Appendix for:
Towards Multimodal Depth Estimation from Light Fields

Titus Leistner, Radek Mackowiak, Lynton Ardizzone, Ullrich Köthe, Carsten Rother
Visual Learning Lab, Heidelberg University

Method	Parameters
EPI-Net	4612166
UPR	4613300
ESE	4613300
DPP	4778872

Table A.1. Number of **trainable parameters** for different models

A. Implementation Details

The architecture of all models in this paper is based on EPI-Net [19]. We input four light field view stacks: horizontal, vertical and two diagonals. Each stack is processed by a separate input stream network. The horizontal and vertical stacks behave similar when one is rotated by 90°. Therefore we effectively share the weights between those two input streams by applying this rotation to the vertical input and revert it before concatenation. Analogously, we also share weights between the two diagonal input streams. Subsequently, we concatenate the inferred features, and feed them to an output stream. All models and streams share the same basic building block which consists of two convolutions with a kernel size of 2×2. We use an alternating padding of one and zero and a stride of one to maintain the image dimensions. In addition, we apply a Rectified Linear Unit (ReLU) non-linearity after the first convolution and a Batch Normalization (BN) as well as a ReLU layer after the second convolution. Table A.1 shows the total number of trainable parameters for each model. A small difference between the four methods is caused by the variable number of output channels. In the following sections, we describe details, specific to one of the architectures.

All four methods, share the same backbone network. The only differences are the variable number of output channels and one additional output ReLU-layer for DPP. Table A.2 shows the detailed architecture for one input stream. This subnetwork infers features from one light field stack containing nine images with three color channels, thus a total number of $9 \times 3 = 27$ input channels. Each input stream consists of three basic blocks. Because the architecture is

Layer	Output Size
LF Stack	$B \times 27 \times H \times W$
2×2 Conv ReLU	$B \times 70 \times H \times W$
2×2 Conv BatchNorm ReLU	$B \times 70 \times H \times W$
Repeat Block $(2\times)$	

Table A.2. **Input stream** of EPI-Net, UPR, ESE and DPP

Layer	Output Size
Concatenate	$B \times 280 \times H \times W$
2×2 Conv ReLU	$B \times 280 \times H \times W$
2×2 Conv BatchNorm ReLU	$B \times 280 \times H \times W$
Repeat Block $(6\times)$	
2×2 Conv ReLU	$B \times C_{\text{out}} \times H \times W$
2×2 Conv (ReLU)	$B \times C_{\text{out}} \times H \times W$

Table A.3. **Output stream** of EPI-Net, UPR, ESE and DPP

based on [19], we chose the same number of 70 output channels. The features of all input channels are concatenated to a total number of $4 * 70 = 280$ feature channels and fed to the output stream which is illustrated in Tab. A.3.

The feed-forward output stream consists of a total number of eight blocks. Both convolutional layers for each block, except the last, output 280 channels. The last block outputs C_{out} channels, depending on the specific model. In case of our baseline, $C_{\text{out}} = 1$, because it directly predicts the disparity for each pixel. For Laplacian distribution prediction, we added a second output channel to also predict b, thus $C_{\text{out}} = 2$ for UPR and ESE. The number of dis-

crete disparity "classes", predicted by DPP, can be chosen arbitrarily. Specifically, we chose $C_{\text{out}} = 108$, thus 108 "classes", motivated by the common BadPix007 metric.

A.1. Sub-Pixel EPI-Shift

Our ESE model utilizes the EPI-Shift transformation, introduced by [13]. This shear transformation allows us to apply a disparity offset Δy to any light field x. We index the 4D light field in horizontal views U, vertical views V, image width W and image height H as x_{uvst} ($u = 1 \dots U$, $v = 1 \dots V$, $s = 1 \dots W$, $t = 1 \dots H$). In contrast to the original method which only applies integer pixel shifts, we also need sub-pixel shifts to ensure the detection of modes that are closer than one pixel. To achieve this, we apply a linear interpolation. Thus the original formulation for a horizontal EPI

$$\mathit{shift}(x_{uvst}, \Delta y) = x_{uv(s-\Delta y \cdot u)t} \tag{A.1}$$

can be generalized to continuous Δy using linear interpolation

$$\mathit{shift}(x_{uvst}, \Delta y) = \alpha x_{uv(\lfloor s-\Delta y \cdot u \rfloor)t} + (1-\alpha) x_{uv(\lceil s-\Delta y \cdot u \rceil)t} \tag{A.2}$$

with an interpolation factor $\alpha = \text{frac}(\Delta y \cdot u)$. This can be adapted trivially to vertical EPIs. For diagonal EPIs, the horizontal and vertical shift is applied successively.

B. Bayesian Interpretation of Opacity

From a Bayesian perspective, the probability $p(y_{ij})$ of each possible ground truth disparity value for a pixel quantifies the "degree of belief" in this value. For a synthetic dataset, in absence of a real ground truth measurement device whose characteristics we can analyze, any definition for $p(y_{ij})$ is valid as long as it leads to stable training and a model that reproduces the different modes with their corresponding probabilities faithfully at test time (as we verify in Sec. 4).

However, there are still some choices which are more sensible or well founded than others. In terms of the opacity η_j, it should be evident to chose

$$\eta_j = 0 \implies p(y_{ij}) = 0 \tag{B.1}$$

$$\eta_j = 1 \implies p(y_{ij}) = 1, \tag{B.2}$$

meaning that if an object is not visible at all in a pixel, its disparity should not be considered, and vice versa, if an object is the only one visible in a pixel, its disparity should be the only valid answer. In between these two points, we argue for the simplest choice of $p(y_{ij}) = \eta_j$. We note that if a setup requires a different definition of $p(y_{ij})$ (e.g. re-weight to increase the dominant mode, up-weight the foreground mode, etc.), the posterior can easily be re-weighted at test time, without retraining the model. This is only possible with methods such as ours that produce a full posterior.

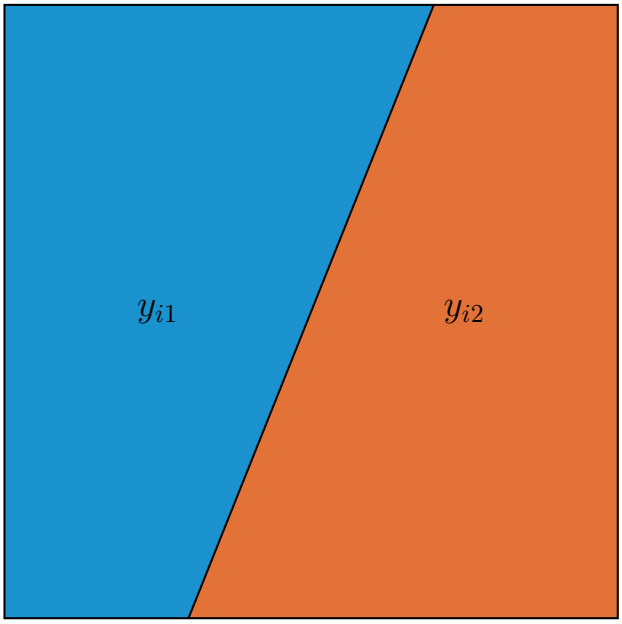


Figure B.1. View of a single idealized square pixel (synthetic case) containing an edge. The opacity values $\eta_{1,2}$ of the rendered pixel correspond to the fraction of the area that the two objects take up within it, and therefore to the probability with which the disparity would be measured at a random point in the pixel.

Despite various valid choices of defining $p(y_{ij})$, we do argue that our definition makes practical sense: the opacity corresponds to the fraction of the area that an object takes up within in a pixel before integration or rendering. It is therefore equal to the probability that the depth of that object would be observed when measuring at a random subpixel position. In other words, if we were to take many physical depth measurements within a pixel, the relative occurrence of each measured depth value y_{ij} (therefore arguably the probability $p(y_{ij})$), would be the same as the opacity η_j. This is illustrated further in Fig. B.1. While this applies exactly to our synthetically rendered dataset, some additional effects such as point spread functions and non-uniform pixel integration functions would apply for real recorded light fields. These effects might make the derivation more complex, but do not change the general idea.

C. Dataset Generation

In the following, we describe the generation of our multimodal light field depth dataset: To maximize occlusions, we generate relatively deep indoor room scenes with a high

number of objects. From a set of 750 3D assets, mainly furniture and accessories, we randomly choose 48 objects per scene and place them in a non-colliding way on the floor. In addition, random materials with a random opacity are chosen to increase the number of semi-transparent surfaces. To maximize the diversity, we also randomly choose one of 750 tileable textures for the walls, ceiling and floor. We then render the created scene by separating it into 128 slices of equal depth, as we observed that this leads to different objects falling into different slices almost always. We then render the color, alpha transparency and depth of each pixel for each slice. Alpha compositing follows the "over operator"

$$C_0 = \frac{C_1\alpha_1 + C_2\alpha_2(1-\alpha_1)}{\alpha_0} \tag{C.1}$$

with C_0 being the resulting color from color C_1 rendered *over* color C_2. The new alpha opacity of color C_0 is

$$\alpha_0 = \alpha_1 + \alpha_2(1-\alpha_1). \tag{C.2}$$

The contribution $p(y_j) = \eta_j$ of the color C_j at disparity y_j is therefore calculated as

$$p(y_j) = \eta_j = \alpha_j \left(1 - \alpha_{j-1}\left(1 - \alpha_{j-2}\left(1 - \ldots \alpha_0\right)\right)\right). \tag{C.3}$$

Lastly, we save all depths for each pixel that are not fully occluded by slices in front. Note that, apart from the multimodal depth ground truth, our synthetic light fields are similar to real light field recordings. The multi-layer color information that real light field cameras could not record is not used as an input to our methods.

D. Additional Experiments

In this section, we first compare our methods to "Image-Based Rendering for Scenes with Reflections" (IBR) [20] and "What Sparse Light Field Coding Reveals about Scene Structure" (SLFC) [11]. Secondly, we present visualizations of exemplary depth posteriors predicted by UPR and DPP. Thirdly, we evaluate our work on the commonly used HCI 4D Light Field Dataset [9] and show additional qualitative results.

D.1. Comparison to IBR [20] and SLFC [11]

We additionally compared our methods to two multimodal depth estimation approaches [20] [11]. These are, to the best of our knowledge, the only previous methods which are able to estimate multiple depth modes. For "Image-Based Rendering for Scenes with Reflections" [20] we implemented the normalized cross-correlation framework for our own dataset, as no source code was publicly available. The method computes the pairwise normalized cross-correlation in a small window ($3\text{px} \times 3\text{px}$) and utilizes it to form a cost volume. In a second step, up to two disparity planes are extracted from the volume using a modified semi-global matching algorithm. We interpret the per-pixel normalized cross-correlations as our disparity posterior distributions. To achieve better results, we first subtract the per-pixel minimum cross-correlation and then normalize the distribution.

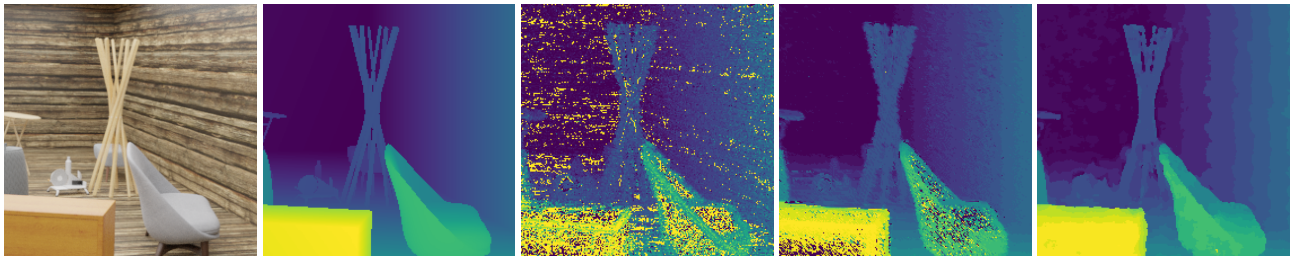
We also compared our methods to "What Sparse Light Field Coding Reveals about Scene Structure" (SLFC) [11]. The method uses a dictionary of small EPI-Patches. Each atom in this dictionary corresponds to a unique disparity. On small EPI windows around each pixel, the Lasso-Optimizer is used to infer the coefficients for each atom. A large coefficient for an atom means that the disparity which corresponds to this atom was observed at this pixel. The vector of coefficients can therefore also be interpreted as a discrete disparity posterior distribution, similarly to DPP. The authors were able to provide us with only a part of the code which we used to create the dictionaries for our multimodal validation dataset. We used the Lasso optimizer from the Python framework "scikit-learn" and set $\alpha = 0.01$ as recommended by the paper authors. Finally, we optimized the posterior distribution for each pixel.

We compared both methods to our four deep learning based models. Please note, that due to the enormous runtime of SLFC (even with our parallel implementation on 128 CPU cores), we run it on a cropped down (0.5×0.5) version of our validation dataset. For a fair comparison, we ran all methods trained on the multimodal posterior distribution with loss functions $\mathcal{L}_x^{MM}$ on the same cropped down scenes and chose a the same number of 108 disparity steps for all methods.

Table D.1 shows the results of our comparison. We notice that IBR and SLFC produce more wrong classifications in non-textured and therefore uncertain areas which leads to more overall noise. We argue that this is due to the local per-pixel optimization. In contrast, our neural networks benefit from a larger receptive field and are therefore capable to deliver smooth results, even within relatively large non-textured areas (compare Fig. D.1). This effect causes an overall worse performance of IBR and SLFC. To compute the unimodal metrics, we chose the discrete disparity with the highest posterior probability for each pixel. Both, the MSE and BadPix score confirm our observations. Note that IBR and SLFC both perform better than our baseline model in terms of multimodal posterior prediction. This clearly shows that the methods are indeed able to correctly predict multiple disparity modes. However, the predicted posterior distributions also suffer from poor performance in uncertain regions. Additionally, due to each pixel being optimized separately, the runtime of SLFC is several orders of magnitudes higher. One $256\text{px} \times 256\text{px}$ scene took approximately 18 minutes to compute in parallel on a dual CPU machine with 128 cores, while DPP runs in approximately one second on a single GPU.

Method	Unimodal Metrics		KL Divergence			AuSE ↓	Time ↓
	MSE ↓	BadPix ↓	Unimodal ↓	Multimodal ↓	Overall ↓		(in sec)
BASE (multi)	**0.435**	0.274	4.807	8.081	6.078	-	**0.557**
UPR (multi)	0.480	0.285	2.028	3.551	2.448	**0.115**	0.578
ESE (multi)	1.204	0.245	4.330	3.769	4.226	0.182	4.502
DPP (multi)	0.608	**0.239**	**1.786**	**3.193**	**2.136**	0.288	1.068
IBR [20]	1.436	0.365	3.835	3.436	3.843	0.617	11.263
SLFC [11]	3.449	0.660	3.694	3.908	3.715	0.324	1054.231

Table D.1. **Comparison to IBR [20] and SLFC [11]**, from left to right: Mean Squared Error and the common BadPix007 score (percentage of pixels with $|y_i - \hat{y}_i| > 0.07$), Kullback-Leibler divergence on unimodal, multimodal and all pixels, Area under Sparsification Error (AuSE), runtime of one forward pass. Our methods were trained using the multimodal loss $\mathcal{L}_x^{\mathrm{MM}}$. Lower is better

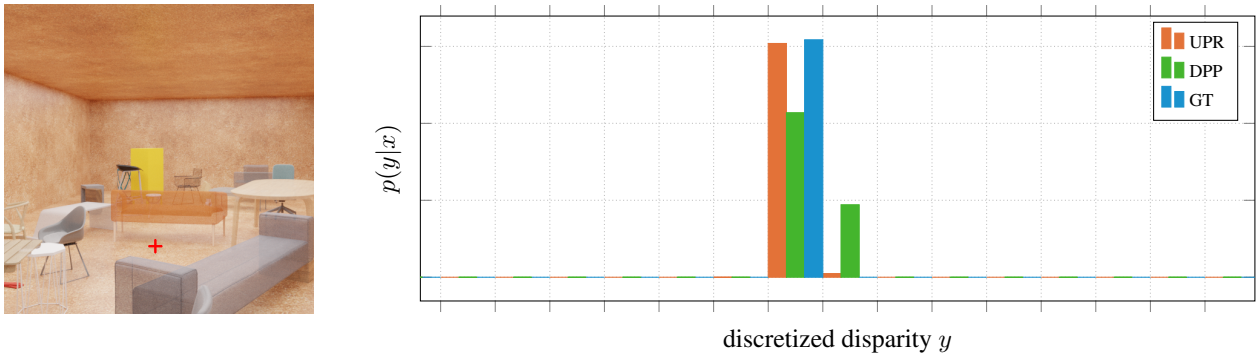


(a) Center view (b) Ground Truth (c) SLFC [11] (d) IBR [20] (e) DPP

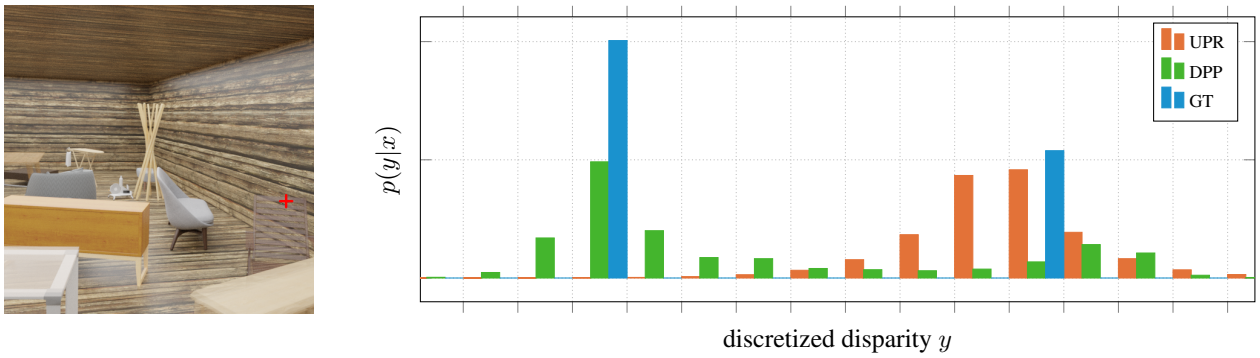
Figure D.1. **Qualitative results of IBR [20] and SLFC [11], compared to DPP** on one of our multimodal validation scenes: We chose the disparity which corresponds to the strongest coefficient for each pixel. Compared to our deep learning based methods, IBR [20] and SLFC [11] tend to wrong classifications in non-textured areas which causes noise. This also has a negative impact on both methods posterior prediction performance.

D.2. Visualization of Disparity Posterior Distributions

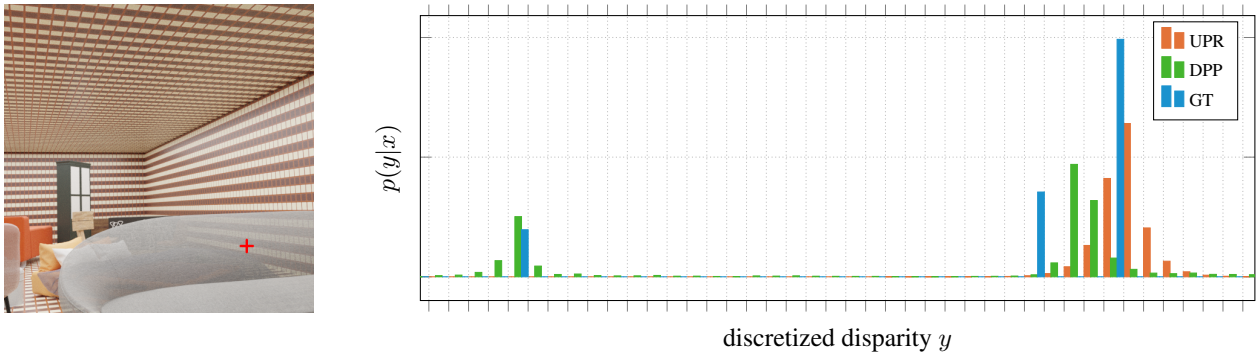
To give some examples of predicted posterior distributions, we visualized estimations of UPR and DPP on our multimodal validation dataset. Similar to our evaluations, we discretized the ground truth disparity posterior using the same number of bins. We chose certain pixels from three validation scenes that contain one, two and three disparity modes respectively. Note that DPP manages to detect both modes in Fig. D.2b, but outputs a high uncertainty due to the similar colors of the foreground and background object. In Fig. D.2c, two of the tree modes collapsed into one. UPR always picked up one present mode with a high uncertainty. Please note that our dataset randomly adds transparency to object materials. This causes some objects that would be opaque in real life to become transparent.



(a) Validation scene 4: pixel (red cross) contains a single disparity mode



(b) Validation scene 3: pixel (red cross) contains two disparity modes

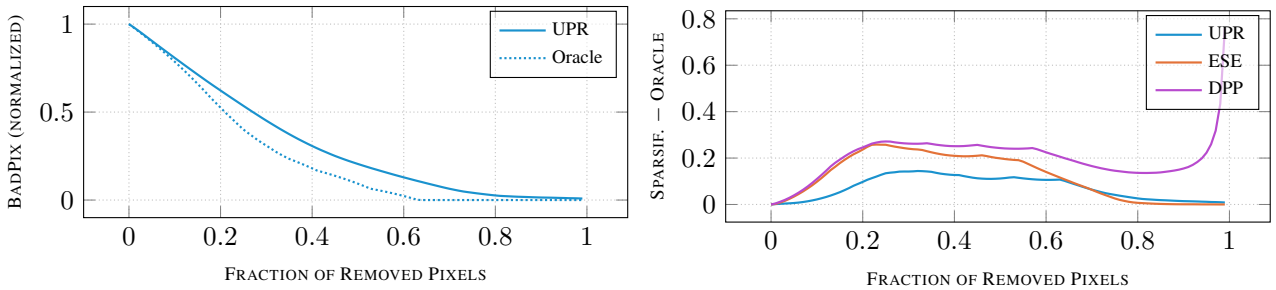


(c) Validation scene 9: pixel (red cross) contains three disparity modes

Figure D.2. **Visualization of disparity posterior distributions** for one pixel (red cross) estimated by UPR (orange) and DPP (green) and discrete ground truth posterior (blue). Note that DPP is able to estimate up to two modes reliably, while UPR only picks up a single mode.

D.3. Evaluation on HCI 4D Light Field Dataset [9]

We also evaluated our methods on the commonly used HCI 4D Light Field Dataset [9]. Like previous methods [19], we used the 16 "additional" scenes as our training dataset and the four "training" scenes for validation. As this dataset only contains a single ground truth depth, we used the unimodal loss functions $\mathcal{L}_x$. All other training parameters remain the same as mentioned in Sec. 4. Note, that we only trained on the HCI dataset for this particular experiment. The methods used in all other experiments were trained solely on our novel multimodal dataset.



(a) Sparsification curve

(b) Sparsification Error for all methods

Figure D.3. **Unimodal uncertainty quantification on HCI 4D Light Field Dataset:** Sparsification results of analyzed methods with respect to the disparity BadPix007.

Method	**Unimodal Metrics**		**AuSE ↓**	**Time ↓**
	MSE ↓	BadPix ↓		(in sec)
BASE	**0.011**	0.065	-	**0.480**
UPR	0.012	0.056	**0.060**	0.481
ESE	0.163	0.088	0.091	14.863
DPP	0.018	**0.044**	0.110	0.783

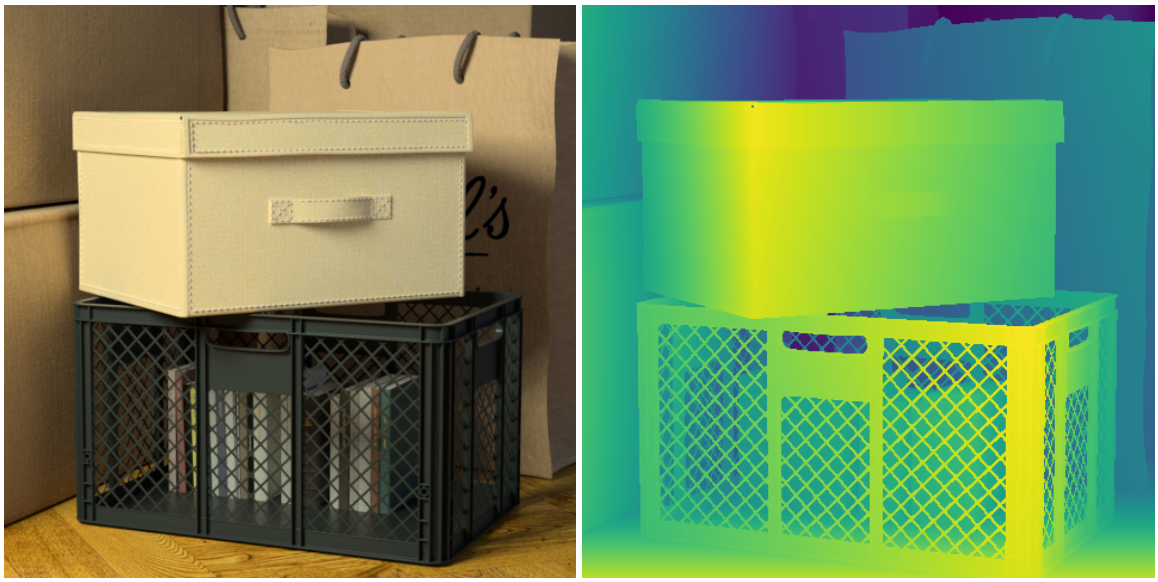
Table D.2. **Evaluation on HCI dataset [9]**, from left to right: Mean Squared Error and the common BadPix007 score (percentage of pixels with $|y_i - \hat{y}_i| > 0.07$), Area under Sparsification Error (AuSE), runtime of one forward pass. Lower is better

Figure D.3 and Table D.2 show our experimental results, which are overall very consistent with the experiments on our randomly generated multimodal dataset. DPP performs best with respect to the amount of accurately predicted pixels (BadPix) but is overconfident which is clearly visible in the sparsification error. In contrast, UPR and ESE deliver a better sparsification performance. Qualitative results are shown in Fig. D.5 to Fig. D.8.

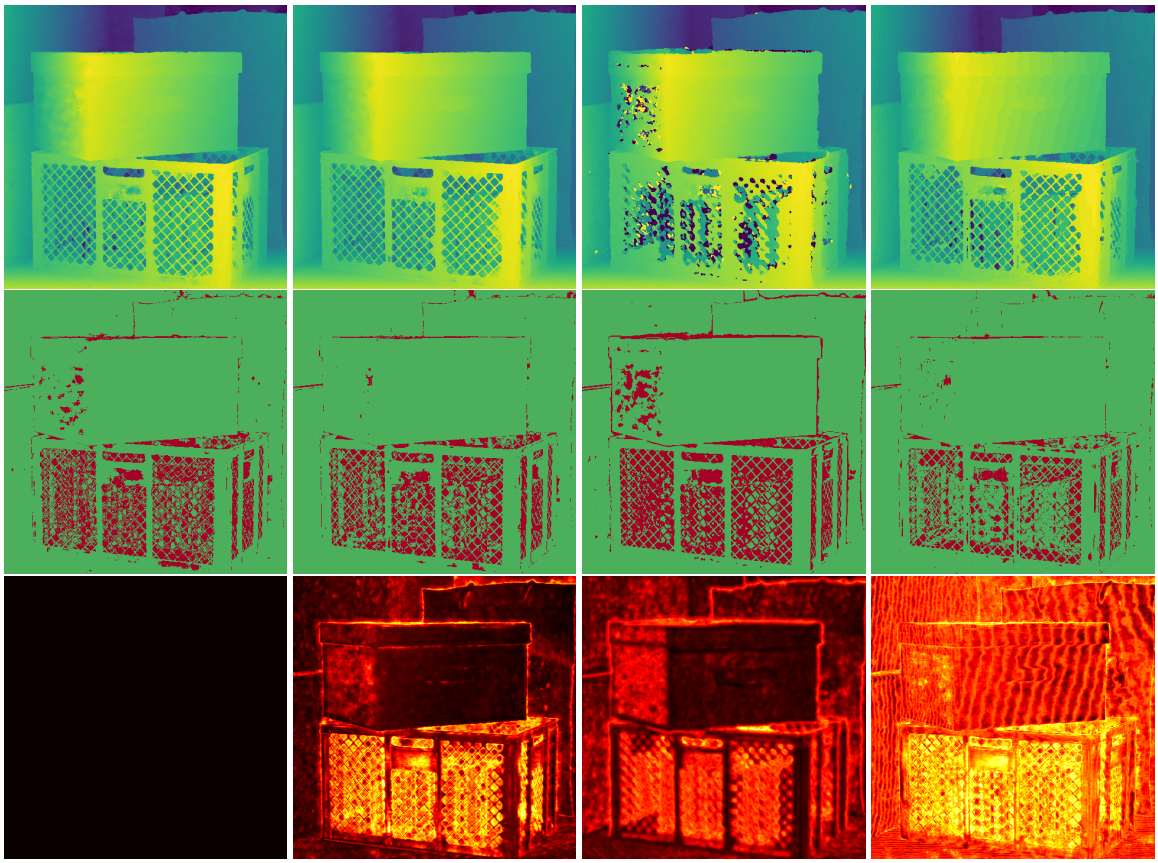
$y_{\min}$ (a) Disparity $y_{\max}$

0 (b) Uncertainty $\sigma^2_{\max}$

Figure D.4. **Color maps** used for results. Disparity and uncertainty maps are normalized to enhance visibility

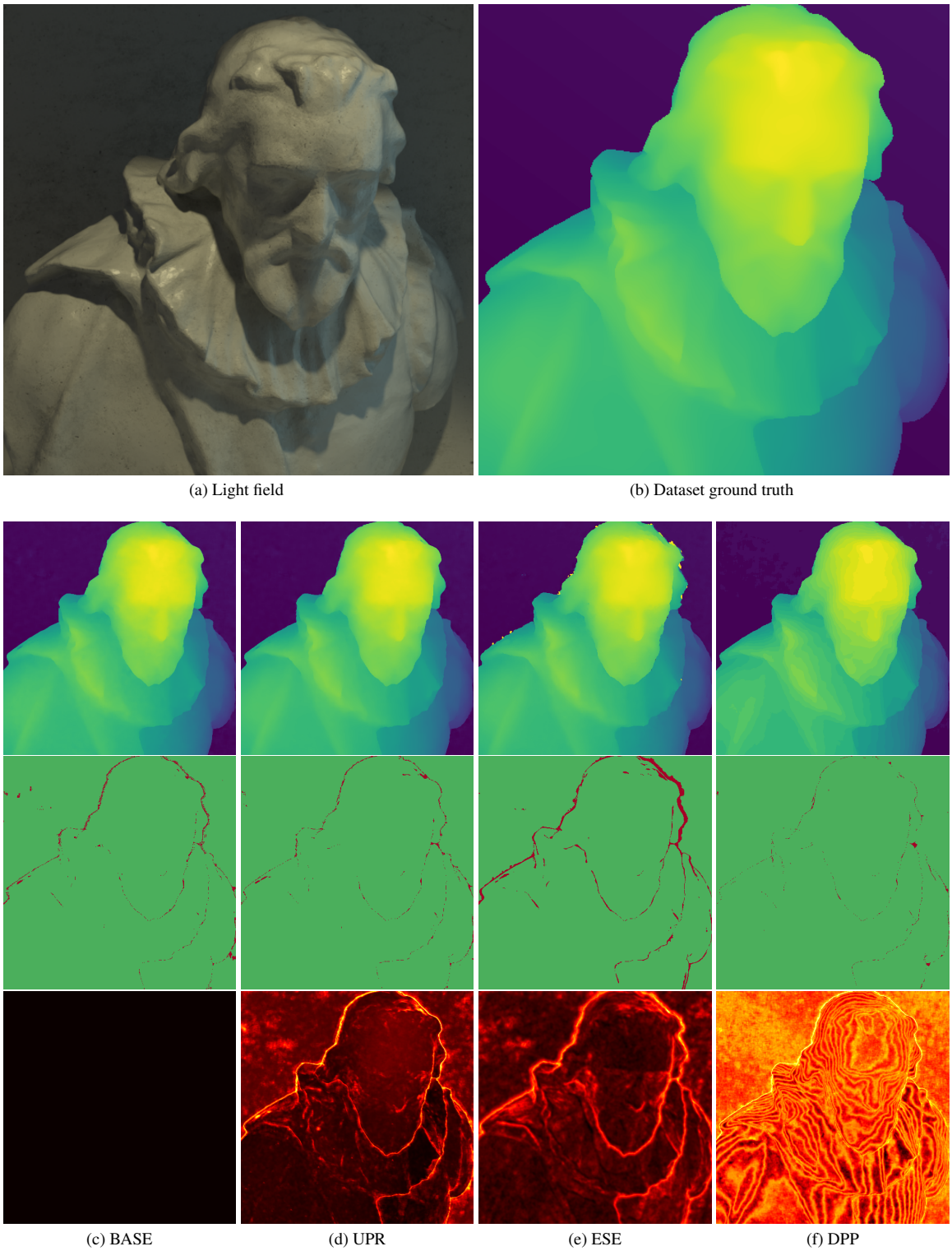


(a) Light field (b) Dataset ground truth



(c) BASE (d) UPR (e) ESE (f) DPP

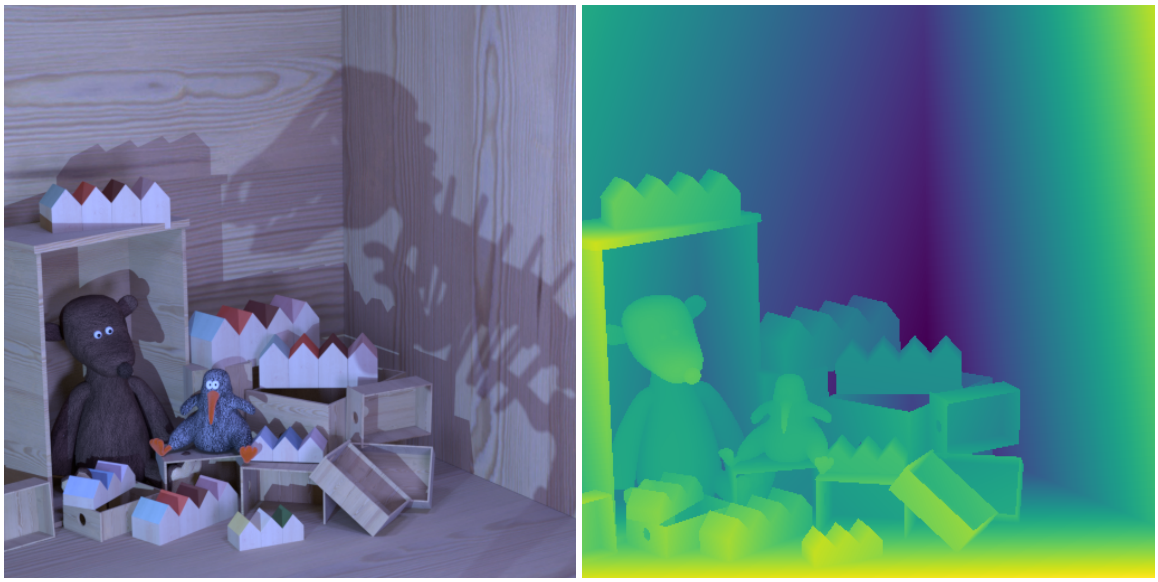
Figure D.5. Results of the four posterior prediction methods ((c) - (f)) for **'boxes'** scene. Top: output disparity (most likely mode). Center: per-pixel BadPix metric (a pixel i is red if $|y_i - \hat{y}_i| > 0.07$). Bottom: per-pixel uncertainty σ^2 (non-existent for baseline method)



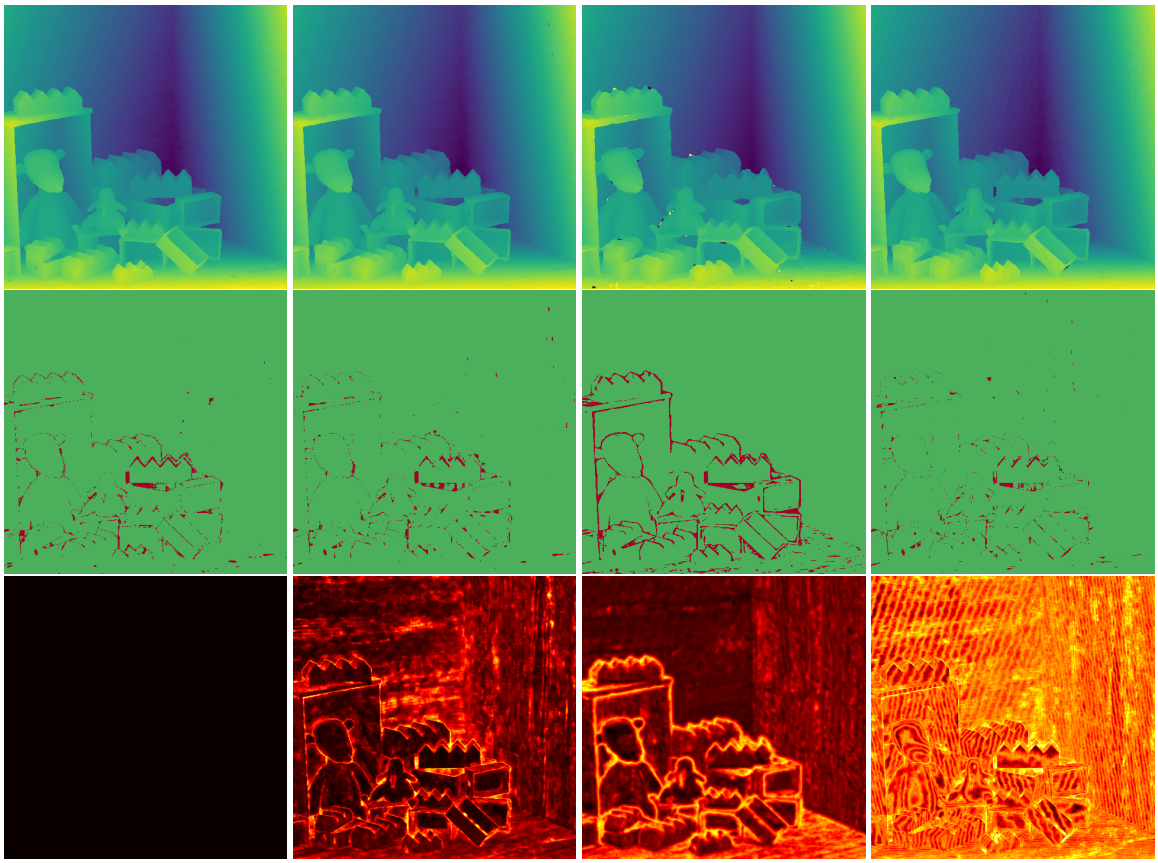
(a) Light field (b) Dataset ground truth

(c) BASE (d) UPR (e) ESE (f) DPP

Figure D.6. Results of the four posterior prediction methods ((c) - (f)) for **'cotton'** scene. Top: output disparity (most likely mode). Center: per-pixel BadPix metric (a pixel i is red if $|y_i - \hat{y}_i| > 0.07$). Bottom: per-pixel uncertainty σ^2 (non-existent for baseline method)

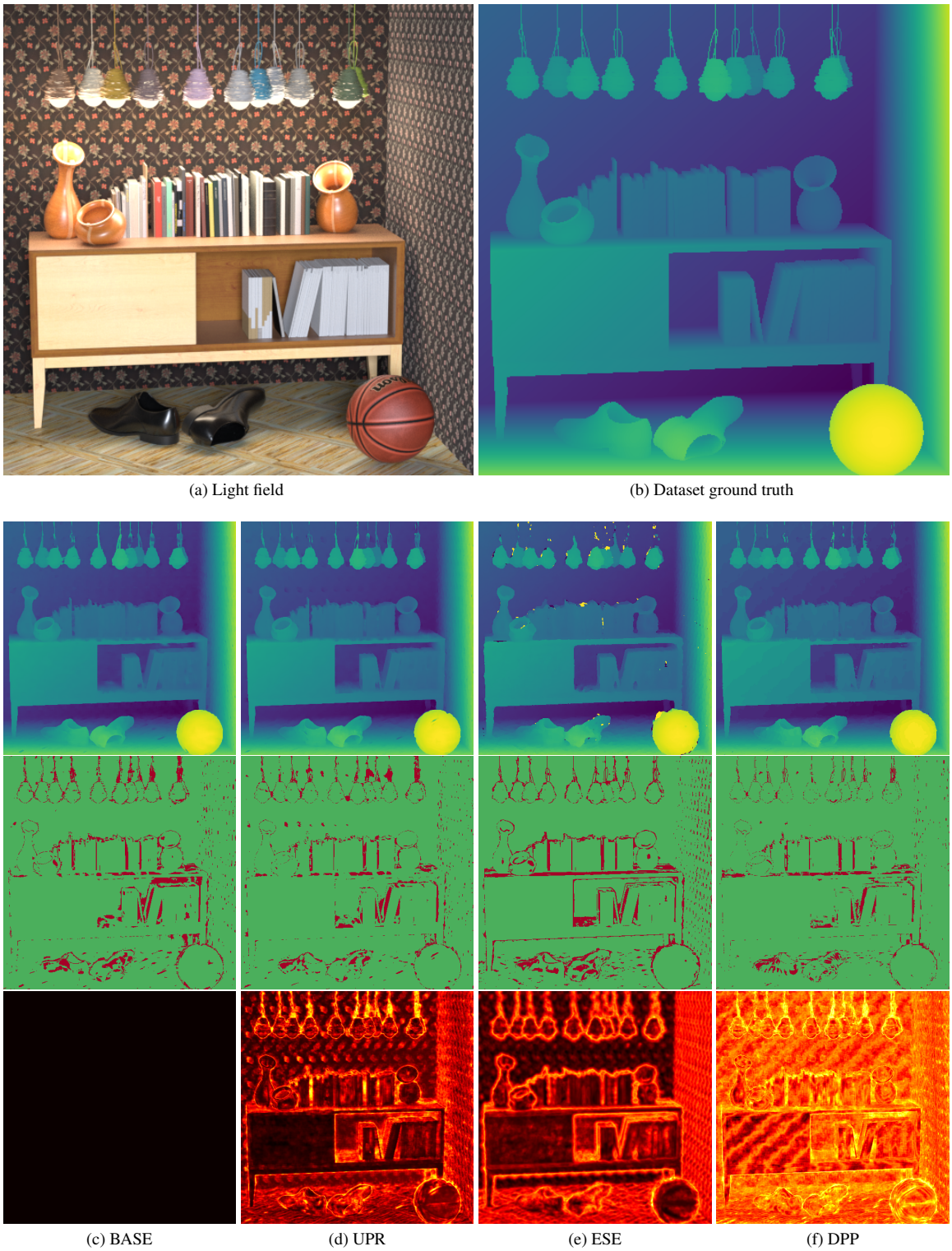


(a) Light field (b) Dataset ground truth



(c) BASE (d) UPR (e) ESE (f) DPP

Figure D.7. Results of the four posterior prediction methods ((c) - (f)) for **'dino'** scene. Top: output disparity (most likely mode). Center: per-pixel BadPix metric (a pixel i is red if $|y_i - \hat{y}_i| > 0.07$). Bottom: per-pixel uncertainty σ^2 (non-existent for baseline method)



(a) Light field (b) Dataset ground truth

(c) BASE (d) UPR (e) ESE (f) DPP

Figure D.8. Results of the four posterior prediction methods ((c) - (f)) for **'sideboard'** scene. Top: output disparity (most likely mode). Center: per-pixel BadPix metric (a pixel i is red if $|y_i - \hat{y}_i| > 0.07$). Bottom: per-pixel uncertainty σ^2 (non-existent for baseline method)