Building a Projection Autostereoscopic Display

Billy Chen

June 11, 2002

This report describes an autostereoscopic display which uses a projector instead of a traditional printed image as a display– trading spatial resolution for dynamic light-field sampling. The display uses a lens array composed of hexagonal lenslets. Such a design opens up a myrid of possibilities, like using the Stanford Light Field Camera Array. This report describes in detail the ideas behind calibration and light field resampling required in creating an image for display. Additionally, an implementation description is also provided. Finally, a discussion of current limitations and future work of this device is described.

1 Introduction

An autostereoscopic display shows an image to a viewer without any use of googles or other devices. The image is appealing because it follows many of the principals of 3D objects in the real world [Hal97]. This viewing model has been around since the turn of the 20th century and has been as popular as 3D movies or holograms. The key contribution is the ability for the viewer to see parallax in an image as he changes his viewing position – yielding the effect of 3D.

Recently in 1996, the introduction of light field/lumigraph rendering allowed viewers to examine a 3D model or scene independent of scene complexity [LH96]. Additionally, the sampled 2D image could be constructed from light rays measured from real irradiance from the scene to the camera's image plane, giving the viewer a highly realistic (synthetic) image of the model or scene. While such convincing images can be rendered, one limitation is the two-dimensionality of the display device.

To date, few researchers have explored the possibilties with viewing a light field on an autostereoscopic display; the notable exception being McMillan and his colleagues at MIT who demonstrated how to construct and print an image for use in combination with a hex lenssheet to achieve an autostereoscopic image of a light field [IMG00].

This report extends the work of McMillan et al. and describes an autostereoscopic display where instead of a printed image, uses a projector – trading spatial resolution for dynamic light-field sampling. Where the MIT group create an autostereoscopic image for a static lightfield, a projector display can handle dynamic light fields, such as those produced by the Stanford Light Field Camera Array [WSLH02].

The report is separated into two parts: theory and implementation. The theory section, entitled, "Display Pipeline", describes the ideas behind building an autostereoscopic display. The implementation section describes the process and parts used to build an autostereoscopic display at the Stanford graphics lab. The report concludes with some results and a discussion of limitations and future work for this display system.

2 Display Pipeline

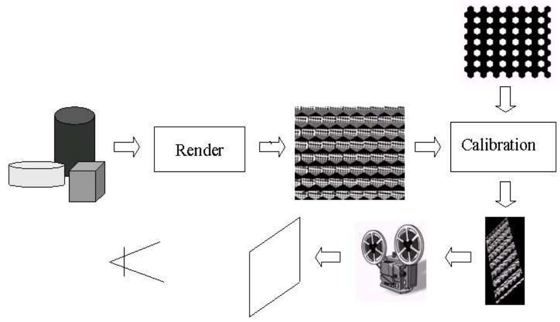

Figure 1: *A high level illustration of the display pipeline. It begins with scene geometry on the left and ends at observer.*

The process of displaying a 3D image on an autostereoscopic display is displayed in Figure 1. Information about the scene, which can be represented as light fields or traditional computer graphics models is piped through a rendering process which outputs an image composed of hexagonal subimages. This composited image goes through a calibration function which warps the image. The calibration function's parameters are set during a calibration phase with an appropriate calibration image. After warping, the display image is sent to the display device, where it is viewed by an observer.

The two critical steps in this pipeline are the calibration and rendering steps. Calibration bridges the gap between world coordinates as observed by a viewer, and the image coordinates. Rendering images resamples the plenoptic function, which is represented explicitly as a light field or implicitly by geometry. The resampled light field is parameterized by the description of the display device.

2.1 Calibration

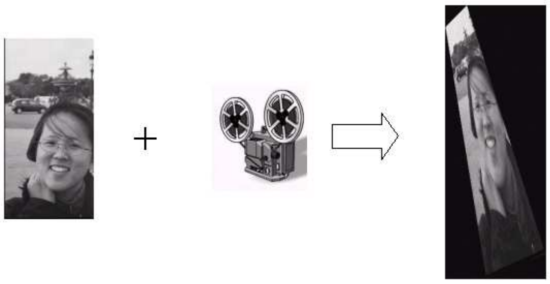

Figure 2: *The problem of calibration. Initially, the image warp is unknown.*

The goal of calibration is to find a mapping between world coordinates and image coordinates. Figure 2 illustrates this problem. In the case of displaying an autostereoscopic image, there are two important variations from direct parameter calibration. First, the color of a lenslet is a function of its angle. Hence translation in the lenslet array becomes a rotation in the solid angle forming the viewing frustum for the display. Second, direct parameter calibration produces extrinsic parameters of rotation and translation and the intrinsic parameters of focal length, pixel size, and coordinates of the image center. For an autostereoscopic image, even the projection matrix isn't entirely necessary (although the camera paramters can be recovered). The projection matrix is used implicitly during calibration.

In practice, calibration is made easier when first calibrating for affine transformations, and in particular scale. All other transformations are handled by a homography describing the mapping between the image coordinates and the lenslet display coordinates. See Section 3 for a description of the evolution of the calibration process in the actual implementation of the display.

2.1.1 Using Homographies

Finding the projection matrix between the lenslet display and the image display would certainly solve the calibration problem. However, a neat insight is that instead of solving for the projection matrix, one can find a homography such that when applied to an image, its projected image in the lenslet array is aligned correctly [And]. Figure 3 illustrates this idea.

Given an image A, its image in the display via projection matrix p is A'. We want to find a homography h such that B' = Ahp is a correctly calibrated image.

Given a point p, projection matrix M and the projected point p' in homogeneous coordinates:

$$\begin{vmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{vmatrix} \begin{vmatrix} x \\ y \\ 1 \end{vmatrix} = \lambda \begin{vmatrix} x' \\ y' \\ 1' \end{vmatrix}$$

Let M_i be the i'th row of M. Then

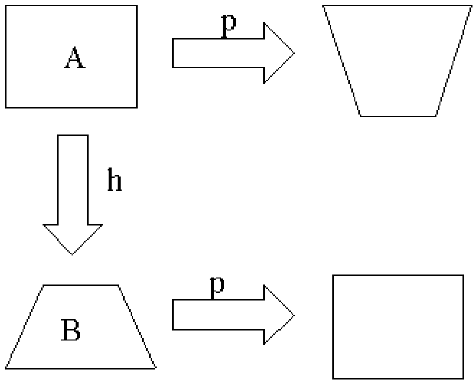

Figure 3: *The goal is to find homography h.*

$$M_1 p = \lambda x' \qquad (1)$$

$$M_2 p = \lambda y' \qquad (2)$$

$$M_3 p = \lambda \qquad (3)$$

The scale factor λ is unknown. However, we can take the ratios of Equations 1 and 2. Similarly for Equations 1 and 3 then we have:

$$y'(M_1 p) - x'(M_2 p) = 0 \qquad (4)$$

$$M_1 p - x'(M_3 p) = 0$$

For each point p there are such two equations. With 9 unknowns, we need 4 points (each with 2 equations) to create a matrix A of rank 8. The matrix A is composed of only point coordinates, and the vector m comprises the values of the matrix M.

$$\begin{vmatrix} xy' & yy' & y' & -xx' & \dots \\ & & \vdots & & \\ & & & & \\ & & & & \end{vmatrix} \begin{vmatrix} M_{11} \\ M_{12} \\ M_{13} \\ M_{21} \\ \vdots \end{vmatrix} = 0$$

The first row of A is shown with respect to equation 4. Since A is rank deficient, its null space exists and in this case is of dimension 1 (the constant scale factor). Vector m can be recovered using SVD techniques as the column of V corresponding to the zero singular value of A [TV98] [HZ00]. In practice, correspondence between points through a homography is known up to a noise factor, so many more points are used, forming an overconstrained system and the column of V corresponding to the smallest singular value of A is used. However, in this case of calibrating the autostereoscopic display, corresponding point are exactly known as explained in section 3.

2.2 Rendering

Rendering images resamples the plenoptic function, which is represented explicitly as a light field or implicitly by geometry. The resampled light field is parameterized by the description of the display device. Isaksen et al. have shown how to sample a light field for autostereoscopic displays [IMG00], and their method is quickly summarized here. The parameterization of the display device is also discussed, in terms of picking the correct camera parameters for each lenslet.

2.2.1 Sampling a Light Field

In their Siggraph paper in 2000 on Dynamically Reparameterized Light Fields, Isaksen et al. discuss an autostereoscopic application for their reparameterized model [IMG00]. Figure 4 is an illustration from that paper.

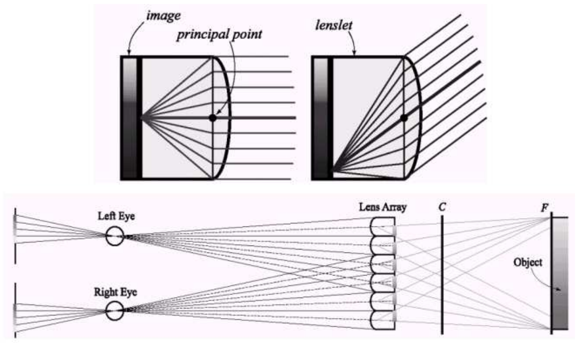

Figure 4: *Top: Rays parallel to a principal ray converge to a point on the image plane. Bottom: An overview of how the observer samples rays from each lenslet.*

If we use a paraxial lens approximation, then rays parallel to a particular lenslet will converge to a single point on the image. In particular, that point is the intersection point of the lenslet's corresponding parallel principal ray with the image plane. If we place a camera's COP at the principal point of a lenslet, facing towards the image plane, then the camera will sample rays from all incident angles to that lenslet – effectively producing the view-dependent color information needed for that one lenslet. This maps angular information for a given point to spatial information. Since spatial resolution is directly effected by the display, the angular resolution for an autostereoscopic display will depend heavily on the resolution of the display device. Section 3 discusses the trade-offs of varying resolution displays.

As a consequence of this parameterization, objects will be rendered as "behind" the autostereoscopic display. Contrary to their statements in [IMG00], simply placing the lens array behind a captured object will not produce an object that appears to float in front of the display. Such an image is pseudoscopic to the viewer, and when using a conventional graphics renderer like OpenGL, the lighting is also incorrect [HK97].

One difference between Isaksen's approach and this implementation is that his reparameterized light field can set a focal distance at the object surface. However, in this implementation each camera is modeled by an infintely small aperture, yielding an infinite depth of field.

Another approach to resampling the light field is to take advantage of the fill rate of the graphics card and to use orthographic cameras in object space instead of perspective cameras within each lens. Section 5 discusses the details and advantages of this accelerated approach.

2.2.2 Computing the Maximum Constrained FOV

Using the lenslet parameterization where camera centers are located at lenslet optical centers, one must be careful to avoid overlap or zoning in the projected images of adjacent lenslets. In order to solve this problem, some notion of a maximum FOV must be computed.

Consider the lens parameterization shown in Figure 5. C_i are the principal points of the lenslets, the bold arc of the circle represents the visable refractive surface (facing upwards). F_i are the focal points of the i'th lens. The lenslets are placed adjacent to each other (lenslets along the orthogonal direction are placed at twice the distance apart due to the hexagonal pattern of the lenslet array). The thickness of the lenslet array is measured from the center of the lenslet. In this case, the thickness is donoted by the horizonal line at z=0.165 in.

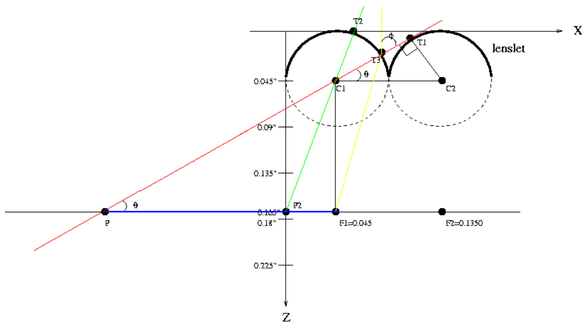

Figure 5: *A lenslet parameterization. The observer looks from the top.*

Given this parameterization, it's useful to find the maximum field of view for any given lenslet. This will also reveal what the pixel coverage is of for given lenslet. The red line $PT1$ is a ray of light. The blue line $PF1$ is the pixel coverage we want to determine – assuming that the pixels are flush with the back of the lenslet array.

Note that $||C1C2||$ and $||T1C2||$ are known. Then $\theta = \arcsin(0.045/0.09) = 30\,\mathrm{deg}$. Hence, the FOV = $180 - 2\theta = 120\,\mathrm{deg}$.

Similarly, $||PF1|| = \frac{||C1F1||}{\tan(30\,\mathrm{deg})} = 0.2078$ in. The lenslet diameter is .09 in., so the pixel coverage at maximum FOV actually overlaps with several pixels over adjacent lenslets. In fact, there is a 0.1628 in. $\frac{2*0.2078-0.09}{2}$ overlap if the maximum FOV is used.

Given that the maximum FOV causes overlap, I want to find the maximum FOV which does not have pixel overlap, which I call the *maximum constrained FOV*. The green line $P2T2$ is a good candidate for the maximum constrained FOV. In fact, $P2T2$ denotes the widest angle ray if we assume a paraxial lens approximation.

If the green line $P2T2$ forms the angle of maximum constrained FOV (measured with respect to the paraxial ray of the lenslet), then any other line forming a larger FOV must overlap adjacent pixel areas. Let's pick any ray forming a larger FOV, then its point of intersection on the focal plane must coincide with a parallel ray that intersects the optical center. In the case of a spherical lens, this ray can be found by finding the point on the sphere where the ray is normal to the sphere surface. Obviously, this point must be to the right of $T2$, since the ray forms a greater angle to the surface pt at $T2$ (since it has a larger FOV). Since this intersection point is to the right of $T2$, and its ray intersects the optical center, then the intersection of this ray and the focal plane must lie to the left of $P2$. But $P2$ is the leftmost pixel under the orthogonal projection of the diameter of the lenslet, so the new intersection must lie in a neighboring pixel area. Hence, any ray creating a larger FOV than the FOV formed by $P2T2$ will be in neighboring pixel areas, so $P2T2$ forms the maximum constrained FOV.

3 Implementation

3.1 Display Design Choices

Before building the autostereoscopic display, one must look at the tradeoffs between various displays. Figure 6 illustrates the advantages and disadvantages of four types of displays, ranging from lowest resolution to highest resolution.

Display Type	Resolution	Advantage	Disadvantage
CRT/LCD	72-115 dpi	cheap, everywhere	low resolution, challenging calibration
Projector	~150 dpi	easier to control pixel depth, adjustable angular resolution	projector distortion
Big Bertha	211.66 dpi	higher resolution	expensive, need special hardware to drive it
Printer	300 dpi	high resolution	static images

Figure 6: *A table illustrating various design display choices, sorted by resolution.*

The most common display is the CRT display, but it is heavily handicapped in resolution density. As a consequence, for smaller lenslets, the number of pixels per lens is alarmingly small. In addition, calibration becomes more challenging since there is a non-projective warp that occurs if the display isn't completely flat and flush to the lens array. The lens array that was used has the property that each lenslet's focal point lies on the back surface of the lenslet array. When the display is not flush with the back surface, then rays that normally converge to a point diverge into an area of pixels, causing overlap. Section 5 briefly discusses the possibilities of such a setup. However, for an initial design, this was too challenging to implement in the time at hand.

Projectors, however, are much easier to use to keep the display flush with the back of the lens array. The ability to adjust the focal distance and scale also allows for various quantities of pixels per lenslet. This trades off angular resolution found in pixels per lenslet to total pixel area coverage of the lenslet array. However, even at the closest focal distance, a maximum of 150 dpi is possible. In addition, the projector adds its own projection distortion when displayed on the back of the lenslet array, and this distortion needs to be accounted for in the calibration.

Big Bertha, or the IBM T220, is a high-resolution display, accomplishing about 211 dpi. It's one of the highest resolution displays on the consumer market but suffers from the same challenge of calibration found in normal CRT or LCD displays. In addition, it requires special hardware to drive the display, and at the time of implementation this hardware was not functional.

Using a printer provides a fairly high resolution at 300 dpi, but one cannot create an animated autostereoscopic display. However, calibration becomes a trivial task.

The final implementation uses a projector for its flexibilty in adjusting the focal plane to the back of the lenslet array. Even at 150 dpi, the author is able to produce convicing images.

3.2 Hardware

Figure 7 illustrates the hardware and setup of the display. A Compaq projector is used, set at its minimum focal length, at 1024x768 resolution. The projector is modestly mounted onto an optical bench, supported by a triangle platform. Other projector display configurations were also considered, including one similar to the mount used in the Stanford Interactive Mural, where a mirror is used for keystone correction. The mirror also flips the image, but since the image is viewed directly (as opposed to off a reflected surface), flipping the image produces the correct orientation when viewed. The author tried to calibrate the image using a mirror setup, but it had many short comings. The obvious one being that the calibrator would need to be behind the display to adjust the mirror, but also in front of the display to see if moire patterns were being formed. Additionally, using a mirror adds more parameters to calibrate. In this case, the mirror has two dials that simultaneously changes the pitch and yaw. In the end, a software solution towards calibration proved more direct and successful.

Perpendicular to the projector is a Fresnel #300 hexagonal lens array. The important dimensions are the focal length (0.12 in.), the thickness (0.12 in.) and the lenslet

Figure 7: *A photograph of the physical setup of the display. The observer views the display from the side of the laptop.*

mean diameter (0.09 in.). Using the equations in Section 2.2.2, the maximum constrained FOV is about 40 degrees. Other lenslet dimensions were also explored, but proved to be too small for a projector-type display. For example, the Frensel #630 has 645 lenslets per square inch, or about 25 lenslets per inch. Using a 150 dpi projector, this gives about 6 pixels per lenslet. Using a 40 degree fov per lenslet (assuming relative dimensions are the same across different lenslet arrays), this yields about 6.7 degrees for each pixel, which is too great for any reasonable autostereoscopic display.

A diffusing surface (tracing paper) is placed flush to the back of the lens array. It's held tightly to the lens array by an adjacent glass surface. Originally, a more diffuse surface was used (white paper), but that caused too much blurring and caused many calibration headaches for the author.

The laptop driving the projector is a Dell Inspiron 8000 with a Pentium 3 chip with 256 megabytes of RAM. Its display resolution is also set at 1024x768 to avoid resampling at the projector.

3.3 Calibration

The task of calibration began as a first order calibration using only affine transforms. An OpenGL program was written that displays a pattern of hexagons separated one hexagon apart. When displayed through the lens array, if not properly calibrated, moire patterns would appear. Based on the orientation and size of the moire pattern, the calibrator can adjust the rotation and scale of the hexagon pattern. As discussed earlier, translation is only a rotation of the solid angle forming the viewing frustum, so it is not explicitly calibrated. The original intention was to use the calibration matrix as one of the MODELVIEW matrices in OpenGL. In the implementation, the scaling factor is 0.49. As a sanity check, the hexagonal images are created for 300 dpi printers, and the projector has about 150 dpi, so the scaling factor is in the correct range.

However, affine calibration is not enough, and although scale and rotation is correctly calibrated with this technique, it cannot fix the projective warp formed from misalignment between the projector and the lenslet array. Figure 8 depicts a correctly calibrated pattern using affine transformations. Notice that the edges of the calibration pattern exhibit a morie pattern indicative of a miscalibration.

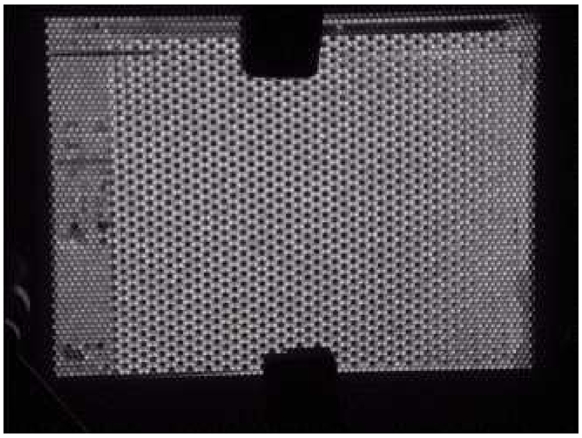

Figure 8: *A photograph of an affine corrected calibration image. However, distortions still persist.*

This problem is solved using a more powerful calibration technique with homographies. Section 2.1.1 discusses the theory behind it. The implementation uses of a form of manual calibration to avoid the noise and uncertainty around using vision techniques for correspondences. In addition, with manual use the implementation becomes straightforward. The user is presented with a calibration object on which four corners are highlighted with green circles. The user can iteractively select corners to move. In addition, keyboard keys are used for subpixel accuracy. When a circle is displaced, a new, warped image is shown. The goal is to warp the image shown on the laptop display so that the image seen through the lenslet display is a correctly aligned image.

The homography interface is written in Matlab because a lot of image transformation tools are built into the image-processing library. In particular, there are functions maketform(...) and imtransform(...) which build homographies based on 8 points and warp images given a transform respectively. Using C and OpenGL was also investigated as a viable alternative, but the overhead of getting an image processing library to work was more costly than using Matlab. In the long run, however, C and numerical recipies is a good, fast implementation of this calibration technique. In addition, the homography can be inserted directly into the OpenGL transform pipeline, following the PROJECTION matrix or as a texture map warp. This would provide a real-time warping engine that Matlab would not be able to handle.

In practice, the affine correction and homography takes about 10-15 minutes each. Additionally, when comparing images which are affine corrected vs. images which are fully corrected, empirical evidence suggests that the distortion correction provides little to negligable visible gains when viewed far away.

3.4 Rendering

The author used a modified version of the simple light field sample generator written in OpenGL by Georg Petschnigg. The application keeps track of an array of projection and pose matrices corresponding to a camera's intrinsic and extrinsic parameters. At each frame, the application iterates through each camera's set of parameters and applies them to the world. Then it samples the plenoptic function as a 2D image. As mentioned above, the author chose to render hexagonal composite images using the same hexagonal dimensions as Isaksen et al., for 300 dpi printers. In this way, hexagonal composite images can be printed on a 300 dpi printer and viewed in the traditional technique of autostereoscopic displays.

Each image resolution is 248x216, which is sampled 8 times larger than the pixel coverage of a single lenslet when pixels are printed at 300 dpi. This effectively supersamples the angular domain of each lenslet. The image is then down-sampled using a box filter to 31x27, where it is multiplied by a hexagonal mask. The final cropped image is then composited next to adjacent haxagonal images to produce a final, composite image. Orginally, the author had implemented this process using multiple files, each file holding a 248x216 image. The files were generated in Windows 2000 and sent over to a Linux OS to convert to tiff and to warp and composite. This process took about an hour to render an image. Most the time came from moving large quantites of small-sized files back and forth. Each frame held about 2500 lenslet images, which is very hard on the Windows file system, for a single directory. A second method was adopted which takes subimages and process them all in memory, so compositing and cropping was performed quickly – reducing frame rendering time to about 15 minutes per frame. Section 5 discusses some further approaches to accelerating the rendering time by taking advantage of hardware fill rates.

Computing the intrinsics of each camera is performed in a straightforward manner using the following relation: 1 unit = 1 pixel = 0.0033333 inches. The pixel to inch relation follows from assuming a 300 dpi display. Since each camera in OpenGL is modeled as a pinhole camera, setting the focal plane doesn't have much importance. In any case, the near plane is set at the focal distance (0.12 in) and the far plane is set at close to infinity.

The original FOV used was 40 degrees, computed using techniques from Section 2.2.2. However, this sized FOV creates a halo around the images. The halo comes from lenslets that can see edges of the centered object due to a larger FOV. In theory, a fully calibrated display would not exhibit such behavior, but any slight deviation will cause rays that normally miss the object to actually hit it near its silhouette edges, hence causing a halo. The author also tried out other degrees for the FOV: 10, 20, 30. With 10 degrees FOV, the projected image is very crisp, but exhibits little to no parallax. In the worst case, the FOV is infinitely small, causing the entire lenslet array to act as an orthographic camera, which has no parallax as the camera translates. At 30 degrees, the image is slightly crisper, with noticeable haloing effects. At 20 degress the halo effects are not very noticeable, but parallax can still be seen as the observer moves around.

4 Results

The first images to be displayed on the autostereoscopic displays were the images by Isaksen et al. Some of the images such as the big flower appeared to have a focal depth where only the frontmost flower is in focus. The background is heavily blurred so seeing the parallax was difficult. Other images such as the toys scene worked sufficiently well. Unfortunately, the results cannot be readily captured on paper.

The second series of images are of a teapot. These images are generated from scratch starting with the glut teapot call. Various degrees of FOV are rendered. Figures 9 and 10 are teapots using 10 and 40 degree FOV respectively. The horizontal jaggy lines are an artifact of pixel accuracy when compositing multiple hexagonal images. One can smooth the image, but in practice these seems are not noticeable to an observer through the autostereoscopic display.

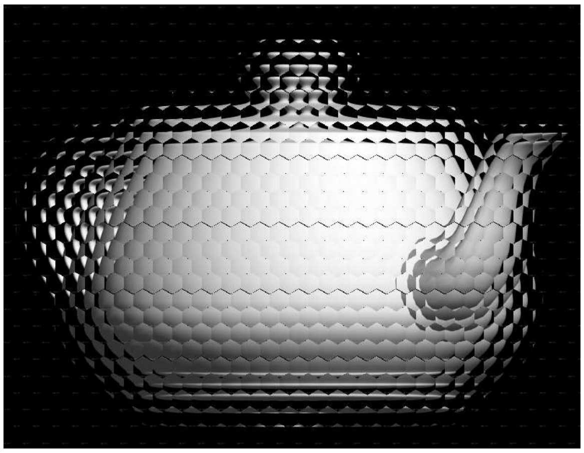

Figure 9: *A synthesized autosterescopic image where each lenslet views a 10 degree FOV.*

Finally, a 360 degree rotation animation of the teapot was also made. The goal is to show an animated autostereoscopic display. However, the parallax is not that evident since the rotation is hiding the disparity that an observer can normally detect by moving his head back and forth. The movie took approximately 14 hours to produce.

A couple nights were also spent getting a variant of the unstructured lumigraph viewer to sample a reparam-

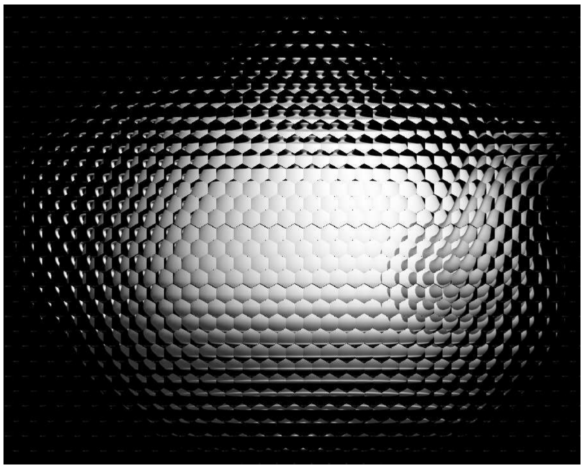

Figure 10: *An autosterescopic image using a 40 degree FOV.*

eterized light field. However a bug in moving the virtual viewpoint prevented such an endevour.

5 Discussion and Future Work

Several improvements could be made to the exisiting calibration process. In addition to the projective distortions introduced by the projector, one could also account for radial distortion. There are well known techniques to correcting for radial distortion, and these techniques could also apply here. In addition, the author wants to explore auto-calibration techniques using cameras. Using manual calibration techniques are useful for a first approximation to alignment. However, as the haloing effects show, the calibration is not perfect. This would be a good application for the Stanford Light Field Camera Array. Since each camera will be pre-calibrated, it would be useful to use such an array to actually compute which rays go to which pixels under the lenslets.

Another area for improvment is the rendering process. With the current setup, real-time rendering isn't currently possible. The process bottleneck is Matlab. While great for prototyping and proof of concept, it's too slow when performing image transformations. OpenGL is better equipped to handle such operations. The image warping can be performed as a texture warp, hence taking advantage of conventional hardware acceleration. An alternative is to take advantage of the OpenGL rendering pipeline and to put the image transformation matrix after the PROJECTION matrix.

Finally, one can also take advantage of graphics card fill rates and redundancy to rendering images more quickly [Ng]. In general, it's faster for a graphics card to scan convert many triangles a few times than a few triangles many times. Across adjacent hexagonal lenslet images many parts of the object will be rerendered. The key idea is that one can render all the necessary pixels once, and perform pixel shuffling to reconstruct all the hexagonal images. Isaksen et al. sample the light field by placing camera centers at optical centers of lenslets in the array. Suppose instead that we take sets of parallel rays going through the optical centers of lenslets in the array. Each set of rays correspond to the rays of an orthographic camera. In this way, the total number of cameras needed is equal to the total number of pixels under a lenslet. For 300 dpi displays, this turns out be 31x27 = 837 cameras. Compare this to the Isaksen approach which places a camera at each lenslet: 200x200 = 40000 cameras. One potential drawback of this new approach is dealing with aliasing. Anti-aliasing by supersampling and averaging corresponds to rendering more orthographic views of the object. There is a point where much anti-aliasing would cause the number of cameras to exceed that of the Isaksen approach. It would be interesting to investigate these kind of trade-offs further.

Another interesting approach is to look at autostereoscopic displays where the individual pixels have multiple rays hitting them. In the aforementioned discussion of this autostereoscopic display, a lot of weight was given to picking a display which keeps pixels flush with the focal plane of the lenslet array. If we relax assumption, then rays would pass the focal point and overlap on the display plane. A similar overlap occurs when we relax the assumption that the display should be viewed from afar. Parallel rays incident to a lenslet would normally converge to a single point. The lenslet acts as a converter from the angular domain to the spatial domain. So when a set of rays emerging from a center of projection are incident to a lenslet, they form an area under the lenslet. Move the center of projection a little, and the area under the lenslet shifts, but will overlap with the previous viewing point area. It would be interesting to investigate techniques to resolve these area overlaps.

One also look at other ways of producing autostereoscopic images. For example, using a reflective setup instead of the refractive one. Some parts are similar, like calibration, while others are different, like the display elements.

In general, autostereoscopic displays have a long way to go before becoming conventional display devices. In particular, display engineers need to consider the bandwidth requirement of such displays. Producing a single image requires computing many small samples of the plenoptic function. While there might be ways to accelerate this process, it's still questionable whether a real-time interative 3D TV application could be possible. Another challenge is simply the spatial resolution that a lenslet array can provide. A smaller lenslet requires a higher resolution display. Unlike a traditional 2D display, when viewed at a great angle, lenslet-type displays provide much distortion and overlap. Finally, while parallax is a good depth cue for stereo, it's questionable whether an observer would continously move his head back and forth to see more parallax.

Despite these limitations, autostereoscopic displays are still an exciting area of research for computer graphics and enthusiasts alike.

6 Acknowledgements

The author would like to thank Vaibhav Vaish for his many insights in the world of calibration. Without his contributions, this project many not have come to being. Thanks also to Georg Petschnigg for his original light field generator. Thanks to Ren Ng for his expertise and knowledge in computer graphics hardware and his contributions to the accelerated approach to autostereoscopic display. Finally, thanks to Sean Anderson, who, despite his own research obligations, took the initiative to show that such an idea could be possible.

References

[And] Sean Anderson. Conversation with sean anderson, may 2002.

[Hal97] Michael Halle. Autostereoscopic displays and computer graphics. In *SIGGRAPH 1997, Computer Graphics Proceedings*, Annual Conference Series. ACM Press / ACM SIGGRAPH, 1997.

[HK97] Michael Halle and Adam Kropp. Fast computer graphics rendernig for full parallax spatial displays. In *Proceedings of the IS&T/SPIE's Symposium on Electronic Imaging*, 1997.

[HZ00] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in computer vision*. Press Syndicate of the University of Cambridge, 2000.

[IMG00] Aaron Isaksen, Leonard McMillan, and Steven J. Gortler. Dynamically reparameterized light fields. In Kurt Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, Annual Conference Series, pages 297–306. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[LH96] Marc Levoy and Pat Hanrahan. Light field rendering. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 31–42. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.

[Ng] Ren Ng. Conversation with ren ng, may 2002.

[TV98] Emanuele Trucco and Alessandro Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice-Hall, 1998.

[WSLH02] Bennett Wilburn, Michal Smulski, Kelin Lee, and Mark A. Horowitz. The light field video camera. In *Proc. Media Processors 2002, SPIE Electronic Imaging*, 2002.