A Multi-viewer Tiled Autostereoscopic Virtual Reality Display

Robert Kooima[1], Andrew Prudhomme[2], Jurgen Schulze[2], Daniel Sandin[2,3], and Thomas DeFanti[2]

[1]Center for Computation & Technology, Louisiana State University (LSU)
[2]California Institute for Telecommunications and Information Technology, University of California San Diego (UCSD)
[3]Electronic Visualization Laboratory, University of Illinois at Chicago (UIC)

Abstract

Recognizing the value of autostereoscopy for 3D displays in public contexts, we pursue the goal of large-scale, high-resolution, immersive virtual reality using lenticular displays. Our contributions include the scalable tiling of lenticular displays to large fields of view and the use of GPU image interleaving and application optimization for real-time performance. In this context, we examine several ways to improve group-viewing by combining user tracking with multi-view displays, and we evaluate channel cross-talk mitigation and depth compression for improved lenticular performance and flexibility.

CR Categories: I.3.1 [Computer Graphics]: Hardware Architecture—Three-dimensional displays; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual reality

Keywords: autostereoscopic, lenticular, immersive, multi-user, virtual reality

1 Introduction

Active stereo viewing technologies hold a valued place in the research laboratory, and passive stereo viewing technologies continue to see increasing adoption in movie theaters and other stable display venues. Yet, there remain many contexts in which the inconvenience of stereo glasses preclude the use of stereoscopic display. Examples include museums, where one or more exhibits in a traditional gallery may benefit from 3D visualization, but where the expected viewer engagement with each is not sufficient to warrant the expense of handling, maintaining, and securing a large collection of stereo glasses, or of cajoling visitors into putting them on and taking them off.

Autostereoscopic display is a viable solution in such contexts. Several multi-viewer, multi-channel autostereoscopic displays are currently on the market, but their small size limits their areas of application. Before such displays will find broader use they must be scaled up to provide wider comfortable viewing areas for larger groups of people.

Our research pursues this goal. Using large arrays of small, off-the-shelf lenticular displays, we have installed a number of scalable autostereoscopic display systems capable of serving sizable groups.

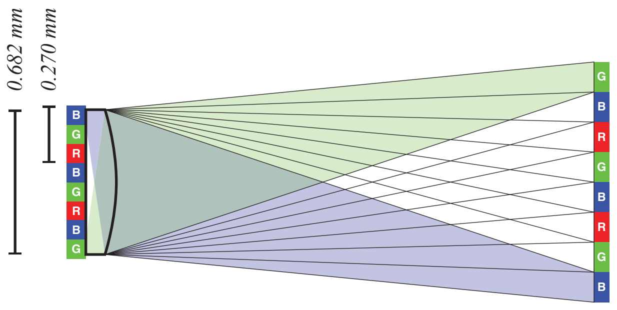

Figure 1: *Detail of an individual lenticule with one scan-line of its underlying sub-pixel grid (left). Each of the eight sub-pixels is magnified and focused to a different area on the plane of focus (right).*

We have devised and implemented an efficient, GPU-accellerated mechanism for performing real-time rendering to lenticular displays, detailed in Section 2. We have established processes for superimposing the autostereo functionality of individual displays and synchronizing rendering to them, resulting in large arrays that behave as a single display, described in Section 3. We have experimented with active measures for the mitigation of the fundamental discontinuities in lenticular displays using user tracking, discussed in Section 4. Finally, we have tested a variety of methods for reducing the visibility of channel-crosstalk and resulting contrast issues common to multi-view lenticular displays, discussed in Section 5.

2 Lenticular Rendering

We begin with an overview of the basics of lenticular display technology sufficient to support the subsequent discussion. We use the Alioscopy 24″ 3DHD as a specific example, though our approaches have been demonstrated on a variety of similar lenticular and parallax barrier displays. This particular display is capable of presenting eight independent image channels, projecting them onto a plane of focus at a distance of $2.95\,m$. Other displays have different sizes, resolutions, lenticular pitches, focal lengths, and channel counts, but the principles remain the same.

2.1 The Lenticular Array

The foundation of the 24″ Alioscopy is a 1920×1200 LCD panel. An array of cylindrical lenses is affixed to the display surface, as shown in Figure 1. Each sub-pixel is magnified by the lenticule and focused in a different direction. As a result, primarily one of these eight sub-pixels is visible from any given view point at the plane of focus.

Apparent in Figure 1, the pitch of the lenticular array differs slightly from the pitch of the sub-pixel grid, but registration of the two is achieved in the perspective projection. The offset of the lenticular array shifts with respect to the sub-pixel grid over the width of the LCD, which has the effect of shifting the direction of focus. In this

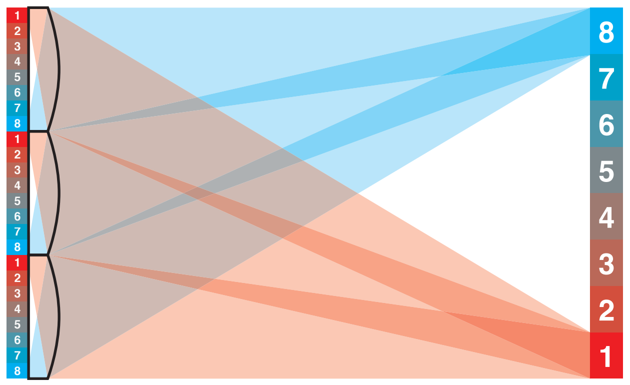

Figure 2: *The off-axis focus of individual lenticules (left) causes all sub-pixels of each channel to converge to a common area on the plane of focus (right).*

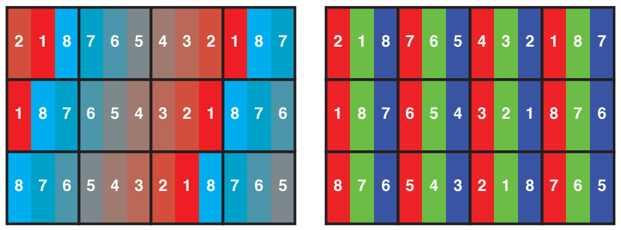

Figure 3: *The mapping of channels onto sub-pixels, showing the* 18.435° *angle (left) and resulting sub-pixel color distribution (right).*

way, the 750+ lenticules converge each of the eight channels to the same area on the plane of focus, as shown in Figure 2.

On the plane of focus, each channel has a projected width of 64 *mm*. This corresponds to the average interocular distance for a human user. Thus, a user at the plane of focus will see one channel in his left eye and an adjacent channel in his right eye. In this way, autostereoscopic viewing is achieved.

Figure 1 shows that each channel has only one sub-pixel per lenticule per scan-line, and thus only one color. To allow all three colors per channel, the lenticular array is rotated relative to the pixel grid and each channel moves one sub-pixel horizontally for each vertical scan-line, as shown in Figure 3 (left). Thus all three colors are present in each set of three scan-lines, Figure 3 (right), evenly distributing the color, averaging to gray, and eliminating color fringing artifacts, and reducing the moiré that would otherwise result from the overlay of the lenticular grid and the pixel grid. This rotation is common in lenticular display design [van Berkel and Clarke 1997].

The lenticular array projects multiple images of its underlying pixel grid. The behavior is similar to Figure 4, which depicts the area covered by the central projection of a single channel, or "primary lobe," with one repetition, or "secondary lobe," to each side. The array of channels of the 24″ Alioscopy repeats continuously to the left and right at eight times the interocular distance. This repetition widens the effective useful viewing area of the display and enables use by multiple simultaneous users, though it introduces a discontinuity in the otherwise smoothly-varying field of views.

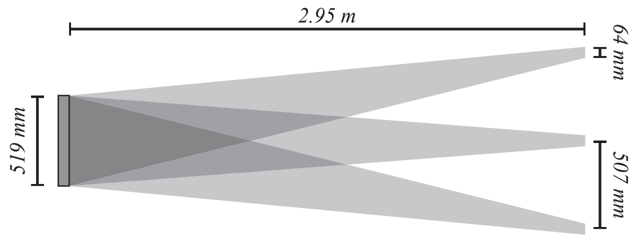

Figure 4: *The coverage of a single channel of the display, with repeating lobes to the left and right.*

2.2 Real-time Rendering

We take real-time interactivity to be a defining characteristic of virtual reality, and rendering efficiency is critically important to the maintenance of interactive frame rates. The lenticular display system we have just defined poses a challenge to rendering efficiency due to the need to render the scene from multiple different view points, plus the added cost of interleaving the individual sub-pixels of these renderings in accordance with the layout of the lenticular array. Despite this complexity, we achieve real-time display using the programmable graphics processor (GPU).

The approach is a generalization of a GPU-based algorithm for parallax barrier autostereo rendering presented at IEEE Virtual Reality 2007 [Kooima et al. 2007]. That algorithm has already been ported by a number of users into a variety of visualization systems and platforms, and we have further generalized it in two ways, both of which address issues raised by the translation from the original work's single-user Varrier platform to the multi-user lenticular.

First, as a single-user system, the Varrier presents exactly two image channels. The lenticular implementation generalizes this to an arbitrary number of views, limited only by the number of texture sampler units supplied by the graphics hardware.

Second, as a user-centered motion-tracked system, the Varrier allows the mitigation of channel cross-talk with the interleaving of dark pixels between channels. This "guard-band" helps eliminate the "ghost" effect, where-in high-contrast content rendered to one channel remains faintly visible in another, ultimately reducing the potential depth of the displayed scene. The common configuration of a multi-user lenticular is untracked, so the display cannot optimize its rendering to the current view position. A user must be allowed to move from one channel to the next with minimal discontinuity. Inter-channel guard-band pixels are thus to be avoided, and alternative methods for ghost mitigation must be sought. Toward this end, a configurable mechanism for the blending of adjacent channels is introduced.

2.2.1 Scene Rendering

Prior to display on-screen, the 3D scene is rendered once per channel into off-screen render buffers. The most straightforward approach is to loop over all channels, bind a frame buffer for each, set the appropriate perspective projection for the channel, and render normally. This approach places a minimum of graphics state requirements upon the application, and simplifies the porting of existing 3D applications for lenticular display.

There are a number of potential optimizations. The first of these is the recognition that only the color buffer of each of these frame buffers is needed by the image interleaving process, and thus only one depth buffer is necessary for all rendering. This optimization is

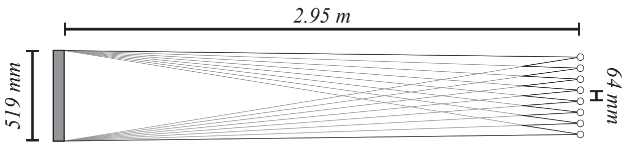

Figure 5: *The view positions (right) and off-axis perspective frusta used to render the eight channels for the* 24″ *Alioscopy (left).*

significant for texture-heavy or otherwise VRAM-limited applications.

Another optimization follows from the use of geometry shading and multiple render targets. The scene is rendered only once, and a geometry shader replicates each scene primitive for each channel, applying the appropriate perspective projection and routing each to its destination render target. In this way, the total vertex cost of lenticular rendering is reduced to equal that of normal rendering, which is significant for vertex-limited applications. This optimization is unfortunately incompatible with the elimination of redundant depth buffers. In our work, we choose to eliminate redundant depth buffers instead of redundant vertex processing.

Perhaps the most significant optimization recognizes that only a fraction of rendered sub-pixels will ultimately appear on-screen. In the case of the 24″ Alioscopy, only one out of every eight sub-pixels is used. It follows that the off-screen render buffer may be shrunk to one eighth its width without a perceptible loss of quality. In this way, the total fragment cost of the lenticular rendering is equal to the fragment cost of normal display, which is significant for fill-limited applications.

Given an understanding of the nature of the bottleneck in any given rendering application, a selection of these optimizations can eliminate the overhead normally associated with rendering from multiple view points. This leaves only the necessary cost of the final on-screen pass performing the image interleaving, described next.

2.2.2 Basic Interleaving

The sub-pixel interleaving of rendered scene images is determined by two sets of parameters. The first is the set of all view positions. In the default case, this is a static array of points on the plane of focus, equally distributed, separated from one another by the interocular distance. These positions give the centers of the projected channels, as depicted in Figure 5. The use of user tracking may cause these positions to be dynamically recomputed per frame, as we will discuss in Section 4.

The second set of interleaver parameters gives a precise definition of the physical nature of the lenticular array. In the simplest case, this definition has five terms, described here along with example values for the 24″ Alioscopy. *Pitch* gives the size of each lenticule ($0.682\,mm$). *Angle* gives the rotation of the lenticular array relative to the pixel grid ($18.435°$). *Duty cycle* gives the fraction of a lenticule dedicated to each channel ($1/8$). *Optical thickness* gives the effective focal length of the lenticular array ($\approx 4.2\,mm$). *Shift* gives a left-to-right calibration value that tunes the direction of focus, allowing multiple lenticular arrays to be brought into phase with one another ($\pm 0.01\,mm$).

Referring again to Figure 3, it is clear that the channel interleaving is equivalent to the *sum* of all channels, with each channel *modulated* by an appropriate mask, light where a channel is visible and dark where it is not. Interleaving is reduced to the computation of that mask.

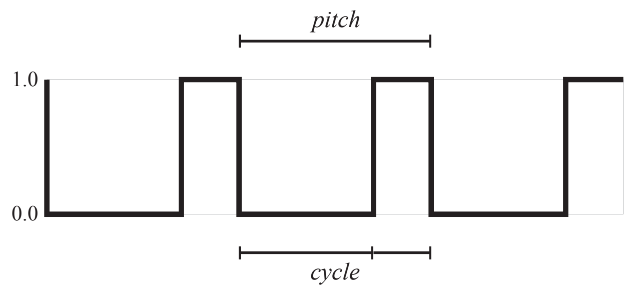

Figure 6: *The basic interleaver waveform*

A channel mask is two-dimensional. But in the coordinate system rotated into alignment with the lenticular array, it is vertically constant and horizontally repeating. From this perspective, the mask is one-dimensional and may be represented by a waveform, with the lenticular pitch determining the wavelength and the duty cycle determining the ratio of light to dark. This is shown in Figure 6.

The process of interleaving follows from this interpretation. Each LCD sub-pixel has a position in the modulating waveform, called its *phase*. This phase is entirely determined by the position of the sub-pixel on the screen, the view position of the associated channel, and the lenticular parameters. We compute this phase per sub-pixel and evaluate the waveform as a simple step function, with the duty cycle giving the edge value.

To begin the interleaving process, a single screen-filling rectangle is rendered, and a vertex program executes for each of its four vertices. Vertex positions are specified in a user coordinate system using the measured locations of the screen corners. These are given in the same real-world coordinate system as the user's eye positions, possibly that given by a user tracking system. An appropriate perspective projection brings these vertices to their proper locations at the corners of the frame buffer. We also transform each vertex from user space into lenticular phase space. The composition of this transformation matrix follows.

Let v be the vector from the center of the display to the view position, in the display-centric coordinate system. Also, let p be lenticular pitch, s be shift, t be optical thickness, and a be angle.

The first consideration of the lenticular transform is the projection of positions on the plane of the pixel grid onto the plane of the lenticular, a distance given by the optical thickness. This introduces a shortened shift value s' and a reduced pitch p',

$$s' = s \cdot (v_z - t)/v_z \qquad p' = p \cdot (v_z - t)/v_z,$$

and a parallax offset along both the horizontal and vertical display axes,

$$d_x = t \cdot v_x/v_z \qquad d_y = t \cdot v_y/v_z.$$

The lenticular transform translates the 3D position of each sub-pixel by the projected shift and offset, rotates them about the center of the screen through the lenticular angle, and scales them by the projected pitch, thus normalizing the wavelength to one.

$$M = \text{Scale}\begin{bmatrix} p' \\ p' \\ 1 \end{bmatrix} \cdot \text{Rotate}\begin{bmatrix} 0 \\ 0 \\ a \end{bmatrix} \cdot \text{Translate}\begin{bmatrix} d_x - s' \\ d_y \\ 0 \end{bmatrix}$$

The x value of the resulting vector gives the normalized 1D phase for each sub-pixel. Corner sub-pixel phases are output from the vertex shader as varying values, one vector per channel. The linearly-

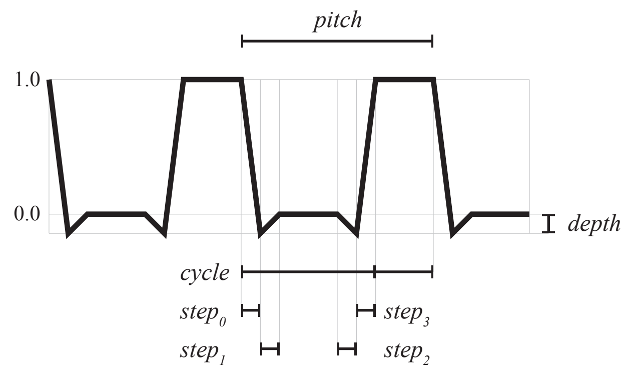

Figure 7: *Complex interleaver waveform*

interpolating rasterizer produces the correct phase for each sub-pixel of the screen. The fragment shader receives the interpolated phases and computes the step function of the fraction of each, giving the RGB masks. Each channel is sampled from its off-screen render buffer and modulated, with the sum of the outputs is written to the frame buffer.

In total, the interleaving process incurs the overhead of rendering a single screen-sized rectangle, plus one coherent texture access per channel per pixel. The cost of this has been negligible since the introduction of the programmable graphics pipeline.

2.2.3 Blended Interleaving

Projected channels are not crisply defined, and the transition from one channel to the next covers is a significant fraction of the width of a projected channel on the plane of focus. Most points on that plane receive a blend of adjacent channels. Given the immediate proximity of channel sub-pixels in the frame buffer, and the need to allow the user to move comfortably from one channel to the next, channels cannot be protected from cross-talk by interleaved guard pixels.

The customary response to this problem is simply to embrace cross-talk as a channel-transition smoothing mechanism. This demands a great deal of similarity between adjacent channels, as any significantly contrasting channel will interfere with nearby view points. In 3D scenes, channel similarity varies inversely with stereo disparity, and channels may be encouraged into coherence by reduced interocular distance and compressed scene depth. These techniques will be discussed in detail in Section 5.

Despite this, we continue to make every effort to reduce the effects of cross-talk at the lowest possible level. Toward this end, we have augmented the interleaving process with a mechanism that allows us to control the shape of the lenticular waveform, flexibly introducing on-demand channel blending and destructive interference of adjacent channels. A new lenticular waveform, shown in Figure 7, includes the addition of a depth parameter which allows the value of a channel to be subtracted from adjacent channels. In addition, four step values allow the specification of blends between channels. This waveform is implemented in terms of the products of step functions and linear interpolations.

Unfortunately, these do not result in significant improvements. The optical blending of adjacent eye views already inherent in the lens system makes the interleaver blending function redundant. The subtraction of adjacent eye views, intended as a form of "ghost busting," does reduce ghosts but produces artifacts. Significant ghost reduction has been achieved by subtraction of up to 40% of the adjacent image, but at this level there are many areas of the images that do not have a large enough value to allow subtraction, resulting in very visible defects. Reducing the coefficient to 10% or 20% does visibly reduce ghosting but not enough to make it disappear. These features of the interleaver will likely be more useful in the future work described below.

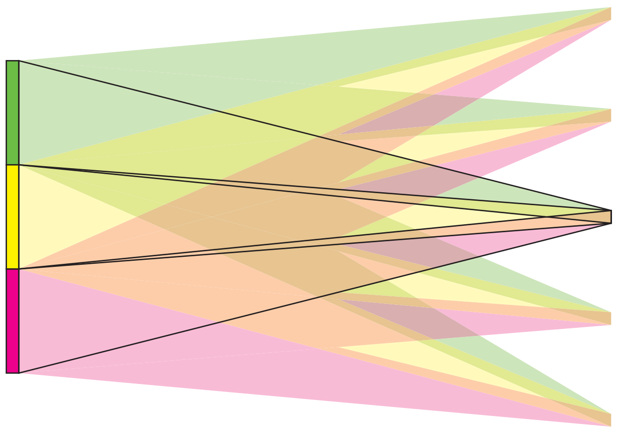

Figure 9: *The tiling of lenticular displays (left) is achieved by shifting the primary lobes of all displays into alignment at the plane of focus (right).*

3 Lenticular Clustering

Immersion is an important property of virtual reality usually achieved by filling the user's field of view with stereoscopic imagery. This is the motivating principle behind large VR environments for multiple users, such as the CAVE [Cruz-Neira et al. 1993]. But the CAVE benefits from the scalability of rear-projection, and lenticular displays of similar size are far from being feasible. So to achieve multi-user immersion with lenticular displays we use an array of them. We apply software techniques to ensure that their stereoscopic display functionalities coincide, and cluster technologies to ensure that their contents synchronize. This effect is highly scalable, as demonstrated by the 6×3 installation shown in Figure 8.

3.1 Autostereo Superimposition

Normally, a lenticular display projects its channels forward, and thus two adjacent lenticular displays project their channels parallel to one another. Given an array of lenticulars, the majority of view positions will fall near the channel repeat discontinuity in one or more displays, and at least one discontinuity will be visible at all times. For multiple lenticular displays to appear to the user as a single continuous display, the projected channels must be brought into alignment at the plane of focus. The shift parameter of the lenticular interleaver described in Section 2.2.2 enables this.

Figure 9 shows the scenario. One display is chosen to be the center and its primary lobe defines the standard. The primary lobes of all other displays are shifted into alignment with it. While the repeat discontinuity is still necessarily present, the alignment of the lobes places the discontinuity at the same view position across the entire display. A user perceives the passage through the discontinuity of the entire array just as he would a single display. As a consequence of this, all secondary lobes come into alignment, enabling group viewing.

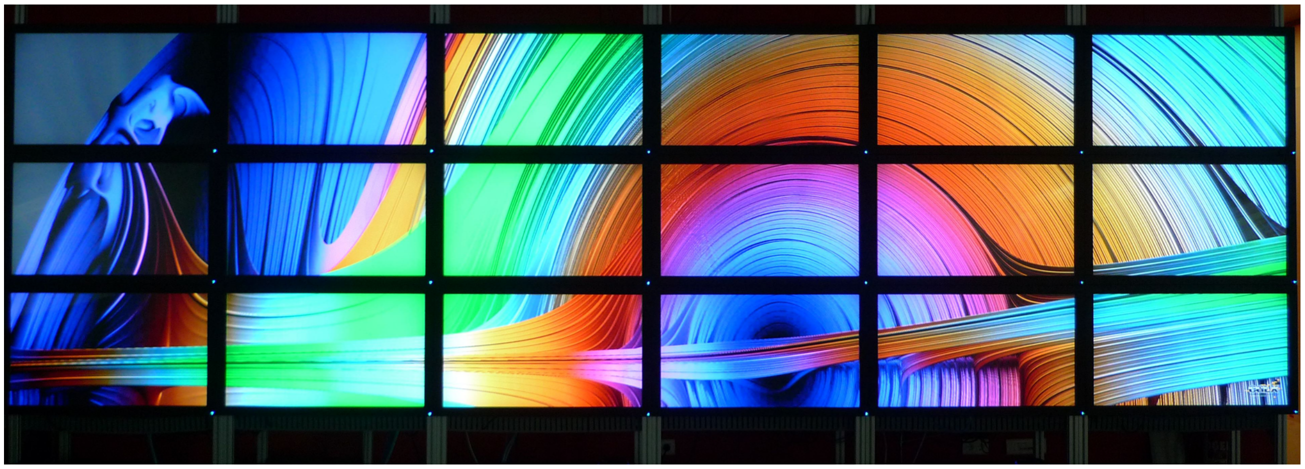

Figure 8: *Julia4D Animation by Dan Sandin on the Rapidly Expandable Virtual Environment (REVE) , a* 6×3 *array of Alioscopy* $42''$ *3DHD displays at the King Abdullah University of Science and Technology (KAUST) using the CGLX-based CineBlaster3D Video playback module. Photo by Tom DeFanti.*

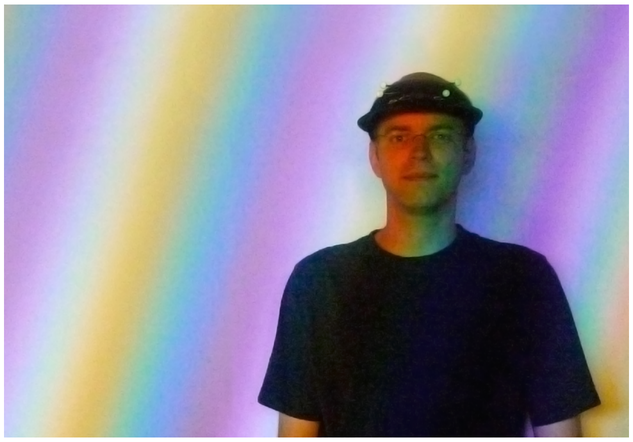

Figure 10: *The repeating channel array, with solid colors projected onto a user and a white card at the plane of focus.*

The calibration procedure for this alignment is interactive and straightforward. To begin, the center display is illuminated with a constant color per channel. The resulting pattern of colored bars is allowed to fall on a white card, as shown in Figure 10, where it remains throughout the calibration. A second display is illuminated and its shift parameter is adjusted until its color bars are visibly aligned with those of the center display. The second display is then darkened and the process is repeated for all other displays, bringing each into alignment with the first.

3.2 Cluster Synchronization

For an array of small displays to appear as a single contiguous large display, the scene rendered to each display must be in synchronization. Fortunately, in our circumstance, the degree of synchronization necessary to provide the illusion of continuity is not high.

Perfectly adjacent displays must have extremely precise synchronization to appear continuous. Any delay in update becomes apparent as a discontinuity in the frame, and hardware genlock synchronization is required. However, if there is even a small physical discontinuity between images, then the temporal discontinuity is masked. A cluster-driven array of lenticular displays takes full advantage of this.

Most LCD displays, including the Alioscopy displays, refresh at a rate of $60 Hz$, so two unsynchronized displays might differ in their update times by as much as $16 ms$. With their combined bezels over $40 mm$ wide, this $16 ms$ delay is perceivable when viewing critically, but is not an obvious defect in normal usage. If the cluster synchronization mechanism and rendering hardware is capable of finishing its task within a refresh period then no significant discontinuity is apparent. Our implementation is compatible with synchronization hardware such as the NVIDIA Quadro G-Sync, and we have demonstrated on small scales that this does eliminate the discontinuity completely. However, the cost of this solution becomes prohibitive at scale of our larger installations.

From a software point of view, synchronization is straightforward, and a number of existing clustering software approaches satisfy the relatively lax timing requirement. There are three fundamental synchronization tasks to be performed. First, the application as a whole must be started across all machines comprising the render cluster. Second, user interactions and other application states must be distributed across the cluster. Third, display updates and buffer swaps must be coordinated.

The most flexible solution is to simply run an independent copy of the rendering application on each node of the cluster, each configured with its own subset of the display as a whole. Start-up is commonly performed using the secure shell with public-key authentication. State synchronization and screen update coordination use TCP socket communication. This approach is fully general, but places the entire burden of synchronization on the application developer. Examples of applications using this approach include the distributed visualization systems COVISE [Rantzau et al. 1996] and CGLX [Doerr and Kuester 2010], and the planetary data visualization tool Thumb [Kooima et al. 2009].

Another effective solution is the Message Passing Interface (MPI) [Gropp et al. 1999]. As an interface for distributed software development, MPI handles application start-up and provides communication primitives enabling application state and display coordination. Applications developed using MPI include the VR scripting framework Electro [Kooima 2005].

Most applications benefit from performing a *glFinish* operation

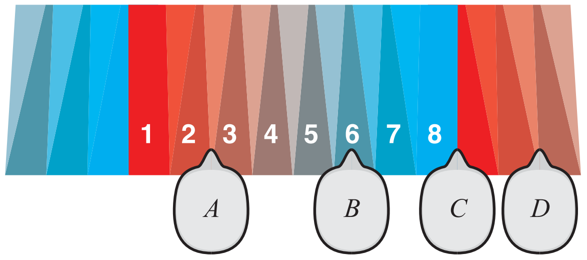

Figure 11: *Four users in different basic positions in front of a display at the plane of focus. A) Normal viewing. B) Between channels. C) On the repeat boundary. D) Normal viewing on the repeat.*

prior to coordinating a group-synchronized buffer swap. This ensures that lightly-loaded render nodes do not swap well in advance of heavily-loaded nodes when the frame rate drops below 60Hz. In all cases, the lenticular rendering is handled by a library implementing the interleaver algorithm described in Sections 2.2.2 and 2.2.3.

4 User Tracking

One of the defining properties of a virtual reality system is the viewer centered perspective, where-in the position of the viewer is tracked and the rendered perspective updated for that position. If the system supports group viewing, other *passive* viewers see the perspective of the tracked *active* viewer. The visual experience of a passive viewer in immersive surround systems such as CAVEs is not perspectively correct, but is compelling if he or she is positioned near the tracked viewer and looking the same direction. Thus, the CAVE is still an effective environment for small groups of people. We have also developed a VR autostereo system called Varrier . With this system the experience of the active viewer is very good but the system produces a very disturbing and possibly nauseagenic experience for passive viewers. Thus, the Varrier is only appropriate for a single user. Our research with multi-view displays is partially motivated by the goal of an autostereo VR system that is appropriate for small groups.

4.1 Stereoscopic Viewing Cases

A user viewing a lenticular display will be in one or more configurations that determine his ability to perceive a stereoscopic image presented by it. These are summarized by Figure 11, which shows the array of repeating channels projected toward the plane of focus, viewed from above, with a set of users in a variety of configurations.

Here, User A is viewing comfortable autostereo with adjacent channels in his left and right eyes. User B is out of phase with the channels, seeing a blend of two viewpoints in each eye. This circumstance is common in lenticular viewing, and User B does perceive a stereo image, though with much more apparent ghost than User A. User C is in the channel repeat and cannot perceive a correct stereo image. Users do feel some discomfort in this circumstance, and they naturally tend to reposition themselves to avoid it. Finally, User D is comfortably viewing the same scene as User A, though in the channel repeats.

All possible combinations of the circumstances of Users B, C, and D do occur in practice, and there are a number of active measures that may be taken to reduce the visibility of their occurrence. Four distinct approaches to the application of user tracking to multi-user immersive lenticular display are discussed.

4.2 Tracking Cases

We implemented four different methods of combining tracking with a multi-view lenticular display and informally evaluated the viewer experience for the tracked viewer and the passive viewers.

4.2.1 Case 0: No Tracking

The default case has no tracking. See Figure 12-0. All viewers are passive. There is limited look-around of approximately eight degrees provided by the eight channels of the Alioscopy display. If the viewer moves between repetitions of the display, stereo is inverted in the scene before the eight views repeat. For most of the viewing area the viewer sees a correct stereo image.

4.2.2 Case 1: Perspective Only

In this case, the perspective is updated based on head-tracking information, but no change is made to the direction in which channels are projected. See Figure 12-1.

One problem with this method is that the perspective is exaggerated, as one effectively adds the perspective change due to viewer tracking to the perspective change provided by the eight different views. The tracked viewer has complete look-around, but he still sees the discontinuity at the channel repeat. During this transition, the perspective of the tracked viewer changes by twice as much as in the non-tracked case. As long as the passive viewer remains within on repetition of channels he or she sees a good stereo image, with the perspective being updated with the position of the tracked viewer. If the passive viewer moves he can still pass through a repeat discontinuity as in the non-tracked case.

4.2.3 Case 2: Channel Adjustment

In this method we move the channels with the tracked positions, as illustrated by Figure 12-2. In our system, this is accomplished by simply extrapolating eight view positions from the tracked position and feeding them into the interleaving system. However, in lenticular autostereo systems such as the Alioscopy, the channels cannot be moved smoothly, but only in discrete steps. Thus, as the tracked viewer moves into the transitional period between one channel and the next, he begins to see the adjacent view before the channel is moved to follow. During that period the viewer is looking one interocular distance over. Eventually the channel adjustment occurs, causing a perspective snap-back. This appears to the viewer as a jarring of the image.

4.2.4 Case 3: Channel Reassignment

In an effort to fix the problems of Case 2, we keep the channels in place and change only the view points mapped to the channel projections, see Figure 12-3. The tracked head position determines the number of channels between the user and the center of the repetition. Views are generated surrounding the channels through which the user is looking, and channels are remapped such that the tracked user sees the center two. To eliminate the perspective snap-back, the center of each channel is used as the view position rather than the user's tracked eye position within that channel. As the user moves to an adjacent channel the view in that channel will be the same as in the previous head position, because the view generation is now offset by one interocular distance and the view mapping changes by one.

The channel reassignment method works best of all the methods we implemented. Because the eight views are generated around the tracked viewer and the center is mapped to this viewer's eyes,

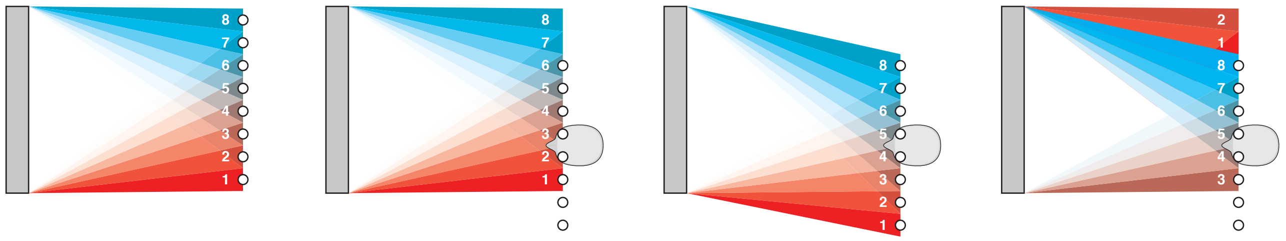

Figure 12: *The four experimental tracking cases. Numbered areas show channel projections. Circles show perspective view positions. The head shows the position of the actively-tracked user. 0) Untracked. 1) Perspective tracking. 2) Channel tracking. 3) Channel reassignment.*

the repeat discontinuity is never perceived and this viewer has full look-around within the tracked space. The experience for the passive viewer is nearly as good as the non-tracked case. The passive viewer does see a similarly solid, unchanging view until the tracked viewer moves through a distance of eight channels. Then, the passive viewer experiences a channel repetition discontinuity, and sees a perspective shift.

5 Depth Compression

Current lenticular and barrier strip multi-view displays exhibit high ghost levels, and thus are typically limited to the display of scenes of very limited depth. The origin of the problem is the very coarse angular sampling of the display space which causes severe aliasing. Different angular views of objects near the image plane are very similar and, with a reconstruction filter, do not present a problem. Objects far from the image plane are very different and without a reconstruction filter would jump or flip from one image to the adjacent image as one moves about. Designers have typically chosen to assign as little as one sub-pixel per view and use the fuzziness of the cylindrical lens as a reconstruction filter. For objects far from the image plane, the channels blend and one see multiple images, or ghosts. There is no hard threshold, but the 24″ Alioscopy displays we use have a maximum depth of around $\pm 15''$. This represents a severe limit to the content one can display. We have developed several techniques to mitigate this problem and allow a wider range of content. All of these methods modify the way the scene is rendered, and do not affect the interleaver algorithm.

5.1 Technique 1: Interocular Distance Control

The first and simplest method for depth compression is based on changing the interocular distance used by the rendering algorithm to offset the camera positions when projecting the scene onto the screen. An interocular distance of zero effectively turns off the stereo effect, and all eight views appear identical. In this case, all parts of the scene appear to be located on the plane of the screen. Our approach allows the user to dynamically adjust the interocular distance to a value appropriate for the depth of the scene.

This method works quite well for data sets which are entirely in front of the user and can be more-or-less centered upon the screen. Examples include protein models, machine parts, and medical data sets. However, this method does not work well for data sets that surround the user, such as architectural or structural engineering models. For those types of models, very low interocular distances can be found which bring them fully into the usable depth range. However, objects both near and far from the viewer are perceived to be close to the screen, misrepresenting the data and undermining the user's sense of immersion in it.

5.2 Technique 2: Bounding-box depth compression

This method does not modify the interocular distance, but instead calculates the maximum extent of the displayed object in the z-direction (the axis perpendicular to the screen) and linearly reduces the z position in eye coordinates before projection to the screen. This linear reduction provides a user-adjustable multiplication factor, and a dial to position the center of the scaling. Both the bounding box computation and the scaling are updated automatically for each frame.

The perceived effect of this method is that objects appear deformed to varying degrees depending on the depth compression factor. This becomes particularly apparent when the factor is large and the objects are rotating. In practice, when the compression factor is small and the object fits on the screen, this method works fairly well. However, it does not work for data sets surrounding the user, as the algorithm will bring all objects into the small usable space around the screen plane, including those which should appear next to and behind the user.

5.3 Technique 3: Box clipping and S-shaped depth compression

By non-linearly compressing the depth within the bounding box of an object, the squeezing and expanding effect of a rotating object observed with Technique 2 can be reduced. The implementation of this method uses a GLSL vertex shader to reposition each vertex of the scene in real-time. The compression function maps a tunable depth range into the usable space around the screen plane. Depth values outside of the usable range are clamped and geometry behind the viewer is clipped.

This method works for isolated compact objects, but produces more visible distortion than bounding-box depth compression due to its non-linear nature. It significantly reduces ghosting in architectural walkthroughs but causes severe spatial distortion at the extremes of the compressed depth range. This is especially noticeable when objects move through the depth range from one extreme to the other, as during a rotation. It is useful for scenes with multiple objects as long as they are not very close to the viewer.

6 Future Work and Conclusion

The current research utilizes a camera-based tracking system that requires retro-reflective targets. In public spaces, having the user wear tracking targets introduces the same problems as the use as glasses for stereo. In future installations we plan to use camera tracking without targets. We have developed such a tracker for use with the Personal Varrier autostereo display system [Girado et al. 2003] . We plan to expand upon that work for applications in public spaces [Sandin et al. 2005]. The depth compression techniques we applied in an effort to control the visibility of ghost have been partially successful for some application areas but have not been successful for applications such as architectural walk-through. We plan to continue to work in this area, including the investigation of some form of non-linear stereo disparity control.

We also plan to develop lenticular autostereo displays that use a greater number of angular samples, spaced more closely together. We also intend to use more sub-pixels in each view along with software anti-aliasing reconstruction filters. This will lower the spatial resolution of the display, but improve the degree of depth that can be displayed. High spatial resolution may be regained using multiple panels. Finally, we plan to pursue techniques developed for the Varrier display that do not depend upon the registration of the lenticulur sheet with the pixel grid. This will allow dynamic refocus the display on the tracked viewer along with other real-time optimizations.

Despite the broad front of future work waiting to be pursued, we have met our initial goals of creating multi-user, large-scale, high-resolution autostereoscopic displays, and we have worked diligently toward their tuning and optimization. The issues detailed throughout this document represent areas of focus for the display researcher, and do not present themselves as obvious flaws to the general user. These installations are big, bright, and immersive. They provide a solid foundation for continued research in autostereo display and virtual reality application development.

Acknowledgements

This publication is based on work supported in part by Award No US-2008-107/SA-C0064, made by King Abdullah University of Science and Technology (KAUST).

UCSD received major funding from the State of California for the StarCAVE. UIC receives major funding from the National Science Foundation (NSF), awards CNS-0420477 (LambdaVision), OCI-0943559 (SAGE) and CNS-0959053 (NG-CAVE), and also receives support from Sharp Laboratories of America and the State of Illinois. Also, UCSD and UIC were lead institutions of the NSF-funded OptIPuter project, award OCI-0225642.

CAVE^{TM}is a trademark of the Board of Trustees of the University of Illinois.

References

CRUZ-NEIRA, C., SANDIN, D., AND DEFANTI, T. 1993. 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, 142.

DOERR, K., AND KUESTER, F. 2010. CGLX: A Scalable, High-performance Visualization Framework for Networked Display Environments. *IEEE Transactions on Visualization and Computer Graphics 99*, PrePrints.

FUKUSHIMA, R., TAIRA, K., SAISHU, T., MOMONOI, Y., KASHIWAGI, M., AND HIRAYAMA, Y. 2009. Effect of light ray overlap between neighboring parallax images in autostereoscopic 3D displays. In *Proceedings of SPIE*, vol. 7237, 72370W.

GIRADO, J., SANDIN, D., DEFANTI, T., AND WOLF, L. 2003. Real-time camera-based face detection using a modified LAMSTAR neural network system. In *Proceedings of SPIE-IS&T Electronic Imaging*, 20–24.

GROPP, W., LUSK, E., AND A., S. 1999. *Using MPI: Portable Parallel Programming with the Message-Passing Interface*. MIT Press.

KOOIMA, R., PETERKA, T., GIRADO, J., GE, J., SANDIN, D., AND DEFANTI, T. 2007. A GPU Sub-pixel Algorithm for Autostereoscopic Virtual Reality. In *IEEE Virtual Reality Conference, 2007. VR'07*, 131–137.

KOOIMA, R., LEIGH, J., JOHNSON, A., ROBERTS, D., SUBBARAO, M., AND DEFANTI, T. 2009. Planetary-scale terrain composition. *IEEE Transactions on Visualization and Computer Graphics 15*, 5, 719–733.

KOOIMA, R., 2005. Electro. http://www.evl.uic.edu/rlk/electro.

PETERKA, T., KOOIMA, R., GIRADO, J., GE, J., SANDIN, D., AND DEFANTI, T. 2007. Evolution of the Varrier Autostereoscopic VR Display. In *IS&T/SPIE Electronic Imaging 2007 Conference Proceedings*.

RANTZAU, D., LANG, U., AND RÜHLE, R. 1996. Collaborative and Interactive Visualization in a Distributed High Performance Software Environment. In *Proceedings of International Workshop on High Performance Computing for Graphics and Visualization, Swansea, Wales, '96*.

SANDIN, D., MARGOLIS, T., GE, J., GIRADO, J., PETERKA, T., AND DEFANTI, T. 2005. The Varrier^{TM}autostereoscopic virtual reality display. *ACM Transactions on Graphics (TOG) 24*, 3, 903.

VAN BERKEL, C., AND CLARKE, J. 1997. Characterisation and optimisation of 3D-LCD module design. In *Proc. SPIE*, vol. 3012, 179–186.