Bachelor Project Journal

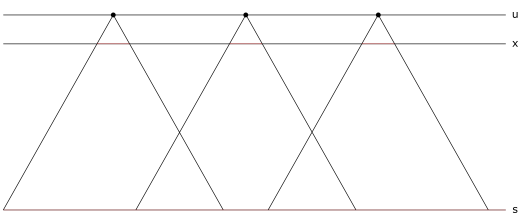
Adrian Wälchli

May 19, 2015

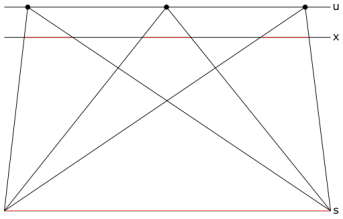
Abstract

This report presents an overview of my bachelor thesis. We discuss several approaches to problems, experiments, ideas and evaluate results. This document will be extended over time as the project evolves.

Contents



(a) Perspective projections, images are shifted



(b) Sheared Projections, cameras have a common image plane

Figure 1: Different methods to capture a lightfield

1 Related work

The basis of this project are the papers from [WLHR11, WLHR12]. Additional papers used for the work are: Light Field Rendering by Marc Levoy and Pat Hanrahan, Fourier Slice Photography by Ren Ng, Light Field Photography with a Hand-held Plenoptic Camera by Ren Ng et al.

2 Types of light fields

In this project, I encountered two types of 4D light fields that are captured using camera grids. The most common way of acquiring a light field is to capture a scene with a 2D-grid of cameras where the optical axes of the cameras are orthogonal to the camera grid. Since the look-at-point of each camera is different, this setup will result in a shift in the images formed on the sensors. An alternative way to capture the scene is to fix the look-at-point for every camera, preferably at the origin of the scene. This is the type of light fields primarily used in the paper from [WLHR11].
In addition, the images can be optained using either perspective or orthographic projections. We can also use sheared projections as mentioned in [LH96, p. 4].

3 A first implementation

In a first step, I (re-)implemented the tomographic light field synthesis for layered 3D-displays in MATLAB, based on the theory in [WLHR11] and their publicly available MATLAB code. The core problem to solve is:

$$\begin{aligned} &\underset{x}{\text{argmin}} && \|Px + \bar{l}\| \\ &\text{subject to} && 0 \leq x \leq 1 \end{aligned} \tag{1}$$

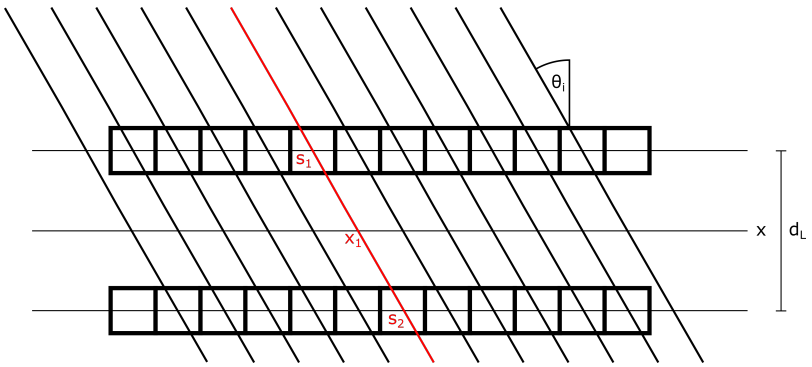


Figure 2: The rays captured by camera i using orthographic projection. All rays from one camera have the same angle θ_i, which is measured relative to the surface normal of the layers. x_1, s_1 and s_2 denote the intersections of the ray with the camera sensor and the layers. d_L is the distance between the layers.

Figure 2 shows the way we can construct P. For each camera we know the angle θ_i. Lets assume we have two layers. We can place the sensor plane between the two layers as shown in figure 2. For a pixel x_1 on the sensor plane, we can compute the positions s_1 and s_2:

$$s_1 = x_1 + \frac{d_L}{2}\tan(\theta_i) \qquad\qquad s_2 = s_1 - d_L \tan(\theta_i)$$

Once we have the positions s_1 and s_2, we compute linear indices k from (i, x), l_1 from $(s_1, 1)$ and l_2 from $(s_2, 2)$. Finally, we set $P(k, l_1) = 1$ and $P(k, l_2) = 1$. The extension for 4D-light fields and more layers is straightforward.

Having constructed the matrix P which defines a system of linear equations, I used the iterative linear least squares solver *lsqlin* in MATLAB to find a solution of equation 1. This method turns out to be too slow when the matrix is very large. I found another iterative method called The Simultaneous Algebraic Reconstruction Technique (SART) that turns out to be efficient for my problem. It is often used in tomography applications. For the definition and convergence analysis of SART, I refer to [Yan10].

4 Moving to light fields of type 1

The next challenge is to support light fields of type 1, as described in section 2. The motivation comes from the fact that most (online) light field archives provide datasets of this type. And there is also the plenoptic camera we can produce light fields with.
The requirements for this setup are the following:

- A camera plane of known size/camera positions
- The distance from the camera plane to the scene origin: z
- The distance from the camera plane/center of projection to the sensor plane: d_s
- The field of view of the cameras
- The size and placement of each layer relative to the scene origin

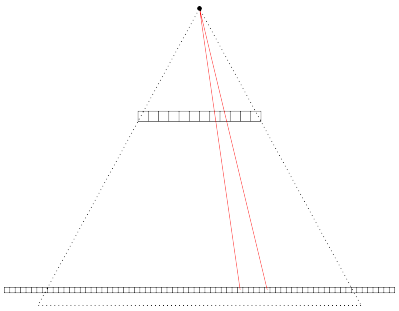
I make the assumption that the cameras are pinhole cameras. The disparity of the images is given by $D = x_1 - x_2 = \frac{d_s d_c}{z}$, where d_c is the distance between two cameras.

4.1 Approach 1: From camera pixels to layer pixels

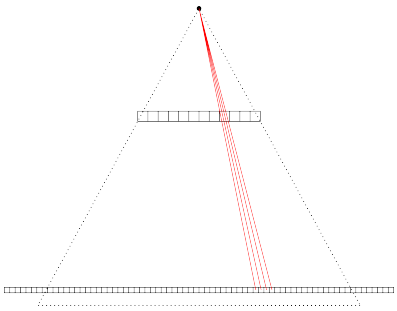
My idea for this approach is to go through each pixel for each camera and trace back the ray going through this pixel and the center of projection. Knowing the ray direction, we can compute the intersection with each layer. This gives us pixel correspondences between camera pixel- and layer pixel coordinates. For the valid intersections, we would then add the value 1 in the matrix at the respective index.
This method does not seem to work. The main problem is that a lot of layer pixels may not be hit by rays for a camera. It heavily depends on the resolution of the cameras and the layers for this method to work.

4.2 Approach 2: Converting the light field

The next idea is to reparametrize the light field $l(c_y, c_x, y, x)$ to a angular (epipolar???) representation $l'(\theta_y, \theta_x, v, u)$. Here, the number of angles corresponds to the resolution of the camera sensor. The motivation behind this approach is that we can fix (θ_y, θ_x) in the latter representation and get an orthogonal view. We would then plug this light field into the old algorithm to solve for the layers.
For the reparametrization, we construct matrices



(a) Two rays hitting a cameras sensor in neighbouring pixels. The two rays intersect with the layer pixels, with multiple pixels in between.



(b) Rays coming from four different layer pixels and hitting the same sensor pixel in the camera.

$$
\begin{aligned}
&C_y\left(\theta_y, :, v, :\right)\\
&C_x\left(:, \theta_x, :, u\right)\\
&I_y\left(\theta_y, :, v, :\right)\\
&I_x\left(:, \theta_x, : u\right)
\end{aligned}
$$

of the same dimension as the light field resolution. The ":" represents replication of the matrix in the respective dimension. We use these matrices to obtain a interpolated light field $l'\left(\theta_y, \theta_x, v, u\right)$.

TODO: Why didn't it work in the end?

4.3 Approach 3: From layer pixels to camera pixels

This idea is basically the opposite of approach 1. For every layer pixel and for every camera we compute the ray intersection on the sensor plane. The positions x_1 and x_2 shown in figure 4 are computed as follows:

$$x_1 = \left(s_1 - u_1\right)\frac{d_s}{z + d_L} \qquad\qquad x_2 = \left(s_1 - u_2\right)\frac{d_s}{z + d_L}$$

These points will be scaled and rounded to their corresponding pixels. We can then construct the propagation matrix P the same way as in section 3. As demonstrated in figure 3b, it often happens that a patch of layer pixels gets mapped to the same camera pixel due to rounding and thus, one pixel from the camera would contribute to multiple layer pixels. This also results in high column sums of the matrix P from equation 1.

5 Creating synthetic light fields

I used light fields from the Heidelberg and Stanford light field archives. In addition, I also rendered synthetic scenes with POV-Ray. The advantage of a synthetic light field is that the cameras can be precisely placed and all the required parameters are known and can be adjusted easily.

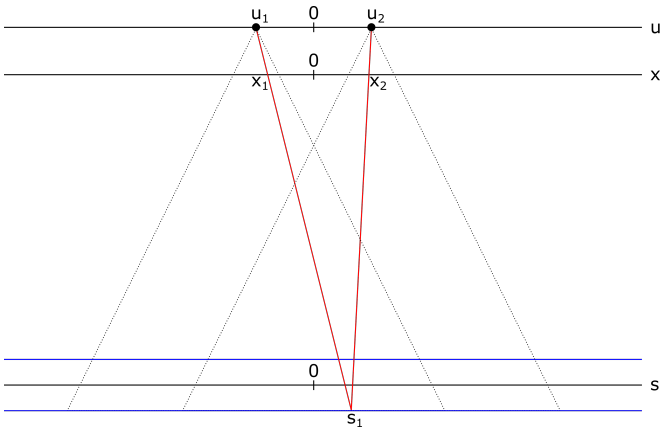


Figure 4: Two rays (red) intersecting the first layer (blue) at position s_1. The rays are captured by different cameras at positions u_1 and u_2, hitting the camera sensor at locations x_1 and x_2. The dotted lines represent the field of view of each camera.

References

[LH96] M. Levoy and P. Hanrahan. Light field rendering. *?*, ?(?):1–12, 1996.

[WLHR11] G. Wetzstein, D. Lanman, W. Heidrich, and R. Raskar. Layered 3D: Tomographic image synthesis for attenuation-based light field and high dynamic range displays. *ACM Trans. Graph.*, 30(4), 2011.

[WLHR12] G. Wetzstein, D. Lanman, M. Hirsch, and R. Raskar. Tensor Displays: Compressive Light Field Synthesis using Multilayer Displays with Directional Backlighting. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 31(4):1–11, 2012.

[Yan10] Ming Yan. Convergence analysis of sart by bregman iteration and dual gradient descent. *?*, pages 1–15, 2010.