Pacific Graphics 2019
C. Theobalt, J. Lee, and G. Wetzstein
(Guest Editors)

Volume 38 (2019), Number 7

Light Field Video Compression and Real Time Rendering

Saghi Hajisharif[1], Ehsan Miandji[1], Per Larsson[1], Kiet Tran[2] and Jonas Unger[1]

[1]Linköping University, Sweden [2]Karolinska Institutet, Sweden

Abstract

Light field imaging is rapidly becoming an established method for generating flexible image based description of scene appearances. Compared to classical 2D imaging techniques, the angular information included in light fields enables effects such as post-capture refocusing and the exploration of the scene from different vantage points. In this paper, we describe a novel GPU pipeline for compression and real-time rendering of light field videos with full parallax. To achieve this, we employ a dictionary learning approach and train an ensemble of dictionaries capable of efficiently representing light field video data using highly sparse coefficient sets. A novel, key element in our representation is that we simultaneously compress both image data (pixel colors) and the auxiliary information (depth, disparity, or optical flow) required for view interpolation. During playback, the coefficients are streamed to the GPU where the light field and the auxiliary information are reconstructed using the dictionary ensemble and view interpolation is performed. In order to realize the pipeline we present several technical contributions including a denoising scheme enhancing the sparsity in the dataset which enables higher compression ratios, and a novel pruning strategy which reduces the size of the dictionary ensemble and leads to significant reductions in computational complexity during the encoding of a light field. Our approach is independent of the light field parameterization and can be used with data from any light field video capture system. To demonstrate the usefulness of our pipeline, we utilize various publicly available light field video datasets and discuss the medical application of documenting heart surgery.

CCS Concepts

• ***Computer graphics*** $\rightarrow$ *Image-based rendering; Computational photography; Image compression;*

1. Introduction

Light field imaging enables a range of post-capture effects such as free-viewpoint rendering, refocusing and image processing. Unlike traditional 2D imaging systems, light fields capture both multiple angles and location making such data ideal for many computer vision applications. It also enables capture for novel 3D display setups [WLHR12] and head-mounted displays (HMDs) such as HTC VIVE where positional tracking is available. Light field rendering is suitable for virtual and augmented reality applications ranging from the entertainment industry to educational platforms. However, a key and still unsolved challenge is handling the very large memory footprint inherent to light field imaging. The amount of data that is required to be captured, processed, and rendered for high resolution and high quality light fields make current solutions difficult to use in practice.

Since the introduction of light fields to the computer graphics community by [LH96] and [GGSC96], a large number of methods have been proposed, ranging from light field acquisition and compression to processing and view synthesis, for an overview, see [WMJ*17]. Light fields can be captured in different ways, including micro-lens array (MLA) setups [NLB*05, AW92], coded masks [VRA*07, WIH13], objective lens arrays, and gantry and array-based camera systems [BSW01, KKSM19]. A few designs have been suggested that can support light field video capture. Wilburn et al. [BSW01] proposed a system where light field videos were captured using an array of cameras. Using similar principles, Ng et al. [NLB*05] presented camera systems where a microlens array was placed in the ray-path in the optics in front of the sensor. This approach was later used in commercial light field video cameras such as Lytro cinema [Lyt17] and Raytrix [Ray19].

In this paper, we present an end-to-end GPU pipeline from compression to the rendering of light field videos. The proposed pipeline is developed to be agnostic to which input device is used to capture the light field video, so that a wide range of different setups can be supported. The key to our solution can be found in a highly efficient dictionary learning for sparse representation. We extend upon the current state-of-the-art in sparse representation of high dimensional visual data, Aggregate Multidimensional Dictionary Ensembles [MHU19], and present a full pipeline for efficient compression, storage, and real-time playback of video light fields. A key aspect in our system is that we encode both color information and depth in the same dictionary representation, leading to efficient reconstruction algorithms supporting parallel computation and GPU implementation. We introduce the concept of dictionary pruning leading to a significantly more efficient encoding algorithm as compared to [MHU19]. Finally, we perform a study investigating how image noise in the light fields affect the dictionary training and encoding and propose a denoising scheme to increase the

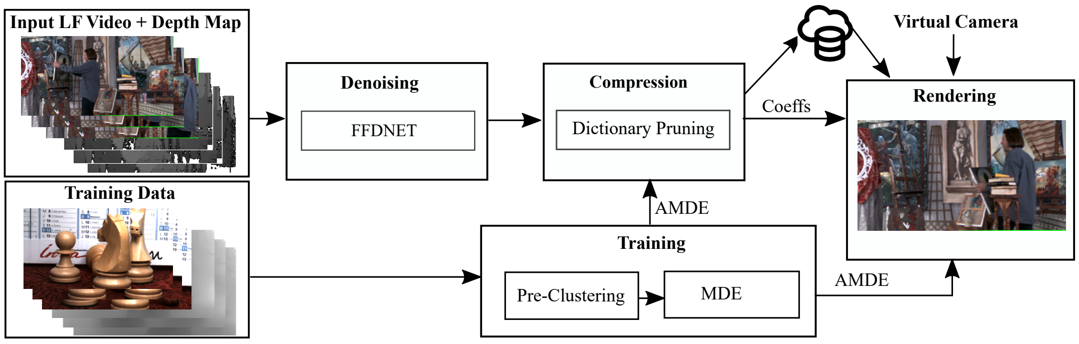


Figure 1: *An overview of the end to end pipeline for Light Field Video (LFV) processing and rendering. The input to the pipeline is a LFV and its estimated depth map/disparity. The noise in LFV is removed from the signal using a denoising algorithm (FFD-NET [ZZZ18]) to improve the compression ratio. In the training process, a LFV dataset is used to train an Aggregate Multidimensional Dictionary Ensembles (AMDE) used for encoding the input LFV and its corresponding depth map. The compressed coefficients together with the AMDE are passed on to the GPU for real-time reconstruction rendering of the LFV from the vantage viewpoint.*

compression ratio and reconstruction quality during rendering. The main contributions can be summarized as:

- Sparse representation of light field videos with auxiliary information such as depth.
- A dictionary pruning algorithm leading to significant speedups of the light field encoding.
- A study on the effects of image noise on training, compression ratio, and reconstruction quality leading to a denoising step to enhance the effectiveness of the pipeline.
- A real-time combined reconstruction/decoding and geometry-guided view interpolation algorithm.

Although our pipeline supports any light field configuration as long as regularly sampled data points can be extracted, the main focus in this paper is on multi-sensor and lenslet capturing systems. Our main objective is to enable the full pipeline for real-time rendering of high-quality compressed light field videos with full GPU support, i.e. the decoding is carried out as part of the rendering algorithm. The pipeline itself, along with the accompanying contributions, relate to a wide body of different technical aspects and scientific areas. The next section, therefore, gives an overview of the design choices made in the development and relates each stage of the pipeline to the relevant previous work.

2. Light field pipeline design and overview

A viable processing chain for high resolution light field imaging needs to enable efficient compression, high quality signal reconstruction, and real-time reconstruction and playback. Since most view interpolation algorithms relies on geometric information such as depth or optical flow, it is also necessary that the compression pipeline can handle such auxiliary information efficiently. In this paper we adhere to a general definition of the light field signal and assume that it consists of: a set of images and the corresponding auxiliary information required for view interpolation, e.g. depth, disparity or optical flow. The images can be captured from different locations using a set of cameras or a single camera equipped with e.g. a lenslet array, or optical setups designed for compressive light field sampling.

The light field video pipeline presented in this paper consists of five main blocks: light field data point extraction, dictionary training, denoising, encoding, lossless coding of the coefficients, and decoding and real-time playback. This is illustrated in Figure 1. The different blocks in the system were developed based on meeting the following design goals:

1. *Quality:* The light field signal should be represented in a basis, or dictionary, which enables high PSNR vs. compression ratio.
2. *Generality:* The basis should be general in a sense that it can represent light fields independently of the underlying image statistics, and support coding of auxiliary information such as depth, disparity or optical flow.
3. *Efficiency:* The basis should support random access, point sampling, and parallel computations.

Point sampling and reconstruction of individual pixels without the computational overhead of recovering the entire data point is crucial for light field rendering since each the reconstructed and the possibly interpolated view is generated from slices of individual pixels from a large set of light field data points.

Light field data points - Throughout the paper, we use light fields consisting of a set of images and auxiliary information in the form of depth maps. The light field video is sliced into 6D data points with $s \times t \times u \times v \times c \times f$ elements each, where s,t is the number of pixels in the image plane u,v the pixels in the angular domain (each from a neighboring camera view), f the number of frames from the video sequence, and c the number of color channels plus the depth channel. The data point size should, in general, be kept as large as possible given the computational resources used. For the experiments presented in this paper we typically keep the number of data point elements around 20K and use e.g. $6 \times 6 \times 7 \times 7 \times 4 \times 5$ or $10 \times 10 \times 4 \times 4 \times 4 \times 4$ elements. Moreover, since each data point includes all the angular information, as well as a subset of consecutive frames, the compression artifacts in the angular and temporal domains are typically negligible, unless the number of coefficients is too low.

Compression and dictionary training - Early works on light field compression with two plane parameterization rely on image and video coding methods such as JPEG, JPEG2000 [TM13], MPEG-2, MPEG-4 that were based on analytical basis functions such as Discrete Cosine Transform (DCT) [CDF92], wavelets, shearlets [ELL08] and the Fourier basis. These methods used information such as disparity [MG00, GCRX03, JSA03] or geometry estimation [XAG03] to propagate information between neighboring views; see [WMJ*17] for an overview. Recently, Overbeck et al. [OEE*18] introduced a compression scheme for still spherical light fields with random access to the memory based on Motion Compensated Prediction (MPC) of VP9 codec [MBG*13]. This method also uses an analytical dictionary and does not exploit coherence in the temporal domain.

It has been shown that data driven dictionary learning algorithms outperform analytical basis functions in terms of compression ratio and reconstruction quality [AEB06, MBPS09, MKU13, SPRE18, APF17]. Miandji et al. [MHU19] presented a dictionary learning method for compression and compressed sensing of light field videos. The training procedure leads to an Aggregate Multidimensional Dictionary Ensemble (AMDE). An AMDE consists of a collection of orthonormal dictionaries that enable sparse representations along different dimensions of a set of *data points* extracted from a light field video dataset. The method can handle various parameterizations including unstructured light fields. AMDE relies on a reduced union of subspaces signal model [EM09, MKU15] such that each data point is represented in one dictionary admitting a high degree of sparsity and low reconstruction error. This means that the sparse representation using AMDE (even without entropy coding) leads to a significant reduction in size, typically in the order of $10:1$ to $100:1$ depending on the input data and the required level of quality. The benefits of the AMDE compared to other data driven signal models [AEB06, RD12, Rak13] are that it performs a 2-layered clustering in the coefficient domain which enhances sparsity with minimal error and that it enables random access to individual pixels in the light field video. The resulting dictionary can in most cases be trained once and then used for a range of different input light fields given that they exhibit similar image statistics (e.g. natural images). Finally, the size of AMDE is orders of magnitude smaller than overcomplete dictionaries such as K-SVD and its variants [MHU19]. Based on the above arguments, we use AMDE as the underlying representation for light field video data points.

AMDE extensions - Although AMDE is the current state-of-the-art in sparse representation of high dimensional visual signals, the memory and bandwidth requirements, and the sheer number of pixels that need to be processed at each step in the pipeline, lead to long processing times and sub-optimal dictionary configurations. To meet the requirements described above, we extend AMDE in a number of important directions to improve the computational efficiency, image quality, and compression ratio:

- We extend the AMDEs from only supporting the representation of the color channels to include also the auxiliary information (depth) as a dimension in each data point
- We present a novel dictionary pruning strategy that analyses the composition of the dictionaries within the AMDE and removes redundant dictionaries, leading to at least 4 times reduction in computational complexity for the encoding of light field videos with negligible effect on quality.
- We investigate the effect of image noise on training and encoding of light field videos using AMDE, and discuss the effect of noise on the reconstruction quality and compression ratio. We enhance the proposed pipeline with a denoising stage prior to encoding to significantly increase the PSNR and compression ratio.

In addition to the above algorithmic extensions and improvements, we have also implemented the full encoding - decoding pipeline with GPU support. For the encoding, we have adapted the GPU-based tensor products for light fields presented in [BMU19] to light field videos, leading up to $10\times$ speedup of the encoding using a GTX 1080 Ti GPU as compared to our multi-threaded CPU version running on 40 cores at a frequency of 2.4GHz.

Denoising - High frequency variations in the input data are likely to reduce the efficiency of the sparse representation and lead to a higher number of coefficients and a decrease in PSNR. We, therefore, conduct a set of experiments to investigate the impact of image noise on the compression algorithm in the proposed pipeline and show how only subtle denoising of the input light field data significantly improves the reconstruction quality and the compression ratio. Any denoising algorithm capable of preserving important features and details in the image can be used. Benchmark methods such as BM3D [DFKE07] and [GZZF14] are effective but not fast enough for our purpose. In recent years convolutional neural networks (CNN) have shown to be able to handle image denoising very well compared to the classical approaches due to their large modeling capacity. One state of the art learning based method, DnCNN [ZZC*17] has achieved competitive denoising result, however, it is limited in the flexibility of handling spatially variant noise and the noise level is constraint to a preset. In this work, a general, efficient and effective noise reduction is a key to improving the training algorithm. We have used a CNN-based algorithm, FFDNet [ZZZ18], which is fast and flexible and suitable for handling large datasets such as light fields. Furthermore, it has the ability to handle a large range of different noise characteristics.

Rendering and view interpolation - Generation of novel, previously unseen views from a sampled light field requires either highly dense sampling or auxiliary information about the geometry of the scene [CTCS00], e.g. depth, disparity, or optical flow [SCK07,CSHD11]. Recent methods based on deep neural networks can estimate the novel view without direct access to the depth map or disparity. However, these methods are still limited to small baseline between the cameras, and in some cases, they can only handle a disparity range of up to two pixels [KWR16]. To demonstrate this issue, an example of a state-of-the-art CNN-based method is shown in Section 6 for light field video with large baseline. Moreover, these algorithms are usually not suitable for real-time applications due to computational complexity. The processing times range from a few seconds, [YHC*18], to minutes, [WZW*17], and more, [FNPS16], per frame for high resolution light fields.

In this work we assume that the light field video is accompanied by some form of auxiliary geometric information and camera calibrations that can be used for efficient real time reconstruction, together with bilinear view interpolation. Since *depth* is most commonly used for this purpose, see e.g. the documentation for the upcoming MPEG-I standard [MPE19], we take this as a representative example. It should be noted that both optical flow and disparity would work equally well within our pipeline. Since the compression method provides fast random access down to the pixel level,

Algorithm 1 The training algorithm that is performed only once

```
c = Train ( $\{\mathcal{L}^{(i)}\}_{i=1}^{N_l}, C, K, \tau_l, N_r, p, \varepsilon$ )

Input: The training set $\{\mathcal{L}^{(i)}\}_{i=1}^{N_l}$, number of clusters $C$, number
    of dictionaries per cluster $K$, training sparsity $\tau_l$, number of
    representatives $N_r$, number of PCA coefficients $p$, and an error
    threshold $\varepsilon$.
Output: A dictionary ensemble $\left\{\mathbf{U}^{(1,k)},\ldots,\mathbf{U}^{(6,k)}\right\}_{k=1}^{CK}$
1: Create data points from input training light field videos
2: Apply pre-clustering on data points
3: for $c = 1 \ldots C$ do
4:     Apply MDE on each clustered data points
5: end for
6: Compute aggregated ensemble $\Psi$ using equation (3)
```

both color intensity and depth information are decoded on GPU for interpolation and refocusing purposes.

3. Sparse representation of light field videos

Light field and light field video datasets are typically very large even at a modest angular resolution. For instance, an hour long light field video at 4K UHD resolution with an angular resolution of 32×32 and a frame rate of 30Hz consumes 5.87 petabytes of storage assuming a half-precision high-dynamic-range format. The storage requirements are expected to increase as new light field imaging modalities emerge. Therefore, efficient compression algorithms are essential for storage and real time playback of light field videos. In many cases, the compression algorithm should reduce the storage cost so that the compressed data can be fitted into the GPU memory for real time playback. Moreover, decreasing the storage cost via compression enables us to transfer the data faster through a network (or via the internet) for a multi-user experience.

As mentioned in Section 2, we use AMDE for sparse representation of light field videos. In addition to high reconstruction quality, AMDE has a very small memory footprint for the dictionary ensemble, which enables real time playback of high resolution light field videos. In this section, we describe various stages of AMDE, as well as our contributions for improving the efficiency of this algorithm. In particular, we will briefly explain the training and encoding stages of AMDE in sections 3.1 and 3.2, respectively. The reader is referred to [MHU19] for further details about the AMDE algorithm. Section 3.3 will present an algorithm for reducing the number of dictionaries in AMDE by a user defined factor with negligible reduction in reconstruction quality, which leads to faster encoding using the GPU.

3.1. AMDE Training

The computation of AMDE requires a training procedure, as a first step, where the training set, consisting of one or more light field video datasets, is divided into small $6D$ data points of dimensionality $s \times t \times u \times v \times c \times f$, as described in Section 2; Algorithm 1 shows an overview of the training procedure. The data points are fed into a pre-clustering algorithm that groups data points with similar sparse representation and error together. The pre-clustering improves the sparsity of the representation while reducing the overall training time. Pre-clustering also reduces the effect of outliers in the training set. In this paper we analyze the effect of noise on the training and the encoding stages of AMDE, see Section 4. After pre-clustering, a Multidimensional Dictionary Ensemble (MDE) is trained for each pre-cluster, obtaining a set of MDEs for the entire training set, which is combined to form an Aggregate MDE (AMDE).

Let $\{\mathcal{L}^{(i)}\}_{i=1}^{N_l}$ be a set of training data points in a pre-cluster, where $\mathcal{L}^{(i)} \in \mathbb{R}^{s\times t\times u\times v\times c\times f}$ and N_l is the number of data points. The goal of MDE is to train a dictionary ensemble $\{\mathbf{U}^{(1,k)},\ldots,\mathbf{U}^{(6,k)}\}_{k=1}^{K}$, where K is the number of dictionaries in a pre-cluster, such that each data point is represented as

$$\mathcal{L}^{(i)} = \mathcal{S}^{(i)} \times_1 \mathbf{U}^{(1,k)} \cdots \times_6 \mathbf{U}^{(6,k)} = \mathcal{S}^{(i)} \bigtimes_{j=1}^{6} \mathbf{U}^{(j,k)}, \tag{1}$$

where $\mathcal{S}^{(i)}$ is sparse and $\mathbf{U}^{(j,k)}$ are orthonormal matrices containing basis functions for dimension j. This is achieved by solving

$$\min_{\mathbf{U}^{(j,k)},\mathcal{S}^{(i,k)},\mathbf{M}_{i,k}} \sum_{i=1}^{N_l}\sum_{k=1}^{K} \mathbf{M}_{i,k} \left\| \mathcal{L}^{(i)} - \mathcal{S}^{(i,k)} \bigtimes_{j=1}^{6} \mathbf{U}^{(j,k)} \right\|_F^2 \tag{2a}$$

subject to

$$\left(\mathbf{U}^{(j,k)}\right)^T \mathbf{U}^{(j,k)} = \mathbf{I},\ \forall k = 1,\ldots,K,\ \forall j = 1,\ldots,6, \tag{2b}$$

$$\left\|\mathcal{S}^{(i,k)}\right\|_0 \leq \tau_l, \tag{2c}$$

$$\sum_{k=1}^{K} \mathbf{M}_{i,k} = 1,\ \forall i = 1,\ldots,N_l, \tag{2d}$$

where τ_l is a user-defined sparsity parameter and the matrix $\mathbf{M} \in \mathbb{R}^{N_l \times K}$ defines the membership of each data point in the ensemble. Hence, equation (2d) ensures that each data point is only sparsely represented in one dictionary.

Once all the MDEs are trained, they are combined to form the AMDE as follows

$$\Psi = \bigcup_{c=1}^{C} \left\{\mathbf{U}^{(1,k,c)},\ldots,\mathbf{U}^{(n,k,c)}\right\}_{k=1}^{K} = \left\{\mathbf{U}^{(1,k)},\ldots,\mathbf{U}^{(n,k)}\right\}_{k=1}^{CK}, \tag{3}$$

where C is the number of pre-clusters.

3.2. AMDE Testing

Let $\{\mathcal{T}^{(i)}\}_{i=1}^{N_t}$ denote a test set containing the data points obtained from a light field video that we would like to compress, where N_t is the number of data point. The procedure of obtaining sparse coefficients of a data point in the test set using a dictionary is often called *testing*. To perform testing using AMDE, we first project the data point onto all the dictionaries of AMDE as follows:

$$\mathcal{S}^{(i,k)} = \mathcal{T}^{(i)} \bigtimes_{j=1}^{6} \mathbf{U}^{(j,k)},\ \forall k \in \{1,\ldots,CK\}. \tag{4}$$

Afterwards, we introduce sparsity by nullifying elements of each $\mathcal{S}^{(i,k)}$ until a user-defined threshold, δ_t, on the representation error is reached. We also set a user-defined upper-bound on sparsity, τ_t, so that the data points in the test set that are not intrinsically sparse get a fixed number of nonzero coefficients, see [MHU19] for more details. Once the sparse coefficients of a data point for all dictionaries in AMDE are computed, we pick the dictionary corresponding to the most sparse coefficients with the least amount of error. The index of the best dictionary, called the membership index, as well as its corresponding coefficients are stored to disk. Indeed the above procedure is repeated for all the data points in the test set.

3.3. AMDE Pruning

In this section, we propose an algorithm for removing a fraction of dictionaries in AMDE to improve the testing speed. Miandji et al. [MHU19] observe that increasing the total number of dictionaries in AMDE, i.e. CK, improves the reconstruction quality, and hence the compression ratio. However, as we described in Section 3.2, the computational complexity of the testing stage is proportional to CK. Hence the number of dictionaries in AMDE provides a tradeoff between the effectiveness of AMDE and the computational complexity of testing. Since the training is done only once, we propose to perform the training with a large number of dictionaries. To improve the testing performance, this section describes a pruning algorithm to remove the dictionaries in AMDE that have a small significance in the final image quality of the light field video.

In order to reduce the number of dictionaries in AMDE, we require a metric to select a subset of dictionaries such that the reconstruction quality or the compression performance is minimally affected. Given the metric, we also need an algorithm to select the dictionary subset. Intuitively, we would like this subset of dictionaries to capture the essential information in the test set in the form of sparse coefficients. Hence the metric should identify the most *diverse* dictionaries in the ensemble. One such metric is Mutual Coherence (MC) [BHEE10]. The mutual coherence of a matrix $\mathbf{X} \in \mathbb{R}^{n \times n}$ is defined as

$$\mu(\mathbf{X}) = \max_{1 \le u \ne v \le n} \left| \mathbf{X}_u^T \mathbf{X}_v \right|. \tag{5}$$

where $\mathbf{X}_u$ is the uth column of $\mathbf{X}$. Therefore, the mutual coherence of a matrix defines the maximum cross-correlation between its columns. Indeed, for an orthonormal matrix, used in e.g. AMDE, mutual coherence of each matrix is zero. Mutual coherence can also be defined for a pair of matrices:

$$\mu(\mathbf{X}, \mathbf{Y}) = \max_{1 \le \forall u, v \le n} \left| \mathbf{X}_u^T \mathbf{Y}_v \right|. \tag{6}$$

In this paper, we propose to use Average Mutual Coherence (AMC) [AFMS10] defined as

$$\mu_{\text{avg}}(\mathbf{X}, \mathbf{Y}) = n^{-2} \sum_{u,v=1}^{n} \left| \mathbf{X}^T \mathbf{Y} \right|_{u,v}. \tag{7}$$

It should be noted that AMC is defined for two matrices, while each dictionary in AMDE, $\{\mathbf{U}^{(1,k)}, \dots, \mathbf{U}^{(6,k)}\}_{k=1}^{K}$, contains 6 matrices in one dictionary trained over a light field video dataset. In order to define a metric for pruning the dictionaries in AMDE, one possibility is to form the Kronecker dictionaries, i.e. $\{\mathbf{U}^{(6,k)} \otimes \mathbf{U}^{(5,k)} \otimes \dots \otimes \mathbf{U}^{(1,k)}\}_{k=1}^{K}$. In this way, each dictionary is defined as one matrix, hence we can use equation (7) to compute the pair-wise distances between the dictionaries. However, asides from the fact that the Kronecker dictionaries impose high storage and computational burden, computing the distances over Kronecker dictionaries does not exploit the distances among various dimensions of two dictionaries individually, but rather over all dimensions. Therefore we propose to compute the pair-wise distances for each dimension individually as follows

$$\mu_{\text{avg}}\left(\mathbf{U}^{(j,k)}, \mathbf{U}^{(j,k)}\right) = n_j^{-2} \sum_{u,v=1}^{n_j} \left| \left(\mathbf{U}^{(j,k)}\right)^T \mathbf{U}^{(j,k)} \right|_{u,v}, \tag{8}$$

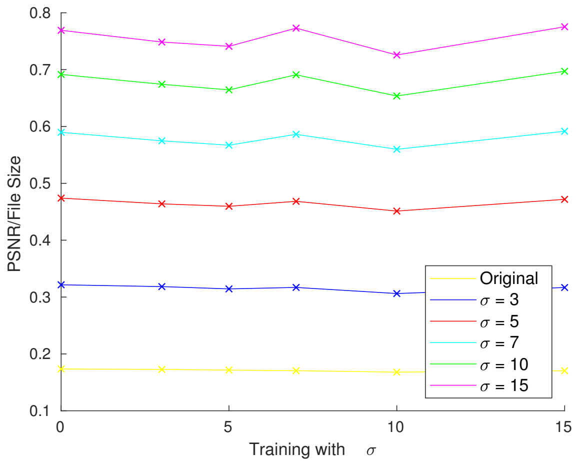
for the jth dimension and $\forall k \in \{1, \dots, K\}$. Given the metric defined in equation (8), we use K-Means++ [AV07] to obtain a subset of dictionaries in a trained AMDE. The K-Means++ algorithm was introduced to improve the performance of the K-Means clustering algorithm [Llo82] by initializing it with the most diverse set of data points rather than random initialization as used by the "vanilla" K-Means algorithm. This is exactly what we would like to achieve for dictionary pruning, however, here we would like to find the most diverse set of dictionaries in AMDE.

The first step in K-Means++ is to randomly select a dictionary in AMDE to start calculating the pairwise distances. We found that such a random initialization leads fluctuations in the reconstruction quality based on the choice of the first dictionary. Instead, we use the membership matrix $\mathbf{M}$, defined in Section 3.1, to perform this task. Recall that the membership matrix of size $N_l \times K$ defines which dictionary in AMDE should be used for a data point. Hence we propose to initialize K-Means++ with the most frequently used dictionary in AMDE. This can be simply implemented by computing the column-wise sum of $\mathbf{M}$, leading to a K-length vector, and finding the index of the largest element, which corresponds to the most frequently used dictionary. Note that we apply this procedure using the training set, while the pruned ensemble is used for testing. Here, we rely on the assumption that the training set is representative of the testing set. Indeed, one can use a validation set to find the most frequently used dictionary, which in turn can improve the pruning results. However, we found the simple approach described above to be sufficient in many cases, see Table 1.

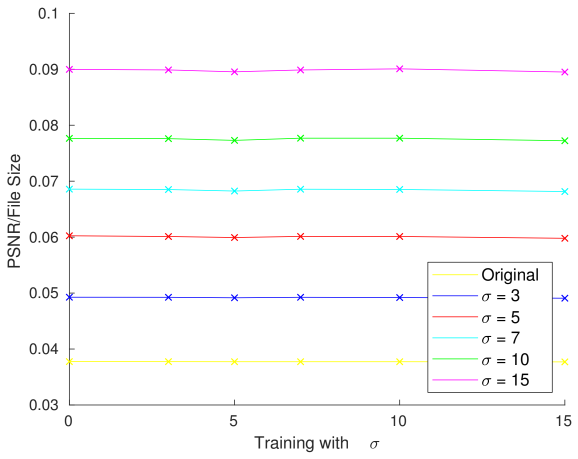
To evaluate the performance of the pruning algorithm, we used two datasets, namely *Chess* and *Heart*, and train an AMDE with 64 dictionaries for each dataset. We prune 75% of the dictionaries in each AMDE, i.e. a reduction by a factor of 4. Because we are analyzing the performance of the training stage, with and without pruning, we use 20% of a dataset for training, while the testing was performed on the entire dataset. For the results presented in the remainder of the paper, the training and testing sets are distinct. Table 1 presents the results for AMDE pruning. As it can be seen, pruning has minimal effect on reconstruction quality and compression ratio. However, we observe a speed up by a factor of 4 in testing.

Chess	Time	PSNR	Size
AMDE Pruning (ours)	140.23s	43.03dB	41MB
AMDE Without Pruning	556.79s	43.01dB	40MB
Heart	Time	PSNR	Size
AMDE Pruning (ours)	96.93s	42.93dB	205MB
AMDE Without Pruning	388.06s	42.96dB	182MB

Table 1: *Ensemble pruning results by a factor of 4 for two datasets: Chess and Heart. The size is calculated as the storage cost of nonzero coefficient values and their corresponding location.*



(a) *Chess dataset*



(b) *Painter dataset*

Figure 2: *The effect of image noise on training. The plots show that the compression efficiency stays relatively flat for all denoising levels on the training set, and that the denoising of the testing data leads to a higher degree of sparsity and better PSNR. For denoising we used FFDNet [ZZZ18] with the parameter* $\sigma = 3, 5, 7, 10, 15$.

This is indeed expected since the testing time is proportional to the number of dictionaries in AMDE.

The results may seem counter-intuitive to the reader since testing over a small subset of dictionaries is performing roughly the same (in terms of quality) as an exhaustive search over all the dictionaries. Indeed, pruning should reduce the reconstruction quality. However, note that according to equation (8), we are allowing the pruning algorithm to select a dictionary element for a certain dimension from other dictionaries in AMDE. This procedure leads to a higher reconstruction quality rather than merely selecting a dictionary regardless of the similarity of a dictionary element in the aforementioned dictionary with the rest of the ensemble.

4. Denoising

Since we are seeking to project data points from the light field onto the AMDE dictionaries, in which the data points are sparse, it is intuitive that the high frequency variations introduced by image noise are going to deteriorate our representation and lead to a higher number of coefficients, i.e. a less sparse representation. Smoothing the input signal by removing image noise should thus lead to a more sparse representation. From a theoretical standpoint, the effect of noise on the efficiency of sparse representations has been thoroughly studied [MEUA17, BHEE10], emphasizing the importance of noise with regards to the reconstruction quality. In this paper, we study the effect of noise on both the training and testing/encoding using an AMDE for the compression of light field videos.

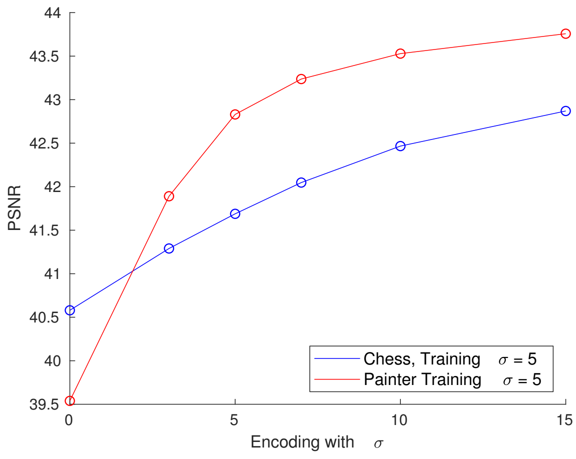


Figure 3: *Shows the effect of image noise on testing (compression) of Painter and Chess datasets. The plot shows that the PSNR increases along with* σ. *Denoising was done using FFDNet [ZZZ18].*

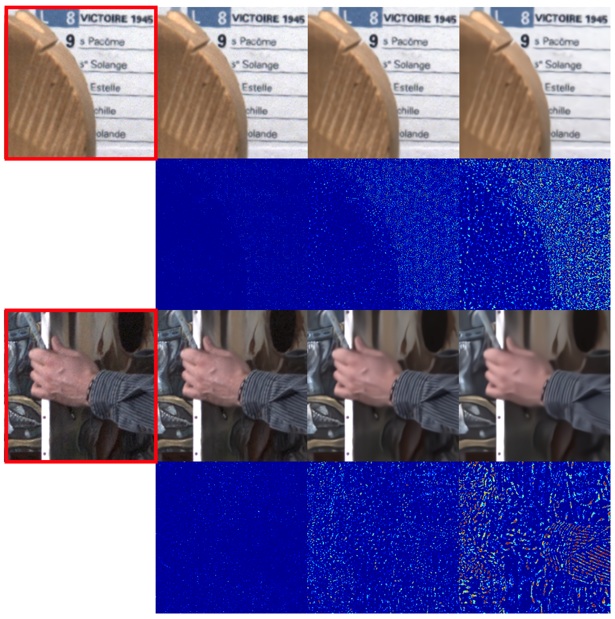


Figure 4: *Image quality comparison for denoising of Chess (top two rows) and Painter (bottom two rows). From left to right: original,* $\sigma = 3$, $\sigma = 7$, $\sigma = 15$, *followed by false color insets. Details such as textures are well preserved even in extreme denoising with* $\sigma = 15$.

To investigate the impact of image noise, we performed a series of experiments in which we compared training and testing performance with respect to the original noisy datasets with those obtained through a denoising procedure. As described in Section 2, there is a large number of denoising methods, and we chose FFDNet [ZZZ18] as being representative for current state-of-the-art in denoising. FFDNet is, due to its efficiency, also suitable for the specific task of denoising video light fields with very large memory footprints. The experiments were carried out by comparing the original data to the data processed with different levels of denoising. We used $\sigma = 3, 5, 7, 10, 15$ as the denoising strength parameter. Figure 4 illustrates the impact of the denoising on data points from two different video light fields as false color images. For a detailed description of σ we refer the reader to [ZZZ18]. For a fair comparison, we measure the effect of different levels of denoising, σ, by computing the PSNR for light field data points with the same σ. The compression efficiency is then measured as PSNR / file size, i.e. taking into account the sparsity achieved by each configuration.

First, we investigate the effect of image noise in the training set on the AMDE dictionary learning process for the *Chess* and *Painter* datasets, see Figure 2. For training, we used 12 frames for each dataset and computed 32 dictionaries with parameters $C = 4$, $K = 8$, $\tau_l = 64$. For testing, we use 50 frames that do not include the training frames. The testing sparsity and threshold are set to $\tau_t = 256$ and $\delta_t = 1e-4$ for *Chess* and $\tau_t = 512$, $\delta_t = 5e-5$ for *Painter*. The plots in Figure 2 show that the compression efficiency is more or less flat for all variations of denoising in the training data, both with noisy and less noisy test data. This can be explained either by that the resulting dictionaries become more expressive when trained on noisy images so that less noisy images still can be accurately represented with a small number of coefficients, or that the representative power of the dictionaries is too weak to fully represent subtle details in neither the noise free nor the noisy images, although the visual quality and PSNR are very high. However, looking at the effect of denoising on compression ratio and PSNR with respect to the testing set, we see that the PSNR / file size in figures 2(a) and 2(b) increases with denoising strength, σ, even for very subtle denoising, keeping image details and structures intact, see Figure 4. The improvement is also confirmed by the two plots in Figure 3, which show that the PSNR increases by around 2dB and 4dB respectively. See the Appendix for tables 3 and 4, where we present all results from the two experiments described in this section.

Based on our investigations and the results from the experiments, we introduce a denoising step in the extended AMDE light field video pipeline as illustrated in Figure 1. An interesting observation regarding Figure 3 is that the effect of noise diminishes significantly if the light field video dataset is intrinsically sparse. For instance, we see that the *Chess* dataset, in contrast to the *Painter*, shows a gradual increase in PSNR when the amount of noise decreases through denoising. We associate this effect with the fact that the *Chess* dataset is significantly more sparse than *Painter* due to a small disparity range over the light field views.

5. Rendering and view interpolation

A virtual camera image computed during rendering corresponds to a 2D slice through the video light field. For a given pose of the virtual camera and a given time stamp, or video frame, f, we thus need to reconstruct a subset of the encoded data points from a large set of patches described by $s \times t \times u \times v \times c \times f$. For the examples generated here, a 6D patch typically spans 3-5 frames, f, in the temporal domain, 6-12 pixel-wide patches, (s,t), in the spatial domain, 4-8 pixel-wide patches, (u,v), in the angular domain, and 4 components for color and depth, c, in the wavelength domain. Since the dictionary size is very small, it is uploaded once to the GPU memory as textures. During rendering the coefficients corresponding to f frames are streamed to the GPU for reconstruction.

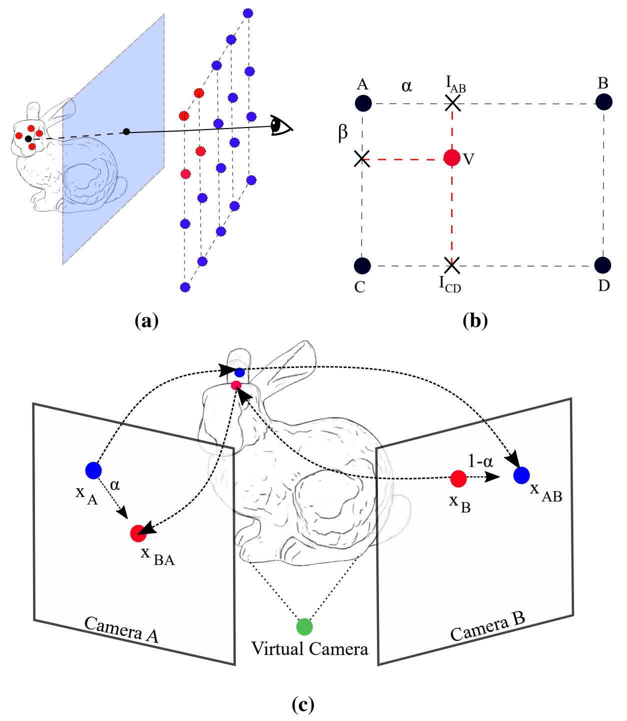


Figure 5: *(a) Proxy plane placed in front of the scene, (b) bilinear interpolation coefficients* α, β *, and (c) view generation by projecting from image coordinate to world coordinate and vice versa, moving point* x_A *towards the projection points from nearby views.*

Our GPU algorithm for rendering and view interpolation works as illustrated in Figure 5. For each fragment, we first render the 3D hit point, P, in world coordinates on a proxy geometry as seen from the virtual camera, see Figure 5(a). The proxy geometry acts as a surface at which the real scene is parameterized, and a plane can be used here, or more or less any other type of shape e.g. sphere, cube, or cylinder. We then compute the ray from the hit point P to the surface spanned by the camera positions. We usually arrange the cameras or sensors so that the vertices corresponding to their positions span an analytical surface, in Figure 5(a) we use a plane. Based on the ray from P to the camera surface, we can then for each fragment find the closest camera and use its calibrated extrinsic and intrinsic matrices to project the point P onto its sensor. Each data point contains the spatial, angular, spectral and time information and its reconstruction $\hat{\mathcal{T}}^{(i)}$ is a simple multiplication of the coefficients $\mathcal{S}^{(i)}$ with a dictionary in the ensemble $\{\mathbf{U}^{(1,1)},\ldots,\mathbf{U}^{(6,CK)}\}$:

$$\hat{\mathcal{T}}^{(i)} = \mathcal{S}^{(i)} \bigtimes_{j=1}^{6} \mathbf{U}^{(j,\mathbf{m}_i)}, \tag{9}$$

where $\mathbf{m}_i$ denotes the index of the best dictionary for the ith data point (obtained during testing, see Section 3.2). During rendering, for each pixel to be computed we only need a single element inside $\hat{\mathcal{T}}^{(i)}$ (assuming no interpolation). However, equation (9) reconstructs the entire data point with several thousands of elements, making it infeasible for real-time rendering. Instead, using the following formula, we can reconstruct a single element in a data point at an arbitrary location $x_1,\ldots,x_n$:

$$\hat{\mathcal{T}}^{(i)}_{x_1,\ldots,x_n} = \sum_{j=1}^{\tau_i} \mathcal{S}^{(i)}_{l_1^j,\ldots,l_6^j} \times_1 \mathbf{U}^{(1,\mathbf{m}_i)}_{x_1,l_1^j} \cdots \times_6 \mathbf{U}^{(6,\mathbf{m}_i)}_{x_6,l_6^j}, \tag{10}$$

where $l_1^j,\ldots,l_6^j$ describe the location of jth nonzero element in the sparse tensor $\mathcal{S}^{(i)}$. This fast, and parallelizable, random access to individual elements of the light field minimizes the computational complexity and enables fast reconstruction of a frame on the GPU.

A key challenge even for densely sampled light fields is to compute novel views. Here we use the encoded depth channel to enable linear interpolation along the spatio-angular dimensions of the light field. For each fragment, see Figure 5(a), we determine the four closest cameras to the viewing ray's intersection point at the camera surface. Based on the position of the virtual camera and its distance to the nearby cameras, A, B, C and D, we reconstruct the novel view from these four cameras. Figure 5(c) illustrates this procedure for two cameras A and B. In order to interpolate between the two points x_A and x_B, we project the points back to the scene using the reconstructed depth information and then compute the projection of x_B on to camera A and the projection of x_A on to camera B as x_{AB} and x_{BA} respectively. Interpolation is then carried out along the vector linking the two points $x_A \rightarrow x_{AB}$ in camera A, and $x_{BA} \rightarrow x_B$ in camera B, based on the interpolation parameter α that is determined from the distance between the virtual camera and cameras A and B. Figure 5(b) illustrates this for the four cameras A, B, C, and D, which is formulated as:

$$\begin{aligned} I = (1-\beta) * (\alpha * I_A(\hat{x}_A) + (1-\alpha) * I_B(\hat{x}_B)) + \\ \beta * (\alpha * I_C(\hat{x}_C) + (1-\alpha) * I_D(\hat{x}_D)), \end{aligned} \tag{11}$$

where β is the interpolation parameter along the second dimension at the surface spanned by the cameras and $\hat{x}_A, \hat{x}_B, \hat{x}_C$, and $\hat{x}_D$ are spatial positions of each camera after the reference point is moved towards the projection points of other cameras. For light field videos, the coefficients are read from disk into a circular buffer in RAM and then asynchronously streamed to the GPU as Pixel Buffer Objects (PBOs) for fast access and continuous playback.

For the reconstruction in equation (9), we first sort the sparse coefficients going from large to small in absolute value. Since this corresponds to how much they contribute to the final reconstruction, we can allow the user to interactively select the number of coefficients, providing a trade off between reconstruction quality and performance. Since some data points require significantly fewer coefficients than others, e.g. a flat region vs. a region with high frequency variations, we base the truncation on the values of the coefficients and sum the contributions down to a user defined threshold. The result is a highly efficient parallel rendering and view interpolation algorithm suitable for GPU implementation.

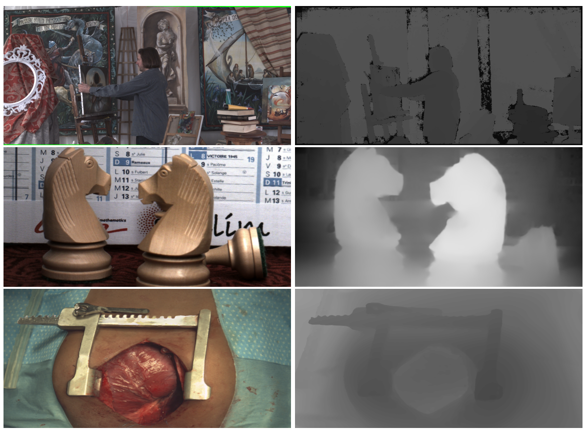


Figure 6: *Exemplars from datasets Painter, Chess and Heart we used for evaluating our framework. Left: single light field view of a frame. Right: its corresponding depth image.*

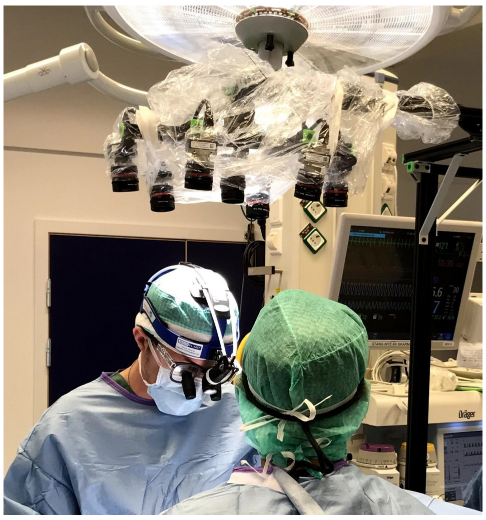


Figure 7: *Light field imaging in heart surgery.*

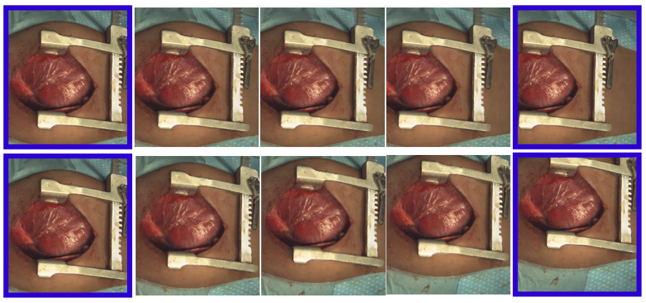
6. Results and applications

In this section, we present a number of example use cases demonstrating and evaluating the compression, processing and playback capabilities presented by the proposed light field video pipeline. See the supplementary video demonstrating a "research prototype" interface for our light field video player used to generate the results. The interface enables the user to playback compressed video light fields streaming from the disk, interactively control the view-point with or without view interpolation, control the coefficient threshold (quality vs. performance), and perform operations such as post-capture refocusing. All examples are generated using a standard PC equipped with 12 cores, 64GB DDR3 RAM, and a GeForce GTX 1080 Ti GPU.

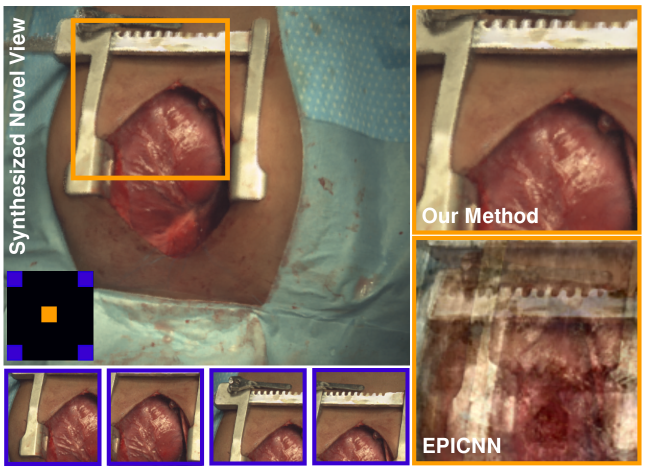
In the evaluation, we use three different datasets with different

	dataset	*Chess*	*Heart*	*Painter*
	Raw Size	118.65GB	7.23GB	101.18GB
	Size	1.69GB	0.84GB	8.19GB
Ours	PSNR	**47.37dB**	**41.93dB**	**38.16dB**
K-SVD	PSNR	42.55dB	39.14dB	38.12dB
HOSVD	PSNR	42.07dB	39.90dB	37.49dB
6D-DCT	PSNR	40.11dB	38.72dB	37.52dB
CDF 9/7	PSNR	36.13dB	39.40dB	34.84dB

Table 2: *Compression comparison between datasets: Chess, Heart and Painter where they are all compressed with depth or disparity information. Note that here we use all the available frames for compression. The Chess has 300 frames, The Heart has 25 frames, and the Painter has 372 frames.*



(a) *Vertical and horizontal interpolation of the Heart dataset.*



(b) *Linear interpolation between cameras A and B.*

Figure 8: *(a) shows vertical and horizontal interpolation of the Heart dataset. The right most and left most images are the sampled views. The top row demonstrates the horizontal interpolation and the bottom row vertical interpolation. (b) View interpolation of the state-of-the-art deep learning method EPICNN [WZW*17].*

characteristics, see Figure 6. The first dataset, named *Painter*, from Technicolor [SBV*17] consists of 372 light field frame captured with a 4×4 camera array system where the cameras are placed 6cm apart with a resolution of 2048×1088 pixels. This dataset is challenging for compression due to the large disparities (approximately 70 pixels) between light field views, hence exhibiting a small coherency along the angular domain. Moreover, this results in inaccuracies in the estimated depth images, as it can be seen in

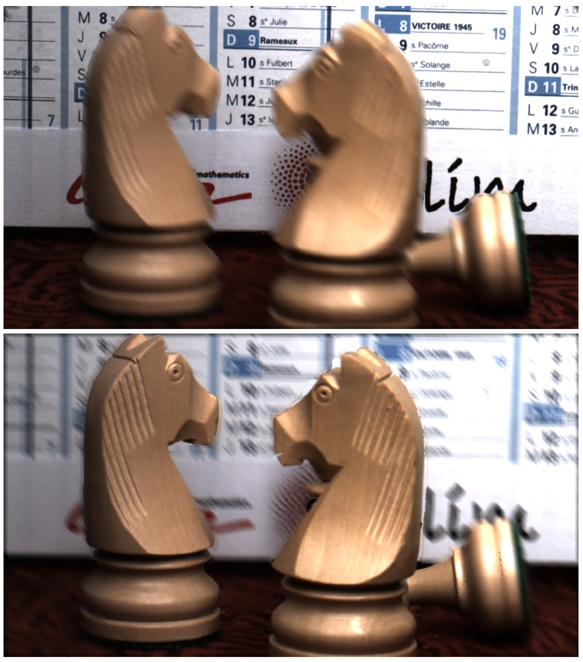


Figure 9: *Refocusing example on the Chess dataset*

Fig. 6, where a lot of noise and outliers are present. The *Painter* dataset was sliced into data points with $(s \times t \times u \times v \times c \times f) = 10 \times 10 \times 4 \times 4 \times 4 \times 4$ elements. The second dataset, named *Chess*, consists of 300 frames captured using an R8 Raytrix camera with a spatial resolution of 1920×1080 pixels and an angular resolution of 5×5 [GjLG18]. Each light field video frame is accompanied by disparity map images computed using the method described in [JPG18]. This dataset has sub-pixel disparity where the minimum disparity is -1.52 and the maximum disparity is 0.45. The overlap between the neighboring views is very large and consequently, it is expected to be easier to compress and render this data. The *Chess* dataset was sliced into data points with $8 \times 8 \times 4 \times 5 \times 5 \times 4$ elements. The third dataset, called *Heart*, comes from an application where we, in collaboration with *Barnhjärtcentrum* (the children's heart clinic at Skåne university hospital), use light field imaging to document live heart surgery. Figure 7 shows an example of camera configuration for capture in the operating theater. Efficient processing and high quality compression are highly important since surgery often times take up to 10 hours or more, which for a high resolution light field video leads to data sizes in the order of petabytes per surgery. Please note that the data is collected with the uttermost respect and with the consent of the patient and follows ethical guidelines and GDPR regulations (https://eugdpr.org). The goal of the initiative is to build an annotated database of light field videos from surgeries captured using different camera configurations for use in the development of teaching material, computer vision algorithms, and machine learning solutions. The light field video consists of 25 frames of a beating heart and is synthetically generated from a temporal 3D reconstruction to simulate a 7×7 camera array with a resolution of 752×1028 pixel perfect depth information. The disparity is around 80 pixels, which means that it is challenging for the compression algorithm. The *Heart* dataset was sliced into data points with $6 \times 6 \times 4 \times 7 \times 7 \times 5$ elements.

Table 2 shows the original and compressed sizes using our method for the three datasets described above. The compression ratios enabled by the sparse AMDE representation are, as expected, varying for different datasets. For *Chess* with only a small disparity the compression ratio is 70:1, and for *Heart* and *Painter* with large baseline, we obtain compression ratio of 8:1, and 12:1 respectively. All datasets can be streamed and rendered in real-time. The supplementary video shows several examples in which the light field videos are played back to demonstrate the performance and user interaction. To compare our method with the state of the art dictionaries used in compression, we adopt the testing methodology of [MHU19]. In particular, we set the parameters for the methods we compare to such that they result in the same compression ratio as our method. We then compare the image quality using PSNR.

For comparisons, we use K-SVD [AEB06], a well-established dictionary learning algorithm, Higher Order Singular Value Decomposition (HOSVD), six dimensional Discrete Cosine Transform (6D-DCT), and the CDF 9/7 wavelet dictionary [CDF92], which is the dictionary used in the JPEG2000 image compression standard [TM13]. Our method obtains higher PSNR for all the datasets we used here. As mentioned earlier, due to the inaccuracies of the depth layer for the *Painter* dataset, we do not observe a significant difference between various methods (except for CDF 9/7). However, for the *Chess* and *Heart* datasets, our method significantly outperforms other algorithms. Note that for a fair comparison, we do not use pruning in our framework since such operation is undefined for the methods we compare to. Table 1 shows the benefits of pruning in reducing the testing time with minimal impact on image quality. Moreover, the training for our method and K-SVD was performed using about 20% of each dataset.

Figure 8(a) illustrates view interpolation along the horizontal and vertical directions for the *Heart* dataset. The leftmost and rightmost images are the sampled light field view points and the intermediate views are generated with steps 0.3, 0.5, and 0.7, respectively. Moreover, Figure 8(b) shows the result of our bilinear interpolation between the four closest cameras using real-time depth map reconstruction. The same view is estimated using the state-of-the-art CNN-based view synthesis algorithm, EPICNN [WZW*17], where the reconstruction of the unseen view took about 36.4 seconds with a kernel size of 21 pixels. It should be noted that EPICNN does not have direct access to the depth map, although it outperforms CNN-based methods that use a depth map [KWR16]. However, as noted [WZW*17], EPICNN and related CNN-based methods [KWR16, YHC*18] fail when the disparity among views is large, see Figure 8. Figure 9 shows an example where the user refocuses the virtual view post-capture.

As described in Section 4, we conducted an extensive experiment to investigate the effect of image noise on the compression and reconstruction quality using AMDE. Tables 4 and 3 show the results from our investigation. The conclusion is that the smoothing of the signal introduced by subtle denoising increases the sparsity and leads to higher compression ratios.

7. Limitations and Future Work

The results from this paper show that data driven sparse representations constitute a key building block in the development of systems and algorithms for multi-sensor and light field imaging. Although the GPU-based pipeline presented in this paper achieves state-of-the-art performance for compression and real-time playback of video light fields, there are still several challenges to be solved. The dictionary pruning algorithm presented in this paper is currently carried out after the AMDE dictionary optimization. In the future, we plan to include dictionary similarity as a constraint in the AMDE optimization. Even though the compression algorithm provides fast access to each data point, it is still challenging to reconstruct multiple points for view interpolation. One way to solve this issue is to perform the interpolation and reconstruction in the coefficient space of data points, which removes the requirement for multiple reconstructions prior to interpolation. Regarding denoising, we believe that a multidimensional algorithm that exploits the intrinsic structures in a light field video is likely to perform better for enhancing the sparsity than image-based techniques, such as the one we utilized in this paper. Further explorations of this topic is left for future work. Moreover, an interesting direction for future research is the estimation of noise parameters of existing capturing systems and data sets. Given the noise parameters, there exists a large body of theoretical research that describe the tolerance of a sparse recovery algorithm to noise power, with respect to the sparsity and dictionary properties [Can08, MEUA17, DET06, EM09].

8. Conclusion

This paper presented a systems pipeline for compression and rendering of light field videos with real time playback. Building upon a previous method, AMDE, for sparse representation of high dimensional visual signals, we proposed a set of extensions to enable a full GPU powered pipeline for compression, processing and playback of high resolution and high quality light field videos. The novel contributions in this paper include: a dictionary formulation combining both color and depth information, a novel dictionary pruning algorithm, and an investigation of effects of noise on sparse representations. The result is a very efficient GPU powered end-to-end pipeline for light field video compression and playback.

9. Acknowledgement

We would like to thank the children's heart clinic at Skåne University hospital, Barnhjärtcentrum, for their support of this project. This work was supported through the strategic research environment ELLIIT, the Swedish Science Council through grant 2015-05180, Vinnova through grant 2017-03728: Surgeon's View, and the Visual Sweden Platform for Augmented Intelligence.

Appendix

The two tables below report all our results from the denoising experiments described in Section 4 for the *Chess* and *Painter* datasets.

References

[AEB06] Aharon M., Elad M., Bruckstein A.: K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation. *IEEE Transactions on Signal Processing 54*, 11 (Nov 2006), 4311–4322. 3, 10

[AFMS10] Abolghasemi V., Ferdowsi S., Makkiabadi B., Sanei S.: On optimization of the measurement matrix for compressive sensing. In *2010 18th European Signal Processing Conference* (Aug 2010), pp. 427–431. 5

Testing	σ	File Size(MB)	PSNR	PSNR/Size
Training Ref	Ref	234	40.53	0.1734
	3	128	41.20	0.3215
	5	88	41.59	0.4740
	7	71	41.96	0.5897
	10	61	42.37	0.6915
	15	**56**	**42.76**	**0.7691**
$\sigma = 3$	Ref	235	40.52	0.1727
	3	129	41.19	0.3183
	5	90	41.63	0.4639
	7	73	42.00	0.5747
	10	63	42.45	0.6742
	15	**57**	**42.86**	**0.7485**
$\sigma = 5$	Ref	237	40.57	0.1715
	3	131	41.29	0.3143
	5	91	41.68	0.4596
	7	74	42.04	0.5671
	10	64	42.46	0.6644
	15	**58**	**42.86**	**0.7409**
$\sigma = 7$	Ref	238	40.53	0.1704
	3	130	41.16	0.3169
	5	89	41.55	0.4683
	7	71	41.91	0.5862
	10	61	42.33	0.6909
	15	**55**	**42.76**	**0.7729**
$\sigma = 10$	Ref	242	40.59	0.1679
	3	135	41.32	0.3062
	5	93	41.76	0.4512
	7	75	42.17	0.5599
	10	65	42.65	0.6536
	15	**59**	**43.09**	**0.7257**
$\sigma = 15$	Ref	238	40.54	0.1703
	3	130	41.16	0.3168
	5	88	41.52	0.4718
	7	71	41.87	0.5916
	10	61	42.30	0.6971
	15	**55**	**42.72**	**0.7753**

Table 3: *Denoising effect on compression of Chess (50 frames).*

Testing	σ	File Size(MB)	PSNR	PSNR/Size
Training Ref	Ref	1048	39.56	0.03775
	3	851	41.91	0.04926
	5	711	42.85	0.06024
	7	631	43.26	0.06857
	10	561	43.55	0.07765
	15	**487**	**43.77**	**0.08998**
$\sigma = 3$	Ref	1048	39.55	0.03775
	3	851	41.91	0.04924
	5	713	42.85	0.06011
	7	631	43.25	0.06850
	10	561	43.54	0.07760
	15	**487**	**43.76**	**0.08988**
$\sigma = 5$	Ref	1048	39.53	0.03775
	3	852	41.89	0.04915
	5	715	42.83	0.05992
	7	634	43.23	0.06823
	10	563	43.52	0.07729
	15	**489**	**43.75**	**0.08954**
$\sigma = 7$	Ref	1048	39.53	0.03775
	3	850	41.88	0.04924
	5	712	42.81	0.06012
	7	630	43.21	0.06856
	10	560	43.49	0.07768
	15	**487**	**43.73**	**0.08987**
$\sigma = 10$	Ref	1048	39.53	0.03775
	3	851	41.88	0.04920
	5	712	42.82	0.06011
	7	631	43.22	0.06852
	10	560	43.50	0.07768
	15	**486**	**43.73**	**0.09007**
$\sigma = 15$	Ref	1048	39.50	0.03775
	3	852	41.84	0.04908
	5	715	42.77	0.05979
	7	634	43.17	0.06815
	10	563	43.47	0.07722
	15	**488**	**43.70**	**0.08951**

Table 4: *Denoising effect on compression of Painter (50 frames).*

[APF17] ABDI A., PAYANI A., FEKRI F.: Learning Dictionary for Efficient Signal Compression. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (March 2017), pp. 3689–3693. 3

[AV07] ARTHUR D., VASSILVITSKII S.: K-means++: The Advantages of Careful Seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (2007), SODA '07, Society for Industrial and Applied Mathematics, pp. 1027–1035. 5

[AW92] ADELSON E. H., WANG J. Y. A.: Single lens stereo with a plenoptic camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence 14*, 2 (Feb 1992), 99–106. 1

[BHEE10] BEN-HAIM Z., ELDAR Y. C., ELAD M.: Coherence-Based Performance Guarantees for Estimating a Sparse Vector Under Random Noise. *IEEE Transactions on Signal Processing 58*, 10 (Oct 2010), 5030–5043. 5, 6

[BMU19] BARAVDISH G., MIANDJI E., UNGER J.: Gpu accelerated sparse representation of light fields. In *14th International Joint Conference on Computer Vision, Imaging and Computer Grahics Theory and Applications* (Feb 2019). 3

[BSW01] BENNETT S. WILBURN MICHAL SMULSKI H.-H. K. L. M. A. H.: Light field video camera. vol. 4674. 1

[Can08] CANDÈS E. J.: The Restricted Isometry Property and its Implications for Compressed Sensing. *Comptes Rendus Mathematique 346*, 9 (May 2008), 589–592. 10

[CDF92] COHEN A., DAUBECHIES I., FEAUVEAU J.-C.: Biorthogonal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics 45*, 5 (June 1992), 485–560. 3, 10

[CSHD11] CHAURASIA G., SORKINE HORNUNG O., DRETTAKIS G.: Silhouette-aware warping for image-based rendering. *Computer Graphics Forum 30*, 4 (2011). 3

[CTCS00] CHAI J. X., TONG X., CHAN S. C., SHUM H. Y.: Plenoptic sampling. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (2000), SIGGRAPH '00, ACM Press/Addison-Wesley Publishing Co., pp. 307–318. 3

[DET06] DONOHO D., ELAD M., TEMLYAKOV V.: Stable recovery of sparse overcomplete representations in the presence of noise. *Information Theory, IEEE Transactions on 52*, 1 (Jan 2006), 6–18. 10

[DFKE07] DABOV K., FOI A., KATKOVNIK V., EGIAZARIAN K.: Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on Image Processing 16*, 8 (2007), 2080–2095. 3

[ELL08] EASLEY G., LABATE D., LIM W.-Q.: Sparse directional image representations using the discrete shearlet transform. *Applied and Computational Harmonic Analysis 25*, 1 (2008), 25–46. 3

[EM09] ELDAR Y. C., MISHALI M.: Robust recovery of signals from a structured union of subspaces. *IEEE Transactions on Information Theory 55*, 11 (Nov 2009), 5302–5316. 3, 10

[FNPS16] FLYNN J., NEULANDER I., PHILBIN J., SNAVELY N.: Deep stereo: Learning to predict new views from the world's imagery. In *IEEE CVPR* (June 2016), pp. 5515–5524. 3

[GCRX03] GIROD B., CHUO-LING CHANG, RAMANATHAN P., XIAOQING ZHU: Light field compression using disparity-compensated lifting. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '03).* (April 2003), vol. 4, pp. IV–760. 3

[GGSC96] GORTLER S. J., GRZESZCZUK R., SZELISKI R., COHEN M. F.: The lumigraph. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1996), SIGGRAPH '96, ACM, pp. 43–54. 1

[GjLG18] GUILLO L., JIANG X., LAFRUIT G., GUILLEMOT C.: Light field video dataset captured by a r8 raytrix camera, April 2018. 9

[GZZF14] GU S., ZHANG L., ZUO W., FENG X.: Weighted nuclear norm minimization with application to image denoising. In *2014 IEEE CVPR* (June 2014), pp. 2862–2869. 3

[JPG18] JIANG X., PENDU M. L., GUILLEMOT C.: Depth estimation with occlusion handling from a sparse set of light field views. In *2018 25th IEEE International Conference on Image Processing (ICIP)* (Oct 2018), pp. 634–638. 9

[JSA03] JAGMOHAN A., SEHGAL A., AHUJA N.: Compression of light-field rendered images using coset codes. In *The Thrity-Seventh Asilomar Conference on Signals, Systems Computers* (Nov 2003), vol. 1, pp. 830–834 Vol.1. 3

[KKSM19] KONIARIS C., KOSEK M., SINCLAIR D., MITCHELL K.: Compressed animated light fields with real-time view-dependent reconstruction. *IEEE Transactions on Visualization and Computer Graphics 25*, 4 (April 2019), 1666–1680. 1

[KWR16] KALANTARI N. K., WANG T.-C., RAMAMOORTHI R.: Learning-based view synthesis for light field cameras. *ACM Transactions on Graphics 35*, 6 (Nov 2016). 3, 10

[LH96] LEVOY M., HANRAHAN P.: Light field rendering. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (1996), SIGGRAPH '96, ACM, pp. 31–42. 1

[Llo82] LLOYD S.: Least squares quantization in PCM. *IEEE Transactions on Information Theory 28*, 2 (1982), 129–137. 5

[Lyt17] The ultimate creative tool for cinema and broadcast, 2017. 1

[MBG*13] MUKHERJEE D., BANKOSKI J., GRANGE A., HAN J., KOLESZAR J., WILKINS P., XU Y., BULTJE R.: The latest open-source video codec vp9 - an overview and preliminary results. In *2013 Picture Coding Symposium (PCS)* (Dec 2013), pp. 390–393. 3

[MBPS09] MAIRAL J., BACH F., PONCE J., SAPIRO G.: Online Dictionary Learning for Sparse Coding. In *Proceedings of the 26th Annual International Conference on Machine Learning* (2009), pp. 689–696. 3

[MEUA17] MIANDJI† E., EMADI† M., UNGER J., AFSHARI E.: On Probability of Support Recovery for Orthogonal Matching Pursuit Using Mutual Coherence. *IEEE Signal Processing Letters 24*, 11 (Nov 2017), 1646–1650. † equal contributor. 6, 10

[MG00] MAGNOR M., GIROD B.: Data compression for light-field rendering. *IEEE Transactions on Circuits and Systems for Video Technology 10*, 3 (April 2000), 338–343. 3

[MHU19] MIANDJI E., HAJISHARIF S., UNGER J.: A unified framework for compression and compressed sensing of light fields and light field videos. *ACM Trans. Graph. 38*, 3 (May 2019), 23:1–23:18. 1, 3, 4, 5, 10

[MKU13] MIANDJI E., KRONANDER J., UNGER J.: Learning Based Compression of Surface Light Fields for Real-time Rendering of Global Illumination Scenes. In *SIGGRAPH Asia 2013 Technical Briefs* (2013), ACM, pp. 24:1–24:4. 3

[MKU15] MIANDJI E., KRONANDER J., UNGER J.: Compressive Image Reconstruction in Reduced Union of Subspaces. *Computer Graphics Forum 34*, 2 (May 2015), 33–44. 3

[MPE19] Coded representation of immersive media, 2019. 3

[NLB*05] NG R., LEVOY M., BREDIF M., DUVAL G., HOROWITZ M., HANRAHAN P.: Light field photography with a hand-held plenopic camera. *Technical Report CTSR 2005-02 CTSR* (01 2005). 1

[OEE*18] OVERBECK R. S., ERICKSON D., EVANGELAKOS D., PHARR M., DEBEVEC P.: A system for acquiring, processing, and rendering panoramic light field stills for virtual reality. *ACM Trans. Graph. 37*, 6 (Dec. 2018), 197:1–197:15. 3

[Rak13] RAKOTOMAMONJY A.: Direct Optimization of the Dictionary Learning Problem. *IEEE Transactions on Signal Processing 61*, 22 (Nov 2013), 5495–5506. 3

[Ray19] RAYTRIX: 3d light field camera technology, 2019. 1

[RD12] RUSU C., DUMITRESCU B.: Stagewise K-SVD to Design Efficient Dictionaries for Sparse Representations. *IEEE Signal Processing Letters 19*, 10 (Oct 2012), 631–634. 3

[SBV*17] SABATER N., BOISSON G., VANDAME B., KERBIRIOU P., BABON F., HOG M., GENDROT R., LANGLOIS T., BURELLER O., SCHUBERT A., ALLIÉ V.: Dataset and pipeline for multi-view light-field video. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (July 2017), pp. 1743–1753. 9

[SCK07] SHUM H. Y., CHAN S., KANG S. B.: *Image-Based Rendering*. Springer US, 01 2007. 3

[SPRE18] SULAM J., PAPYAN V., ROMANO Y., ELAD M.: Multilayer Convolutional Sparse Modeling: Pursuit and Dictionary Learning. *IEEE Transactions on Signal Processing 66*, 15 (Aug 2018), 4090–4104. 3

[TM13] TAUBMAN D., MARCELLIN M.: *JPEG2000 Image Compression Fundamentals, Standards and Practice*. Springer Publishing Company, Incorporated, 2013. 2, 10

[VRA*07] VEERARAGHAVAN A., RASKAR R., AGRAWAL A., MOHAN A., TUMBLIN J.: Dappled photography: Mask enhanced cameras for heterodyned light fields and coded aperture refocusing. vol. 26. 1

[WIH13] WETZSTEIN G., IHRKE I., HEIDRICH W.: On plenoptic multiplexing and reconstruction. *International Journal of Computer Vision 101*, 2 (Jan 2013), 384–400. 1

[WLHR12] WETZSTEIN G., LANMAN D., HIRSCH M., RASKAR R.: Tensor Displays: Compressive Light Field Synthesis using Multilayer Displays with Directional Backlighting. *ACM Trans. Graph. 31*, 4 (2012), 1–11. 1

[WMJ*17] WU G., MASIA B., JARABO A., ZHANG Y., WANG L., DAI Q., CHAI T., LIU Y.: Light field image processing: An overview. *IEEE Journal of Selected Topics in Signal Processing 11* (2017), 926–954. 1, 3

[WZW*17] WU G., ZHAO M., WANG L., DAI Q., CHAI T., LIU Y.: Light field reconstruction using deep convolutional network on epi. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (July 2017), pp. 1638–1646. 3, 9, 10

[XAG03] XIAOQING ZHU, AARON A., GIROD B.: Distributed compression for large camera arrays. In *IEEE Workshop on Statistical Signal Processing, 2003* (Sep. 2003), pp. 30–33. 3

[YHC*18] YEUNG H. W. F., HOU J., CHEN J., CHUNG Y. Y., CHEN X.: Fast light field reconstruction with deep coarse-to-fine modeling of spatial-angular clues. In *Computer Vision – ECCV 2018* (2018), Springer International Publishing, pp. 138–154. 3, 10

[ZZC*17] ZHANG K., ZUO W., CHEN Y., MENG D., ZHANG L.: Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing 26*, 7 (July 2017), 3142–3155. 3

[ZZZ18] ZHANG K., ZUO W., ZHANG L.: Ffdnet: Toward a fast and flexible solution for cnn-based image denoising. *IEEE Transactions on Image Processing 27*, 9 (Sep. 2018), 4608–4622. 2, 3, 6, 7