CS635: Assignment #3 – Fun with Lightfields

Out: February 21, 2025
Due: March 14, 2025

Objective

The objective of this assignment is for you to play with Lightfields which is a fundamental component of computational imaging/photography and even Neural Radiance Fields. To this end, you are free to seek whatever resources you can find, short of downloading some existing implementation, of course. Aside from your rendering engine, the amount of code for this assignment is quite small. Its mostly about camera control, ray casting, and then accessing weighted combinations of pixels.

Detailed Description

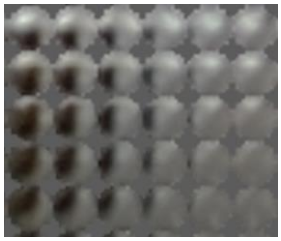
Step 0 – Lightfield Imagery

There are many repositories of already captured lightfield images; for example:
https://github.com/lightfield-analysis/resources

From that website you can reach the Stanford Archive for Lytro Ilum images:
http://lightfields.stanford.edu/LF2016.html

These are images captured by the Lytro Illum camera which was 40MB but had an array of microlenses, so that the effective resolution claimed to be about 4MP.

Here is a full view and a close-up of one the Lytro Illum images:

Full resolution is 7574 x 5264. Calibrating and decoding such a microlens array setup is not trivial. This work proposes a toolset:
http://www-personal.acfr.usyd.edu.au/ddan1654/PlenCal.pdf
https://github.com/doda42/LFToolbox

but its non-trivial. So instead, lets create our own simple lightfield dataset.

Step 1 – Render Lightfield (50%)

You need to create a simple rendering engine able to render something of interest to you. With this renderer, you need to specify the camera location, viewing direction, and FOV, and be able to save the rendered image.

Using the renderer, define an imaginary plane on which the camera will translate. Translate the camera along this plane (e.g., XY plane and looking down -Z). For 32x32 camera positions observing an indoor scene/object, the translation baseline is probably quite small (a few centimeters). At each camera position (S,T), you render a FOV of pixels, call them the (U,V) plane – it should be the same UV plane. One way to do this is to use the glFrustum-style matrix where you specify the left, right, bottom, top, near and far matrices (https://learn.microsoft.com/en-us/previous-versions/ms537094(v=vs.85)). So even though your camera moves, the image plane can remain constant (e.g., as you "move to the left", the frustum stays looking "a bit to the right" and so forth. Another approach is to use a homography (e.g., image transformation) to reproject the same corresponding subset of the scene to a common image rectangle.

Please create at least a lightfield of at least one scene of your choosing and with at least 16x16 camera positions (32x32 is a better number).

Step 2 – Reconstruct Novel View (50%)

Now, for a novel view, you choose a new viewpoint/direction/FOV and then shoot rays through all pixels and they will hit ST and UV values. Use nearest neighbor or quadrilinear interpolation. Repeat for each ray. Background (empty pixels) can just be color black and will work just fine. This novel view creator does not need to run at highly interactive rates but it should generate a result "fairly quickly" so that we can alter the viewpoint and see the new image relatively quickly.

Nearest neighbor: 25%
Quadrilinear interpolation: 25%

Step 3 – Focal Distance (Extra credit: 25%)

As an optional step, you can choose a focal plane at some distance from the camera. Then, for each ray rather than a quadrilinear interpolation (which essentially assumes a focal plane at infinity), you chose neighboring rays that converge to the focal plane. You should also define an aperture which means do not just combine immediately neighboring pixels but some circular range of neighboring pixels, perhaps using a Gaussian weighted fall-off.

During your demo session, I will use the provided zip-file to grade your program. Your grade will be influenced by how well your particular solution works, by the presentation and usability of your program, and by how well you complete the assignment requirements.

In this assignment, you may collaborate only to help with the mechanics of the assignment. *Everybody must use their own images however*! Practically speaking, this means that nobody should have the same pictures or results.

If you have questions, please come see me ASAP – do not wait until the last moment. Have fun and good luck!