are received. We acknowledge this as a limiting factor in the scalability of our system and discuss our plans to improve this in Section 7.

In order to absorb network latency, each IPU collects and processes local correspondences while it waits to receive image responses from the other IPUs. It then enters an inner loop where it processes camera image responses and requests until all IPU's have responded or a timeout condition has been reached. This timeout condition provides fault tolerance by allowing system operation to continue should an IPU be unable to provide a response.

The final step is to update the Kalman filter to produce a new pose estimate that is communicated to the rendering process to allow the new estimate to affect the image correction that takes place.

6 Results

We have tested our framework for distributed cooperative pose estimation using a two-IPU display and two real-time applications. The first application displays a rotating panorama of real-world imagery that contains many strong features, while the second application is an open source flight simulator called Flight Gear, whose synthetic imagery contains far fewer features. Figure 1 shows the capability of our system to continuously estimate the poses of both projectors when both are moved simultaneously using the panorama application. Similar results for the flight simulator application are shown in Figure 5. Figure 6 shows a close-up of the projector image overlap as the projectors are moved and recalibrated using our technique. Since we currently do not re-estimate intensity blending masks for IPUs after they have been moved, bright bands are visible in the images where the projectors overlap.

Figure 7 shows the results of pose estimation over time for one IPU in a two-IPU display using the panorama application. In this sequence, captured over roughly four minutes, the IPU is moved once about half-way through the sequence. We have extracted frames from the corresponding video sequence that show the display configuration before, during, and after the movement of the projector. Rotation of the panorama was disabled during this experiment for comparison purposes. The ringing effect in the plots as the IPU is moved is a result of the temporary violation of our

Algorithm 1 ContinuousPoseEstimation

```
while true do
    [I_l^p, I_l^s] = Get-Local-Camera-Images
    BroadCast-Request(I_l^p.time)
    Process-Local
    repeat
        if responseReceived then
            Process-Remote;
        end if
        if requestReceived then
            Process-Request
        end if
    until timeout ∨ allResponsesReceived
    Update-Kalman-Filter
end while
```

Algorithm 2 Process-Local

```
Image I_l^p, I_l^s
Intrinsics κ_l^p, κ_l^s
Extrinsics x_l
CoordinateTransform Δ_l
DisplaySurface S
z_l^p = Detect-Features(I_l^p)
z_l^s = Match-Features(z_l^p, I_l^p, I_l^s)
[z_l^p, z_l^s] = Verify-Epipolar-Constraint(z_l^p, z_l^s, κ_l^p, κ_l^s, x_l, Δ_l)
[H_l, z̃_l^p] = Compute-Filter-Mats-L(x_l, z_l^s, κ_l^p, κ_l^s, Δ_l, S)
Add-Local-Correspondences-To-Filter(z_l^p, z_l^s, H_l, z̃_l^p)
```

Algorithm 3 Process-Remote

```
Image I_l^p, I_{r_i}^p
Intrinsics κ_l^p, κ_{r_i}^p
Extrinsics x_l, x_{r_i}
DisplaySurface S
Ĩ_{r_i}^p = Predict-Remote-Image(I_l^p, κ_l^p, κ_{r_i}^p, x_l, x_{r_i}, S)
F̃ = Detect-Features(Ĩ_{r_i}^p)
z_{r_i,l}^p = Match-Features(F̃, Ĩ_{r_i}^p, I_{r_i}^p)
z_{l,r_i}^p = Warp-Features(F̃, κ_l^p, κ_{r_i}^p, x_l, x_{r_i}, S)
[H_{l,r_i}, H_{r_i,l}, z̃_{l,r_i}^p] = Compute-Filter-Mats-R(x_l, x_{r_i}, z_{r_i,l}^p, κ_l^p, κ_{r_i}^p, S)
Add-Remote-Measurements-To-Filter(z_{l,r_i}^p, z_{r_i,l}^p, H_{l,r_i}, H_{r_i,l}, z̃_{l,r_i}^p)
```

Algorithm 4 PROCESS-REQUEST

1: Intrinsics κ_l^p
2: Extrinsics x_l
3: Covariances P_l,Q_l
4: Time-Stamp t
5: I = Search-Camera-Buffer(t)
6: Send-Response(I,κ_l^p,x_l,$P_l + Q_l$)

assumption that the IPU is stationary. Also, a slight drift over time may be observed in the y component of the IPU's position. Due to the vertical ambiguity in the shape of our display surface, which corresponds to the y axis, the y component of projector pose is unconstrained in our experimental set-up.

7 Discussion and Future Work

We have presented a novel distributed calibration framework for multi-projector displays where intelligent projector units interact to cooperatively re-estimate the poses of all projectors during actual display use. By making use of features in the projected imagery itself, our technique can be applied to any type of projector and operates without altering the projected imagery or affecting its quality.

We believe our technique is amenable to stereoscopic projection as well. In this situation, the offset left- and right-eye images will appear superimposed in the camera images, which would seem to degrade the sharpness of features. We have simulated this effect in a monoscopic two-projector display by overlapping the two projectors as completely as possible and moving one of them slightly to create a blurred-image effect. In all such experiments, the poses of both projectors were successfully recovered. We attribute this to the robustness of the feature matching algorithms used in our system.

While we have shown that our framework can operate successfully when feature-sparse synthetic imagery is projected, it is possible for calibration accuracy to be quite poor for some applications where the imagery is too lacking in strong features for calibration to be successful. In this case, since our computational framework imposes no requirements on the origin of the image measurements, it could be used in conjunction with techniques for embedding imperceptible patterns into projected imagery [4, 5, 6].

It is also possible that the configuration of measured features cannot fully constrain the pose of one or more projectors. There is such an ambiguity in the examples we have shown, where due to the shape of the display surface, the vertical component of projector location is unconstrained. When the pose of at least one IPU is constrained in a direction that may be unobservable for others, this will be reflected in its error covariance matrix and will allow the IPU to propagate constraints to other IPUs via remote correspondences.

When the poses of all IPUs are unconstrained in some direction, as in our examples, the behavior of our technique is to estimate poses for the IPUs that result in their projected imagery being registered together. While this is a visually appealing result, it may not be consistent with the viewing position, resulting in an inability of the system to correctly compensate for the shape of the display surface. This could be remedied by continuously estimating the viewer position with respect to one of the projectors or adding a fiducial to the display surface and continuously estimating its position.

Our framework can impose a significant amount of computational overhead that competes with the rendering process for resources. We would like to overcome these performance implications by fully integrating a computational unit with each IPU whose sole responsibility is to estimate its pose. The rendering application could then operate on a separate machine that periodically receives updates as to the current pose of the IPU.

Except for additional computational load, we do not anticipate difficulties in applying our framework to displays consisting of more than two projectors. Since the cameras of any IPU are likely to overlap with the cameras of only a small number of remote IPUs, we believe it is possible to mitigate the additional computation load in larger displays to a great extent by limiting the number of remote IPUs that each IPU communicates with to those whose cameras have an overlapping field-of-view.

References

[1] Y. Abdel-Aziz and H. Karara. Direct linear transformation into object space coordinates in close-range photogrammetry. In *Symposium on Close-Range Photogrammetry*, pages 1–18, 1971.

[2] E. S. Bhasker, P. Sinha, and A. Majumder. Asynchronous distributed calibration for scalable and reconfigurable multi-projector displays. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1101–1108, 2006.

[3] J.-Y. Bouguet. Pyramidal implementation of the lucas kanade feature tracker description of the algorithm. Technical report, Intel Corporation, 1999.

[4] D. Cotting, M. Naef, M. Gross, and H. Fuchs. Embedding imperceptible patterns into projected images for simultaneous acquisition and display. In *International Symposium on Mixed and Augmented Reality*, pages 100–109, 2004.

[5] D. Cotting, R. Ziegler, M. Gross, and H. Fuchs. Adaptive instant displays: Continuously calibrated projections using per-pixel light control. In *Eurographics*, pages 705–714, 2005.

[6] A. Grundhöfer, M. Seeger, F. Häntsch, and O. Bimber. Coded projection and illumination for television studios. Technical Report 843, Bauhaus-University Weimar, 2007.

[7] R. Haralick and L. Shapiro. *Computer and Robot Vision*, volume 2. 1993.

[8] T. Johnson and H. Fuchs. Real-time projector tracking on complex geometry using ordinary imagery. In *Workshop on Projector-Camera Systems (PROCAMS)*, 2007.

[9] R. Kalman. A new approach to linear filtering and prediction problems. *Transaction of the ASME - Journal of Basic Engineering*, pages 35–45, 1960.

[10] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of Imaging Understanding Workshop*, pages 121–130, 1981.

[11] K. Myers and B. Tapley. Adaptive sequential estimation with unknown noise statistics. *IEEE Transactions on Automatic Control*, 21:520–523, 1976.

[12] P. Quirk, T. Johnson, R. Skarbez, H. Towles, F. Gyarfas, and H. Fuchs. Ransac-assisted display model reconstruction for projective display. In *Emerging Display Technologies*, 2006.

[13] A. Raij, G. Gill, A. Majumder, H. Towles, and H. Fuchs. Pixelflex2: A comprehensive, automatic, casually-aligned multi-projector display. In *IEEE International Workshop on Projector-Camera Systems (PROCAMS-2003)*, 2003.

[14] A. Raij and M. Pollefeys. Auto-calibration of multi-projector display walls. In *International Conference on Pattern Recognition*, 2004.

[15] R. Raskar, J. van Baar, P. Beardsley, T. Willwacher, S. Rao, and C. Forlines. ilamps: Geometrically aware and selfconfiguring projectors. In *ACM SIGGRAPH*, 2003.

[16] J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 1994.

[17] R. Yang, D. Gotz, J. Hensley, H. Towles, and M. Brown. Pixelflex: A reconfigurable multi-projector display system. In *IEEE Visualization*, 2001.

[18] R. Yang and G. Welch. Automatic projector display surface estimation using every-day imagery. In *9th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2001*, 2001.

[19] J. Zhou, L. Wang, A. Akbarzadeh, and R. Yang. Multi-projector display with continuous self-calibration. In *Workshop on Projector-Camera Systems (PROCAMS)*, 2008.

[20] S. Zollmann, T. Langlotz, and O. Bimber. Passive-active geometric calibration for view-dependent projections onto arbitrary surfaces. In *Workshop on Virtual and Augmented Reality of the GI-Fachgruppe AR/VR*, 2006.

Figure 5: A two-projector display after both projectors have been moved (left) and after roughly ten seconds of calibration (middle,right).

Figure 6: A close-up of calibration accuracy in projector image overlap.

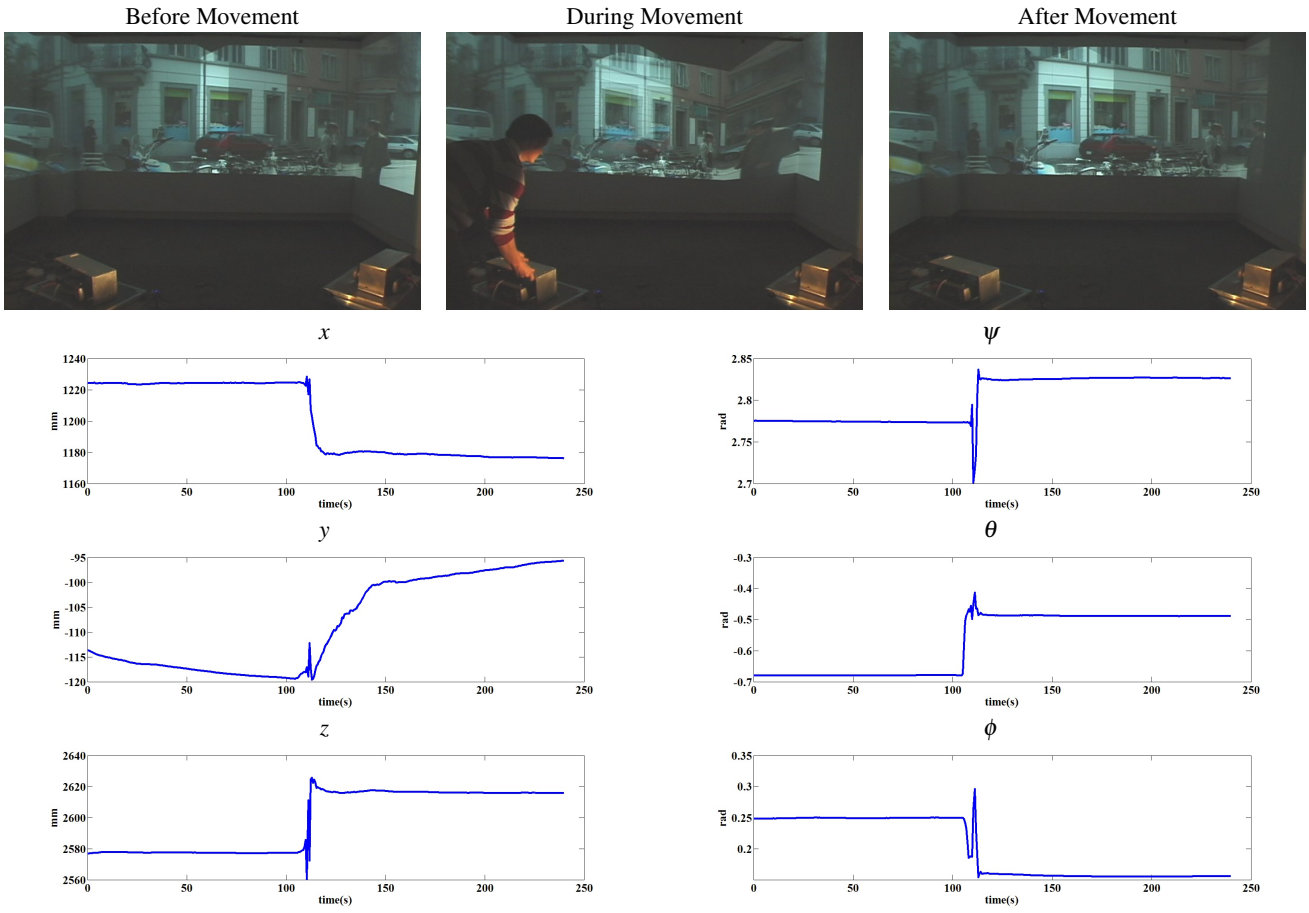

Figure 7: Results of estimating the pose of a single projector as it is moved once over the course of four minutes. Frames from the corresponding video sequence show the configuration of the display before, during, and after the projector is moved.