CS276 Fall 2020

Computational Photography

Assignment 1: Light Field Rendering

Due: October 3rd 11:59 p.m.

Implement a Light Field Renderer

Light fields are simple but useful image-based rendering primitives. In this assignment, you will be asked to implement a light field renderer that support various rendering effects, such as narrow and wide aperture field, synthetic depth of view effects, and translational motion of the virtual viewing camera. It will also familiarize you with GUI and file IO.

Part I. IO and GUI

A light field consists of an array of images. You should start with an IO interface that reads in the light fields. You should write some code to generate the light field script. One sample light fields will be provided to class. (link:https://pan.baidu.com/s/1bzhkjO pass: imqs, or http://pan.shanghaitech.edu.cn/cloudservice/outerLink/decode?c3Vnb24xNjAwMzM2ODIyMzM5c3Vnb24=)

1. Your program should read in an array of images and store them in the memory system. Since the light field usually consists of 256~1K images, your program should work on a machine with reasonably big memory. As is discussed in class, an efficient way to store LF is to use the stuv coordinate system, where st represents the camera index and uv represents the pixel index.
2. Your program should have a UI to allow you change the location of the camera, the focal plane, and the aperture size. You are required to implement the translational motion of the virtual viewing camera (along x, y, and z directions). You do not need to implement the rotation of the camera, although doing so will make your renderer much more interesting. You need to set up your translation step small enough so as to observe the interpolated views between the data cameras.

Part II. Interpolation Scheme

3. You should start with the simplest interpolation scheme, i.e., quadralinear (or bilinear if assume uv is sufficiently sampled) interpolation. What do you observe if you use bilinear interpolation on undersampled light field?
4. Introducing a new control of the focal plane (e.g., a new scrollbar in your program window). Map the depth of the focal plane to the disparity and repeat 4 with the variable focal depth. What happens when you move your focal plane from far to near? Which focal plane gives you the optimal results (least aliased reconstruction)?
5. Next, implement the wide aperture filter. You can use Gaussian weight as described in Dynamically Reparametrized Light Field. Add another control for the size of the aperture. What happens when you increase the size of the aperture?
6. Finally, implement the z directional motion of the camera. I would suggest that all camera motions be controlled by mouse (or keyboard) while the aperture size and focal depth by a scroll bar.

Extra Credit

Implement the z directional motion of the camera. I would suggest that all camera motions be controlled by mouse (or keyboard) while the aperture size and focal depth by a scroll bar.

You need to Submit

1. Code (Python, C/C++, Matlab), DO NOT COPY & PASTE, We will check.
2. A Short Report in English containing overall design and what you have done.

   a) Using LaTeX template would be great, elegant formula derivation and composing can make your report get higher scores.
   b) Basic sections of your report: Introduction / Implement(Derivations of rendering principals) / Experiments / Conclusion(analyzing your program and the algorithms, is there any possible way to improve your result or performance?)

3. Video showing that your light field renderer is working correctly.
4. Upload **your report** to Gradescope, and send **your video** to cp2020fall@126.com, email title is: Assignment 1 Video [Your name] [Student number], then your video name is: [Your name]_[Student number].mp4(or any fashion format)

Start early and happy coding!