Gaze Contingent Depth Of Field Display

Kushagr Gupta
Stanford University
kushagr@stanford.edu

Suleman Kazi
Stanford University
sbkazi@stanford.edu

Abstract

This project aims to demonstrate a gaze contingent content display technique. We use a light field camera to generate a focal stack with the images in the stack focused at different depths within the scene. We display an image on a conventional digital display and use an eye tracking system to determine the user's eye gaze location in the image. Using a depth map generated from the light field we are then able to determine the depth plane that the user's gaze is at and pick the corresponding image from the focal stack to display to the user.

1. Introduction

Gaze contingent display techniques dynamically update the images shown on a screen based on the user's eye gaze location. The change in content can take on different forms. Some examples are degradation of resolution in the areas that the user is not directly looking at i.e. blurring the regions in peripheral vision and maintaining the sharpness of regions in primary field of view, interactive user interface elements based on eye gaze location, AI reactions in video games when the users look at them.

Applications for these techniques include enhanced realism in Virtual Reality (VR) or Augmented Reality (AR) headsets, flight / driving simulators and tele-operation, lessened user fatigue due to possibly reduced vergence-accommodation conflict (which can be rigorously tested with user studies), increased interactivity in gaming and bandwidth conservation in video or image transmission as relevant content based on eye gaze can have higher transmission bit rate as compared to regions in peripheral vision.

In this project we develop a practical test setup as illustrated in Fig. 1, wherein the gaze location on screen determines the depths that come into focus which inherently dictates the defocus blur. Our test setup comprises of integration of existing hardware (Tobii-EyeX Eyetracker and the Lytro Illum Light Field Camera) and software tools along with development of new programs to fulfill the task at hand. Although we wanted to develop a system for HMD, but due to lack of readily available eye tracking hardware mounted in HMDs we developed a platform for 2D-digital LCD screens and added the ability to perceive depth using anaglyph.

Figure 1: Our test setup showing Eye Tracker mounted on a LCD screen and anaglyph glasses worn by a user

2. Related Work

The fact that defocus blur increases realism of a 3D scene has been studied and tested with user studies. Mauderer in [1] investigates gaze contingent depth of field (DOF) as a method to produce realistic images and analyzes how effectively people use it to perceive depth i.e. to perceive ordering and distance between objects. An approach followed widely is by providing depth through binocular disparity wherein different images are presented to each eye and the visual system derives depth from differences.

The problems as mentioned in [1] with this approach are increased visual fatigue especially as it conflicts with other depth cues. [1] reports from the user study they conducted that in a gaze contingent condition participants judged objects to be more realistic and appeared to stick out of the screen and have a more defined depth and appeared separate from each other. [2] and [3] mention how relative blur by itself helps in ordinal depth perception but not quantitative depth which can be improved by using perspective or binocular disparity.

3. Light Fields Data-Set

Three sources of light fields captured using Lytro Illum were used for this project. The first dataset consisted of

stereo light field images taken on Stanford Campus while the rest being samples available at [8] and [7] which were taken from single viewpoint. Light fields from [7] were used for our demo during the poster session.

The reason behind shifting to a single viewpoint light field was the fact that stereo light fields had a large baseline which resulted in fusion problems, i.e. when they were coupled to create stereoscopic effect using anaglyph (red-cyan concept), the resulting image had large disparity range which our eyes couldn't fuse properly,as illustrated in Fig:2. Thus, we chose to use single viewpoint light fields and created novel views for them to get the stereo effect in order to perceive depth.

Figure 2: Stereo Fusion Problem due to large disparity range in scene

4. Method

We used the light fields captured using the Lytro Illum Camera and processed them digitally to generate a focal stack of images i.e. a series of images focused at different depths in the captured scene. Focal stack was converted into a stereo stack using the relative depth information, with detailed description in the sections to follow.

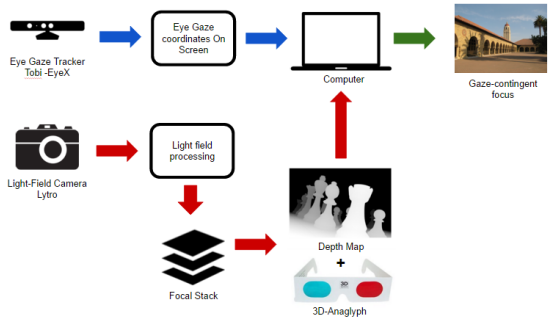

Figure 3: System flowchart

Now, user's gaze (captured using Tobii-EyeX), depth information and the stereo focal stack was fed as input to another program which displayed a focal plane on the screen based on gaze co-ordinates, giving an enhanced realistic viewing experience. The process flow is illustrated in Fig:3

4.1. Focal Stack Generation

The light field captured from Lytro has 14x14 perspective images of the scene due to the micro lens array arrangement. Considering the central image as reference all other 195 images are shifted accordingly to get a different depth in focus. Each image is assigned two lables u and v referring to the two dimensions and ranging from [-6,7]. The amount of shift is decided by the relative distance between the reference image and the perspective image identified by values u and v, multiplied by a constant shift parameter to bring different objects in focus.

In order to focus nearby depths in the scene, images were shifted away from the reference image and when the far away objects had to be focused images were moved towards the reference image. As the actual depths were not known in the scene captured, we manually tweaked the shift parameter until we could focus on the objects of interest. Two different depths in focus can be seen in Fig. 4a and Fig. 4b, for a light field from first dataset.

(a) Image focused on objects at shallow depth (flower)

(b) Image focused on objects at large depth (Main Quad)

(c) Image with everything in focus

Figure 4: Images from the Focal Stack

An all in focus image, as shown in Fig. 4c was also generated from this focal stack. We looked at gradients in the stack at each pixel and chose a pixel from a focal plane with highest gradient. This process was followed for each pixel in the image and at the end we got an image where every object at its respective depth was in focus.

4.2. Depth Estimation

In order to estimate the depth of each pixel in the image, we generated a 2D map of the same size as image. The value at every location in the map was the index of the focal plane that was taken for that pixel from the focal stack for

generating the all in focus image. This gave us an estimate of the relative depths of objects in the scene, but was very noisy and therefore we transformed this result to HSV space and normalized it too to bring all values between [0,1].

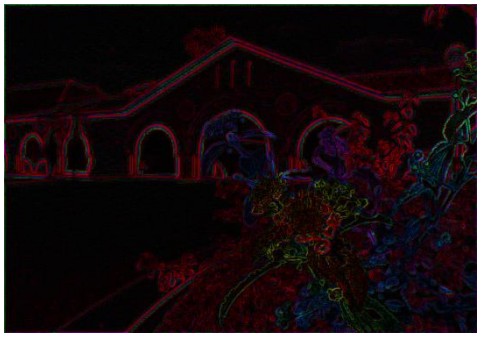

Figure 5: Depth Map Generated using Gradients

The result was a noise free estimate of different depths in the scene but this method disregards the regions with zero gradients. As ac lot of regions turned out to be black due to zero gradients, as can be seen in Fig. 5, we couldn't use this depth map explicitly as an input for gaze contingent display since we wanted each pixel to be annotated to a depth. This is due to the fact that depth map behaved as a look up table to map gaze coordinates to a particular focal plane which needed to be displayed.

Figure 6: Lytro Depth Map

Hence we used the inbuilt lytro depth map estimate as shown in Fig. 6 but quantized it to fewer number of levels, as can be seen in Fig. 7. This had to be done in order to remove flickering between focal planes as the eye gaze output jittered even in a fixated state (when looking at an object for a prolonged time), if there were too many depth values. When there were too many edges or holes in the depth map, this jittering became significant and caused erratic flickering between planes. Thus, we smoothed the gaze output to reduce the noise, compromising on the agility of tracking, but making it more robust. By quantizing the depth map and thereby the number of planes in the focal stack we got rid

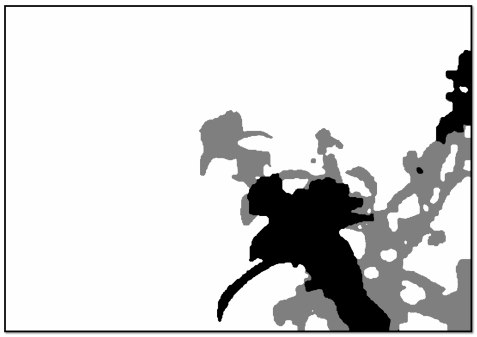

Figure 7: Quantized Lytro Depth Map

of the flickering effect such that the user had a comfortable viewing experience.

4.3. Anaglyph Stereo-Pair

In order to be able to preceive depth on a conventional LCD screen we made use of the anaglyph effect. Given a light field from a single viewpoint and its corresponding depth map and focal stack we generated a focal stack from a novel view. This was done as follows:

1. A copy of the existing focal stack was made, say *[Novel-View]*.

2. Each focal plane in *[Novel-View]* was shifted with respect to its corresponding focal plane in the original stack as per the following calculation:

$$S = \frac{f}{d}B$$

   where S is the amount of shift in pixels for the focal plane, f is the focal length in pixels, d is the depth of that plane in physical units and B is the Baseline in physical units (6.5 cm).

   Thus, focal plane wherein shallow depth was in focus was shifted more as compared to the focal plane where far away objects were in focus.

3. In the overlaping portions of the two corresponding focal planes, in one of the planes only red channel was activated and in the other green and blue. This composite image was generated for each depth in the scene and the resulting anaglyph focal stack was stored.

4. This focal stack was used as input to the program which used eye gaze coordinates and displayed the corresponding focal plane, hence allowing a user to perceive depth by wearing red-cyan glasses.

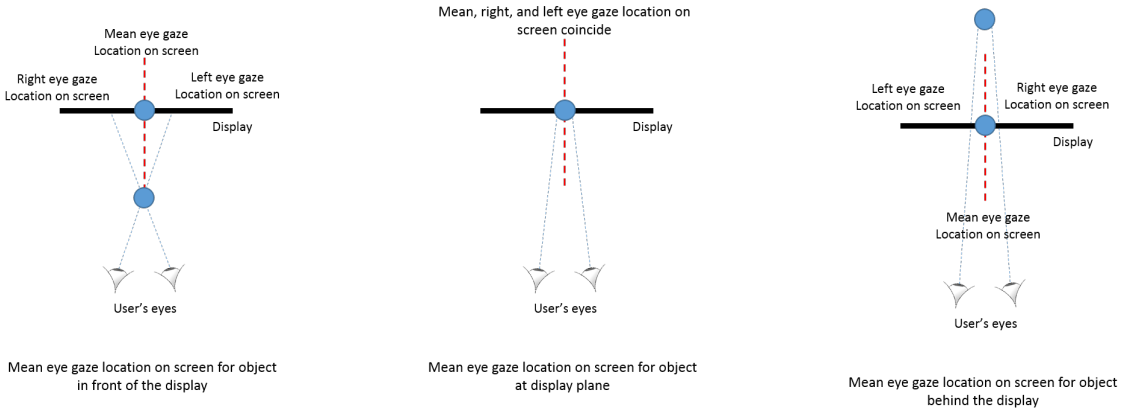

Figure 8: Left, right and mean eye gaze locations on screen

4.4. Eye Gaze Tracking

To determine the eye gaze position of the user on a computer display we use a commercially available eye tracker. This uses infrared micro projectors to create reflection patterns on the eye. Images captured by the eye tracker are then used to localize this pattern to determine eye gaze direction and eye position. We use the SDK supplied with the tracker to interface it with our application which in written in C++ using the QT Framework for UI elements.

Since the light source used by the tracker is infrared wearing anaglyph 3D glasses with red / cyan filters does not affect the functionality of the eye tracker. Although the eye tracker SDK provides a filtered version of the eye gaze location it was found that an additional low pass filter inside our application greatly reduced the noise and jitter in the eye gaze location.

4.4.1 Focal Plane Selection

The output from the eye tracker is the mean eye gaze location of both the eyes on the screen. This does not present a problem even when the user is looking at an object that appears in 3D space to not be in the plane of the display. This can be seen from Fig. 8 which shows the mean eye gaze location for the user looking at objects that appear to be both in front of and behind the plane of the display. In each case, the mean eye gaze location on the screen is still at the object that the user is looking at.

Once we have the user's eye gaze location we can see which depth level on the depth map that the user is looking at, once this is known we select the corresponding focal plane from the focal stack and display it to the user. This process happens near instantaneously with there being no fading effect or blending in the transition between the two images. Another method adopted to reduce the jitter and enhance the viewing experience was the use of the eye gaze locations corresponding only to fixations only and discarding the eye gaze locations obtained during which the eyes move substantially. The idea behind this is that if two gaze points are within a minimum distance from each other or have a speed below a certain threshold they are considered to be a fixation.

5. Implementation Details

In the actual implementation the depth map and the focal stack are generated offline using the algorithm described in the previous section. These are then loaded into memory by the C++ application to allow fast switching between focal planes. For the prototype implementation an image with three clearly distinguishable objects at different depths of focus is selected. This image used for demonstration purposes is shown in 9.

The three focal planes selected (front to back) are at the toy aeroplane, the face, and the background respectively. The depth map (Fig. 13) is quantized to contain only three different values each representing one of the focal planes mentioned above (Fig. 14). This means depending on the user's eye gaze location the program switches between three images displayed to the user.

Besides the one final image selected for the demonstration several other images were also tried taken with a lytro camera. It was observed that when the depth map has small holes in it as in (Fig. 7) when the user's eye gaze moves over the holes a flickering effect is caused by the program rapidly switching between the two depth planes corresponding to the area inside the hole and outside it. Similarly if the depth map contains irregular edges, these may also cause the image to switch quickly and rapidly between two focal plane when the user looks at the edges due to inherent uncertainty in the eye gaze location.

In order to reduce both these effects which are undesirable we fill the holes in the depth map and also artificially

increase the size of objects in the depth map so that the edges in the depth map do not correspond exactly to edges in the image. This means that even if the user looks at the edge of an object in the image, the image will not flicker between the two depth planes. This does lead to a slight inaccuracy in changing the depth planes (since the depth map now does not exactly correspond to the image) but it was observed that this effect is negligible and the increase in user comfort achieved by reduction in flicker is more substantial than the slight inaccuracy which is not noticeable by most users.

6. Results

A live demonstration for the gaze contingent display system was shown during the poster presentation for the course, wherein participants could refocus different depths in the scene based on their eye gaze. There were explicitly three different focal planes corresponding to three depths in the scene and as results presentable on paper we show some of the anaglyph images focused at different location and depth maps generated from these images.

7. Future Work

Our work in this project represents a prototype to which there could be many possible additions and enhancements, a few of them are mentioned below:

- In this project the transition between focal planes was instantaneous, however different types of transition functions can be explored. Subjective user studies can be carried out to determine which function produces the most natural looking and visually pleasing transitions.

- This project deals with changing of focal planes to simulate depth of filed effects however the light field can also be used in conjunction with a head tracker to simulate parallax when the user moves their head in 3D space. Head tracking capabilities are also present in the hardware we use for eye tracking so no additional hardware is required for this extension.

- Different depth map estimation algorithms can be tried to determine the one which gives optimal results for this problem.

References

[1] Mauderer, Michael, et al. "Depth perception with gaze-contingent depth of field." Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, 2014.

[2] Marshall, Jonathan A., et al. "Occlusion edge blur: a cue to relative visual depth." JOSA A 13.4 (1996): 681-688.

[3] Mather, George. "Image blur as a pictorial depth cue." Proceedings of the Royal Society of London B: Biological Sciences 263.1367 (1996): 169-172.

[4] Jacobs, David E., Jongmin Baek, and Marc Levoy. "Focal stack compositing for depth of field control." Stanford Computer Graphics Laboratory Technical Report 1 (2012): 2012.

[5] Duchowski, Andrew T., Nathan Cournia, and Hunter Murphy. "Gaze-contingent displays: A review." CyberPsychology & Behavior 7.6 (2004): 621-634.

[6] Reingold, Eyal M., et al. "Gaze-contingent multiresolutional displays: An integrative review." Human Factors: The Journal of the Human Factors and Ergonomics Society 45.2 (2003): 307-328.

[7] Lytro Images used for demo: https://s3.amazonaws.com/lytro-corp-assets/sample.zip

[8] Lytro Sample LFP Files for Download: http://lightfield-forum.com/2012/08/lytro-sample-lfp-files-for-download/

Figure 9: All in focus image. Light field taken from [7]

Figure 10: Image focused on the background.

Figure 11: Image focused on the face.

Figure 12: Image focused on the toy.

Figure 13: Depth Map from Lytro's algorithm

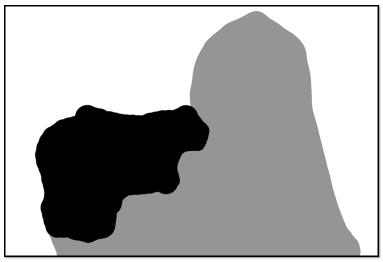

Figure 14: Depth map quantized to three levels