A multi-projector CAVE system with commodity hardware and gesture-based interaction

Carlos Andujar, Pere Brunet, Àlvar Vinacua, Miguel A. Vico and Jesús Díaz García
MOVING research group. Jordi Girona 1-3
08034 Barcelona, Spain

Abstract

Spatially-immersive systems such as CAVEs provide users with surrounding worlds by projecting 3D models on multiple screens around the viewer. Compared to alternative immersive systems such as HMDs, CAVE systems are a powerful tool for collaborative inspection of virtual environments due to better use of peripheral vision, less sensitivity to tracking errors, and higher communication possibilities among users. Unfortunately, traditional CAVE setups require sophisticated equipment including stereo-ready projectors and tracking systems with high acquisition and maintenance costs. In this paper we present the design and construction of a passive-stereo, four-wall CAVE system based on commodity hardware. Our system works with any mix of a wide range of projector models that can be replaced independently at any time, and achieves high resolution and brightness at a minimum cost. The key ingredients of our CAVE are a self-calibration approach that guarantees continuity across the screen, as well as a gesture-based interaction approach based on a clever combination of skeletal data from multiple Kinect sensors.

Keywords:
Virtual Reality Systems, Immersive Systems, Projector Calibration, Gesture-based interaction

1 Introduction

CAVE systems are known to be highly immersive back-projected VR rooms. Classical CAVE systems from the 90's and beginning of the past decade were using expensive CRT projectors and magnetic head and hand tracking. Unfortunately, most projection systems of the existing CAVE's are becoming obsolete after several years, and updating them can be rather expensive. In the meantime, however, multi-projector systems and cheap motion capture systems have appeared. These recent technologies can be efficiently used for CAVE updates, resulting in higher image resolution, better brightness and lower costs.

In this paper, we present a novel four wall multi-projector CAVE architecture which is powered by 40 off-the-shelf projectors controlled by 12 PCs. It operates in passive stereo, providing high brightness at 2000×2000 pixel resolution on each of the 4 walls. We have achieved high resolution while significantly reducing the cost and increasing versatility: the system works with any mix of a wide range of projector models that can be individually substituted at any time for more modern or cheaper ones. The uniformity of the final image is achieved using a specially designed self-calibration software which adapts each of the 40 projectors and guarantees concordance and continuity. The main contributions of our approach are:

- The design and construction of a passive stereo, four-wall CAVE system with commodity hardware is presented. It is based on 40 off-the-shelf DLP projectors and 12 PCs. The CAVE design achieves higher resolution and brightness than alternative installations, at significantly lower total cost.
- The system is versatile: it works with any mix of a wide range of projector models that can be individually substituted at any time for more modern or cheaper ones. The software and system architecture can be easily adapted to different screen configurations.
- Uniformity of the final image is guaranteed by a

specially designed self-calibration software which adapts each of the 40 projectors and guarantees concordance and continuity. Independent self-calibration of the different CAVE walls is sufficient.

- A gesture-based, ergonomic interaction paradigm, based on dynamically merging the information from two nearly-orthogonal Kinect sensors. Interaction is intuitive and cable-less.

The paper is organized as follows. After reviewing the previous work in next Section, Section 3 presents the architecture of the CAVE system. Sections 4 and 5 are devoted to presentation of the self-calibration algorithms and the software infrastructure for application development. Section 6 presents the Kinect-based gestural interaction scheme, while Section 7 discusses the system performance through a number of examples. Conclusions and future work are listed in Section 8.

2 Previous work

Multi-projector systems and calibration

The work on multi-projector displays using camera-based registration started at the end of the 90's, see for instance [21]. The design of projection-based tiled display systems was considered shortly afterwards [12], and the use of homographies and hierarchical homography settings for geometric automatic calibration started in 2002 with the work of Wallace et al. [7]. Since then, a number of papers have addressed the problem of designing tiled displays for planar Power Walls, but, with few exceptions, like [23], almost no work has been done for CAVE systems.

The color variation in multi-projector displays is due to the spatial variation in the color gamut across the display [17, 18]. This variation can be classified into three different categories: intra-projector variation (within a single projector), inter-projector variation (across different projectors), and overlap variation. For an analysis of perceptual aspects, see for instance [19]. In [22] a new method was presented which addressed spatial variation in both luminance and chrominance in tiled projection-based displays. Their approach morphs the spatially-varying color gamut of the display in a smoothly constrained manner while retaining the white point.

The system which is probably closest to our proposal is the Star CAVE project [8]. It uses passive stereo and multi-projection on a five-sided small room with three horizontal display layers with a total of 15 projection surfaces and two projectors per surface. However, the Star CAVE is not using commodity hardware and it is not able to benefit from existing CAVE settings. Our proposal, instead, works with any mix of commodity projector models and results on a higher resolution and brightness at a significantly reduced total cost.

Kinect-based interaction

The Microsoft Kinect provides a convenient and inexpensive depth sensor and skeleton tracking system for VR applications [29, 26]. Different authors have measured and reported the performance of the Kinect tracking software. Most of these studies refer to the first version of the sensor [15, 16, 6, 9] whereas only a few [11, 1] report on the Kinect V2 sensor. Livingston et al. [15] have conducted multiple tests to evaluate the noise, accuracy, resolution, and latency of the Kinect V1 skeleton tracking software. At 1.2 m the position noise was found to be 1.3 mm (sd=0.75 mm), whereas at 3.5 m, the noise increased to 6.9 mm (sd=5.6 mm). Noise was found to differ by dimension (z values were noisier than x and y values) and joint (wrist and hand exhibited more noise than other joints). Average end-to-end latency on a machine equipped with two Intel Core2 6600 2.4 GHz processors was found to be approximately 125 ms [15]. Minimum latency has been reported to be 102 ms for V1 and 20-60 ms for V2 [1]. In all these tests users were facing a single Kinect sensor during the data acquisition.

Several authors have explored the use of multiple Kinect cameras for a number of applications ranging from real-time 3D capture of scenes [16] to gesture-based interaction [6]. The infrared dot patterns projected by multiple Kinects [10] are known to interfere with each other [9]. Several works have addressed the multi-Kinect interference problem, including hardware solutions for time-multiplexing [24, 3, 9] and software solutions [16, 6]. Maimone and Fuchs [16] have observed that the difference between the depth images with and without interferences corresponds mostly to the missing data rather than differences in depth values. They propose a modified median filter to fill missing points (using data from other units), allowing the 3D reconstruction of dynamic scenes for telepresence systems. Maimone and Fuchs also show how to combine 2D eye recognition with depth data to provide head-tracked stereo.

Caon et al. [6] use two Kinect sensors to get skeletal data from multiple users. Their skeleton fusion algorithm calculates the coordinates of every joint of the final skeleton as a weighted average of the joints at the individual skeletons, where weights are assigned according to the number of joints detected by each

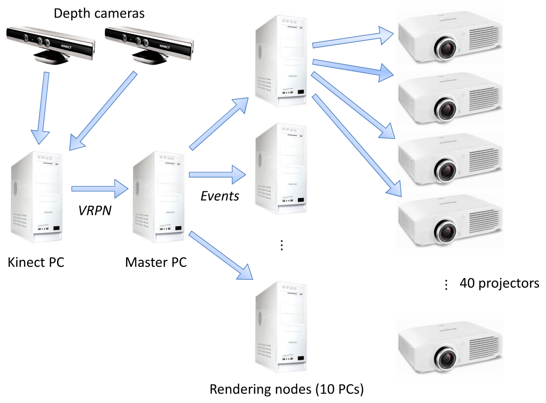


Figure 1: Architecture of the system. It consists of two Kinects controlled by a dedicated PC, a master PC which handles interaction and orchestrates rendering, 10 rendering PCs and 40 DLP projectors. The system also includes four digital cameras for calibration, not shown in the image.

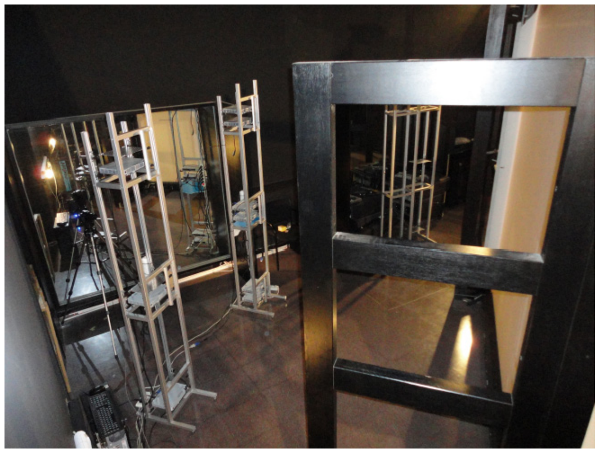


Figure 2: The supporting towers, holding 12 projectors per vertical CAVE wall.

sensor. We also use two Kinects but our fusion algorithm uses more elaborated, per-joint data to estimate position reliability.

3 System Architecture

The architecture of the system is shown in Figure 1. A first PC receives events from two tracking Kinects and sends them to the Master PC via VRPN. The Master PC sends camera events to a total of 10 rendering PCs. Rendering PCs have 12–16 GB of RAM, 2.66 GHz and two Nvidia GTX 580 boards, driving 4 projectors. Rendering PCs are network synchronized, each drawing the same scene with different cameras and viewports. Display on each vertical CAVE wall is performed by 3 PCs which manage 12 projectors. Display on the CAVE floor is performed by a single PC, the output being directed to four HD projectors. We used Stewart FilmScreen 100 screens for the CAVE walls.

The projection setup includes two towers per vertical CAVE wall, supporting a total of 12 DLP, 1024×768 projectors per wall, Figure 2. Direct (without mirrors) rear projection is used. Four digital cameras (Canon EOS 1100-D) are used for the calibration step. Cameras capturing vertical walls have been fixed to the room walls to minimize the need for repeating the first phase of the calibration step (Section 4). We use front-projection on the floor with four HD DLP projectors at the top of the front wall and a mirror. The camera that captures the floor surface is fixed at the top of the wooden structure at the CAVE entrance. A mix of 26 Mitsubishi XD550U (1024×768, 3000 lm), 12 Casio XJ-S32(1024×768, 2300 lm) and 2 Mitsubishi FD630U (1920×1080, 4000 lm) projectors is used in the present setup, see Figure 3. New DLP projectors like the BenQ MX768, the ViewSonic PJD7333, or similar models could also be used. Led-laser DLP technology, however, is not recommended, because of the interaction between polarization and color.

The overall system achieves higher resolution and brightness while significantly reducing the total construction and maintenance cost, as discussed in Section 7.4

Two Kinect devices are located at the top of the two edges between the side walls and the front wall. Our current setup uses Kinect V1 sensors (with a 57.5 horizontal fov) although we plan to test also V2 sensors (with a 70 horizontal fov). The sensors are positioned in a way that users at the center of the CAVE are in the area of maximum tracking resolution for both of them. The location of the Kinects at a height of three meters is not optimal, but it is the best option to minimize optical disturbances. Lower limbs are hardly detected, but a good tracking of the head and upper limbs is achieved. A MultiKinect software module receives the 3D coordinates of the user skeleton joints as detected by both Kinects, and merges them according to the user position and orientation, as detailed in Section 6. The VRPN client in the Master PC receives the merged skeleton, estimates the user gesture type and computes the new rendering settings for the rendering PCs.

The next three sections describe the self-calibration software, our software infrastructure for application development and the gesture interaction module.

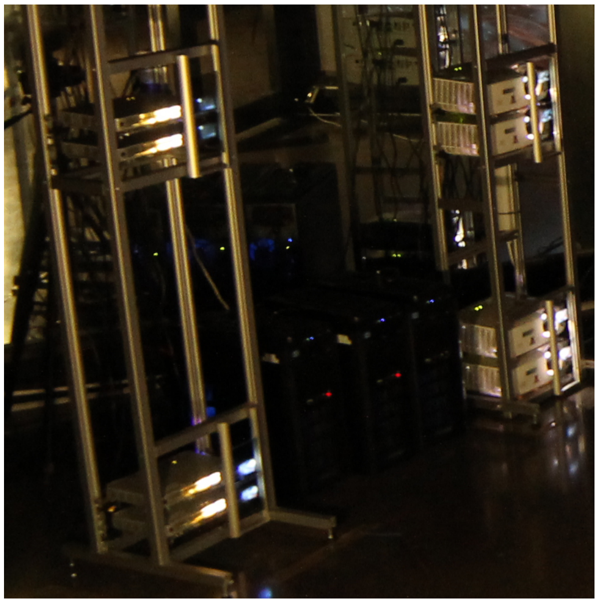


Figure 3: The auto-calibration algorithm supports mixed projectors from different suppliers, as shown in these towers holding the projectors of one wall.

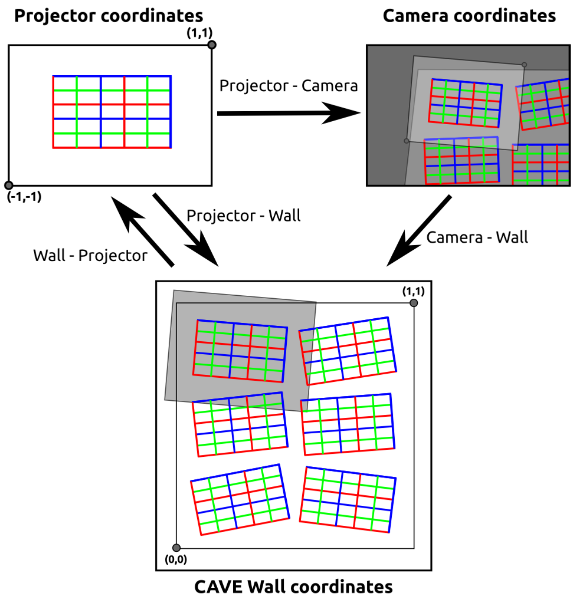


Figure 4: Geometric calibration involves three types of coordinates: projector coordinates, image coordinates and CAVE coordinates.

4 Adaptive auto-calibration system

Manual calibration in a projection system with 40 projectors is unaffordable. Solving the calibration problem requires finding a set of suitable transformations to be used in each rendering PC, the final goal being that users should perceive a unique, smooth and surrounding image along the CAVE walls. We now describe an algorithm for the automatic calibration of the whole set of projectors which succeeds in obtaining a unique image in the CAVE without user intervention. The algorithm is able to calibrate any mix of different projectors even from different suppliers, as shown in Figure 3.

The auto-calibration is an off-line process which can be run at any time (although, in our experience, once a week is sufficient). It is performed in three steps, as detailed in the next paragraphs. During calibration, projectors are made to project specific patterns on the CAVE walls which are captured by four computer-controlled cameras (one per CAVE wall). Calibration involves three types of coordinate systems, as shown in Figure 4: the projector coordinate systems (40 in total), the camera coordinate systems (four cameras in total) and the CAVE coordinate system. Figure 2 shows the setup for one of the vertical CAVE walls, involving two towers with 12 projectors and one digital camera to capture the projected patterns.

Auto-calibration output consists of a deformation matrix and a texture mask per projector. Deformation matrices will be inserted in the rendering pipeline just after the projection matrix, in order to produce a projection adapted to the targeted region in the CAVE wall. On the other hand, texture masks will be used by fragment shaders as detailed below to blend the projections of neighboring projectors. Calibration involves two basic steps: geometric calibration [7] and chromatic calibration. Deformation matrices are produced by the geometric calibration whereas the chromatic step creates the texture masks. Our method, well adapted to planar screen configurations and consumer hardware, is simpler than that proposed in [23].

Before calibration, projectors are uncoordinated and neighbor images do not match, Figure 5(a). Geometric calibration starts by detecting the eight vertices of the CAVE cube. A lamp is located inside (Figure 5(b) and images of the CAVE walls are captured by the four cameras, Figure 5(c). Camera distortions are then removed, and we use implicit information to link vertices that appear in two or in three images: the cube has 4 vertices which belong to a single wall, 2 vertices which belong to two neighbor walls and 2 vertices belonging to three walls (two vertical walls and the floor). We then estimate the positions of the relevant wall vertices in each of the undistorted camera images by computing them as the intersections of the wall edge lines in camera coordinates. These refined positions are used to compute the Homographies [7] $H(C, W)$ for each CAVE wall W. The Homographies

$H(C, W)$ convert from camera image coordinates of W to normalized CAVE coordinates (Figure 4), allowing us to use the cameras as an optical measuring system for the physical CAVE walls. We take special care in minimizing errors when computing the vertex positions in the captured wall images and the four camera-wall Homographies $H(C, W)$. Captured images of the front wall and floor have four relevant vertices each, whereas the two side walls have three of them each, as they have one unshared vertex. Proper matching among CAVE walls in the final result depends on the 2D image coordinates of these 14 relevant vertices. We therefore iterate the geometric calibration algorithm to optimize the 2D coordinates of the 14 relevant vertices: we estimate them, we complete the second geometric calibration phase, we detect mismatches between neighbor walls, and we use them to improve the estimates of the 2D image coordinates of the 14 vertices and to run again the second geometric calibration phase. We rely on the fact that 2D coordinates of the relevant vertices and Homographies $H(C, W)$ are quite stable, as they only depend on the camera and CAVE wall locations and cameras are usually well fixed. Recomputation of camera-wall Homographies $H(C, W)$ is therefore unnecessary in most cases and should only be performed from time to time. It should be observed that, once proper Homographies $H(C, W)$ have been computed, the geometric calibration problem has been decoupled and simplified: the second geometric calibration phase consists on four independent CAVE wall calibrations.

In the second phase of the geometric calibration algorithm we sequentially visit the individual CAVE walls. The algorithm for each wall runs in two iterations, the first one involving all projectors for the left eye and the second one including all projectors which generate images for the right eye. The process involves six projectors for the vertical walls and two projectors in the case of the floor. Each projector P_k displays a pattern grid with 30 vertices, as shown in Figure 5(d), and this projection is captured by the corresponding camera. After removing image distortions, the coordinates of the 30 grid vertices in the captured image are computed at subpixel resolution. We use standard image processing algorithms for line detection followed by the analytical computation of line intersections in image space. By solving a least squares problem, we compute the Homography matrix $H_{P_k,C}$ that converts from projector P_k coordinates to camera C coordinates (we have 60 equations for the 8 unknown components of $H_{P_k,C}$, as any grid vertex generates two independent equations). At this point, the Homography $H(W, P_k) = (H_{P_k,C} * H_{C,W})^{-1}$ relates points in the CAVE wall coordinates to their images in projector coordinates. Now, by just imposing suitable target positions in CAVE coordinates of the 4 corner vertices of the pattern grid, we obtain the deformation matrix for P_k which precisely ensures a projection on the desired rectangular region of the CAVE wall. Figure 7 shows the result after running the calibration algorithm on the four CAVE walls. Observe that matrices $H_{P_k,C}$ are used to calibrate the projections within each of the CAVE walls, whereas the four matrices $H(C, W)$ are responsible for seamless image continuity between neighbour walls.

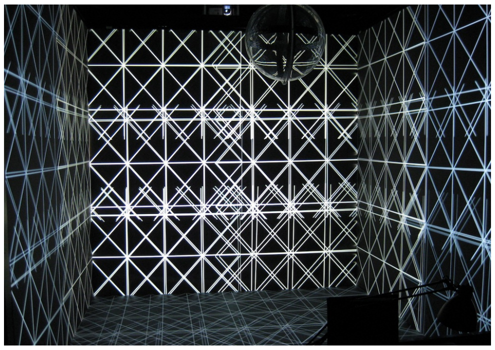
For the chromatic calibration we have implemented the algorithms from Sajadi et al. [22]. Individual color gamuts are measured for each projector and these gamuts are morphed along horizontal and vertical overlap bands. The result is a texture mask for each one of the individual projectors. During the interactive visualization, fragment colors will be multiplied by the value of the corresponding texel in the mask before being written to the frame buffer, to compensate gamut differences and to smoothly blend overlapping areas between projectors. Figure 6 shows the CAVE with two projected models after this final calibration step. The complete calibration procedure —geometric and chromatic— takes less than half an hour; most of that time is consumed by the chromatic calibration, which seldom needs to be repeated.

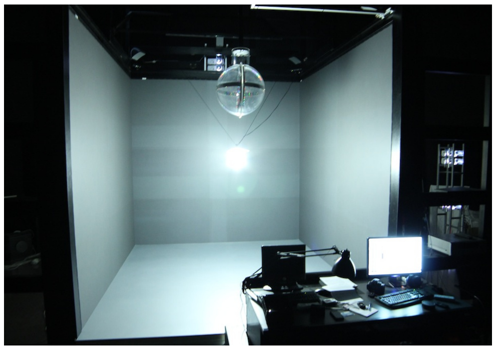
Our auto-calibration software is flexible, and it is straightforward to adapt it to the calibration of VR systems with different screen configurations in terms of size, relative positions and number of projection walls. It is also possible to use other projector combinations. For example, each wall may be served by only four FullHD projectors, splitting each wall in two halves, simplifying the architecture at the expense of brightness.

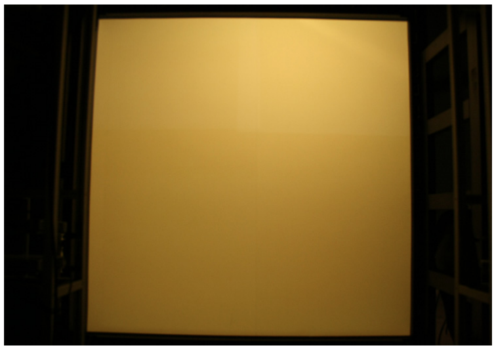
5 Software infrastructure for application development

To maximize the ease with which new developers can write software for this CAVE —or adapt old applications— we developed a generic rendering engine that abstracts the multi-projector architecture and the calibration process, providing a relatively lean API.

In fact, this API sits between the user application and whichever other software is being used for controlling the projectors, so an installation may choose to use VRJuggler [4], for instance, and another may use Unity3D [14], or yet another Qt framework (`http://www.qt.io`). (for a recent review of frameworks for VR applications see [5]). A developer may be inter-


(a)


(b)


(c)

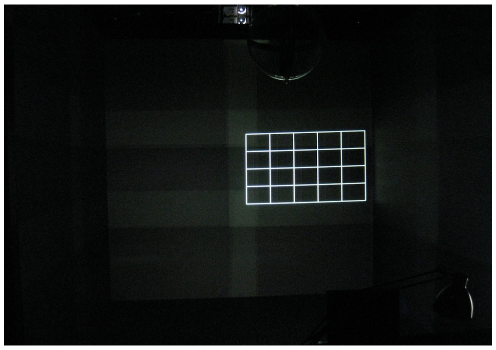

(d)

Figure 5: Calibration process: (a) uncalibrated projections, (b) interior lighting for wall image capturing, (c) capture of one CAVE wall by the corresponding camera, (d) pattern as displayed by one of the projectors.

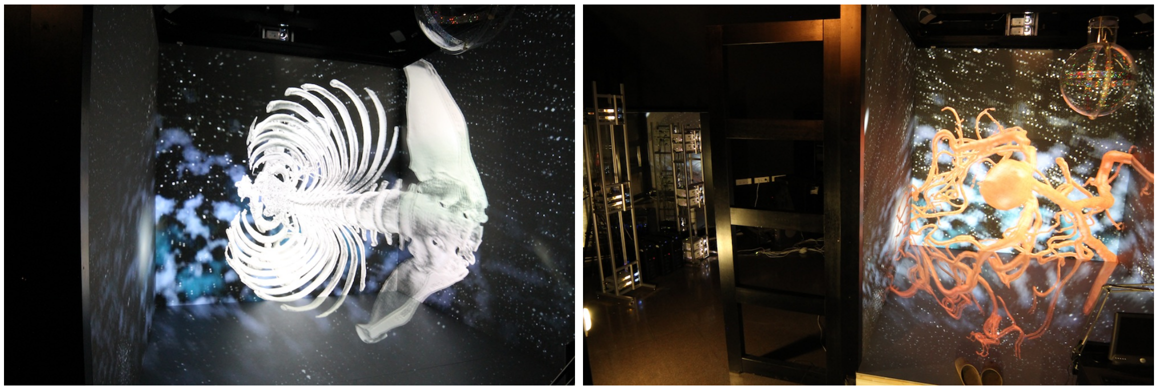


Figure 6: Two medical models, displayed after the calibration process. The camera does not have a filter, so the picture shows the left-eye and right-eye images overlapped.

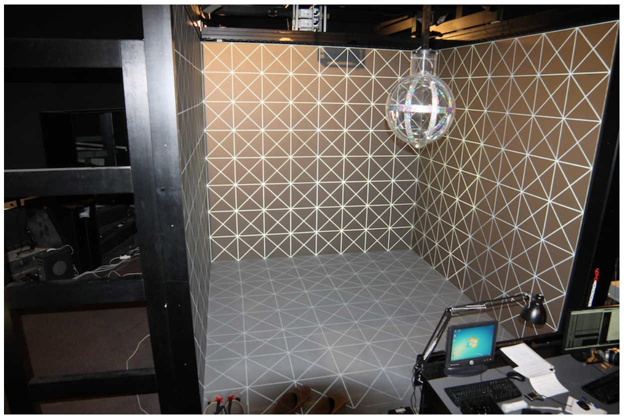


Figure 7: Patterns in Figure 5(a), redisplayed after geometric calibration.

ested in developing using one of these platforms, perhaps on a desktop, and then deploy it in the CAVE using a different one. Because of this, the abstractions the API offer have been conceived to make this unimportant and transparent.

To create a new application, the user creates an application object of class 'App' and assigns a scene and a set of events to which the application shall respond. The scene class contains implementations of all of its methods (most of them empty) so the user needs only to refine the ones relevant to his application. The two main things one must redefine are the initialization (an opportunity to initialize any resources the application will need later) and the render method (which must only take care of sending the adequate information to the pipeline; the library will otherwise concern itself with the low level details of integrating this stream with whichever rendering platform is being used).

Events are handled through a callback system that allows the user to subscribe a function to a certain event (all input and all relevant user actions trigger events that can be handled in this way. The App class provides a method *associateEvent*() to declare this connection. Furthermore, there is a plugin mechanism to reuse navigation or interaction modes. The user may dynamically load or replace these plugins to alter the behavior of the application without exiting, and may easily develop his own plugins to test new interaction modes.

Our auto-calibration process puts minimum requirements in terms of shader programming. For example, our basic shader for Unity has 7 lines of code for the vertex program (this includes the application of the deformation matrix) and 9 lines of code for the fragment program (the blending mask is encoded as a texture). Closed-source applications relying on OpenGL can also be supported via function call interception [30] to activate the appropriate shaders, although we have not experimented with this option yet.

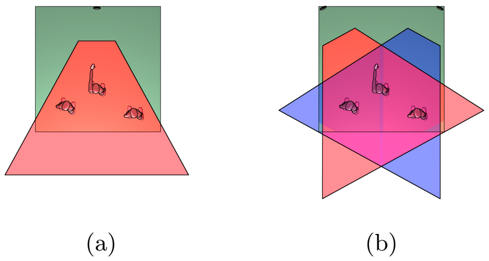


Figure 8: Interaction space with one Kinect (a) and two Kinects (b).

6 Kinect-based Gestural Interaction

In this section we discuss how we support head-tracking and gesture-based interaction in our CAVE by means of Kinect cameras. Although the Kinect does not reach the position accuracy of traditional tracking systems, it provides full-body motion capture at a much lower cost.

The Kinect V1 sensor has a field of view of 57.5 degrees (horizontal) and 43.5 degrees (vertical), with a physical depth range of 0.8 m to 4 m [20]. Depth values of objects beyond the 4 m limit can also be retrieved but at the expense of tracking noise.

6.1 Analysis of kinect placement

Figure 8(a) shows a single-Kinect setup along with the resulting interaction space. It turns out that this setup has two important limitations. On the one hand, a large portion of the CAVE is outside the field of view of the sensor, resulting in a rather limited interaction space. Since the head position provided by the Kinect is used for head-tracking, this fact limits the freedom of movement of the tracked user. On the other hand, skeletal tracking is stable when the user faces the sensor, but gets noisier and unreliable as the user turns left or right, or gets partially occluded by other users. Since CAVE users often face the side screens, a single Kinect provides unreliable tracking data even with a single user. In practice, existing CAVEs using a single Kinect force the user to face the front screen, thus severely reducing the advantages of head-tracking.

The above limitations can be addressed using additional depth cameras with overlapping fields of view. However, the infrared dot patterns projected by multiple Kinects are known to interfere each other. Experimentally we have found that using two Kinects to

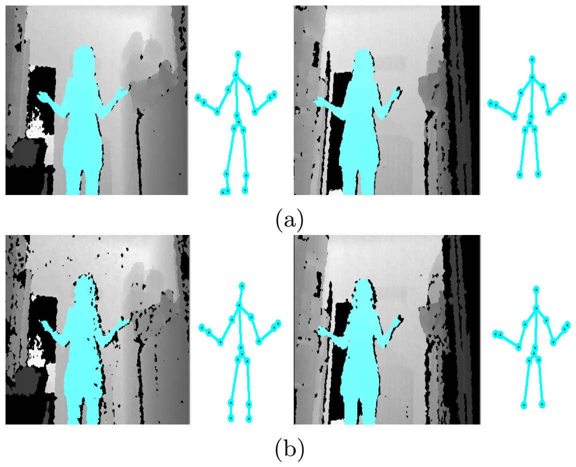


Figure 9: Depth images and reconstructed skeleton of a static user from two slightly rotated Kinects. The images in (a) were taken sequentially, with one Kinect being covered while the other captured the subject, whereas those in (b) were taken simultaneously and thus suffer from mutual interferences.

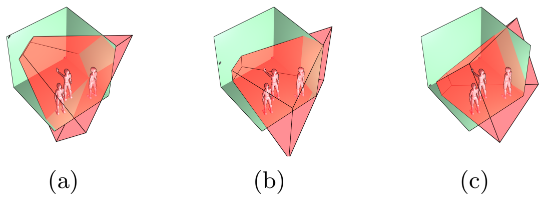


Figure 10: Different setups with two Kinects. Both Kinects are placed on the top (a), middle (b) or bottom (c) of the vertical edges of the front screen. Only one frustum is shown for clarity; the other frustum is placed symmetrically on the opposite vertical edge of the front screen.

illuminate the target area does increase noise in the depth images but only at a moderate level (having almost no effect on skeletal tracking accuracy) whereas using three or more sensors results in more severe artifacts. This is also in agreement with previous studies on Multi-Kinect interference issues [9, 6]. Figure 9 illustrates the effect of such mutual interference on the depth images. Despite the depth images in Figure 9(b) are clearly noisier, the skeletal tracking of the Kinect [25] seems to be quite resilient to noise, as evidenced by the nearly identical skeletons reconstructed with and without interference noise.

Since we aim at improving both the physical area covered by the Kinects and the stability of gesture recognition, a reasonable option is to place the Kinects on the edges of the front screen (Figure 10). We tested several combinations of height and tilt angles: (a) at 3 m from the floor with a tilt angle of -21.75 degrees (half the vertical field of view, Fig. 10a), (b) at 1.8 m with no tilt rotation (Fig. 10b), and (c) at 0.04 m from the floor with a tilt angle of 21.75 degrees (Fig. 10c). In all three cases the rotation around the vertical axis was approximately $\pm$28.75 degrees (half the horizontal field of view of the Kinect), as shown in Figure 8b. Since we aim to get skeletal data through the Kinect for Windows SDK, which predicts 3D positions of body joints through a classifier optimized for people facing the sensor [25], we did not consider alternative configurations (such as placing one Kinect on top of the CAVE looking downwards).

An informal user study with a variety of selection, manipulation and navigation gestures revealed that placing the Kinects at floor level (configuration c) is the worst option, because the head is often occluded by the arms, both during interaction gestures and inadvertent gestures. The other two options provided more robust skeletal tracking. In our experiments we adopted configuration (b) because the physical space with full-body tracking better matches the CAVE space, and because it better mimics the front-facing scenario the Kinect software body-part classifier has been trained for [25].

6.2 Gesture interaction with two kinects

The relative pose between the two Kinects can be determined in a number of ways. We adopted the joint depth and color calibration procedure described in [13] which, as most camera calibration procedures, is based on capturing a checkerboard pattern at different poses. We first register both Kinects with respect to a high-resolution camera (placed at the center of front screen), and then we compose the resulting rigid transformations to get the relative pose between the Kinects.

Since each Kinect provides its own skeleton reconstruction, the two skeletons need to be combined into a single one before joint data can be delivered to the VR application. Let P_i^L (resp. P_i^R) be the (x, y, z) position of the i-th joint of the skeleton as reported by the Kinect placed on the left (resp. right) edge. From now on we will use P_i^j with $j \in \{L, R\}$ to refer to these joint positions. We use a simple skeleton combination algorithm based on assigning a confidence value to each joint. This confidence value is the product of the confidence values described below.

We define the *detection confidence* of P_i^j as

$$cf_D(P_i^j) = 2^{-n(P_i^j)} \quad (1)$$

where $n(P_i^j)$ is the number of frames the i-th joint has not been detected by the j-th Kinect (this value

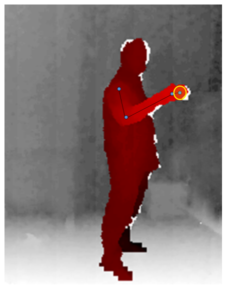


Figure 11: Depth image from a Kinect sensor. The right hand appears unoccluded, and thus most depth values in the vicinity (yellow circle) approximately match the depth of this joint as reported by the Kinect software. Conversely, the left hand is completely occluded; if the Kinect software still estimates its position behind the user, its visibility confidence value would be close to zero.

is reset when the joint is detected again). The joint detection flag is provided by the Kinect software. Notice that Equation 1 halves the confidence of a joint each time it is not detected.

We define the *within-frustum confidence* of P_i^j as

$$cf_F(P_i^j) = 1 - \max(0, (|\text{NDCx}(P_i^j)| - 0.8)/1.1, \quad (2)$$

where $\text{NDCx}(P_i^j)$ returns the x coordinate of point P_i^j in the normalized device space of the j-th Kinect. The bounds 0.8 and 1.1 have been found empirically. Equation 2 assigns a confidence value of 1 to joints with NDC $|x|$ in $[0, 0.8]$ (thus reasonably apart from the frustum boundary) and linearly decreases the confidence to 0 in the interval $[0.8, 1.1]$. We use a 1.1 value (which indicates that the joint is actually outside the frustum) because when some body parts leave the frustum, the Kinect software often still continues to predict the clipped parts, but with much less accuracy.

Let A_i^j be the screen projection of a small sphere centered at P_i^j with respect to the j-th Kinect (we used a sphere with a radius of 4 cm in our tests). The *visibility confidence* $cf_V(P_i^j)$ is defined as the ratio of pixels (labeled as part of the user) in A_i^j whose depth value (from the j-th Kinect's depth image) is less than the depth value corresponding to P_i^j. In the comparison we use a 12 cm offset to account for the fact that the Kinect returns joints at some learnt distance from the body surface, rather than on the surface [25]. This confidence value attempts to identify poorly-recognized joints due to self-occlusion (see Figure 11).

We also compute an *alignment confidence* value that measures roughly to which extent the upper torso of

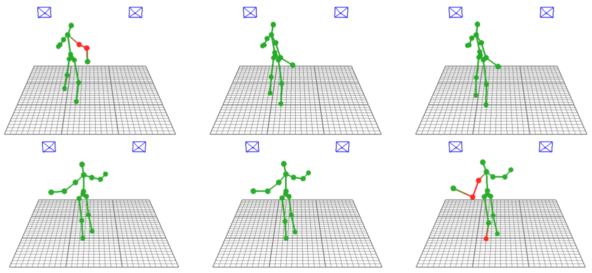


Figure 12: Pose reconstruction examples. The skeleton shown on the left (resp. right) corresponds to the Kinect located on the left (resp. right) edge. The middle skeleton is the combined skeleton. Joints in red correspond to non-detected nodes (shown at the last known position). In (a), the left Kinect is unable to detect the right arm due to occlusion, but the combined skeleton uses joint data from the other Kinect. A symmetric situation is shown in (b), this time affecting also the left ankle. In both cases the combined skeleton provides an accurate estimation of the pose.

the j-th skeleton is facing the sensor. We compute $cf_A(P_i^j)$ as $1 - |\vec{s_j} \cdot \vec{v_j}|$ where $\vec{s_j}$ is the vector joining the left and right shoulder joints (according to the j-th Kinect), and $\vec{v_j}$ is the view vector of the j-th Kinect.

Finally, the confidence value of a joint $cf(P_i^j)$ is the product of the four confidence values defined above. The combined position of the i-th joint P_i is defined as the weighted average $(cf(P_i^L)/W)P_i^L + cf(P_i^R)/W)P_i^R$, where $W = cf(P_i^L) + cf(P_i^R)$.

Source code and binaries of our software for combining two or more Kinect skeletons is publicly available [28]. For maximum portability, the skeletal tracking is implemented as a VRPN [27] server, allowing an arbitrary number of clients (running on Linux in our CAVE) to connect to the VRPN server (currently running on Windows 7 and Kinect for Windows 1.8) to receive joint data. Figure 12 shows different pose estimations from two Kinects configured as in Figure 10(b).

7 Results and Discussion

7.1 Calibration

Geometric calibration results are shown in Figure 7 and in the accompanying video. The feedback from users about the calibration is discussed in section 7.3.

Regarding chromatic calibration, however, there is still room for improvement, as discussed in Section 8. The main problem is related to the differences between wall and floor projections, as we use four HD DLP projectors for front-projection on the floor and 12 stan-

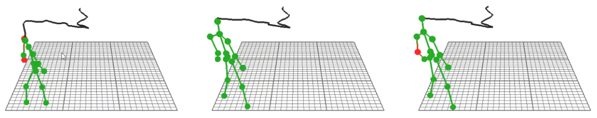


Figure 13: Path followed by the head of a moving user. The middle skeleton is the combined skeleton, which is resilient to the severe head misplacements when the user is partially outside the Kinect's frustum.

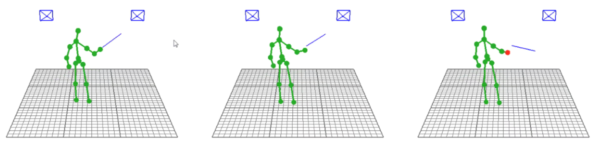


Figure 14: Pointing with two Kinects. The combined skeleton (middle) is resilient to non-detected joints on one of the Kinects.

dard 1024 x 768 DLP projectors in the back of each vertical wall. The result is that the floor is darker than the walls as seen in Figure 6. We have decided to accept some degree of brightness discontinuity between walls and floor in order not to darken excessively the wall projectors, with the goal of keeping the maximum CAVE brightness. Tuning this brightness discontinuity can be a good avenue for future research. However, the main fact is that the brightness discontinuity does not affect the immersive experience. In our user tests, brightness and chromatic calibration discontinuities were perceived by less than 2% of the users.

7.2 Interaction

We now discuss some performance measures of the Kinect-based tracking system in our CAVE. For the experiments we used two Kinects V1 placed on the vertical edges of the front screen, as in Figure 10(b).

The physical space with head tracking (that will be referred to as *target space*) covers the most relevant portion of the CAVE. For an average adult user standing up, head tracking is available in 78% of the CAVE floor (about $7\,\mathrm{m}^2$). This contrasts with the 53% ($4.7\,\mathrm{m}^2$) available in a single-Kinect configuration.

Figure 13 shows the path followed by the head of a moving user. The relative pose of the Kinects guarantees robust head tracking also near the side walls of the CAVE (see the accompanying video).

We did not measure head-tracking accuracy since a minor shift in the head location does not cause important artifacts in the rendered images. Instead, we compared the noise of the head position coordinates of a still user adopting several poses. We measured the noise as $\sqrt{\sigma_x^2 + \sigma_y^2 + \sigma_z^2}$ where σ_x is the standard deviation of the x coordinate of the reported head position in a 400 ms window (12 samples on a 30fps Kinect). We found the average head noise within the target space to be less than 0.1 mm, but this increased up to 5 mm in poses with the arms partially occluding the head. Using data from a single Kinect also provided an average deviation of about 0.1 mm (on a restricted target space though), but noise increased up to 56 mm in the problematic poses, making the single-Kinect option less usable.

We also evaluated the suitability of Kinect-based tracking for virtual pointing, as this is the prevalent interaction metaphor for 3D selection tasks [2]. Again, the use of two Kinects was found to be of great value (Figure 14). We conducted an informal user study to measure the accuracy of the path followed by the pointing tool (a virtual ray) when the user was asked to follow a predefined path (both the predefined path and the pointing tool were shown on the screen). The path consisted of five horizontal segments at elevation angles ± 45, ± 22.5 and 0, with heading angle $-45 \leq \theta \leq 45$. We tested three different pairs of joints for defining the pointing direction: the elbow-wrist pair, the shoulder-wrist pair, and the hip-wrist pair. We will refer to these conditions as ELBOW, SHOULDER and HIP. The hip point considered was the central hip joint returned by the Kinect, whereas the elbow, shoulder and wrist correspond to the user's dominant hand, which was selected manually. Figure 15 shows the paths followed by the pointing tool for a representative right-handed user. We measured the standard deviation of the angle between the actual pointing vector and its closest match in the path. The ELBOW condition was the noisiest option, with an average deviation among the six participants of $\sigma = 2.4$ degrees. The SHOULDER condition was more stable ($\sigma = 2.0$ deg), although it still had significant jittering, especially when pointing at or above the horizon. The HIP condition was slightly more stable ($\sigma = 1.7$ deg).

In practice, the suitability of these options for 3D selection depends on a number of factors [2]. When pointing accuracy is not critical (e.g. large, unoccluded targets) the ELBOW condition is a suitable option requiring little physical effort from the user. In other scenarios the SHOULDER or HIP conditions are more suitable, at the expense of less natural movements and higher physical effort.

In terms of gesture-based interaction, our two-Kinect configuration provides two significant advantages: (a) a relaxation of the front-facing constraint, increasing by about 28 degrees the range of orientations for which most gestures can be recognized (with

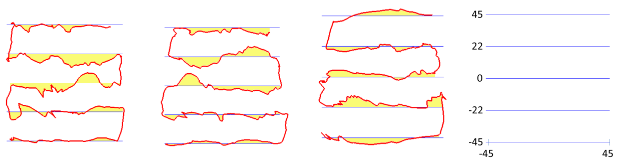


Figure 15: Path followed by the pointing tool using the ELBOW (a), SHOULDER (b) and HIP (c) conditions. We use an equirectangular projection to map unit vectors to the plane (d).

respect to a single Kinect), and (b) a pose estimation more stable on the presence of other users inside the CAVE, gestures being recognized as long as the joints involved are seen by at least one of the two Kinects. According to our experiments, most typical gestures can be recognized within a ±90 degree rotation with respect to the front-face direction, which in practice is enough for casual CAVE interaction. The accompanying video shows how the combined skeleton facilitates pose reconstruction for different gestures and at different facing conditions.

7.3 User feedback

A large number of persons (well over a hundred) have tested our CAVE enviroment over the last months. The vast majority of these users were not practitioners. Across the board, the informal feedback gathered validates both the geometric calibration and image blending, as no users complained of any defects, and when asked, they concurred that the visualization was satisfactory and perceptually correct.

The immersive experience was usually rated as 'high', and users were only aware of the chromatic calibration discontinuities at the end, after having been informed by us of the problem. Our main conclusion is that users are more tolerant to chromatic calibration biases during immersive experiences, and that perceptive metrics for chromatic calibration discontinuities in immersive VR settings should be investigated.

Users with previous CAVE experiences unanimously considered that this Multi-resolution CAVE was brigther than the previous ones, with a resolution and visual quality significantly higher. The main comment was related to the gesture-based interaction, which is still not intuitive, and to the Kinect latency.

7.4 Cost analysis

We have compared the multi-projector CAVE system presented in this paper with a standard commercial system. This commercial system uses active stereo with four projectors and four front surface mirrors. Synchronization is achieved through Nvidia Quadro boards with Nvidia Sync cards. The results are shown in Table 1.

The table shows that we achieve a much higher luminosity (with a factor of ≈ 5.25) at a higher resolution while having a significantly reduced cost. The projectors maintenance cost is the cost of replacing a projector when it crashes. This cost is remarkably low in the proposed system as a consequence of using off the shelf DLP projectors. Moreover, we have observed that circular polarization in conjunction with bright images projected in the walls produces a high immersive perception and a significant presence level, probably similar to active stereo VR systems. Experiments to confirm this last hypothesis will be part of our future work.

8 Conclusions and Future Work

We have presented a novel four wall multi-projector CAVE architecture, powered by 40 off the shelf projectors and controlled by 12 PCs. It operates in passive stereo, providing high brightness at 2000×2000 pixel resolution on each of the four walls. We have achieved high resolution while significantly reducing the cost and having increased versatility: the system works with any mix of a wide range of projector models that can be substituted at any time for more modern or cheaper ones. The uniformity of the final image is achieved using a specially designed self-calibration software which guarantees concordance and continuity. Interaction is intuitive and cable-less, operating on a gesture-based, ergonomic interaction paradigm. We dynamically merge the information from two nearly-orthogonal Kinect sensors. Having a two-Kinect configuration results in a more stable pose estimation and a relaxation of the front-facing constrain. Most typical gestures are well recognized within any face and body rotation up to 90 degrees with respect to the frontal CAVE direction.

In the future we plan to optimize the chromatic calibration differences between wall and floor projections, in order to have a maximum brightness in the vertical walls while keeping a high immersive perception. Perceptive metrics for chromatic calibration discontinuities in CAVE settings should also be investigated. We also plan to position the Kinects in configuration (b) —Figure 10— for optimal gesture recognition, as well as to test Kinect V2 sensors.

	Cost of projectors and computers	Cost of projectors maintenance	Resolution per wall	Lumens per wall
Standard commercial CAVE	299 K€	n.a.	1920×1200	4000
Multi-projector CAVE	52 K€	0.6 K€	2000×2000	≈ 21000

Table 1: Comparison between a commercial CAVE-like system and the proposed setup

Acknowledgements

We would like to thank Oscar Argudo, Pedro Hermosilla, Jordi Moyés, Isabel Navazo and Adrià Vernetta for their valuable contributions. This work has been partially funded by the Spanish Ministry of Science and Innovation under grants TIN2010–20590–C01–01 and TIN2014–52211–C2–1–R.

References

[1] Clemens Amon and Ferdinand Fuhrmann. Evaluation of the spatial resolution accuracy of the face tracking system for kinect for windows v1 and v2. In *Proceedings of the 6th Congress of Alps-Adria Acoustics Assosiation*, October 2014.

[2] Ferran Argelaguet and Carlos Andujar. A survey of 3d object selection techniques for virtual environments. *Computers & Graphics*, 37(3):121 – 136, 2013.

[3] Kai Berger, Kai Ruhl, Yannic Schroeder, Christian Bruemmer, Alexander Scholz, and Marcus A Magnor. Markerless motion capture using multiple color-depth sensors. In *VMV*, pages 317–324, 2011.

[4] Allen Bierbaum, Christopher Just, Patrick Hartling, Kevin Meinert, Albert Baker, and Carolina Cruz-Neira. Vr juggler: A virtual platform for virtual reality application development. In *Virtual Reality, 2001. Proceedings. IEEE*, pages 89–96. IEEE, 2001.

[5] Rozenn Bouville, Valérie Gouranton, Thomas Boggini, Florian Nouviale, and Bruno Arnaldi. #FIVE : High-Level Components for Developing Collaborative and Interactive Virtual Environments. In *Eighth Workshop on Software Engineering and Architectures for Realtime Interactive Systems (SEARIS 2015), conjunction with IEEE Virtual Reality (VR)*, Arles, France, March 2015.

[6] Maurizio Caon, Yong Yue, Julien Tscherrig, Elena Mugellini, and Omar Abou Khaled. Context-aware 3d gesture interaction based on multiple kinects. In *AMBIENT 2011, The First International Conference on Ambient Computing, Applications, Services and Technologies*, pages 7–12, 2011.

[7] H. Chen, R. Sukthankar, G. Wallace, and K. Li. Scalable alignment of large-format multi-projector displays using camera homography trees. In *IEEE Visualization 2002*, pages 339–346, 2002.

[8] Thomas A. DeFanti, Gregory Dawe, Daniel J. Sandin, Jurgen P. Schulze, Peter Otto, Javier Girado, Falko Kuester, Larry Smarr, and Ramesh Rao. The starcave, a third-generation CAVE and virtual reality optiportal. *Future Generation Computer Systems*, 25(2):169 – 178, 2009.

[9] Florian Faion, Simon Friedberger, Antonio Zea, and Uwe D Hanebeck. Intelligent sensor-scheduling for multi-kinect-tracking. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 3993–3999. IEEE, 2012.

[10] D. Freedman, A. Shpunt, M. Machline, and Y. Arieli. Depth mapping using projected patterns. US patent 2010/0118123, 05 2010.

[11] Hermann Fürntratt and Helmut Neuschmied. Evaluating Pointing Accuracy on Kinect V2 Sensor. In *Proceedings of the 2nd International Conference on Human-Computer Interaction*, pages 124–1–124–5, August 2014.

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

[13] Daniel Herrera C., Juho Kannala, and Janne Heikkilä. Joint depth and color camera calibration with distortion correction. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(10):2058–2064, 2012.

[14] J. Jerald, P. Giokaris, D. Woodall, A. Hartbolt, A. Chandak, and S. Kuntz. Developing virtual reality applications with unity. In *IEEE Virtual Reality (VR), 2014 Tutorials*, pages 1–3, March 2014.

[15] M.A. Livingston, J. Sebastian, Zhuming Ai, and J.W. Decker. Performance measurements for the microsoft kinect skeleton. In *Virtual Reality Short Papers and Posters (VRW), 2012 IEEE*, pages 119–120, 2012.

[16] A. Maimone and H. Fuchs. Encumbrance-free telepresence system with real-time 3d capture and display using commodity depth cameras. In *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*, pages 137–146, 2011.

[17] Aditi Majumder and M Gopi. Modeling color properties of tiled displays. *Computer Graphics Forum*, 24(2):149–163, 2005.

[18] Aditi Majumder and Rick Stevens. Color nonuniformity in projection-based displays: Analysis and solutions. *IEEE Transactions on Visualization and Computer Graphics*, 10(2):177–188, March 2004.

[19] Aditi Majumder and Rick Stevens. Perceptual photometric seamlessness in projection-based tiled displays. *ACM Trans. Graph.*, 24(1):118–139, January 2005.

[20] Microsoft Corporation. *Kinect for Windows. Human Interface Guidelines v1.8*, 2013.

[21] R. Raskar, M.S. Brown, Ruigang Yang, Wei-Chao Chen, G. Welch, H. Towles, B. Scales, and H. Fuchs. Multi-projector displays using camera-based registration. In *Proceedings of Visualization'99.*, pages 161–522, 1999.

[22] Behzad Sajadi, Maxim Lazarov, M. Gopi, and Aditi Majumder. Color seamlessness in multi-projector displays using constrained gamut morphing. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1317–1326, 2009.

[23] Behzad Sajadi and Aditi Majumder. Autocalibration of multiprojector CAVE-like immersive environments. *IEEE Transactions on Visualization and Computer Graphics*, 18(3):381–393, March 2012.

[24] Yannic Schröder, Alexander Scholz, Kai Berger, Kai Ruhl, Stefan Guthe, and Marcus Magnor. Multiple kinect studies. Technical Report 09-15, ICG, October 2011.

[25] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124, 2013.

[26] Evan A. Suma, David M. Krum, Belinda Lange, Sebastian Koenig, Albert Rizzo, and Mark Bolas. Adapting user interfaces for gestural interaction with the flexible action and articulated skeleton toolkit. *Computers & Graphics*, 37(3):193 – 201, 2013.

[27] Russell M Taylor II, Thomas C Hudson, Adam Seeger, Hans Weber, Jeffrey Juliano, and Aron T Helser. Vrpn: a device-independent, network-transparent vr peripheral system. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 55–61. ACM, 2001.

[28] Miguel Angel Vico. Multi-kinect VRPN server, 2013. github.com/mvm9289/multikinect.

[29] Zhengyou Zhang. Microsoft kinect sensor and its effect. *Multimedia, IEEE*, 19(2):4–10, 2012.

[30] David J. Zielinski, Ryan P. McMahan, Solaiman Shokur, Edgard Morya, and Regis Kopper. Enabling Closed-Source Applications for Virtual Reality via OpenGL Intercept-Based Techniques. In *IEEE Workshop on Software Engineering and Architectures for Realtime Interactive Systems (SEARIS)*, pages 59–64. IEEE, March 2014.