Analysis and Practical Minimization of Registration Error in a Spherical Fish Tank Virtual Reality System

Qian Zhou[1], Gregor Miller[1], Kai Wu[1], Ian Stavness[2] and Sidney Fels[1]

[1]Electrical and Computer Engineering, University of British Columbia, BC, Canada
[2]Department of Computer Science, University of Saskatchewan , SK, Canada

Abstract. We describe the design, implementation and detailed visual error analysis of a 3D perspective-corrected spherical display that uses calibrated, multiple rear projected pico-projectors. The display system is calibrated via 3D reconstruction using a single inexpensive camera, which enables both view-independent and view-dependent applications, also known as, Fish Tank Virtual Reality (FTVR). We perform error analysis of the system in terms of display calibration error and head-tracking error using a mathematical model. We found: head tracking error causes significantly more eye angular error than display calibration error; angular error becomes more sensitive to tracking error when the viewer moves closer to the sphere; and angular error is sensitive to the distance between the virtual object and its corresponding pixel on the surface. Taken together, these results provide practical guidelines for building a spherical FTVR display and can be applied to other configurations of geometric displays.

1 Introduction

As computer graphic and display technology advances rapidly, the interaction and visualization of 3D information is becoming increasingly important. Volumetric displays have shown promise [1] for interacting with and visualizing 3D data as they provide voxel-based 3D imagery. Many types of volumetric displays have been proposed in a variety of shapes[2, 3]. Among those types of volumetric displays, spherical displays have seen significant interest with research prototypes and commercial products. While a true volumetric display provides pixels in actual 3D space, perspective-corrected 3D display provides this illusion by projecting the correct perspective view of the scene for the viewer on the surface of the sphere. Thus, as the viewer moves, the scene maintains the correct 3D perspective; this is known as Fish Tank Virtual Reality (FTVR).

Spherical displays embody the metaphor of a "crystal ball" and provide non-occluded views from all viewpoints around it. However, providing uniform pixels appearing on the surface of the sphere is one of the requirements for constructing spherical FTVR. One approach to achieve this effect uses an array of rear-projection projectors from within the sphere, providing a scalable system with

high resolution and uniform pixel density [4]. However, geometry calibration and blending of multiple projectors on the curved screen surface to achieve seamless imagery is a challenge.

We present a practical approach for building a spherical multi-projector display to overcome this challenge. Our system works in both view-dependent and view-independent modes as we can project a seamless image over the entire surface. We introduce our system design along with the workflow for our 3D reconstruction-based, single-camera display calibration approach. We blend projected images based on geometry reconstruction, creating a seamless undistorted imagery using two-pass rendering. Using this approach, perceptual discrepancies arise when virtual objects do not remain correctly aligned based on the actual viewpoint due to error in the system pipeline. This misalignment may cause artifacts like distortions of rendered objects, making the visualization unacceptable to the viewer for a given application. To mitigate this issue, higher fidelity calibration can be performed, however, without knowing which calibration component accounts for the most perceptual error, it is difficult to know what to improve. Thus, as part of our practical guide, we provide a mathematical model of different calibration error sources. Using this model, we conduct error analysis for the spherical multi-projector FTVR system in terms of display calibration error and head-tracking error. As we discuss, we found: 1. Tracking error causes significantly more angular error than display error; 2. Angular error becomes more sensitive to tracking error when the viewer moves closer to the sphere; 3. Angular error is sensitive to the distance between the virtual object and its corresponding pixel on surface; 4. Our calibration approach is more sensitive to the error in sphere pose than projector error, making it necessary for us to improve the sphere pose estimation to reduce calibration error; 5. Calibration error is not spatially homogeneous. These results provide a guide to establish system component and calibration fidelity that can be matched to different application needs.

2 Related work

2.1 FTVR and Spherical Displays

Fish Tank Virtual Reality (FTVR) [5] is a type of 3D head-tracked display, providing motion parallax cues to improve user's understanding of the 3D virtual scene. This technique has been widely applied to various systems and applications. The CAVE [6] is one of these well-known systems, extending the traditional FTVR by projecting on multiple screens to form a geometric shape display. Besides the CAVE, Stavness et al. [7] developed pCubee, a perspective handheld cubic display. They arranged five small LCD panels to form the sides of a cube with head-coupled perspective-corrected rendering. However, a pCubee study [8] revealed occlusion caused by the seam between screens discouraged users from changing their view from one screen to another, making the shape of the display an important form factor.

A spherical display has a promising shape for a volumetric display as it has no seam between screens. A number of spherical displays have been proposed with different implementations. One of the early work in this field is the Perspecta Spatial 3D System from Actuality Systems, Inc [1]. It utilizes an embedded stationary projector to project imagery onto a rotating screen. Although it can generate volume-filling imagery, this type of system is expensive to build and has a limitation in resolution and scaling. As an alternative method, some systems use rear-projection directly onto a spherical screen. Sphere [9], Snowball [10], and commercial products like Pufferfish use one projector for rear-projection onto the spherical screen. While simple and fairly effective, these systems offer low resolution and lack of scalability. Spheree [4] extends this approach by utilizing multiple pico-projectors to increase resolution, making the system scalable to spherical screens with various sizes. We use the same approach to design our system, making it scalable and flexible.

2.2 Multi-projector System Calibration

The main challenge with multi-projector systems is the calibration of the system. The calibration includes geometry calibration and photometric blending to achieve seamless imagery. While there has been substantial work on calibration of planar multi-projector display using a single camera via linear homography transformations [11, 12], non-planar multi-projector calibration is still under explored.

An early work on non-planar calibration was published by Raskar et al. [13], in which a stereo camera pair is used to recover projector-camera parameters and reconstruct a non-planar surface. They achieved registration using both structured light and additional surfaces. They improved their work by focusing a subset of non-planar surfaces called quadric surfaces using the stereo camera to recover a quadric transformation [14].

Harville et al. [15] proposed a calibration method for a class of shape which can be made by transforming a plane through folding, blending etc. They attached a physical checkerboard pattern to the display and used an uncalibrated camera to compute a composition of 2D-mesh-based mappings. Since the 3D geometry is not recovered, their application space is limited.

More recently, Sajadi and Majumder have published a series of papers on non-planar calibrations [16][17]. They use an uncalibrated camera to compute a rational Bezier patch for geometric correction to account for the distortion of the display surface. They extend their work for various shapes such as vertical extruded surfaces, swept surfaces, dome surfaces and CAVE-like surfaces. Their calibration methods mostly aims at large-scale displays and thus mount the camera on a pan-tilt unit to cover the entire display.

Teubl et al. [4] developed a multi-projector system library to automatically calibrate multi-projector systems for different types of display surfaces. Their approach uses fixed warping without full 3D reconstruction. For a spherical shape they use a linear assumption by using a homography transformation between

camera and projector without reconstructing 3D geometry. This causes misalignment artifacts in overlapping areas.

Drawing inspirations from previous work, we also use the spherical geometry assumption as prior knowledge to calibration the system. However, our system is relatively small-scale (i.e. $< 1m$ diameter with a hole cut at the bottom) compared to those large-scale display ($> 1m$). Rear-projection through a small projection hole at the bottom of the sphere makes calibration difficult since the view of camera is mostly blocked by the edge of small hole. To overcome those problems, we propose a calibration pipeline similar to Raskar's work [13, 14] but with modifications to make it adapted to our system.

2.3 Virtual Reality System Error Analysis

Substantial research analyzes and handles the error in virtual reality and augmented reality systems [18–21]. Holloway et al. [19] analyzed the causes of registration error for a Head Mounted Device (HMD) using a set of parameters. MacIntyre et al. [21] presented a statistical method to estimate the error and further use the estimated error to improve the AR interface. However most of them are for see-through HMD systems. Though there are many descriptions on FTVR systems and geometric displays, very little work has dealt with error analysis for these systems. Cruz et al. discussed tracking noise and delay for the CAVE system [6], but mainly aimed at comparing to HMD and normal monitor systems to show the reduced effect of these errors on the CAVE system. In this paper, we analyze errors in our spherical system in terms of display calibration error and tracking error using a mathematical model, which can be useful when choosing devices for system built-up and improvement.

3 Calibration of Multi-Projector FTVR System

A basic FTVR system consists of two essential parts to make the view-dependent display work: a display system and a tracking system. The display system is made of a set of projectors (we use pico-projectors) and a spherical screen. Projectors are set under the spherical screen, rear-projecting through a projection hole onto the screen as illustrated in Figure 1. Calibration is required to make the system work. This includes display calibration and tracker calibration.

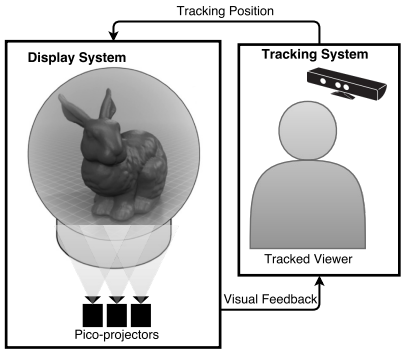

Fig. 1. System diagram of a 3D perspective-corrected spherical display.

3.1 Display Calibration

To generate seamless imagery on the spherical display, we need to reconstruct

the 3D geometry of projected pixels on sphere, which is used to register projected images and then blend the intensity in overlapping areas.

As discussed in related work, cameras are commonly used to calibrate multi-projector systems. The calibration of our FTVR system is challenging and differs from other multi-projector systems in the following aspects:

1. System scale. Most multi-projector calibration methods are designed for large-scale display, where space is plenty for projectors, cameras and mirrors. In our case, we have a relatively small spherical screen of diameter 29 cm with projectors set in limited space under the screen. It is difficult to put multiple cameras in the limited space to see the screen. A single small camera is preferred in our system.
2. Visibility. The projection hole of diameter 14 cm on the spherical screen will block the view of camera, making a large portion of the projection invisible to the camera.
3. As a FTVR system, it should be able to support both view-independent and view-dependent applications to enable multi-person VR display.

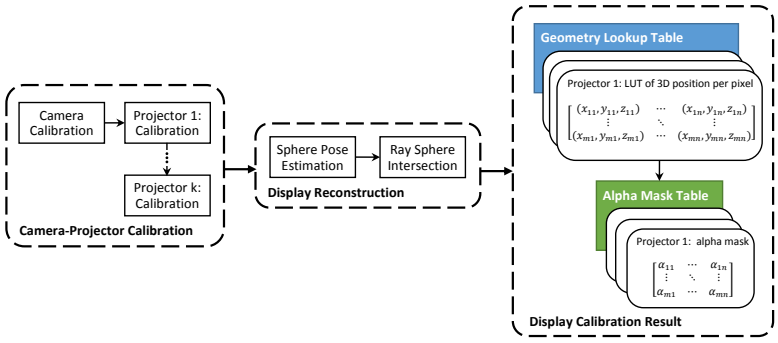

Fig. 2. Display calibration pipeline

Being able to support both types of applications forces us to reconstruct the 3D geometry of the projection on the sphere. Fixed warping with linear assumptions [4] offers low accuracy thus not applicable to our case. Although the parametric approach using quadric transformation [14] appears to be promising, it does not support view-independent applications. For view-dependent applications, we still have to update the quadric transformation for each frame since the viewpoint will be moving. The visibility problem makes most calibration methods using patch-based or mesh-based interpolations [16] not workable in our system. Finally, our system scale suggests a single camera approach.

As a result, we propose our display calibration approach as illustrated in Figure 2. This approach begins with the calibration of camera and projectors, followed by a pose estimation of the spherical display surface, then ending with ray-sphere intersection to locate 3D position of each pixel on the surface.

Camera-Projector Calibration Each projector is modeled as an inverse pinhole camera. We calibrate camera C and projectors P using a plane-based calibration approach [22], which is essentially an extension of Zhang's calibration technique [23] for camera-projector system. With the removal of the spherical screen, the planar pattern placed at different positions and orientations is used to avoid degenerate case for projector calibration [13]. After this step, the intrinsic and extrinsic parameters for the camera and projectors are recovered.

Sphere Pose Estimation In this step, each projector P_i is paired with the same camera C, forming a stereo pair S_i as illustrated in the left image of Figure 3. For each projector P_i, we rear-project an array of blobs onto the sphere, with the camera seeing part of the projected patterns. We triangulate to find the 3D position of the blobs in the camera-centered coordinate system. This procedure is repeated for each stereo pair S_i to obtain more points on the sphere. The right images of Figure 3 show the partial projection from two projectors seen by the camera.

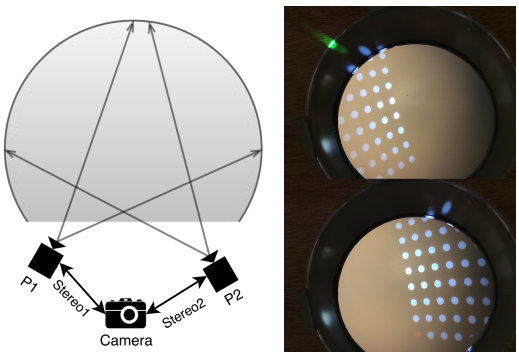

Fig. 3. Stereo pairs and projected blob patterns

We fit the sphere equation with these 3D points using Weighted Least Square [24]. The weighting matrix W comes from the re-projection error in triangulation. Points with large re-projection error will have a relative small weighting when fitting a sphere. After this step, the sphere pose with four parameters is recovered.

Ray-Sphere Intersection Our goal is to find the 3D position for each pixel on the surface. The ray-sphere intersection is used to find the 3D position for each pixel. This begins by computing the ray equation $Ray(\lambda)$ associated to the pixel:

$$\begin{aligned} Ray(\lambda) &= C_{P_i} + V_{uv} \cdot \lambda \\ &= -R_{P_i}^{-1} T_{P_i} + \lambda (K_{P_i} R_{P_i})^{-1} \begin{pmatrix} u & v & 1 \end{pmatrix}^T , \end{aligned} \tag{1}$$

where C_{P_i} represents the center of projector P_i in camera-centered coordinate system. V_{uv} is the vector pointing from C_{P_i} to the pixel (u, v) in projector P_i. The intrinsic matrix of projector P_i is K_{P_i} and corresponding extrinsic matrix is $\begin{pmatrix} R_{P_i} & T_{P_i} \end{pmatrix}$.

$Ray(\lambda)$ is intersected with the recovered sphere for each pixel (u, v) in projector P_i to find the intersection point in the camera-centered coordinate system. This results in the 3D position of each pixel on the sphere. For the convenience of rendering, we transform those 3D points from the camera-centered coordinate system to the display coordinate system through translation. The display coordinate system has its origin set at the center of the sphere. The geometry result of each projector is stored in a look-up table as shown in Figure 2, which is later used to compute alpha mask for blending.

Intensity Blending For the photometric blending of the overlapping area, we compute the alpha mask for each projector based on a weighted average[13], assigning an intensity weight from 0 to 1 to each pixel in the projector. The alpha-weight $A_m(u, v)$ for the (u, v) pixel in projector m is computed:

$$A_m(u, v) = \frac{\alpha_m(m, u, v)}{\sum_i \alpha_i(m, u, v)},$$

where $\alpha_i(m, u, v) = w_i(m, u, v) \cdot d_i(m, u, v)$. $w_i(m, u, v) = 1$ if the pixel (u, v) of projector m is inside the hull of projector P_i; otherwise zero. $d_i(m, u, v)$ is the distance of the pixel (u, v) of projector m to the nearest edge of projector P_i.

$w_i(m, u, v)$ is computed based on whether the pixel is inside the view frustum of projector P_i. The distance $d_i(m, u, v)$ is computed as the length of the shortest arc that connects pixel to its nearest edge arc.

3.2 Tracker Calibration

The goal of tracker calibration is to compute a similarity transformation between tracker coordinate system and display coordinate system. We first project blob patterns onto the sphere. Then we use the tracker to detect and recover their 3D positions in tracker coordinate system. Since the display system has been calibrated, we know the 3D position for each blob in display coordinate system. We estimate the similarity transformation using SVD as an initial guess, followed by Levenberg-Marquardt method for refinement [25] .

4 Error Analysis of the Spherical System

While many spherical display systems have been proposed, none has included a formal analysis of visual error to our knowledge. However, error analysis is important for two main reasons. First, when building the system, one may expect

different error tolerance for each component of the system based on the purpose of application. Error analysis provides guidelines when choosing these components. Second, knowing the nature and sensitivity of the error also helps us to eliminate artifacts cause by the error. These artifacts include, but are not limited to: distortion (straight lines appear to be curved), ghosting effect (double-image) and floating effect (virtual objects that are supposed to locate at a fixed location appear to swim about as viewer moves head) [18].

4.1 Metric for FTVR system

We use eye angular error [6] as the metric for evaluating the performance of a FTVR system. Eye angular error quantifies the registration between viewer and the display surface. Qualitatively, eye angular error creates artifacts like distortions when rendering 3D objects. Quantitatively, the angular error can be defined to be the displacement between a pixel shown on the spherical screen D and its desired location $\bar{D}$ which should be perspective-corrected according to the viewer position.

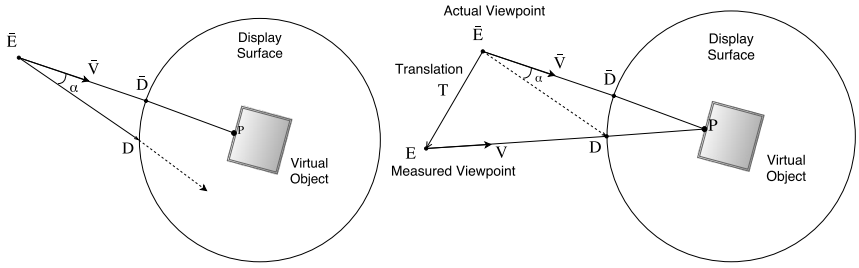

Fig. 4. (a) eye angular error in our system (b) angular error caused by tracking error

Let $\bar{E}$ represent the viewer position, looking at a virtual point P inside the sphere as shown in Figure 4(a). Then the angular error α can be computed using the following equation:

$$\alpha = arccos(\frac{(D-\bar{E})(P-\bar{E})}{\|(D-\bar{E})\|\|(P-\bar{E})\|}), \tag{2}$$

Since our FTVR system consists of display system and tracking system, it is natural to analyze the system error in terms of display error and tracking error. We now discuss how the tracking error and display error will influence α in the following sections.

4.2 Tracking Error

Tracking error represents the error between the actual viewpoint and the measured viewpoint. Sources of this error include: tracker error, tracker latency and

transformation error. We use translation T to represent the tracking error between the actual viewpoint $\bar{E}$ and the measured viewpoint E in Figure 4(b). Due to the translation T, $\bar{D}$ is the desired pixel corresponding to the actual viewpoint $\bar{E}$, while D corresponding to E is the pixel shown on the display.

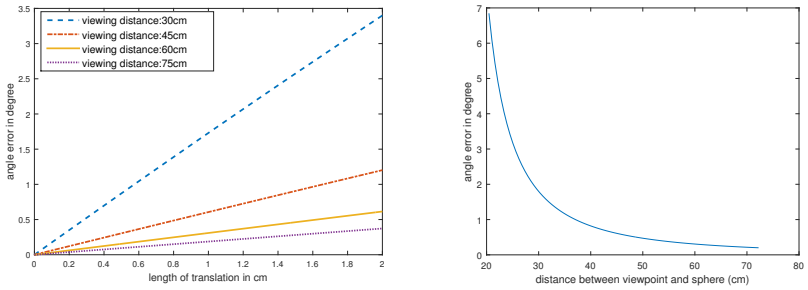

Fig. 5. (a) tracking error caused by translation (b) by viewpoint

The ray starting at the measured viewpoint E pointing in the direction of V can be expressed as $Ray(\lambda) = E + V \cdot \lambda$. Intersecting the ray with the sphere, we can express D with respect to E and V, while the intermediate variables E and V can be expressed using $\bar{E}, P$ and T. Thus, from point D, we see the angular error α in the equation (2) is a function of the virtual point P, translation T and actual viewpoint $\bar{E}$.

Effect of translation α is maximized when the direction of T is perpendicular to the EP. In the following sections, we always assume EP is tangent to T so that we are evaluating based on the worst case. Figure 5(a) shows the simulation result of α as a function of $\|T\|$ while the viewer is looking at the virtual point P placed at the sphere center. The slope increases as the viewer gets closer to the sphere, meaning α becomes more sensitive to translation error if the viewer is closer to the display.

Effect of viewpoint Figure 5(b) illustrates the simulation result of how the viewpoint influences α when the viewer is looking towards a virtual point P placed at the center of the sphere. As the viewpoint $\bar{E}$ moves away from the sphere along $\bar{E}P$, α decreases dramatically. To control α, a minimum viewing distance can be established depending on the applications. For example, for interactive applications, the viewing distance is likely shorter than ones only needing visualizations. Hence, the interactive system will require a tracking device with higher accuracy.

Effect of virtual point Consider that virtual point P can be placed both inside and outside the sphere. For a viewpoint $\bar{E}$, several viewing directions pointing

to different virtual points P_i are illustrated in Figure 6. As a function of the virtual point P_i, the angular error α first decreases when P_i travels towards $\bar{D}_i$, so that the virtual point away from its corresponding display pixel will cause larger α. When P_i arrives at $\bar{D}_i$, there is no angular error since D_i overlaps with $\bar{D}_i$, meaning points on the display surface towards the viewer will not cause angular error even if there is translation error in the tracker. As P_i moves out of the sphere further toward $\bar{E}$, α starts to increase rapidly, meaning points out of the sphere are more sensitive to translation error.

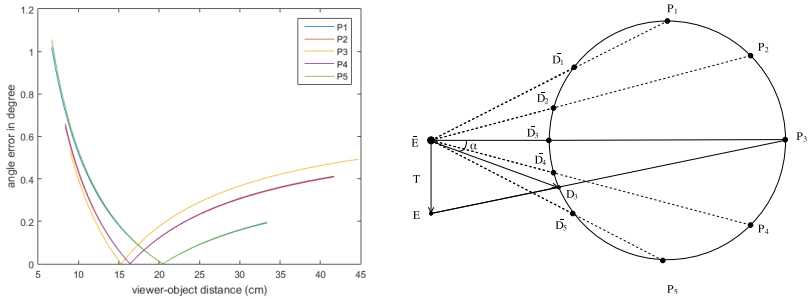

Fig. 6. Effect of the virtual point position on angular error

Since α depends on the position of virtual point P_i, there exists an "optimal" rendering region with relatively small angular error. For instance, if we want to have a maximum angular error as 0.4°, according to Figure 6 we can only render a virtual object inside the sphere that is less than 18 cm away from its corresponding pixel, or less than 5 cm away for an object outside the sphere, denoted as -5cm to +18cm.

Result As a summary, the eye angular error α increases: 1. as tracking error increases, with its slope influenced by viewing distance; 2. as viewer moves closer to the sphere; 3. as the virtual point is further from the pixel on display surface.

Table 1. Quantitative result of head-tracking error

Device	Head-Tracking Error	Angular Error	Viewing distance (object in center)	Rendering Range (at min view distance)
Kinect v2	8cm [26]	<2 degree	>60cm	-8cm to +11cm
Fastrak	0.2cm [27]	<0.5 degree	>30cm	-6cm to +30cm

Based on this analysis, we have quantitative results for two tracking systems we use for our FTVR system. According to Xu et al. [26], the average joint accuracy for Kinect v2 using its joint tracking SDK is around 8cm. If we set angular error to be no more than 2°, the minimum viewing distance should be no less than 60 cm assuming a viewer is looking at a virtual object at the center. At this minimum viewing distance, the rendering range should be -8cm to +11cm. Naturally, improvements over the joint tracking library in Kinect v2 SDK for head tracking can improve these bounds. For the Polhemus Fastrak with a tracking accuracy of 0.2 cm within 150 cm of the transmitter, it is possible to yield a 0.5° maximum angular error system, making it more appropriate for an *interactive* FTVR so viewers can get close enough to the display.

4.3 Display Error

Assuming the head-tracking is perfect, then the display error can be represented by the displacement between the desired 3D position of pixel $\bar{D}$ and its actual location D on the display surface in Figure 4. This error depends on the accuracy of the calibration workflow. Ideally, with ground truth we can compute the accuracy of our calibration algorithm. Unfortunately, ground truth is hard to acquire in this system. Instead, we use covariance as a measure of the calibration accuracy [25]. Alternatively, an empirical method could be used, in which error can be measured using a camera [28].

The 3D positions of projector pixels are computed using ray-sphere intersection in the calibration pipeline, which can be expressed as:

$$\boldsymbol{X} = f(\boldsymbol{x}; \boldsymbol{p}), \tag{3}$$

where $\boldsymbol{X}$ is the 3D position, $\boldsymbol{x}$ is the 2D projector pixel and $\boldsymbol{p}$ is a column vector that contains all parameters including projector parameters $\boldsymbol{p_1}$ and sphere parameters $\boldsymbol{p_2}$, with all parameters written in the form of column vector. Any error in $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$ will disturb the 3D position of projector pixel.

According to forward propagation [25], we can compute the covariance matrix of X as: $\Sigma_X = J_{X(p)} \Sigma_p {J_{X(p)}}^T$, where $J_{X(p)}$ is the Jacobian matrix of the vector function $X(\boldsymbol{p})$ with respect to the parameter vector $\boldsymbol{p}$ and Σ_p is the covariance matrix of $\boldsymbol{p}$.

Though the Jacobian matrix $J_{X(p)}$ can be computed analytically as the partial derivative of the vector function $X(\boldsymbol{p})$, the covariance matrix Σ_p is not so straight-forward. The parameter vector $\boldsymbol{p}$ contains both projector parameters and sphere parameters. The covariance matrix of projector parameters can be estimated using backward propagation [25] of the re-projection error once we calibrate the projector. The covariance matrix of sphere parameters can also be computed in a similar manner using a least square covariance computation. However, the correlation between projector parameters and sphere parameters is difficult to obtain. It is not appropriate to make the assumption that those parameters are independent since we used projector parameters to estimate sphere pose. So instead of computing a composite Σ_X in terms of all parameters, we

compute Σ_X respectively to projector parameters and sphere parameters as described next.

Error in projector parameters Assume $\boldsymbol{p_1}$ is the vector that contains projector intrinsic K_{P_i} and extrinsic parameters R_{P_i}, T_{P_i}. The covariance matrix Σ_{p_1} of $\boldsymbol{p_1}$ is computed using backward propagation from the projector equation (4): $\Sigma_{p_1} = ({J_{m_{p_1}}}^T \Sigma_{\mathbf{m}}^{-1} J_{m_{p_1}})^{-1}$, where Σ_m is the re-projection error during projector calibration and $J_{m_{p_1}}$ is the Jacobian matrix of the equation (4) with respect to the projector parameter $\boldsymbol{p_1}$.

$$\boldsymbol{m} = K_P \begin{pmatrix} R_P & T_P \end{pmatrix} \boldsymbol{M} \tag{4}$$

We plug the covariance matrix Σ_{p_1} into the equation (3) to compute covariance matrix Σ_X. Using our calibration result, this yields to an average 3D Euclidean distance error of 0.315 mm for pixels on surface.

Error in sphere pose parameters Assume $\boldsymbol{p_2}$ is the vector that contains sphere parameters s_1, s_2, s_3 and s_4. The covariance matrix Σ_{p_2} can be estimated [29] based on a weighted least square. Using the computed Σ_{p_2}, we get an average of 1.344mm, indicating that error in sphere pose tends to be more influential than the error in projector calibration.

Result From the above analysis, we see the following results. First, if we consider sphere pose and projector parameters as error sources that influence the calibration error, then our calibration is more sensitive to changes in sphere pose estimation than projector parameters. Thus, it is best to improve the sphere pose estimation to reduce calibration error.

Second, the calibration error is not spatially homogeneous. Figure 7 shows calibration errors taken at equally spaced locations on the projector using equation (3) when considering respectively the projector parameters and sphere pose parameters as error sources. The result is based on the implementation described in Section 5. The calibration error reaches peaks on the corners and falls off from the center by a factor up to ten times the error at the center. The fringe pixels have significantly more errors than pixels at the center possibly resulting in a noticeable misalignment in the overlapping projector area since overlaps happen mostly along the projection fringe. Hence, it is useful to have a pixel-by-pixel adjustment after the euclidean reconstruction to minimize those local errors.

Lastly, if we compare the display calibration error with the tracking error, the calibration error of 1mm on the display surface only yields a 0.38° angular error with 30cm viewing distance. So tracking error causes significantly more angular error than calibration error. This matches with Holloway's result [19] for an HMD system. While tracking error can be the major error source for artifacts like distortion and floating effect, the display calibration error accounts for the ghosting effect in overlapping area. In fact, calibration error is the only error source to cause ghosting. Ghosting happens when we stitch images from

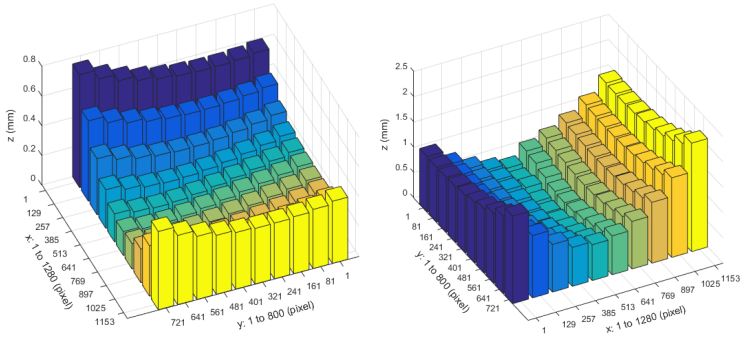

Fig. 7. Space dependency of the calibration error on pixels with (a) projector as error source (b) sphere as error source

adjacent projectors and is caused by the error in geometry calibration. This is independent of the viewpoint position. Thus, improving calibration error is important, especially for view-independent applications in which imagery is wall-papered on the entire sphere so that overlapping pixels are always used for rendering.

5 Implementation

Our system includes a display system and tracking system. The display system consists of two pico-projectors, an acrylic spherical display and a chassis which holds the display surface and projectors. The sphere size is 29 cm diameter and the projection hole size is 14 cm diameter. We use two ASUS P2B projectors with the resolution of 1280x800. A host with a NVIDIA Quadro K5200 graphic card directly sending rendering content to projectors. We use OpenGL to render graphics. To generate perspective-corrected images on the curved screen, we use a two-pass rendering method [11]. The two-pass rendering is chosen since the projection from 3D objects to the curved screen is non-linear. Though we evaluate the proposed approach with only two projectors, the result generalizes to more than two projectors. To add new projectors into the system requires adding another stereo pair directly registered to the world coordinate system for each projector. Doing so does not cause cascading error as the scale goes up as each projector is registered independently.

For the tracking system evaluation, we use a Kinect for Windows v2 and its SDK joint tracking APIs to track head position at 30Hz. For comparison, we also use a Polhemus Fastrak, which is a wired magnetic tracking system with the update rate up to 120Hz.

The calibration of the system is done once as a pre-processing step using Matlab. Figure 8(a)(b) shows the result for a view-independent application. This supports a wall-papered rendering for multiple users. Figure 8(c) shows the result for a view-dependent application, in which we track the viewer position and do a

two-pass rendering based on the pixel geometry lookup table and viewer position. This supports a perspective-corrected view for a single viewer and scales to more than two projectors.

6 Summary and Future Work

We have presented our work for designing and implementing a spherical FTVR system. We describe several practical methods to build the system, including the display calibration, the tracker calibration and rendering procedure. We presented an error analysis for the spherical multi-projector FTVR system in terms of display calibration error and head-tracking error. Our error analysis shows the tracking error causes significantly more angular error than calibration error. Thus, based on the application needs, we can select an appropriate tracking device to match the angular error requirements. Likewise, based on the tracking error of the device used, we can establish a minimum viewing distance and rendering region to control the angular error. Although the calibration error does not incur large eye angular error, it can still cause a double-image effect in the overlap region between adjacent projectors when blending. Thus, improving calibration in terms of pose estimation and the minimization of the local misalignment error in the overlap region is important. Future work involves taking into account the compound estimation for calibration error to provide an end-to-end estimate to further improve bounds on performance for spherical FTVR display. Our results provide an important contribution to enable the construction of non-planar FTVR displays that use a matrix of projectors. We are able to demonstrate the relative error contributions from the different parts of the compound system to help designers select components to match the needs of their 3D applications.

Fig. 8. View-independent application (a) after geometry calibration (b) after blending (c) View-dependent application.

Acknowledgement. We thank B-Con Engineering, NVIDIA and NSERC Canada for providing financial and in-kind support and Dr. Marcelo Zuffo and his group at University of Sao Paulo for helpful discussions.

References

1. Favalora, G.E.: Volumetric 3d displays and application infrastructure. Computer (2005) 37–44
2. Downing, E., Hesselink, L., Ralston, J., Macfarlane, R.: A three-color, solid-state, three-dimensional display. Science **273** (1996) 1185
3. Blundell, B.G., Schwarz, A.J.: Volumetric three-dimensional display systems. Volumetric Three-Dimensional Display Systems, by Barry G. Blundell, Adam J. Schwarz, pp. 330. ISBN 0-471-23928-3. Wiley-VCH, March 2000. **1** (2000)
4. Teubl, F., Kurashima, C.S., Cabral, M., Lopes, R.D., Anacleto, J.C., Zuffo, M.K., Fels, S.: Spheree: An interactive perspective-corrected spherical 3d display. In: 3DTV-Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON), 2014, IEEE (2014) 1–4
5. Arthur, K.W., Booth, K.S., Ware, C.: Evaluating 3d task performance for fish tank virtual worlds. ACM Transactions on Information Systems (TOIS) **11** (1993) 239–265
6. Cruz-Neira, C., Sandin, D.J., DeFanti, T.A.: Surround-screen projection-based virtual reality: the design and implementation of the cave. In: Proceedings of the 20th annual conference on Computer graphics and interactive techniques, ACM (1993) 135–142
7. Stavness, I., Lam, B., Fels, S.: pcubee: a perspective-corrected handheld cubic display. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM (2010) 1381–1390
8. Lam, B., Tang, Y., Stavness, I., Fels, S.: A 3d cubic puzzle in pcubee. In: 3D User Interfaces (3DUI), 2011 IEEE Symposium on, IEEE (2011) 135–136
9. Benko, H., Wilson, A.D., Balakrishnan, R.: Sphere: multi-touch interactions on a spherical display. In: Proceedings of the 21st annual ACM symposium on User interface software and technology, ACM (2008) 77–86
10. Bolton, J., Kim, K., Vertegaal, R.: Snowglobe: a spherical fish-tank vr display. In: CHI'11 Extended Abstracts on Human Factors in Computing Systems, ACM (2011) 1159–1164
11. Raskar, R.: Immersive planar display using roughly aligned projectors. In: Virtual Reality, 2000. Proceedings. IEEE, IEEE (2000) 109–115
12. Raij, A., Gill, G., Majumder, A., Towles, H., Fuchs, H.: Pixelflex2: A comprehensive, automatic, casually-aligned multi-projector display. In: IEEE International Workshop on Projector-Camera Systems, Nice, France (2003) 203–211
13. Raskar, R., Brown, M.S., Yang, R., Chen, W.C., Welch, G., Towles, H., Scales, B., Fuchs, H.: Multi-projector displays using camera-based registration. In: Visualization'99. Proceedings, IEEE (1999) 161–522
14. Van Baar, J., Willwacher, T., Rao, S., Raskar, R.: Seamless multi-projector display on curved screens. In: Proceedings of the workshop on Virtual environments 2003, ACM (2003) 281–286
15. Harville, M., Culbertson, B., Sobel, I., Gelb, D., Fitzhugh, A., Tanguay, D.: Practical methods for geometric and photometric correction of tiled projector. In: Computer Vision and Pattern Recognition Workshop, 2006. CVPRW'06. Conference on, IEEE (2006) 5–5
16. Sajadi, B., Majumder, A.: Automatic registration of multi-projector domes using a single uncalibrated camera. In: Computer Graphics Forum. Volume 30., Wiley Online Library (2011) 1161–1170

17. Sajadi, B., Majumder, A.: Autocalibration of multiprojector cave-like immersive environments. Visualization and Computer Graphics, IEEE Transactions on **18** (2012) 381–393
18. Azuma, R., Bishop, G.: Improving static and dynamic registration in an optical see-through hmd. In: Proceedings of the 21st annual conference on Computer graphics and interactive techniques, ACM (1994) 197–204
19. Holloway, R.L.: Registration error analysis for augmented reality. Presence: Teleoperators and Virtual Environments **6** (1997) 413–432
20. You, S., Neumann, U., Azuma, R.: Orientation tracking for outdoor augmented reality registration. Computer Graphics and Applications, IEEE **19** (1999) 36–42
21. MacIntyre, B., Coelho, E.M., Julier, S.J.: Estimating and adapting to registration errors in augmented reality systems. In: Virtual Reality, 2002. Proceedings. IEEE, IEEE (2002) 73–80
22. Falcao, G., Hurtos, N., Massich, J.: Plane-based calibration of a projector-camera system. VIBOT master **9** (2008) 1–12
23. Zhang, Z.: A flexible new technique for camera calibration. Pattern Analysis and Machine Intelligence, IEEE Transactions on **22** (2000) 1330–1334
24. Forbes, A.B.: Least-squares best-fit geometric elements. National Physical Laboratory Teddington (1989)
25. Hartley, R., Zisserman, A.: Multiple view geometry in computer vision. Cambridge university press (2003)
26. Xu, X., McGorry, R.W.: The validity of the first and second generation microsoft kinect for identifying joint center locations during static postures. Applied ergonomics **49** (2015) 47–54
27. Polhemus, F.: 3space fastrak users manual. F. Polhemus Inc., Colchester, VT (1993)
28. Chen, H., Sukthankar, R., Wallace, G., Li, K.: Scalable alignment of large-format multi-projector displays using camera homography trees. In: Proceedings of the conference on Visualization'02, IEEE Computer Society (2002) 339–346
29. Strang, G.: Introduction to Applied Mathematics. Wellesley-Cambridge (1986)