Bachelor Project Journal

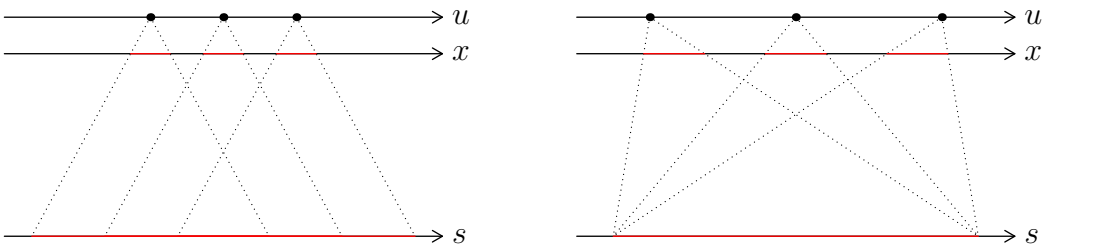
Adrian Wälchli

August 20, 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. Sheared projections can also be used as mentioned in [LH96, p. 4].

3 Notation

The two-plane representation is used to describe a 4D-light field $l(u, v, s, t)$, where (u, v) is the coordinate for the camera plane and (s, t) for the focal plane.

Symbol	Meaning
d_c	Distance between two cameras
d_s	Distance between image plane and center of projection
z	Distance between the camera plane and the scene origin
u_j	Position of camera j
x_j^i	Position of pixel j on the image plane of camera i
s_j^i	Position of pixel j on layer i
d_L	Distance between two layers

4 A first implementation for orthographic projections

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

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

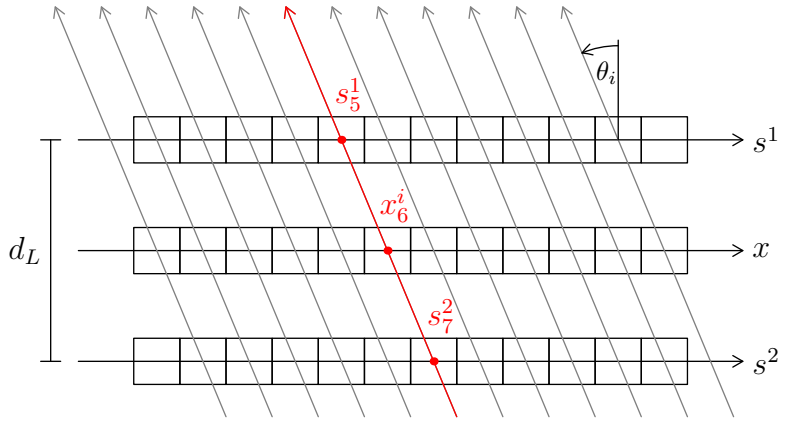


Figure 2: The rays captured by camera i using orthographic projection. All rays from one camera have the same angle θ_i, which is measured relative to the surface normal of the layers. x_6^i, s_5^2 and s_7^1 denote the intersections of the ray with the camera sensor and the layers. d_L is the distance between the layers.

where $\bar{l}$ is the log of the original light field l. The solution x is in log-space as well, therefore the constraints are chosen accordingly since the layers have finite contrast, i.e. $0 \leq e^x \leq 1$.

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 pixels x^i on the sensor plane, we can compute the positions

$$s^1 = x^i + \frac{d_L}{2} \tan(\theta_i) \qquad \text{and} \qquad s^2 = s_1 - d_L \tan(\theta_i).$$

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

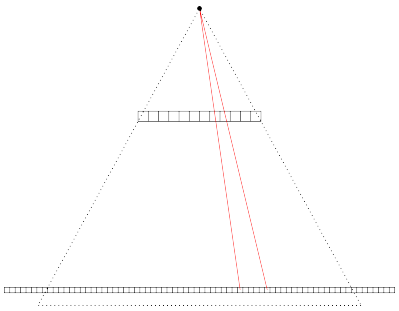
Having constructed the matrix P which defines a system of linear equations, I used the iterative linear least squares solver *lsqlin* in MATLAB to find a solution of equation 1. This method turns out to be too slow when the matrix is very large. I found another iterative method called The Simultaneous 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].

5 Moving to light fields of type 1

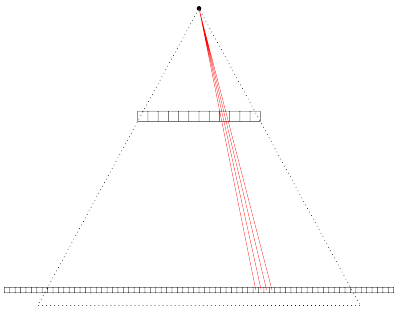
The next challenge is to support light fields of type 1, as described in section 2. The motivation comes from the fact that most (online) light field archives provide datasets of this type. And there is also the plenoptic camera we can produce light fields with.
The requirements for this setup are the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



(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.

Figure 3: Problems that can arise from different resolution in image- and layer space.

5.1 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. The problem is shown in figure 3a.

5.2 Approach 2: Converting the light field

The next idea is to reparametrize the light field $l\left(c_y, c_x, y, x\right)$ to an angular representation $l'\left(\theta_y, \theta_x, v, u\right)$. 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

$$C_y\left(\theta_y, :, v, :\right) \qquad C_x\left(:, \theta_x, :, u\right) \qquad I_y\left(\theta_y, :, v, :\right) \qquad I_x\left(:, \theta_x, : u\right)$$

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?

5.3 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_j^1 and x_j^2 shown in figure 4 are computed as follows:

$$x_j^1 = \left(s_i^1 - u_1\right)\frac{d_s}{z + d_L} \qquad x_k^2 = \left(s_i^1 - u_2\right)\frac{d_s}{z + d_L}$$

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 4. 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.

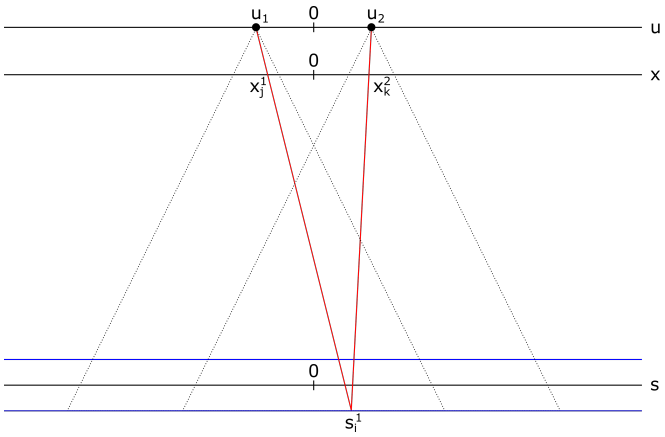


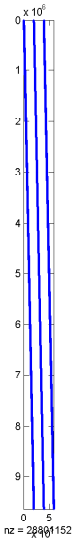
Figure 4: Two rays (red) intersecting the first layer (blue) at position s_i^1. The rays are captured by different cameras at positions u_1 and u_2, hitting the camera sensor at locations x_j^1 and x_k^2. The dotted lines represent the field of view of each camera.

6 Creating synthetic light fields

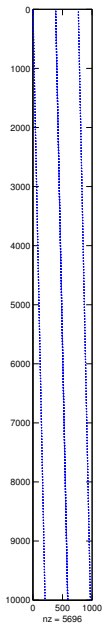
I used light fields from the Heidelberg and Stanford light field archives. In addition, I also rendered synthetic scenes with POV-Ray. The advantage of a synthetic light field is that the cameras can be precisely placed and all the required parameters are known and can be adjusted easily.

7 Review of the two implementations

7.1 Orthographic projections

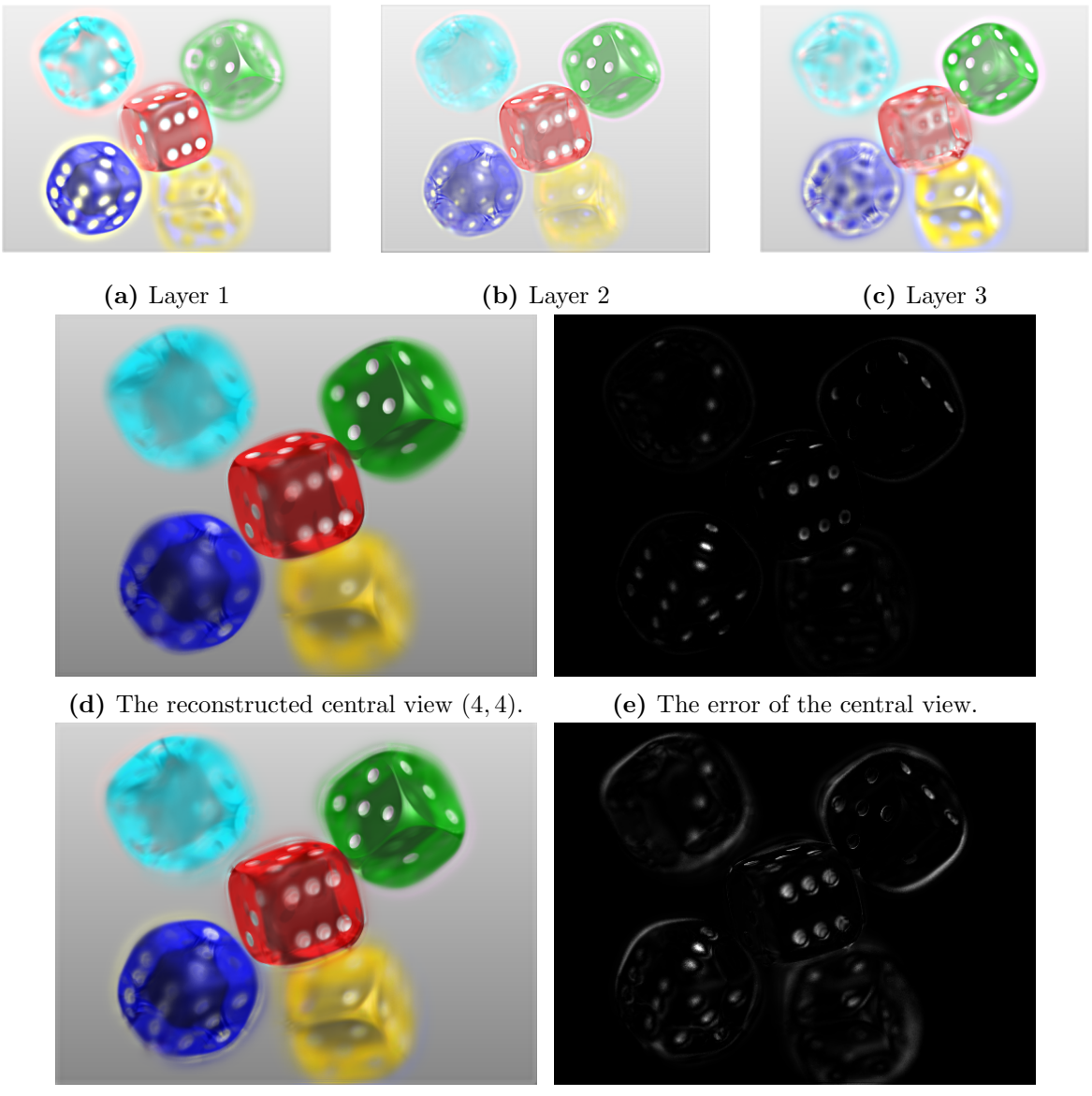


(a) Full matrix.



(b) The upper left section of the matrix.

Figure 5: The structure of the propagation matrix P. The non-zero elements are marked as blue.



(a) Layer 1 **(b)** Layer 2 **(c)** Layer 3

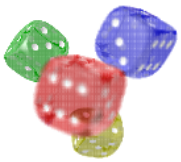
(d) The reconstructed central view $(4, 4)$. **(e)** The error of the central view.

(f) Reconstruction of the view $(7, 7)$, the bottom-rightmost view. **(g)** The error of the view $(7, 7)$.

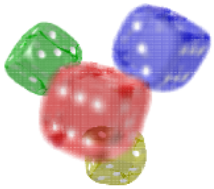
Figure 6: Optimization for three layers. The light field has an angular resolution of 7×7 views. The blur in the reconstruction is also in the original light field.

7.2 Perspective projections

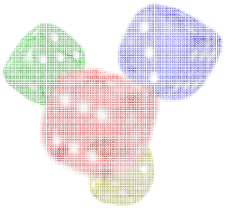
(a) Layer 1



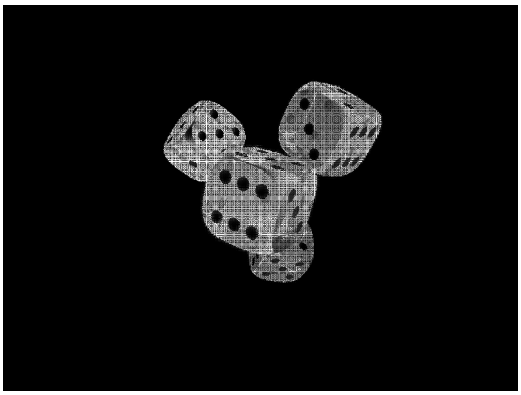
(b) Layer 2



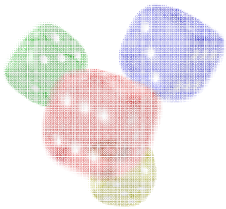
(c) Layer 3



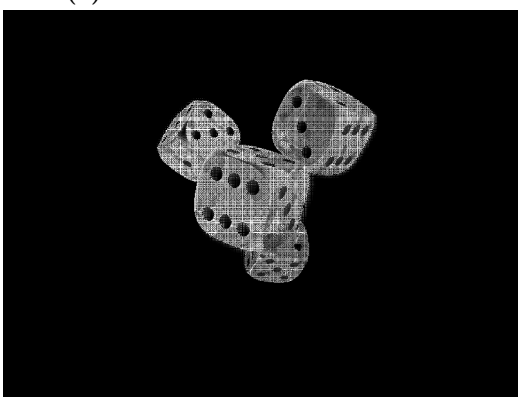
(d) The reconstructed central view $(3, 3)$.



(e) The error of the central view.



(f) Reconstruction of the view $(5, 5)$, the bottom-rightmost view.



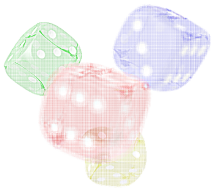
(g) The error of the view $(5, 5)$.

View	orthographic	perspective
$(3, 3)$	?	54.925458
$(5, 5)$	?	127.421736

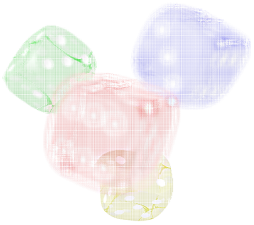
(h) Root-mean-square-error between the original and the reconstructed view.

Figure 7: Optimization for three layers. The light field has an angular resolution of 5×5 views. Parameters: $z = 8$, $d_c = (0.05, 0.05)$, fov $= (60°, 45°)$, $d_L = 1.5$

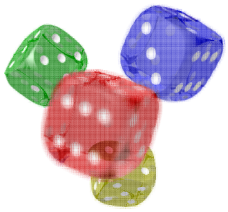
(a) Layer 1



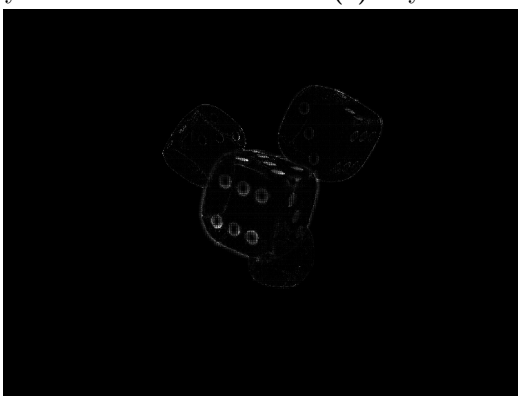
(b) Layer 2



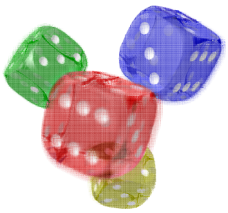
(c) Layer 3



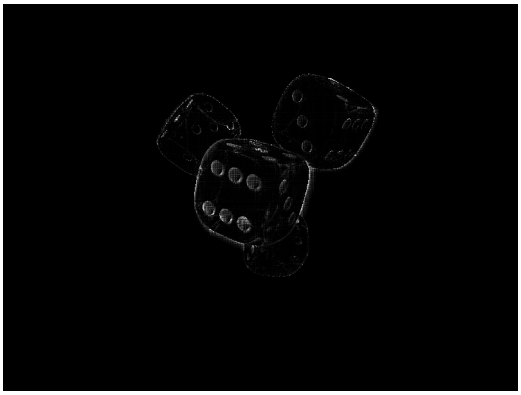
(d) The reconstructed central view $(3, 3)$.



(e) The error of the central view.

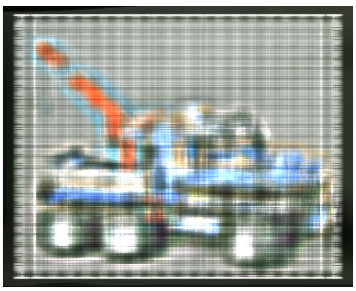


(f) Reconstruction of the view $(5, 5)$, the bottom-rightmost view.

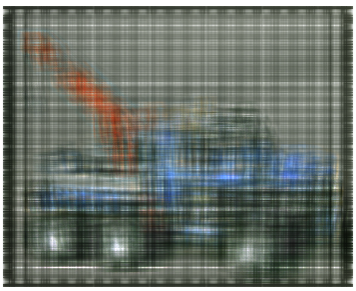


(g) The error of the view $(5, 5)$.

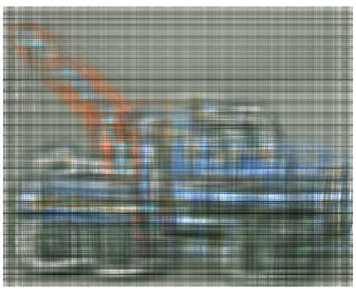
Figure 8: Optimization for three layers. The light field has an angular resolution of 5×5 views.



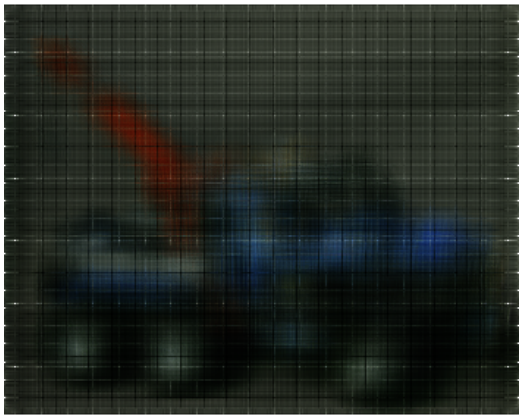
(a) Layer 1



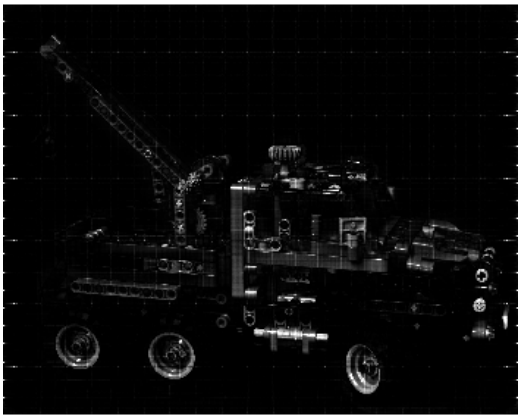
(b) Layer 2



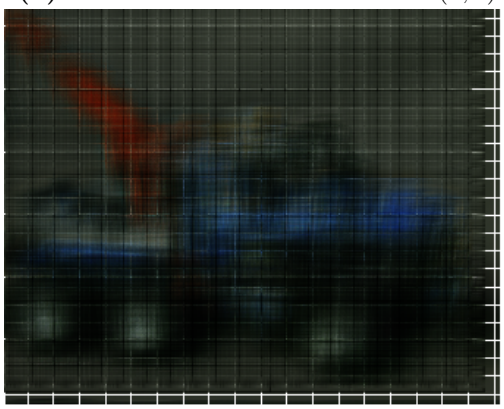
(c) Layer 3



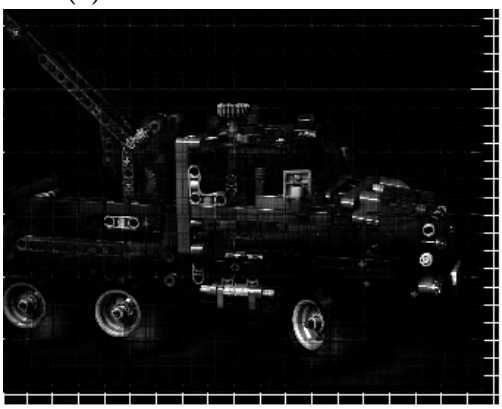
(d) The reconstructed central view $(5, 5)$.



(e) The error of the central view.

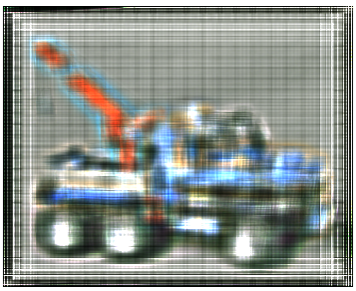


(f) Reconstruction of the view $(9, 9)$, the bottom-rightmost view.

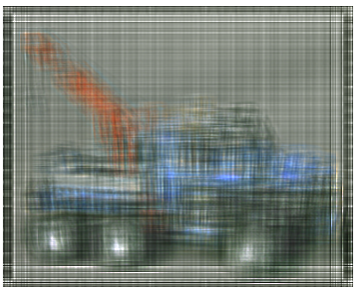


(g) The error of the view $(9, 9)$.

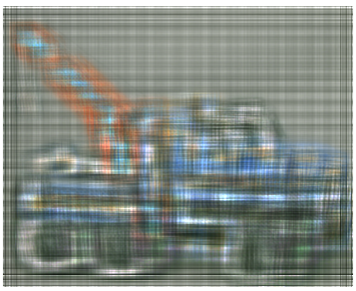
Figure 9: Optimization for three layers. The light field has an angular resolution of 9×9 views.



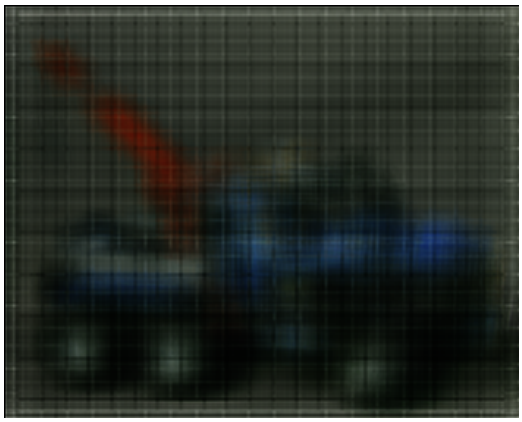
(a) Layer 1



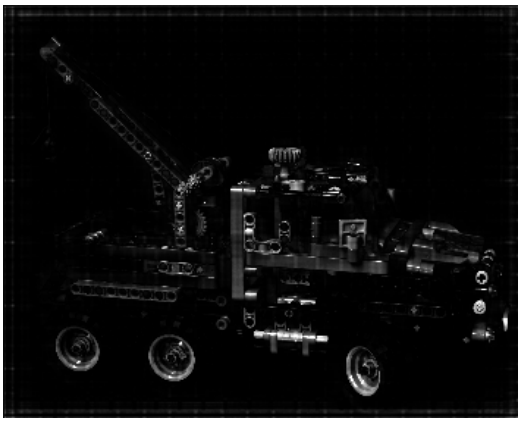
(b) Layer 2



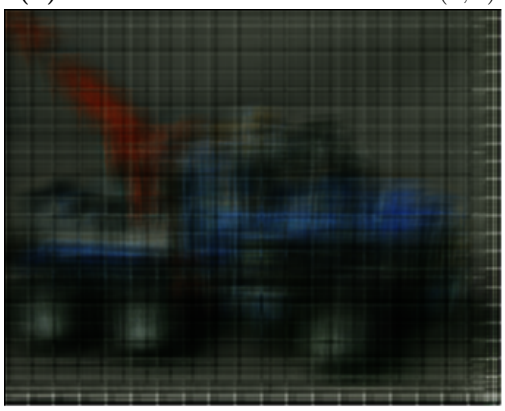
(c) Layer 3



(d) The reconstructed central view $(5, 5)$.



(e) The error of the central view.

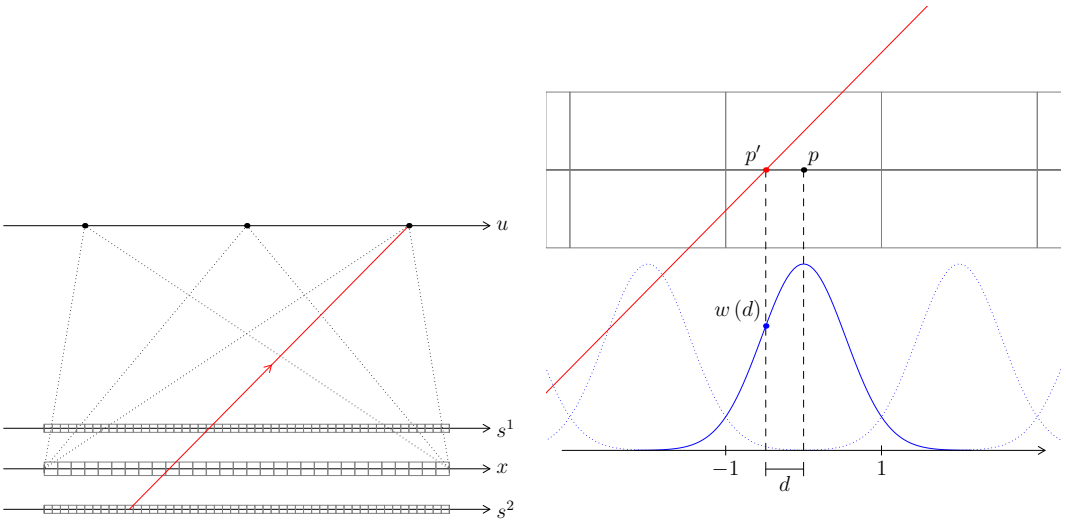


(f) Reconstruction of the view $(9, 9)$, the bottom-rightmost view.

(g) The error of the view $(9, 9)$.

Figure 10: Optimization for three layers. The light field has an angular resolution of 9×9 views.

8 Fixing the artefacts that occur in the reconstruction



(a) Ray (red) coming from a pixel on layer s^2 intersecting the sensor plane x and the topmost layer s^1.

(b) The ray intersects at p'. The nearest pixel with center p is considered and a weight $w(d)$ is computed from the deviation $d = p - p'$.

Figure 11: Calculation of the intersections and weights.

The main idea is to use a kernel to weight the ray-pixel-correspondences. As depicted in figure 11, the weight is computed from the deviation of the pixel center. For any ray, the nearest pixel gets selected and a weight is calculated. I considered and tested the following configurations:

- Weights only for the sensor plane x
- Combination of weights on sensor plane and each layer
- Box-Filter for the sensor plane only
- Combination of Box-Filter on sensor plane and each layer

During the implementation of the box-filter it came to my attention that the propagation matrix P should be normalized along the rows. This normalization alone seems to have a drastic impact on the artefacts we encountered. Figure 12 compares the impact of normalization on the reconstruction quality. Notice that figure 12b shows the best result, albeit no weights were calculated, and all other settings have no visible influence. The conclusion is: Adding weights to the system seems to have little to no impact on the optimization of the layers and the reconstruction quality.
It is also worth noticing that the row-normalization cancels out the sensor weights. This is due to the fact that during the construction of P, each entry (layer weight) of a row gets multiplied by the corresponding sensor weight.

8.1 Comparison of the three methods implemented so far

1. Rays from sensor to layers
   - – Layer pixels missed by rays (figure 3a)
   - – Artefacts/Aliasing
   - – Sensitive to small parameter changes
   - \+ No need to adjust/shear the camera images
2. Rays from layers to sensor (figure 3b)

(a) No normalization, box radius 0, $w \equiv 1$.

(b) Row-normalization, box radius 0, $w \equiv 1$.

(c) No normalization, box radius 0, weights from normal distribution $\mathcal{N}(0, 0.3)$.

(d) No normalization, box radius 1, weights from normal distribution.

Figure 12: Reconstructions of the central view $(2, 2)$ from the 3×3 dice scene with different parameters.

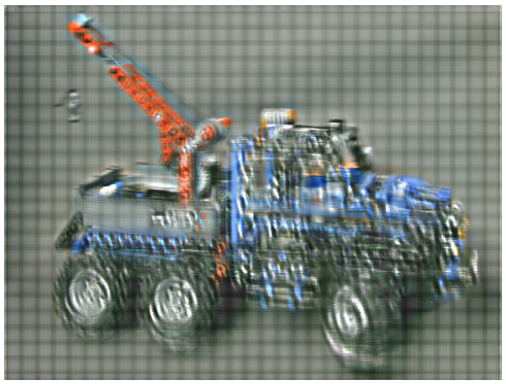
- – Weights only for intersections on sensor
- – Artefacts/Aliasing
- – Sensitive to small parameter changes
- – Applying box-filter is computationally expensive
- \+ Adjustable sampling density of rays
- \+ No need to adjust/shear the camera images

3. Rays from bottom-most layer, weights for intersection on layers + sensor (figure 11a)
   - – Normalization of weights along ray cancels out sensor weights
   - – Require rectified images for a common focal plane
   - – Applying box-filter is computationally expensive
   - \+ Adjustable sampling density of rays
   - \+ Fewer camera/scene parameters required, less sensitive (if the light field consists of rectified images)

8.2 Interpolation on the sensor plane

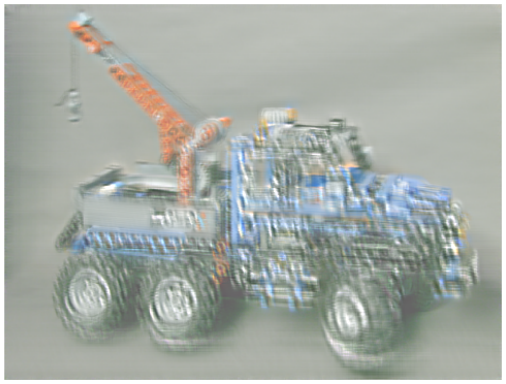
Instead of rounding the ray intersection points on the sensor to the nearest pixel center and assigning weights to the intersections, we apply linear interpolation to resample the light field. Here are the observations we can make when using the new version of the pipeline:

(a) Reconstruction of view $(9, 9)$, weights from normal distribution, RMSE: 33.816189.



(b) Layer 5 (topmost)

(c) Reconstruction of view $(9, 9)$, $w \equiv 1$, RMSE: 32.503661.



(d) Layer 5 (topmost)

Figure 13: Comparison of the reconstruction with weights on layers on and off for the legotruck-scene. 9×9 views, 5 Layers, box-radius 0.

- Increasing the box radius does not seem to improve the quality of the reconstruction (RMSE)
- Row normalization is not needed anymore
- Applying weights on the layers according to a gaussian or a tent function introduces line artefacts similar to the ones seen in figure 12 or figure 9
- Using constant weights $w \equiv 1$ on the layers instead makes the artefacts disappear
- Increasing the resolution of the layers does not have an impact on the reconstruction quality (RMSE)

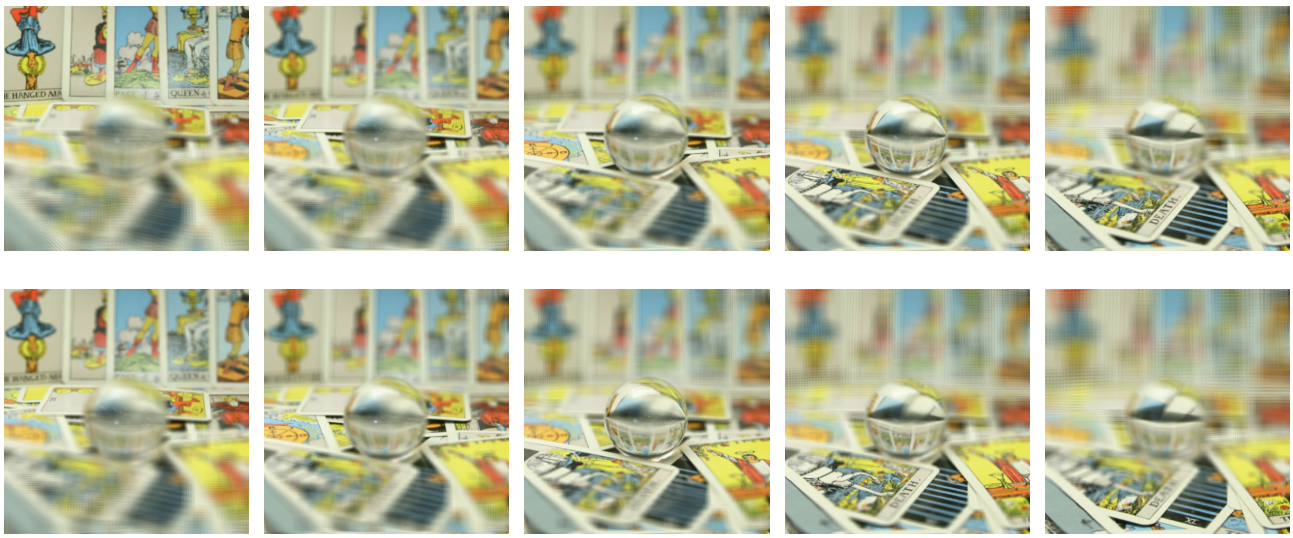


Figure 14: Comparison of two back-projections (top and bottom row) onto five layers (left to right). The first example has the sensor plane at $z_{sensor} = -0.5$, thus zero disparity is mapped onto the middle layer(s). The example below has the sensor plane at $z_{sensor} = -3$. This means that the zero-disparities are mapped onto a layer closer to the observer. The first example is better because it covers the entire depth-range of the scene (the cards in the back are in focus on the first layer and the card in the front is in focus on the 5th layer, not so in the example below).

8.3 Introduction of the sampling plane

From now on, the ray density is no longer tied to the resolution of the attenuator or the sensor plane. The introduction of a new and independent plane allows the sampling to be as dense as needed. This new *sampling-plane* has the attributes:

- $z_{sampling}$: It can be placed at any depth. By default, it is set to coincide with the first layer.
- Size: The size of the plane is variable. The default size is the layer size.
- Resolution: The resolution together with the size of the sampling plane define the ray-density.

8.4 Back-Projection and positioning of the layers

Each layer of the attenuator corresponds to a specific disparity value in the scene. The goal is to optimally place the layers such that all disparities from the light field can be represented by the attenuator. The sensor plane acts as the plane with zero disparity and thus, if the scene has a symmetric disparity range ($D_{min} = -D_{max}$), then the sensor plane should be placed at the center of the layer stack. In general, scenes that do not have symmetric disparity range and the sensor plane needs to be shifted inside the layer stack.

Back-Projection can be used to make sure that the layers are placed correctly. Once the propagation matrix P has been computed, the back-projection onto the layers can be performed by

$$b = P^T l. \tag{2}$$

The vector b holds the result of the back-projection. Equation 2 corresponds to the sum over all rays passing through a layer pixel. Figure 14 shows two back-projections of a scene with a symmetric disparity range.

8.5 Tile-based optimization for attenuation layers

High resolution light fields can take up a significant amount of space in memory. For example, a light field taken with a Full HD camera from 17×17 angles would take up $1920 \cdot 1080 \cdot 17^2 \cdot 3 \cdot 8 / \left(1024^3\right) =$

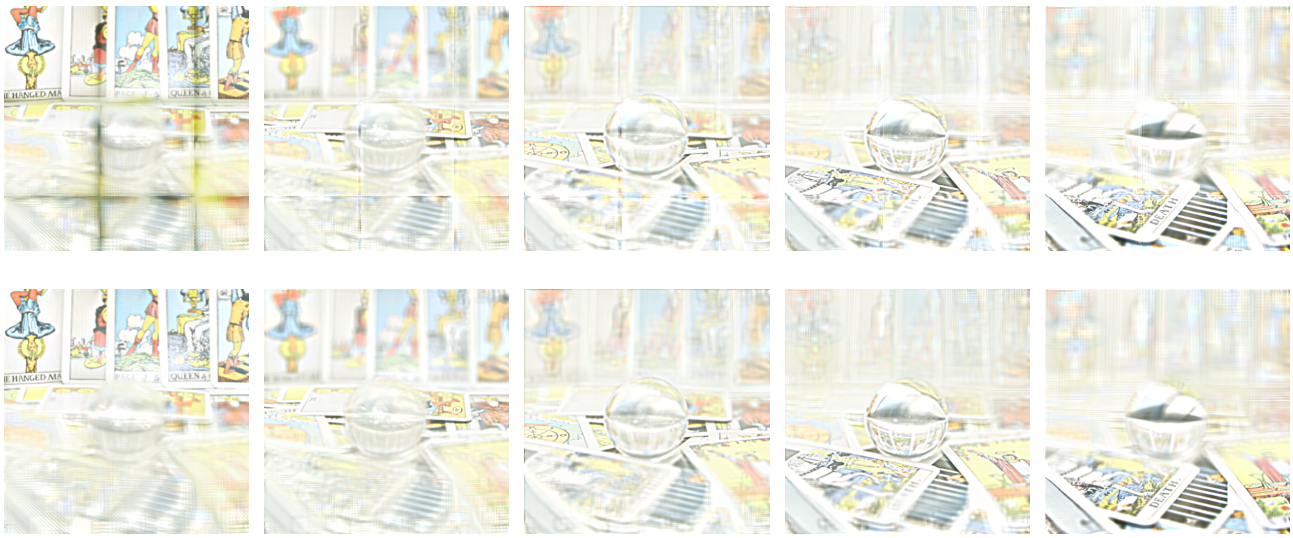
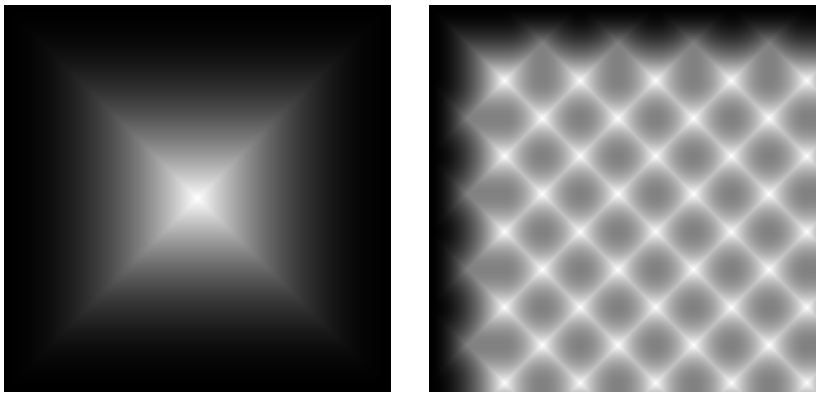


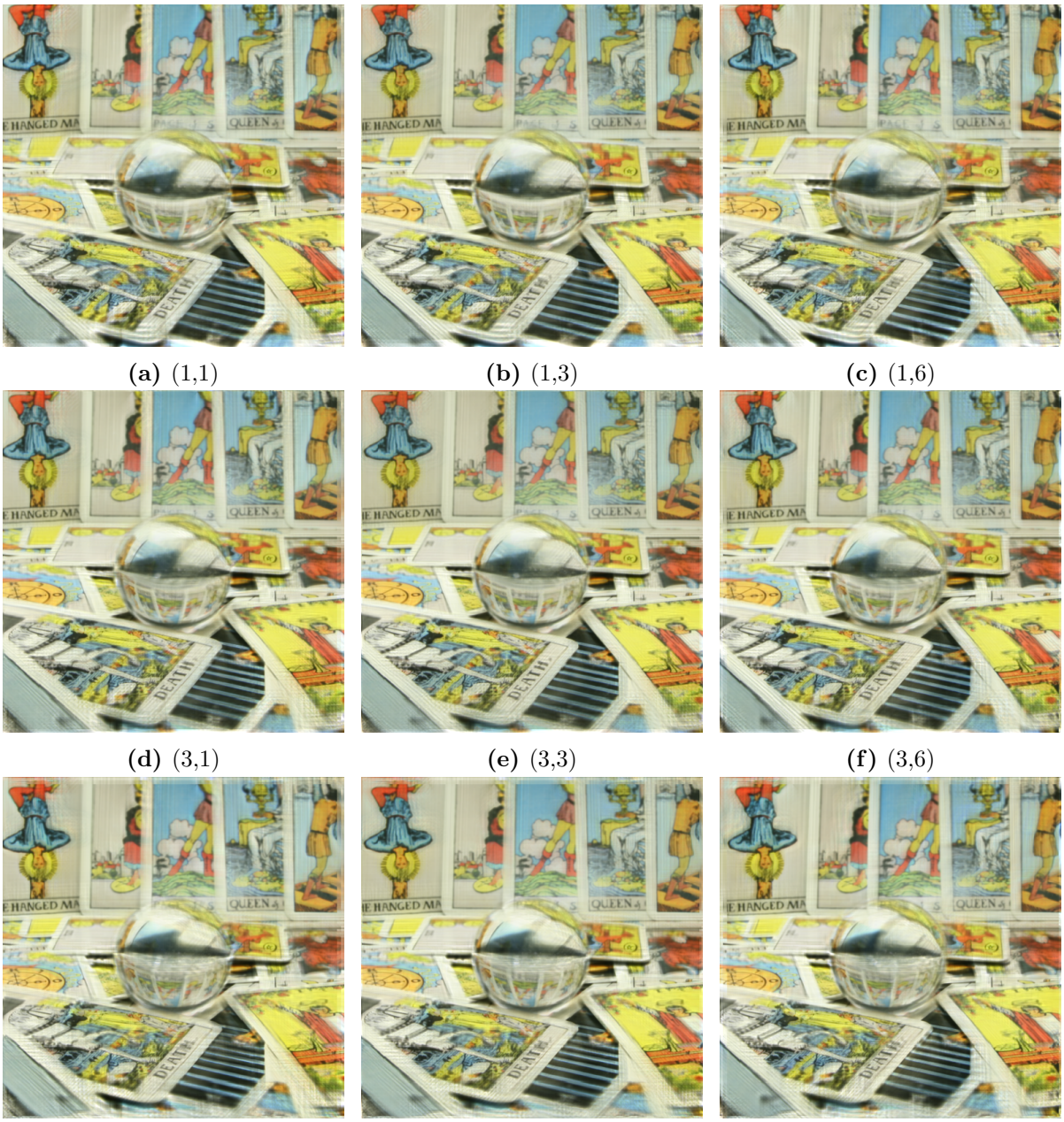
Figure 15: Top row: Tiles without overlap. Bottom row: Tiles with 50% overlap. The layers have a resolution of 512×512 pixels and the tiles have 200×200 pixels. The two examples use 3×3 tiles and 5×5 tiles respectively.



(a) **(b)**

Figure 16: A quadratic mask (a) is used to blend the overlaps of the tiles. The sum of the masks is shown in (b).

13.3947 Gigabyte of memory. In addition, the propagation matrix stores information about every pixel in the light field and thus, can take up gigabytes of space depending on the resolution of the attenuation layers. The new approach divides the attenuation layers into tiles. The combination of the same tile from each layer forms a new attenuator of smaller size and lower resolution. The optimization is then performed for every tile separately, resulting in a smaller propagation matrix per tile. In the end, the optimized tiles are put together to form the attenuation layers. In general, the borders of the attenuator contain less ray-propagation information and thus, introduce a higher degree of freedom for the optimization. This introduces artefacts that are clearly visible in the reassembled layers as shown in figure 15. To solve this issue, the tiles have to overlap. In this case, when reassembling the layers from the tiles, the overlaps need to be blended with a mask: After the optimization, each tile gets multiplied with a mask shown in figure 16a. The finished layers are then obtained by summing the tiles and dividing by the sum of the blending masks shown in figure 16b.

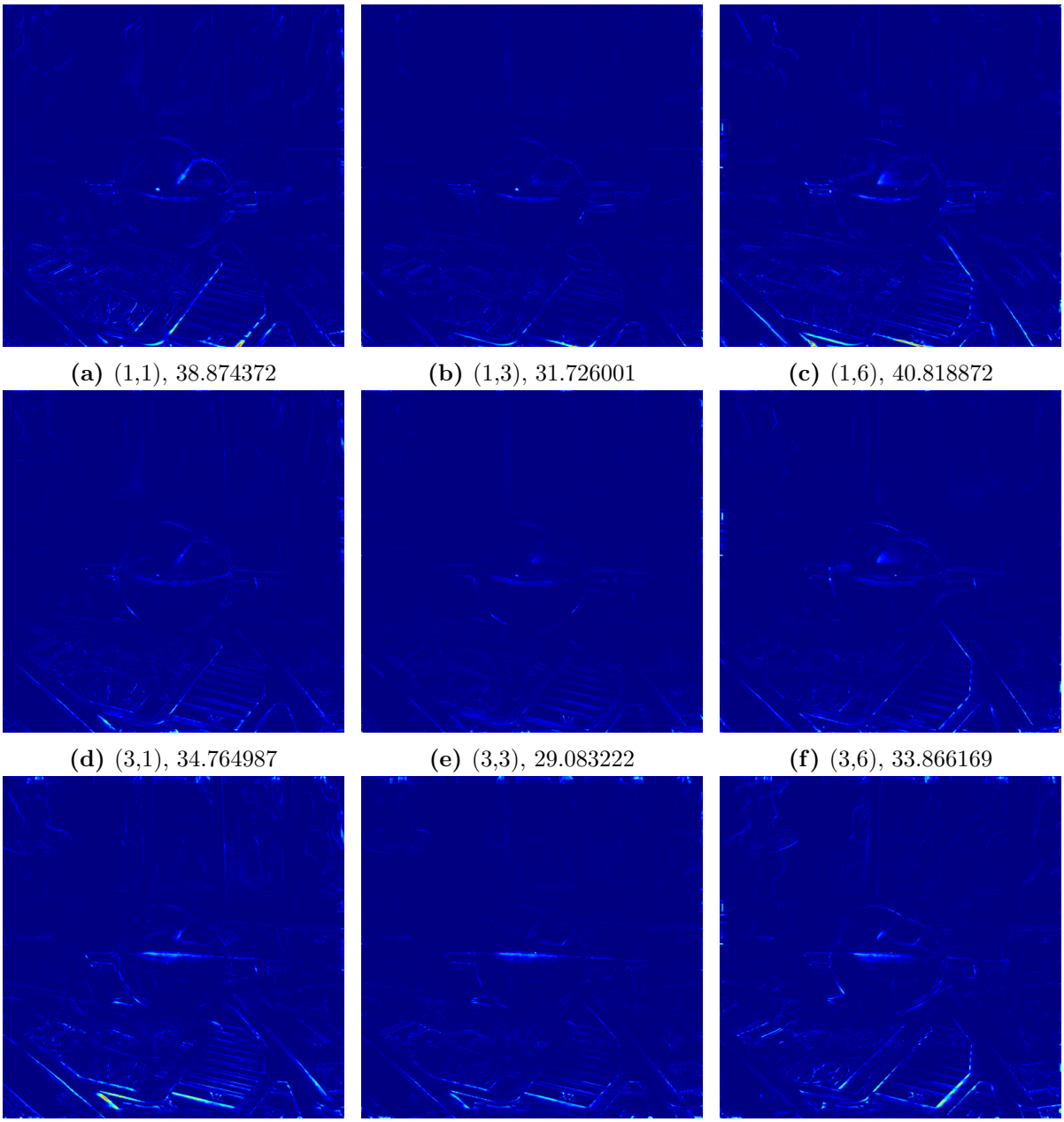


(a) (1,1) **(b)** (1,3) **(c)** (1,6)

(d) (3,1) **(e)** (3,3) **(f)** (3,6)

(g) (6,1) **(h)** (6,3) **(i)** (6,6)

Figure 17: Reconstructed views from the attenuation layers created with tiles shown in figure 15.

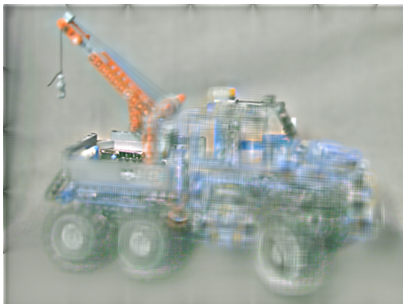


(a) (1,1), 38.874372 **(b)** (1,3), 31.726001 **(c)** (1,6), 40.818872

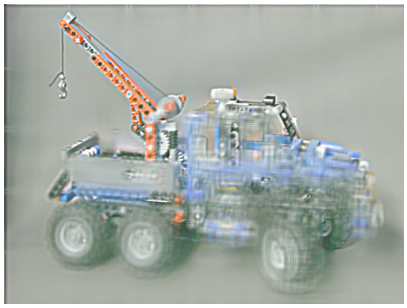
(d) (3,1), 34.764987 **(e)** (3,3), 29.083222 **(f)** (3,6), 33.866169

(g) (6,1), 46.360238 **(h)** (6,3), 37.760786 **(i)** (6,6), 42.153688

Figure 18: Mean square error (MSE) images for the reconstructed views in figure 17. The root mean square error (RMSE) is written besides the angular coordinate.



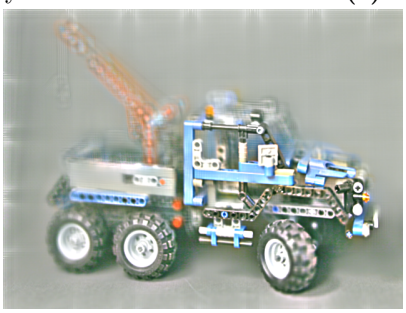
(a) Layer 1



(b) Layer 2

(c) Layer 3

(d) Layer 4



(e) Layer 1

(f) (1,1)

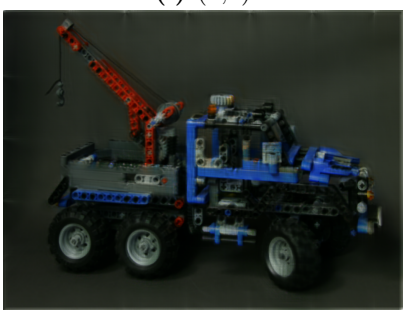
(g) (1,3)

(h) (1,6)

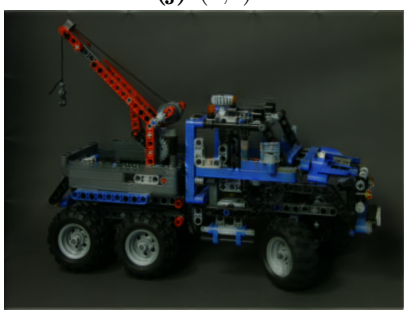
(i) (3,1)

(j) (3,3)

(k) (3,6)



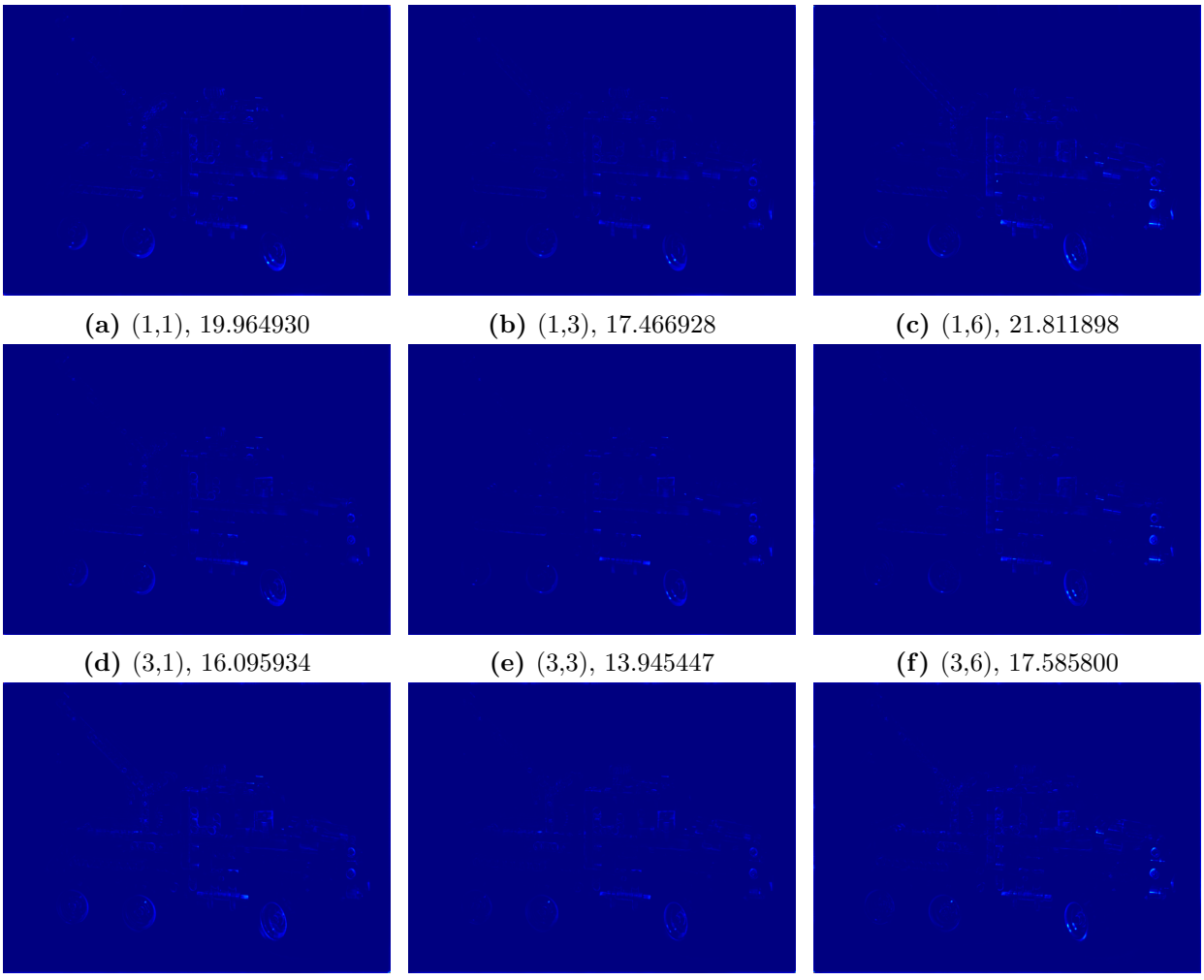
(l) (6,1)



(m) (6,3)

(n) (6,6)

Figure 19: The legotruck scene, produced by 5 attenuation layers (a) to (e) using 4×6 tiles of size 200×200 with an overlap of 50%. Below are the reconstructions of the views (f) to (n) from the 6×6 camera grid (every other left out).



(a) (1,1), 19.964930 **(b)** (1,3), 17.466928 **(c)** (1,6), 21.811898

(d) (3,1), 16.095934 **(e)** (3,3), 13.945447 **(f)** (3,6), 17.585800

(g) (6,1), 20.407914 **(h)** (6,3), 17.096632 **(i)** (6,6), 20.560457

Figure 20: Mean square error (MSE) images for the reconstructed views in figure 19. The root mean square error (RMSE) is written besides the angular coordinate.

References

[LH96] M. Levoy and P. Hanrahan. Light field rendering. pages 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.