Effective Light Field Rendering for Thumbnails

Muhammad M. Almajid
School of Earth, Energy & Environmental Sciences
Energy Resources Engineering Department
Stanford University
Email: majimm0a@stanford.edu

Kuy Hun Koh Yoo
School of Earth, Energy & Environmental Sciences
Energy Resources Engineering Department
Stanford University
Email: kohykh@stanford.edu

***Abstract*—Light field (LF) cameras are gaining popularity due to advances in lens technologies and new post-processing capabilities. Consequently, new products are constantly being developed that are designed to attract more consumers. For example, there are currently several projects to develop and apply LF technology on phone cameras (i.e. Linx and Pelican Imaging). As the technology becomes more readily available to everyday consumers, new and creative rendering procedures will be required to maximize the utility of the captured data. For the technology to be adopted, the rendered images must provide fun/creative attributes that are beyond conventional image manipulation (i.e. social media). In this project, we exploit the fact that light field images have more data than conventional images to come up with rendered images that could be animated and saved in Graphics Interchange Format (gif).**

I. Introduction

Lightfield images are different than conventional images in that they capture the same scene from multiple perspectives. The light field is usually parameterized by the two-plane parameterization that was originally proposed by Levoy and Hanrahan [1]. In this parameterization, each ray is described by its intersection with the s, t plane and the u, v plane. These two planes are parallel to each other and separated by an amount that is equal to d as seen in Fig. 1. The s, t plane is much sparser than the u, v plane as it includes the lentslets of the camera (i.e. 7 $\times$7 versus 380$\times$380). Even though the s, t plane contains only a small number of perspectives of the scene, other perspectives can be derived from the measured ones using various rendering methods. In this report, we explore some methods used for light field rendering and shed light on future additions that could improve our algorithms.

II. Perspectives and Ray Tracing

In order to trace the rays captured by any given perspective on the s, t plane, we first parameterize the rays for an arbitrary point in space. For simplicity we choose our reference point to be [s$=$ 0,t$=$ 0,d$=$ 0]. We then determine all vectors that connect our reference point to the center of each pixel of an image and normalize. We also compute the projection of the normalized vectors onto a line passing through our reference point and perpendicular to both the s, t and u, v planes. Given our choice of reference point, this value is simply the z-component of our normalized vector. The collection of rays that pass through any point s, t, z is then given by:

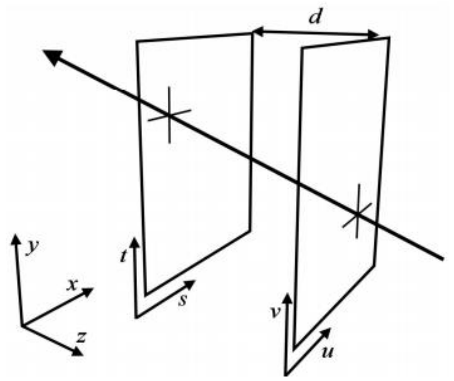


Fig. 1: Two-plane parameterization of light rays. Figure was taken from [2]

$$(u_1, v_1, d) = (s, t, z) + n_k \frac{D}{v_z} \tag{1}$$

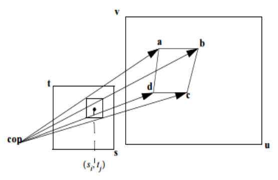
Where (s, t, z) represents the position of our virtual camera, n_k is the collection of normalized vectors, D is the distance between our virtual camera and the u - v plane and v_z is the z-component of our normalized rays.

Any angular rotations are achieved by multiplying by the corresponding rotation matrix. This parameterization automatically takes into account the field of view and original image resolution. We can now compute the intersection points of the rays entering a given perspective on the u - v plane (u_1, v_1, d) constrained to the FOV and image resolution.

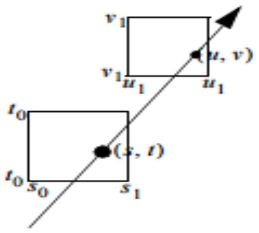
III. Rendering

A. Texture Mapping versus Quadrilinear Interpolation

Given our u, v plane ray intersection points for a given perspective, we can render our image using several techniques, Fig. 2a. Using textural maps involves selecting the pixels closest to the new incident u, v points. This technique produces some aliasing artifacts that are not visually appealing as shown in Fig. 3a. The alternative approach is based on interpolation over the 4 dimensions of our light field (i.e. quadrilinear interpolation) using the closest physically captured rays as shown in Fig. 3b. This method is effective in removing aliasing effects (see GIF 1).

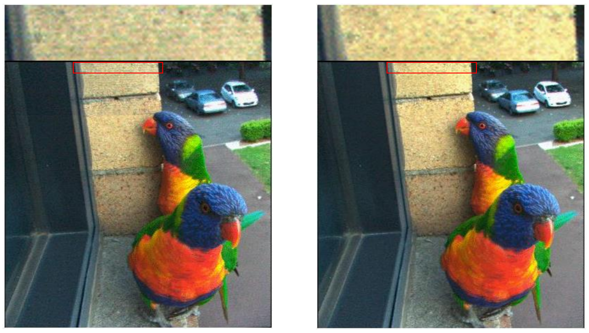


(a) Texture



(b)

Fig. 2: (a) Texture mapping, and (b) quadrilinear interpolation.



(a) (b)

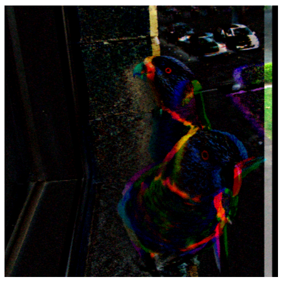
Fig. 3: (a) Texture mapping, and (b) quadrilinear interpolation. The red box indicates where the images were zoomed to illustrate aliasing effects.

Although we are able to produce clear images, quadrilinear interpolation is only effective for scenes that have no depth variations. When we have depth variations, we expect objects to move proportionally to their distance from the camera as we shift perspectives.

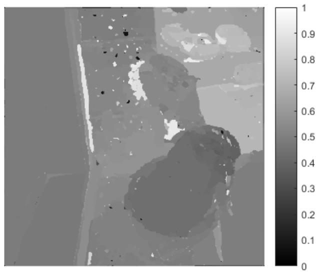
B. Depth-corrected Interpolation

The naive way to correct for depth is to assume a uniform relative depth across the entire scene. Although this is not a valid assumption for most cases, we explore the alternative to understand the importance of depth. To visualize the differences, we show image subtractions in Fig. 4. Figure 4a shows the difference between two images that have uniform depth but are scaled by increasing factors, in this particular case we use scaling factors of 2 and 5. We observe that all the pixels are equally shifted proportionally to the scaling factor. When we look at the GIFs generated from these uniform depth rendered images, we can see that it exaggerates the shift for all pixels (see GIF 2). Having this problem, we knew we had to estimate a depth map for the scene to have a more realistic shifts based on that.

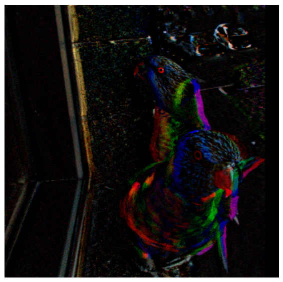
It is possible to estimate a qualitative depth for certain objects using the images captured on the light field. These are determined by the movement of pixels as we move from one lenslet to another. Nevertheless, this method requires that we have high contrast changes to determine an acceptable depth value. This results in sparse depth maps for most images. A dense depth map is, however, needed to correct for perspective



(a)



(b)



(c)

Fig. 4: (a) Uniform depth, (b) depth map estimate, and (c) after depth consideration.

since we would expect closer objects to move less as we move horizontally in the$s$, t plane.

To compute a dense depth estimate, we use an occlusion-aware method [3]. In this method, the candidate occlusion pixels are first computed by applying a Canny edge detection and dilating the resulting image. Then, an initial depth estimate is computed by shearing the light field data [4]. This gives a sparse depth map that needs to be filled. To achieve that, the algorithm performs depth regularization given an initial depth estimate and occlusion cues using Markov Random Field (MRF). Figure 4b shows the obtained dense depth estimate that we used.

Figure 4c shows the difference between a uniform relative depth versus the depth estimate in Fig. 4b. We notice that the further the object, the more shift it has, that is very similar to the uniform depth case. While when the object is close (like the front bird), the relative depth map adjusts its shift such that it gives the correct perspective (see GIF 3).

IV. Validation

In order to validate our developed methods, we compare how our methods perform against an example in the LF toolbox that chooses 2D slices of the light field image simulating a circular path [5] (see function "LFDispVidCirc" in the LF toolbox). To make this concrete, we plot the s, t lenslets chosen by the LF toolbox function against the lenslets we use by our rendering methods as shown in Fig. 5. Clearly, the LF toolbox function is limited to choosing between 81 (9×9) lenslets while we are free to choose any point in the s, t plane.

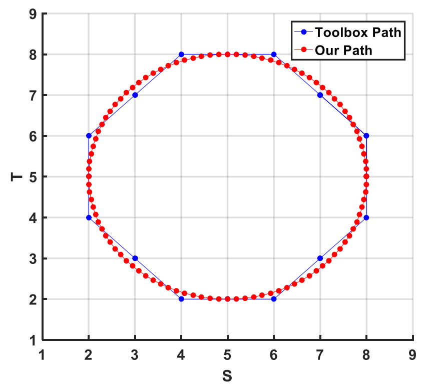
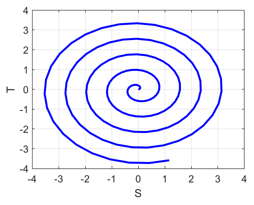
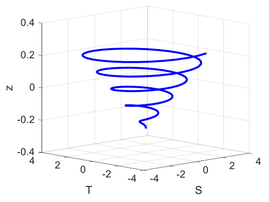


Fig. 5: Paths taken by the toolbox's method and our method overlaid on the s, t plane



(a) Spiral



(b) Tornado

Fig. 6: Perspective paths.

Our methods are able to generate a smoother circular path as proven by the generated (see GIF 4).

V. GIF Manipulation

A. Defining Perspective Path

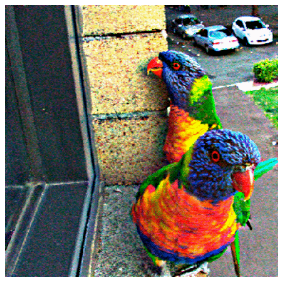
Now that we have increased our degrees of freedom to go in and out of and anywhere on the s, t plane, we are able to follow any defined path to generate different effects. We have generated two interesting paths as shown in Fig. 6. The first path is a spiral path that goes in circles in the s, t plane starting from small circles and ending at wider ones. The second path we chose was a tornado path that goes in circles and in and out of the s, t plane. An example for the spiral and tornado paths can be found in GIF 5. We notice that there are some aliasing effects at the end of the tornado path which are due to the inaccuracy of our depth estimate.

B. Vertigo Effect

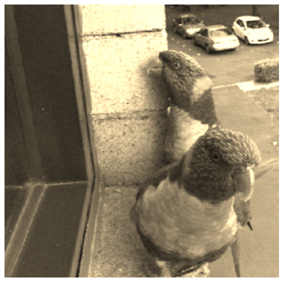
We were also able to perform the vertigo effect (a.k.a dolly zoom) from the light field image. This was achieved by adjusting the field of view as we go in and out of the s, t plane. All of the GIFs provided in the attached folder have a portion that shows the vertigo effect.

C. Fun Filters

In order to make the pictures more interesting, we can apply filters to the images before we generate the GIFs. We have tried only two filters here but other manipulations could be easily applied. The first filter we used is a morphological one as shown in Fig. 7a. The second filter changes the opacity of the image as shown in Fig. 7b. These filters were randomly generated by the authors by playing around with different parameters (i.e. RGB channel scaling, dilation/erosion by structural elements, etc.) until visually appealing images were achieved.



(a) Morphological



(b) Opacity

Fig. 7: Different filters applied to the images for fun visualizations.

VI. Conclusion and Future Work

Light field camera typically have a sparse s, t plane that includes a small number of lenslets. During this project, we were able to produce virtual camera perspectives from locations on the s, t, d that are not measured when the image is captured. Three methods were tested. Texture mapping was the simplest method but it gave us undesirable aliasing. Quadrilinear interpolation fixed the aliasing effects but it did not capture the right perspective of objects according to their depth. The perspective issue was fixed by the last method, which is the depth-corrected interpolation where we used a uniform depth and a relative dense depth map estimate. The best results were obtained using the relative depth-corrected interpolation. Our results were validated against the LF toolbox function "LFDispVidCirc" and showed much smoother views.

For future work, we recommend finding better estimates of the depth map such that the rendering becomes even smoother. Moreover, better rendering could potentially be achieved by a patch-based approach, where the light field patches are modeled using a Gaussian mixture model (GMM) [6]. Lastly, if we wish to have these rendering techniques reach the everyday consumers, then we need to make these algorithms more computationally efficient.

Acknowledgment

The authors would like to thank Donald Dansereau for his mentorship, feedback, and useful discussions during this project. Also, we would like to thank the teaching staff: Gordon Wetzstein, Jean-Baptiste Boin, and Hershed Tilak for a very fun and enjoyable class.

List of GIFs

All GIF files can be found in the GIFS folder. Instructions on how to generate the GIFs and use different parameters is outlined in the code's README.pdf file.

1) **GIF 1**: Textural Mapping vs. 4D Interpolation
2) **GIF 2**: Constant Depth Comparison
3) **GIF 3**: Depth Map Consideration
4) **GIF 4**: Validation Circular Path
5) **GIF 5**: Spiral and Tornado
6) **GIF 6**: Image Filters and Effects

References

[1] M. Levoy and P. Hanrahan, "Light field rendering," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM, 1996, pp. 31–42.

[2] D. Dansereau and L. Bruton, "Gradient-based depth estimation from 4d light fields," in *Circuits and Systems, 2004. ISCAS'04. Proceedings of the 2004 International Symposium on*, vol. 3. IEEE, 2004, pp. III–549.

[3] T.-C. Wang, A. Efros, and R. Ramamoorthi, "Occlusion-aware depth estimation using light-field cameras," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015.

[4] R. Ng, M. Levoy, M. Brédif, G. Duval, M. Horowitz, and P. Hanrahan, "Light field photography with a hand-held plenoptic camera," *Computer Science Technical Report CSTR*, vol. 2, no. 11, pp. 1–11, 2005.

[5] D. G. Dansereau, O. Pizarro, and S. B. Williams, "Decoding, calibration and rectification for lenselet-based plenoptic cameras," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 1027–1034.

[6] K. Mitra and A. Veeraraghavan, "Light field denoising, light field superresolution and stereo camera based refocussing using a gmm light field patch prior," in *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, 2012, pp. 22–28.

Appendix A
Work Breakdown

- Light field data acquisition - Muhammad
- Textural and 4D interpolation - Kuy
- Depth-corrected interpolation - 50/50
- Validation - 50/50
- Paths - 50/50
- Filters - 50/50
- Final report and poster - 50/50