A Unified Framework for Compression and Compressed Sensing of Light Fields and Light Field Videos

Ehsan Miandji, Saghi Hajisharif and Jonas Unger

A Unified Framework for Compression and Compressed Sensing of Light Fields and Light Field Videos

EHSAN MIANDJI, Linköping University
SAGHI HAJISHARIF, Linköping University
JONAS UNGER, Linköping University

In this paper we present a novel dictionary learning framework designed for compression and sampling of light fields and light field videos. Unlike previous methods, where a single dictionary with one dimensional atoms is learned, we propose to train a Multidimensional Dictionary Ensemble (MDE). It is shown that learning an ensemble in the native dimensionality of the data promotes sparsity, hence increasing the compression ratio and sampling efficiency. To make maximum use of correlations within the light field data sets, we also introduce a novel nonlocal pre-clustering approach that constructs an Aggregate MDE (AMDE). The pre-clustering not only improves the image quality, but also reduces the training time by an order of magnitude in most cases. The decoding algorithm supports efficient local reconstruction of the compressed data, which enables efficient real-time playback of high resolution light field videos. Moreover, we discuss the application of AMDE for compressed sensing. A theoretical analysis is presented which indicates the required conditions for exact recovery of point-sampled light fields that are sparse under AMDE. The analysis provides guidelines for designing efficient compressive light field cameras. We use various synthetic and natural light field and light field video data sets to demonstrate the utility of our approach in comparison with the state-of-the-art learning based dictionaries, as well as established analytical dictionaries.

CCS Concepts: • **Computing methodologies → Rendering**; **Image compression**;

Additional Key Words and Phrases: light field video compression, light field photography, dictionary learning, compressed sensing

ACM Reference format:
Ehsan Miandji, Saghi Hajisharif, and Jonas Unger. 2019. A Unified Framework for Compression and Compressed Sensing of Light Fields and Light Field Videos. *ACM Trans. Graph.* 1, 1, Article 1 (June 2019), 18 pages.
https://doi.org/10.1145/1122445.1122456

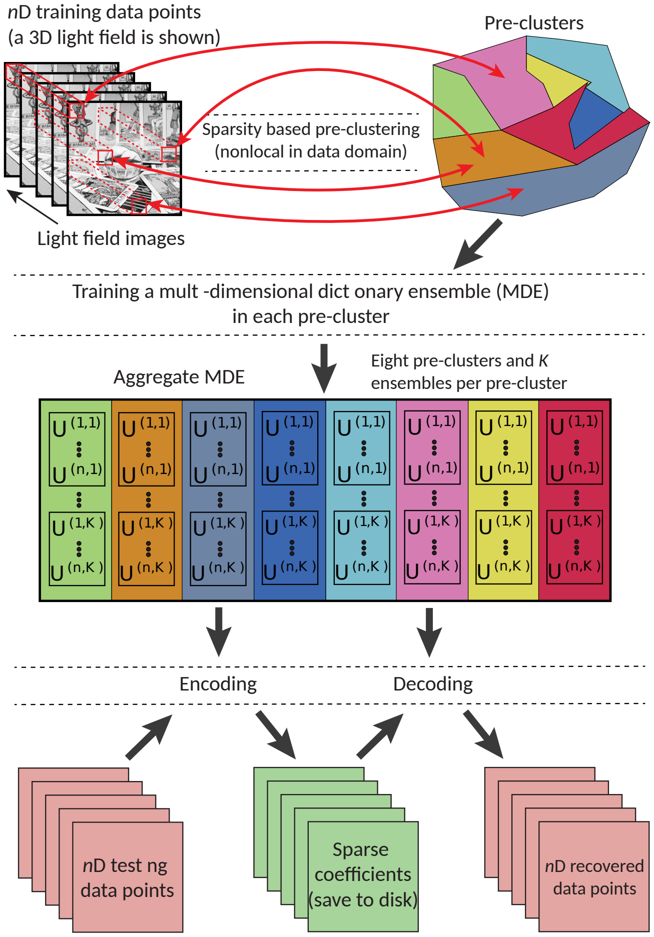

*n*D training data points
(a 3D light field is shown)
Pre-clusters
Sparsity based pre-clustering
(nonlocal in data domain)
Light field images
Training a multi-dimensional dictionary ensemble (MDE)
in each pre-cluster
Aggregate MDE
Eight pre-clusters and *K*
ensembles per pre-cluster
U (1,1)
U (n,1)
U (1,K)
U (n,K)
Encoding
Decoding
*n*D testing
data points
Sparse
coefficients
(save to disk)
*n*D recovered
data points

Fig. 1. Overview of our compression method. An *n*D training data set, in this example a monochrome light field (3D), is divided into a set of *data points*. After performing pre-clustering on the data points, a Multidimensional Dictionary Ensemble (MDE) is trained in each pre-cluster (color coded in the figure). Next, we combine MDEs to form an Aggregate MDE (AMDE). The AMDE is used for sparse representation of data points, which in turn can be used for compression and compressed sensing.

1 INTRODUCTION

Over the last decade we have seen the field of computational photography, especially light field and multi-view imaging, emerge and mature as a new paradigm in imaging technology. These technologies enable a range of novel applications ranging from advanced multidimensional image processing such as refocusing [Ng et al. 2005] and depth estimation [Vaish et al. 2006] to cinematic editing [Jarabo et al. 2014], glasses free 3D display systems [Jones et al. 2016; Lee et al. 2016; Wetzstein et al. 2012b,a], single sensor light field cameras [Babacan et al. 2012; Marwah et al. 2013; Miandji et al. 2018], spectral imaging [Arguello and Arce 2014; Kittle et al. 2010], and appearance capture [Gortler et al. 1996; Levoy and Hanrahan 1996; Müller et al. 2005]. A highly important and still unsolved challenge inherent to capture, storage and processing of such high dimensional data is to handle the very large data sizes. For example, the multi-view video captured using 30 HD cameras used to drive

0730-0301/2019/6-ART1 $15.00
https://doi.org/10.1145/1122445.1122456

the 3D display in [Jones et al. 2016] at 30 Hz corresponds to more than 5 GB of data per second.

The majority of existing compression algorithms, e.g. the well-established JPEG standard, rely on a simple framework. First, a data set is divided into a set of small elements. For instance, the JPEG algorithm divides an image into distinct patches of size 8×8 pixels. The patches are transformed using a *dictionary* to obtain a small number of *coefficients*. In the case of the JPEG standard, the Discrete Cosine Tranform (DCT) is used as the dictionary. Indeed, since the number of coefficients is lower than the number of pixels in the patch, the obtained representation is the compressed form of the image patch. The coefficients are then quantized and compressed further using an entropy coding algorithm.

In this paper we present a novel framework for the compression of light field and light field video data sets based on a highly efficient dictionary learning approach, see Fig. 1 for an outline of the framework. A training set is first divided into small nD data points (typically 5D or 6D), see Section 2. The resulting data points are fed into an one-time training process that learns a Multidimensional Dictionary Ensemble (MDE), as described in Section 3. Each nD dictionary in the ensemble is a collection of basis functions, also known as atoms, that independently represent each data point along its various dimensions in a transformation domain. The trained ensemble enables a high degree of sparsity in the transformation domain, which has been shown to be an important factor for efficient compression [Mallat 2008; Miandji et al. 2013; Zepeda et al. 2011] and compressed sensing [Candès et al. 2006; Candes and Tao 2006]. While we only focus on light fields and light field videos, the framework can be readily applied for compression and compressed sensing of commonly used high dimensional data sets in graphics such as BTFs [Dana et al. 1999], measured BRDFs, hyperspectral images and videos (see [Miandji 2018] for a few examples).

To further improve the representation power of MDE and enable training on large-scale data sets, we propose a novel nonlocal pre-clustering approach to divide the data points of the training set into groups with similar sparsity, while minimizing the reconstruction error. For each pre-cluster an MDE is trained, then combined to form an Aggregate MDE (AMDE). AMDE has a very small memory footprint (less than 1MB), hence encoding and decoding is substantially more efficient than previous methods (see Section 6). Our proposed pre-clustering improves the training process with regards to two important aspects: 1. It promotes sparsity for each MDE, which in turn improves the compression and compressed sensing performance, and 2. It reduces the training time, including pre-clustering, by an order of magnitude (see Section 3.2).

Having the dictionary ensemble, encoding of the unobserved light fields, i.e. the testing set, is achieved by projecting each data point onto the dictionary in AMDE that produces the most sparse coefficients with the least error. The coefficients are then coded and stored to disk. The decoding process is as simple as multiplying the decoded coefficients by their corresponding dictionary in AMDE. Indeed for real-time rendering of light fields, one requires random access to the compressed data. For instance, to reconstruct a single view in one frame of a light field video, only a small portion of the data set needs to be decompressed. We propose a method for reconstruction of compressed multidimensional light field data sets down to a single pixel level. Due to the small memory footprint of the compressed data and the random-access property, our method achieves real-time reconstruction and rendering of light field videos at a Full HD resolution using consumer-level hardware. In Section 5, we perform a thorough parameter analysis to show the effect of each parameter on image quality. This can be used as a guideline in different applications for obtaining a desired compression ratio or image quality.

An important property of AMDE is *universality*. The efficient sparse representation of AMDE, achieved by nD dictionary training and pre-clustering, allows us to perform the training only once for each application. The trained AMDE can then be used for compression and compressed sensing of any unobserved data set (see Section 3.3). This is in contrast with prior work on compression [Bilgili et al. 2011; Maimone et al. 2013; Miandji et al. 2013; Tsai 2015; Tsai and Shih 2012; Wang et al. 2005], where a dictionary has to be learned/computed for any unobserved data set.

The sparsity of coefficients introduced by AMDE has an additional application, namely Compressed Sensing (CS) [Donoho 2006]. Compressed sensing states that if a signal is sufficiently sparse in a dictionary, it can be reconstructed exactly from only a few measurements, much less than what is required by the Nyquist criterion [Candes and Wakin 2008; Dragotti and Lu 2014; Gribonval and Nielsen 2003; Tropp 2004]. We show in Section 4 that AMDE can be used for efficient sampling of light fields using an extension of the framework proposed in [Miandji et al. 2015] to higher dimensions. This is particularly useful for designing light field coded-aperture video cameras, where a mask is placed on the aperture [Babacan et al. 2012] or at a small distance from the sensor [Marwah et al. 2013; Miandji et al. 2018]. Indeed designing such systems are essential given the excessively high data rate of a high-resolution light field video camera. Our main contribution regarding compressed sensing is a theoretical analysis for the uniqueness of the solution obtained using the proposed CS framework, see Section 4.1. In particular, we will describe conditions under which one can *exactly* recover an incomplete light field that is sparse under an AMDE. Our results show that for a sufficiently sparse light field one can uniquely recover the light field with high probability. This analysis can provide guidelines for designing efficient light field cameras.

Recently, Convolutional Neural Networks (CNN) have been utilized for light field processing. For instance, several methods for spatial super-resolution [Wang et al. 2018; Yoon et al. 2015], angular super-resolution (also known as view synthesis) [Choudhury et al. 2017; Kalantari et al. 2016; Wu et al. 2017b], and joint spatio-angular super-resolution [Yoon et al. 2017] have been proposed. However, light field compression and multidimensional compressive point sampling (Section 4) have not been considered.

The main contributions of this paper are summarized below:

- A training-based approach that identifies intrinsic self-similarities present in light fields to enable sparse representation using an ensemble of nD dictionaries, with a negligible memory footprint.
- A novel nonlocal pre-clustering method for nD data sets that not only accelerates the training process by about an order of magnitude, but also reduces reconstruction error.
- We show that sparsity induced by AMDE can be utilized for compressed sensing.

- The proposed method resides on a strong theoretical foundation. We prove required conditions on AMDE, sparsity, and the number of samples such that one can *exactly* recover a light field from a few point samples with high probability. These conditions can be used as guidelines for designing efficient light field cameras.

We evaluate AMDE by comparing to both analytical and training-based dictionaries. Analytical dictionaries are based on a fixed equation derived from a mathematical model of the data. Typical examples are discrete cosine transform, used in the JPEG standard, and wavelets, used in the JPEG2000 standard. In contrast, training-based dictionaries infer an explicit dictionary from a set of examples. The most widely used training-based dictionary is K-SVD [Aharon et al. 2006]. Our method achieves much higher image quality in most cases compared to both analytical and training-based dictionaries. Moreover, the encoding and decoding time for AMDE is competitive with analytical dictionaries while being substantially faster than K-SVD. Additionally, the storage cost for AMDE is negligible and is orders of magnitude smaller than a K-SVD dictionary.

Notations

Throughout the paper, we use the following notational convention. Vectors and matrices are denoted by boldface lower-case ($\mathbf{a}$) and bold-face upper-case letters ($\mathbf{A}$), respectively. Tensors are denoted by calligraphic letters, e.g. $\mathcal{A}$. A finite set of objects is indexed by superscripts, e.g. $\left\{\mathbf{A}^{(i)}\right\}_{i=1}^{N}$ or $\left\{\mathcal{A}^{(i)}\right\}_{i=1}^{N}$. Individual elements of $\mathbf{a}$, $\mathbf{A}$, and $\mathcal{A}$ are denoted $\mathbf{a}_i$, $\mathbf{A}_{i_1,i_2}$, $\mathcal{A}_{i_1,\dots,i_n}$, respectively. The nth column and row of $\mathbf{A}$ are denoted $\mathbf{A}_{.,n}$ and $\mathbf{A}_{n,.}$, respectively. Given an index set, I, the sub-matrix $\mathbf{A}_{.,I}$ is formed from columns of $\mathbf{A}$ indexed by I. The unfolding of a tensor $\mathcal{A}$ along mode j is denoted $\mathcal{A}_{[j]}$. The ℓ_p norm of a vector $\mathbf{s}$, for $1 \leq p \leq \infty$, is denoted by $\|\mathbf{s}\|_p$. Frobenius norm is denoted $\|\mathbf{s}\|_F$. The ℓ_0 pseudo-norm of this vector, $\|\mathbf{s}\|_0$, defines the number of non-zero elements.

2 LIGHT FIELD DATA POINTS

Let the function $l(\mathbf{r}_i, \mathbf{t}_j, \mathbf{u}_\alpha, \mathbf{v}_\beta, \lambda_\gamma)$ describe the two plane light field parametrization [Gortler et al. 1996] defined by a pair of spatial locations $(\mathbf{r}_i, \mathbf{t}_j)$, angular locations $(\mathbf{u}_\alpha, \mathbf{v}_\beta)$, and a spectral parameter λ_γ. We divide the light field into small elements, which we call *data points.* Let $m_1 \times m_2 \times m_3 \times m_4 \times m_5$ be the dimensionality of each data point. For a seamless angular representation of the light field, we include all the angular ($m_3 \times m_4$) and spectral information (m_5) in each data point, while dividing the spatial domain (e.g. individual light field view images in a multi-camera setup) into small patches of size $m_1 \times m_2$. The spatial patches are constructed using a non-overlapping sliding window. Indeed for light fields with high angular resolution, one has to also divide the angular domain in each data point into smaller patches for exploiting the coherence in angular domain. In this case, we obtain 7D data points. Furthermore, if the spectral domain is divided (e.g. for hyperspectral light fields), we obtain 8D data points. Since there are various approaches for constructing data points from light fields, we do not assume the dimensionality of data points and present our method for nD data points.

A light field video can be represented by the function $l(\mathbf{r}_i, \mathbf{t}_j, \mathbf{u}_\alpha, \mathbf{v}_\beta, \lambda_\gamma, f)$, where f is the time domain. To exploit the coherence in the time domain, each data point contains a small number of consecutive frames. Hence each data point is 6D, with dimensionality $m_1 \times m_2 \times m_3 \times m_4 \times m_5 \times m_6$, where m_6 is the number of consecutive frames used in a data point. One can also construct overlapping data points from light fields and light field videos along e.g. spatial dimension. This will indeed have negative effect on compression. However, significant improvements can be achieved in compressed sensing [Elad and Aharon 2006; Miandji et al. 2015].

3 COMPRESSION

In this section, a novel multidimensional dictionary learning method suitable for compression and compressed sensing of multidimensional data sets such as light fields and light field videos is introduced. We start by describing the requirements for dictionary learning of multidimensional data. In Section 3.1, the training algorithm for an MDE is presented, which takes into account the aforementioned requirements. In Section 3.2 our novel pre-clustering method is presented. By training an MDE for each pre-cluster, we obtain an AMDE, which will be described in Section 3.3. Encoding and decoding of light field and light field video datasets using the AMDE will be described in sections 3.4 and 3.5, respectively. In Section 5, we perform a thorough parameter analysis and provide guidelines for choosing suitable values for each parameter depending on the application at hand.

Our method for constructing the dictionary ensemble is learning based. Hence we consider two data sets: a *training* set and a *testing* set. The training set is utilized for learning an MDE, or AMDE, while the testing set represents the data sets we would like to compress. We denote the training set as $\{\mathcal{L}^{(i)}\}_{i=1}^{N_l}$, where $\mathcal{L}^{(i)} \in \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_n}$ and N_l is the number of data points. Similarly, the testing set is denoted $\{\mathcal{T}_i\}_{i=1}^{N_t}$. Indeed the dimensionality of the data points in the training and testing sets should be identical. Moreover, the training is performed only once, unless the data dimensionality changes.

We train an ensemble of $K \ll N_l$ orthonormal dictionaries of order n, denoted $\left\{\mathbf{U}^{(1,k)}, \dots, \mathbf{U}^{(n,k)}\right\}_{k=1}^{K}$, such that each data point $\mathcal{L}^{(i)}$ is represented by *one* dictionary as follows

$$\mathcal{L}^{(i)} = \mathcal{S}^{(i)} \times_1 \mathbf{U}^{(1,k)} \times_2 \cdots \times_n \mathbf{U}^{(n,k)} = \mathcal{S}^{(i)} \bigtimes_{j=1}^{n} \mathbf{U}^{(j,k)}, \qquad (1)$$

where $\mathbf{U}^{(j,k)} \in \mathbb{R}^{m_j \times m_j}$ is the jth dictionary element of kth dictionary. The coefficient tensor $\mathcal{S}^{(i)} \in \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_n}$ is sparse and the symbol $\times_j$ defines tensor-matrix product along the jth mode.

In addition to sparsity, we would like our representation of the data set to possess the following properties:

- ***nonlocal clustering***: Each data point in our model is represented by one dictionary. Since $K \ll N_l$, the training algorithm also performs clustering. Hence a collection of points share a multidimensional dictionary. Such clustering has been theoretically and empirically shown to promote sparsity [Eldar et al. 2010; Elhamifar and Vidal 2013; Miandji et al. 2015; Soltanolkotabi et al. 2012]. Since the metric for this clustering is based on sparsity, data points that are similar in an ℓ_2 sense may not be grouped together. Hence the clustering is nonlocal.

- ***representative power***: The training is performed only once. Hence AMDE should be effective enough to represent any unobserved data set (of the same dimensionality). This property is not possessed by tensor decomposition based methods such as [Gutheet al. 2009; Tsai 2015; Tsai and Shih 2012; Vasilescu and Terzopoulos 2004; Wang et al. 2005].
- ***small dictionary size***: Since we use a training based approach, the dictionary should be stored for reconstruction. Previous methods create relatively large dictionaries that are challenging to transmit to the decoder [Aharon et al. 2006; Guthe et al. 2009; Tsai 2015].
- ***arbitrary dimensionality***: The dictionary ensemble should be flexible enough to be applied to the many variations of the Plenoptic function [Adelson and Bergen 1991], as well as different data point dimensionalities.
- ***local reconstruction***: For real-time light field rendering, efficient local reconstruction of the compressed data is essential.

In the remainder of this section, we will describe our approach for training an MDE, followed by the proposed pre-clustering method. Together, these two steps construct AMDE, which satisfies the above criteria.

3.1 Multidimensional Dictionary Ensemble

We start by formulating the problem of learning an ensemble of nD orthonormal dictionaries that enable sparse representation:

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

subject to

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

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

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

where $\boldsymbol{\mathcal{S}}^{(i,k)}$ contains coefficients of the ith data point when projected onto the kth dictionary in the ensemble. The ensemble consists of K dictionaries, where each dictionary is a collection of n matrices (which we call *dictionary elements*) operating along different modes. The orthogonality of each dictionary is ensured in (2b). The constant τ_l in (2c) is a user-defined sparsity value for training. We call $\mathbf{M} \in \mathbb{R}^{N_l \times K}$ a *membership matrix*, a binary matrix associating each training data point $\boldsymbol{\mathcal{L}}^{(i)}$ to its corresponding dictionary $\{\mathbf{U}^{(1,k)},\dots,\mathbf{U}^{(n,k)}\}$. Since each data point is represented by one dictionary, each row in $\mathbf{M}$ has only one non-zero component. This is enforced by the constraint in (2d). In fact, $\mathbf{M}$ is a clustering matrix which groups data points based on the dictionary they are represented in.

The two-dimensional (2D) variant of (2) was first introduced in [Gurumoorthy et al. 2010] for the compression of facial images. They show the superiority of the 2D dictionary ensemble over vectorized dictionaries [Aharon et al. 2006; Bryt and Elad 2008], since images are intrinsically 2D objects. In other words, representation error diminishes if the dimensionality of the dictionary matches that of data. Hence we expect the learning based MDE presented in (2) to be more effective for high dimensional data sets such as light fields. Our numerical results confirms this (see sections 3.2 and 6).

We denote the training process, i.e. solving (2), using the following function:

$$\left\{\mathbf{U}^{(1,k)},\dots,\mathbf{U}^{(n,k)}\right\}_{k=1}^{K} = \textbf{Train}\left(\left\{\boldsymbol{\mathcal{L}}^{(i)}\right\}_{i=1}^{N_l}, K, \tau_l\right). \tag{3}$$

The input to this function is the training set, the number of dictionaries to be trained, and the sparsity. The output is an MDE.

Equation (2) can be solved efficiently using an iterative approach. First the matrices $\left\{\mathbf{U}^{(1,k)},\dots,\mathbf{U}^{(n,k)}\right\}_{k=1}^{K}$ are initialized with random orthonormal matrices. To do this we compute higher order singular value decomposition (HOSVD) of a tensor with uniform random elements. The elements of the membership matrix $\mathbf{M}$ are initialized with $1/K$. Then the following update rules are applied consecutively (see the supplementary document for the derivation of update rules):

$$\boldsymbol{\mathcal{S}}^{(i,k)} = \boldsymbol{\mathcal{L}}^{(i)} \times_1 \left(\mathbf{U}^{(1,k)}\right)^T \cdots \times_n \left(\mathbf{U}^{(n,k)}\right)^T, \tag{4}$$

$$\mathbf{Z}^{(j,k)} = \sum_{i=1}^{N_l} \mathbf{M}_{i,k} \boldsymbol{\mathcal{L}}_{[j]}^{(i)} \left(\mathbf{U}^{(n,k)} \otimes \cdots \otimes \mathbf{U}^{(j+1,k)} \otimes \mathbf{U}^{(j-1,k)} \otimes \cdots \otimes \mathbf{U}^{(1,k)}\right) \left(\boldsymbol{\mathcal{S}}_{[j]}^{(i,k)}\right)^T, \tag{5}$$

$$\mathbf{U}^{(j,k)} = \mathbf{Z}^{(j,k)} \left(\left(\mathbf{Z}^{(j,k)}\right)^T \mathbf{Z}^{(j,k)}\right)^{-1/2}, \tag{6}$$

$$\mathbf{M}_{i,k} = \left(\sum_{b=1}^{K} e^{\beta(\delta_k^i - \delta_b^i)}\right)^{-1}, \tag{7}$$

where

$$\delta_k^i = \left\| \boldsymbol{\mathcal{L}}^{(i)} - \boldsymbol{\mathcal{S}}^{(i,k)} \times_1 \mathbf{U}^{(1,k)} \cdots \times_n \mathbf{U}^{(n,k)} \right\|_F^2, \tag{8}$$

and $\boldsymbol{\mathcal{L}}_{[j]}^{(i)}$ is the unfolding of $\boldsymbol{\mathcal{L}}^{(i)}$ along the jth dimension (mode). We nullify $(\prod_{j=1}^{n} m_j) - \tau_l$ elements of $\boldsymbol{\mathcal{S}}^{(i,k)}$ with smallest absolute value after each update to $\boldsymbol{\mathcal{S}}^{(i,k)}$. Equations (4)-(7) are computed for all N_l data points and all K dictionaries in the ensemble. To avoid matrix inversion in (6), we observe that this equation is related to the solution of the orthogonal Procrustes problem:

$$\min_{\Omega} \|\mathbf{A}\Omega - \mathbf{B}\|_F \quad s.t. \quad \Omega^T\Omega = \mathbf{I}. \tag{9}$$

If we set $\mathbf{A} = \mathbf{I}$ and $\mathbf{B} = \mathbf{Z}^{(j,k)}$, the solution of (9) is the closest orthogonal matrix to $\mathbf{Z}^{(j,k)}$; i.e. we are seeking an orthogonalization of $\mathbf{U}^{(j,k)}$. To do this, we first compute the SVD of $\mathbf{Z}^{(k,j)}$, i.e. $\mathbf{Z}^{(j,k)} = \mathbf{L\Sigma R}^T$, then $\mathbf{U}^{(j,k)} = \mathbf{LR}^T$ is the closest orthogonal matrix to $\mathbf{Z}^{(j,k)}$, see [Schönemann 1966].

The temperature parameter, β, in equation (7) is initialized with a small value. For a fixed value of β, the sequential updates are repeated until the changes to the matrices $\left\{\mathbf{U}^{(1,k)},\dots,\mathbf{U}^{(n,k)}\right\}_{k=1}^{K}$ are minimal. In practice we put an upper bound for the number of such iterations. Then the temperature parameter is increased and the same update procedure is repeated. The algorithm converges when $\mathbf{M}$ is binary or near binary, e.g. satisfying the following criteria: $\|\mathbf{M} - \lfloor\mathbf{M}\rceil\|_2 < \epsilon$, where ϵ is a small user-defined constant. The operator $\lfloor . \rceil$ rounds its argument to the closest integer value. In Algorithm (1) of the supplementary document we present an implementation of the iterative method described above. Highly scalable parallelization can be achieved thanks to the fact that the data points are small and unordered.

3.2 Pre-clustering

The proposed pre-clustering algorithm groups the data points of the training set into *pre-clusters*. This algorithm ensures that the data points in each pre-cluster are similar in terms of their sparse representation. After pre-clustering, we train an MDE, described in Section 3.1, independently in each pre-cluster. We first describe our motivation for pre-clustering. This is followed by a description of the algorithm and preliminary results for pre-clustering.

For the training algorithm, a larger training set (N_l) and number of dictionaries (K) leads to a more accurate and representative dictionary ensemble for compression of a high-frequency data set (see Section 5 of the supplementary document). Overfitting is not a big issue in practice due to the high nonlinearities in the data. As a result, increasing N_l and K will lead to a more representative dictionary ensemble. On the other hand, the computational cost of the training is directly proportional to N_l and K. We propose a pre-clustering algorithm that significantly accelerates the training process for a predefined N_l and K. In addition to the reduction in computational cost, we show that in some cases the pre-clustering algorithm improves the reconstruction quality.

As discussed in Section 3.1, the learning algorithm for MDE performs a clustering based on sparsity and reconstruction error, see (7) and (8). Moreover, data sets used in graphics often exhibit a global nonlinearity, along with locally low-dimensional structures [Elhamifar and Vidal 2013; Mahajan et al. 2007; Mairal et al. 2009; Miandji et al. 2013, 2015; Schröder et al. 2013; Sloan et al. 2003; Tsai 2015; Zhou et al. 2016]; in other words, these data sets are only locally sparse. Indeed, if we pre-cluster the training set using a sparsity-based criteria, fewer iterations are required for finding the sparsifying dictionaries in each pre-cluster. The pre-clustering stage combined with the clustering performed in the training phase ensure less variations of sparsity in each cluster. As a result, we can reduce the number of dictionaries for each pre-cluster. This significantly reduces the computational cost during training and encoding.

Pre-clustering has been utilized in graphics for compression. For instance, in [Sloan et al. 2003], the authors use an iterative variant of K-Means [Lloyd 1982] for pre-clustering of light transport matrices. Furthermore, K-Means has been used for pre-clustering of surface light fields [Miandji et al. 2013]. According to our discussion above, we argue that K-Means does not accelerate the convergence of our training based approach since the distance metric is the ℓ_2 norm rather than the sparsity promoting ℓ_0 norm. Consider the following two data points (matrices in this case):

$$\mathbf{L}^{(1)} = \begin{bmatrix} 0.01 & 0.1 & 1 \\ 0.01 & 0.1 & 1 \\ 0.01 & 0.1 & 1 \end{bmatrix}, \quad \mathbf{L}^{(2)} = \begin{bmatrix} 0.0001 & 0.001 & 0.01 \\ 0.0001 & 0.001 & 0.01 \\ 0.0001 & 0.001 & 0.01 \end{bmatrix}. \tag{10}$$

If the matrix components are in $[0, 1]$, we see that the ℓ_2 distance, $\|\mathbf{L}^{(1)} - \mathbf{L}^{(2)}\|_F = 1.7234$, is relatively large compared to the maximal distance, which is 3.0. However, both matrices can be described with one coefficient in a suitable basis (note that $\text{rank}(\mathbf{L}^{(1)}) = \text{rank}(\mathbf{L}^{(2)}) = 1$). Therefore, the ℓ_0 distance is minimal while the ℓ_2 distance is relatively large.

We propose a novel pre-clustering algorithm that satisfies the above criteria. In particular, the pre-clustering algorithm groups the data points based on their sparsity and representation error. Moreover, the proposed pre-clustering reduces the effect of noise and outliers in the data. We have summarized our method in Algorithm 1. As a first step for pre-clustering, we reduce the dimensionality of the training set. Formally, we perform the decomposition $\mathbf{F} = \mathbf{A}\mathbf{B}^T$, where $\mathbf{F} \in \mathbb{R}^{N_l \times m_1 m_2 \ldots m_n}$ contains vectorized data points as rows, $\mathbf{A} \in \mathbb{R}^{N_l \times p}$ contains coefficients in each row, and $\mathbf{B} \in \mathbb{R}^{m_1 m_2 \ldots m_n \times p}$ has the basis vectors as columns, where $p \ll \prod_{j=1}^{n} m_j$. An obvious choice for performing this task is Principal Component Analysis (PCA). However, PCA is known to be notoriously sensitive to outliers and noise [Xu et al. 2013], which are extremely common in real-world light field data sets. Instead, we use Coherence Pursuit (CP) [Rahmani and Atia 2017], a Robust-PCA algorithm that is simple and fast, with strong theoretical guarantees. The dimensionality reduction will accelerate the next step in pre-clustering.

Given the coefficients obtained from CP, i.e. $\mathbf{A}$, we use K-Means to compute a small subset of the training set that is representative of the entire set. To do so, we first apply K-Means with N_r clusters on the p-dimensional rows of the coefficient matrix obtained from CP, where N_r is the number of representatives. Then, using the obtained cluster indices, the centroids of the training set, $\left\{\mathcal{L}^{(i)}\right\}_{i=1}^{N_l}$, are calculated. We call these centroids, $\left\{\bar{\mathcal{L}}^{(i)}\right\}_{i=1}^{N_r}$, the *representative set*.

The representative set is then used to train an MDE with C dictionaries, where C is the number of user-defined pre-clusters. Since $N_r \ll N_l$, the computational cost of this stage is negligible compared to the training process applied on the entire training set. As a final stage, the trained MDE, together with the entire training set, are used to produce a membership vector $\mathbf{c} \in \mathbb{R}^{N_l}$, which associates each data point in the training set to a pre-cluster. This stage is also a part of our encoding process which will be described in Section 3.4. To summarize, we use the following function to denote the pre-clustering:

$$\mathbf{c} = \textbf{Precluster}\left(\left\{\mathcal{L}^{(i)}\right\}_{i=1}^{N_l}, C, N_r, p, \epsilon\right), \tag{11}$$

To demonstrate the usefulness of our pre-clustering algorithm in practice, the remainder of this section reports quality and performance results. To facilitate a direct comparison with [Gurumoorthy et al. 2010] and [Miandji et al. 2013], we first use a 2D training set

Algorithm 1 The pre-clustering algorithm implementing $\mathbf{c} = \mathbf{Precluster}\left(\{\boldsymbol{\mathcal{L}}^{(i)}\}_{i=1}^{N_l}, C, N_r, p, \epsilon\right)$

```
Rearrange $\{\boldsymbol{\mathcal{L}}^{(i)}\}_{i=1}^{N_l}$ into $\mathbf{F} \in \mathbb{R}^{N_l \times m_1 m_2 \ldots m_n}$.
Compute $\mathbf{F} = \mathbf{A}\mathbf{B}^T$ using CP with $p$ principal components.
Apply K-Means to rows of $\mathbf{A}$ with $N_r$ clusters.
The representative set $\{\bar{\boldsymbol{\mathcal{L}}}^{(i)}\}_{i=1}^{N_r}$ is computed as the centroids
    of $\{\boldsymbol{\mathcal{L}}^{(i)}\}_{i=1}^{N_l}$ using cluster indices returned by K-Means.
$\left\{\bar{\mathbf{U}}^{(1,c)}, \ldots, \bar{\mathbf{U}}^{(n,c)}\right\}_{c=1}^{C} = \mathbf{Train}\left(\{\bar{\boldsymbol{\mathcal{L}}}^{(i)}\}_{i=1}^{N_r}, C, \tau_l, \epsilon\right)$.
$\mathbf{c} = \mathbf{Test}\left(\{\boldsymbol{\mathcal{L}}^{(i)}\}_{i=1}^{N_l}, \left\{\bar{\mathbf{U}}^{(1,c)}, \ldots, \bar{\mathbf{U}}^{(n,c)}\right\}_{c=1}^{C}, \tau_l, \epsilon\right)$.
```

Table 1. The effect of pre-clustering for a training set of 2D data points. For Algorithm 1 we use $N_r = 2048$, $p = 8$, $\tau_l = \tau_t = 8$, and $\epsilon = 10^{-5}$.

				time (minute:sec)	
	C	K	PSNR	pre-cluster	train
SC1 (no pre-clustering)	1	32	65.00	-	62:31
SC2 (Algorithm 1)	8	4	64.65	01:24	6:51
SC3 (Algorithm 1)	16	8	66.66	02:28	14:12
SC4 ([Miandji et al. 2013])	16	8	63.75	00:32	16:27

Table 2. Timing results for the proposed pre-clustering

	time (in seconds)			
	CP	K-Means	Train	Test
scenario 2	15	12	56	1
scenario 3	15	13	118	2

with 65536 data points of dimensionality 32 × 16 representing a surface light field data set[1]. The following scenarios are considered:

- Scenario 1 (SC1): We do not perform pre-clustering. The training is performed on all the data points in the training set. Since we use 2D data points, this is essentially the compression method of [Gurumoorthy et al. 2010].
- Scenario (SC2): We perform pre-clustering and training as described in algorithms 1 and 2. This scenario represents our method. The values for C and K are chosen such that the total number of dictionaries, CK, is equal to that of SC1.
- Scenario (SC3): This is the same as SC2 but we use $2C$ pre-clusters and $2K$ dictionaries per pre-cluster.
- Scenario (SC4): We use the same parameters as SC3 applied on [Miandji et al. 2013], which uses K-Means for pre-clustering.

Performance and image quality results (using PSNR) are reported in Table 1. In Table 2, we break down the timing results of the pre-clustering method according to the steps of Algorithm 1.

Comparing SC1 and SC2 we see that pre-clustering makes the training phase more than an order of magnitude faster. This is particularly important because the time for pre-clustering is negligible compared to training. However, there is a slight reduction in quality. This is mainly associated to the approximations done in the pre-clustering stage. In SC3, we double the number of clusters and dictionaries per cluster, i.e. we have 4 times more dictionaries than SC2. One expects that the compression stage should become 4 times slower. But we see that SC3 is only about 2 times slower. This is because more clusters are present in SC3, so that the inter-cluster data points are more similar in terms of sparsity and reconstruction error. Hence the training requires fewer iterations. Apart from this, we see that SC3 achieves higher quality compared to SC1, while there is still a large improvement in terms of speed (about 4.4 times speedup). This demonstrates the significance of pre-clustering in terms of reconstruction quality when compared to the image compression method described in [Gurumoorthy et al. 2010]. Comparing SC3 and SC4, our method (SC3) achieves a much higher PSNR, while the overall training time is similar. In addition, although the number of dictionaries are equal, our method makes training phase faster. This is because of the proposed sparsity-based metric for pre-clustering compared to the ℓ_2 metric used in [Miandji et al. 2013].

Table 3. The effect of pre-clustering for a training set of 5D data points. For Algorithm 1 we use $N_r = 1000$, $p = 64$, $\tau_l = 10$, $\tau_t = 128$, and $\epsilon = 10^{-4}$.

				time (minute:sec)	
	C	K	PSNR	pre-cluster	train
SC1 (no pre-clustering)	1	16	31.77	-	164:21
SC2 (Algorithm 1)	8	2	31.82	02:49	24:25

To show the effectiveness of pre-clustering for high dimensional data sets, we report results for a light field data set in Table 3. We use the *Dragon and Bunnies* data set used in [Marwah et al. 2013] and create 55440 non-overlapping data points of dimensionality $m_1 = 3$, $m_2 = 3$, $m_3 = 5$, $m_4 = 5$, and $m_5 = 3$, where m_1 and m_2 represents the spatial, m_2 and m_3 the angular, and m_5 the spectral dimensions. We see that not only the training time decreases significantly, the PSNR slightly increases.

3.3 The Aggregate MDE

Algorithm 2 summarizes the training process combined with the pre-clustering stage. After performing the pre-clustering, an MDE is trained for each pre-cluster. Hence the total number of dictionaries will be CK. We then take the union of all the ensembles in different pre-clusters, leading to one ensemble. Formally,

$$\boldsymbol{\Psi} = \bigcup_{c=1}^{C} \left\{\mathbf{U}^{(1,k,c)}, \ldots, \mathbf{U}^{(n,k,c)}\right\}_{k=1}^{K} = \left\{\mathbf{U}^{(1,k)}, \ldots, \mathbf{U}^{(n,k)}\right\}_{k=1}^{CK} \tag{12}$$

We call the final ensemble $\boldsymbol{\Psi}$ an *aggregate* ensemble, or AMDE. In Section 5, we will discuss the effect of different parameters used for the pre-clustered training on the reconstruction quality.

The training of an AMDE is performed once for each application, as long as the dimension of data points in the testing set matches that of AMDE. Even if the dimension does not match, one can change the extraction of data points from the testing set (Section 2) to match AMDE. For instance, assume that we have an AMDE trained on light fields with 4 × 4 views. For compression of a testing set with 16 × 16 views, it is only required to create 4 × 4 patches on the angular domain and use the existing AMDE; note that, in the case of non-divisible patch sizes, one we can create overlapping patches.

[1] Specifically, the glossy plane of the cornell box scene in Figure 1 of [Miandji et al. 2013] was used. We observed similar results for different data sets.

Algorithm 2 Pre-clustered Training: this is performed only once for each application

```
Input: The training set $\left\{\mathcal{L}^{(i)}\right\}_{i=1}^{N_l}$, 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 $\epsilon$.
Output: A dictionary ensemble $\left\{\mathbf{U}^{(1,k)},\dots,\mathbf{U}^{(n,k)}\right\}_{k=1}^{CK}$
$\mathbf{c}$ = Precluster $\left(\left\{\mathcal{L}^{(i)}\right\}_{i=1}^{N_l}, C, N_r, p, \epsilon\right)$ using Algorithm 1
for $c = 1 \dots C$ do
    $\left\{\mathcal{X}^{(j)}\right\} \subset \left\{\mathcal{L}^{(i)}\right\}_{i=1}^{N_l}$ such that $\mathbf{c}_j == c$
    $\left\{\mathbf{U}^{(1,k,c)},\dots,\mathbf{U}^{(n,k,c)}\right\}_{k=1}^{K}$ = Train $\left(\left\{\mathcal{X}^{(j)}\right\}, K, \tau_l, \epsilon\right)$
end for
Compute aggregated ensemble $\mathbf{\Psi}$ using (12)
```

3.4 Encoding

The input to encoding is the testing set $\left\{\mathcal{T}^{(i)}\right\}_{i=1}^{N_t}$, i.e. the data set we would like to compress, as well as the AMDE. Because each $\mathcal{T}^{(i)}$ should be represented using one dictionary in AMDE, we need to find the *best* dictionary for each data point. Since our goal is compression, we choose a dictionary that leads to the most sparse coefficients with the least error. We can calculate the coefficients by projecting the data point $\mathcal{T}^{(i)}$ onto the chosen dictionary (see Eq. (4)). Let us represent this procedure by the following function:

$$\left[\left\{\mathcal{S}^{(i)}\right\}_{i=1}^{N_t}, \mathbf{m}, \mathbf{z}\right] = \mathbf{Test}\left(\left\{\mathcal{T}^{(i)}\right\}_{i=1}^{N_t}, \mathbf{\Psi}, \tau_t, \epsilon\right). \tag{13}$$

The function **Test**(.) takes as input a data point in the testing set, AMDE, and thresholds for sparsity and error, respectively. Note that the testing sparsity, τ_t, is an upper bound for sparsity; i.e. the number of nonzero coefficients is typically smaller than this upper bound. The output of the function is a collection of sparse coefficient tensors, $\mathcal{S}^{(i)}$, and a vector $\mathbf{m} \in \mathbb{R}^{N_t}$ that associates each data point with a dictionary in AMDE. The sparsity of each data point is stored in $\mathbf{z} \in \mathbb{R}^{N_t}$. We call the index of the best dictionary for data point $\mathcal{T}^{(i)}$, denoted $\mathbf{m}_i \in [1 \dots CK]$, a *membership index*. The vector $\mathbf{m}$ is called a *membership vector*, which describes the clustering of the testing set based on a given AMDE. Algorithm (2) in the supplementary document implements (13) using a greedy approach. This stage is significantly faster than the training stage, see Table 2.

As an additional step, we perform Huffman coding [Huffman 1952] prior to saving to disk. This is specifically useful in scenarios where there are limited resources for transmission or storage of the compressed data. The vectors $\mathbf{m}$ and $\mathbf{z}$ can be directly coded, while the coefficients need to be discretized first. To do this, we first use the Fisher Natural Breaks Classification (FNBC) [Fisher 1958] to determine a set of partitions for the coefficient values. The FNBC is indeed the one dimensional variant of the K-Means algorithm. The mean of each partition is then used for defining the codebook for Huffman coding. The number of partitions is a user-defined parameter, denoted q. In Section 5 of the supplementary document, we will discuss the effect of different parameters used in the encoding stage on the reconstruction quality.

3.5 Decoding

Reconstructing a data point in the testing set is done by evaluating (1) using the aggregate ensemble and the coefficients calculated in Section 3.4, i.e. we have

$$\mathcal{T}^{(i)} = \mathcal{S}^{(i)} \times_1 \mathbf{U}^{(1,\mathbf{m}_i)} \cdots \times_n \mathbf{U}^{(n,\mathbf{m}_i)}. \tag{14}$$

Since each data point includes all the angular and spectral information, random access into the compressed data point during reconstruction is of utmost importance for real-time rendering of a light field. This is because only a single direction per spatial location is needed to be reconstructed from the vantage point of the camera. Even when view interpolation is required, we reconstruct a very small fraction of the views per spatial location. Hence, we also propose a method for reconstructing a *single* element of a data point in the testing set. The element at location $x_1, x_2, \dots, x_n$ of a data point $\mathcal{T}^{(i)} \in \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_n}$ is calculated as follows

$$\mathcal{T}^{(i)}_{x_1,x_2,\dots,x_n} = \sum_{z=1}^{\mathbf{z}_i} \mathcal{S}^{(i)}_{l_1^z,l_2^z,\dots,l_n^z} \mathbf{U}^{(1,\mathbf{m}_i)}_{x_1,l_1^z} \mathbf{U}^{(2,\mathbf{m}_i)}_{x_2,l_2^z} \cdots \mathbf{U}^{(n,\mathbf{m}_i)}_{x_n,l_n^z}, \tag{15}$$

where the index tuple $(l_1^z, l_2^z, \dots, l_n^z)$ defines the location of z-th nonzero element in $\mathcal{S}^{(i)}$. The computational complexity of recovering a single element in $\mathcal{T}^{(i)}$ is $O(n\mathbf{z}_i)$, which is competitive to that of precomputed radiance transfer for diffuse surfaces [Liu et al. 2004; Sloan et al. 2002], where a dot product of two vectors with a user-defined size is calculated.

For a GPU-based implementation, the nonzero elements of $\mathcal{S}^{(i)}$, together with their corresponding locations, are stored in textures. The sparsity and membership index vectors, $\mathbf{z}$ and $\mathbf{m}$, as well as the AMDE are also uploaded as textures. For 5D light fields, the aforementioned textures are uploaded only once. However, for the light-field video, the textures are updated every f frames, where f is the number of frames contained in each data point. We have used multi-threading and pixel buffer objects (PBO) in order to accelerate the data transfer between the CPU and GPU. For rendering with a resolution of 2048×1088 pixels, and using the *Painter* light field video data set (see Fig. 6), where $\tau_t = 1024$, we achieve 40 frames per second on an Nvidia Titan Xp GPU, see the supplementary video.

4 COMPRESSIVE POINT SAMPLING

For decades, the Shannon-Nyquist theorem [Nyquist 1928; Shannon 1949] had been the standard for sampling signals. This theorem states that any function with no frequencies higher than f can be exactly recovered with equally spaced samples at a rate larger than $2f$ (the Nyquist rate). Compressed Sensing (CS) [Donoho 2006] sparked a paradigm shift in the field of signal processing. Seminal work in the field [Candes et al. 2006; Candes and Tao 2005, 2006] show that a signal of length m with at most $\tau \leq m$ nonzero elements (i.e. sparsity) can be recovered with overwhelming probability using Gaussian or Bernoulli sampling, provided that $s \geq C\tau \ln(m/\tau)$, where s is the number of samples and C is a universal constant. Note that the number of samples is linearly dependent on the sparsity and only logarithmically on the signal length. For an introduction to the field of CS, we refer the reader to [Candes and Wakin 2008; Elad 2010].

It has been shown that an ensemble of 2D dictionaries can be used for reconstructing incomplete measurements of visual data using a compressed sensing framework [Miandji et al. 2015]. In this section, we generalize this framework for arbitrarily high dimensional data sets using an AMDE, which is particularly useful for designing coded-aperture light field cameras [Ashok and Neifeld 2010; Babacan et al. 2012; Marwah et al. 2013; Miandji et al. 2018]. Our main contribution in this regard is a theoretical analysis for the uniqueness of the solution obtained from our proposed *n*D CS framework (see Section 4.1). In particular, we describe conditions under which one can *exactly* recover an incomplete light field that is sparse under an AMDE. These conditions can be used as guidelines for designing efficient light field cameras.

Let us formulate the problem of reconstructing a signal from its random measurements. Consider a one dimensional data point, $\mathbf{p} \in \mathbb{R}^{m}$, and define a sampling operator, $\mathbf{\Phi} \in \mathbb{R}^{s \times m}$, $s < m$, that samples $\mathbf{p}$, i.e. $\mathbf{y} = \mathbf{\Phi}\mathbf{p}$, where the number of linear samples is s. The matrix $\mathbf{\Phi}$ is known as a *measurement* or *sensing* matrix. On the other hand, if $\mathbf{p}$ is τ-sparse, then it can be represented using a dictionary as $\mathbf{p} = \mathbf{D}\mathbf{s}$, where $\|\mathbf{s}\|_0 = \tau$. Therefore the measurement model becomes $\mathbf{y} = \mathbf{\Phi}\mathbf{D}\mathbf{s}$, i.e. an underdetermined system of linear equations for the unknowns $\mathbf{s}$ with an infinite number of solutions. Since $\mathbf{s}$ is sparse, one can reduce the number of solutions or even obtain a unique one by solving

$$\min_{\mathbf{s}} \|\mathbf{s}\|_0 \quad \text{s.t.} \quad \|\mathbf{y} - \mathbf{\Phi}\mathbf{D}\mathbf{s}\|_2^2 \leq \epsilon, \tag{16}$$

where ϵ is related to noise power and possible representation error introduced by the dictionary. Once the coefficients are recovered, the data point is reconstructed by computing $\mathbf{D}\mathbf{s}$. As we will see later, the higher the sparsity, the lower the number of measurements for exact recovery.

The recovery problem (16) for a 2D dictionary ensemble can be formulated as follows [Miandji et al. 2015]:

$$\min_{\mathbf{s}} \|\mathbf{s}\|_0 \quad \text{s.t.} \quad \left\|\mathbf{y} - \mathbf{\Phi}(\mathbf{U}^{(2,k)} \otimes \mathbf{U}^{(1,k)})\mathbf{s}\right\|_2^2 \leq \epsilon, \tag{17}$$

where $\mathbf{y} = \mathbf{\Phi}\text{vec}(\mathbf{P})$, for a 2D data point $\mathbf{P} \in \mathbb{R}^{m_1 \times m_2}$; moreover, the sensing matrix acts on the vectorized data point, hence $\mathbf{\Phi} \in \mathbb{R}^{s \times m_1 m_2}$. Since no information on the underlying data point is available, one has to solve (17) for all the dictionaries in the ensemble and choose the most sparse coefficient vector $\mathbf{s}$ as the solution.
It is straightforward to extend (17) for higher dimensions:

$$\min_{\mathbf{s}} \|\mathbf{s}\|_0 \quad \text{s.t.} \quad \left\|\mathbf{y} - \mathbf{\Phi}\left(\mathbf{U}^{(n,k)} \otimes \mathbf{U}^{(n-1,k)} \dots \otimes \mathbf{U}^{(1,k)}\right)\mathbf{s}\right\|_2^2 \leq \epsilon, \tag{18}$$

where $\mathbf{\Phi} \in \mathbb{R}^{s \times \prod_{j=1}^{n} m_j}$, and $\mathbf{y} = \mathbf{\Phi}\text{vec}(\boldsymbol{\mathcal{P}})$, for a data point $\boldsymbol{\mathcal{P}} \in \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_n}$. When it is required to sample different modes independently, equation (18) can be reformulated as

$$\min_{\mathbf{s}} \|\mathbf{s}\|_0 \quad \text{s.t.}$$
$$\left\|\mathbf{y} - \left(\mathbf{\Phi}^{(n)}\mathbf{U}^{(n,k)} \otimes \mathbf{\Phi}^{(n-1)}\mathbf{U}^{(n-1,k)} \dots \otimes \mathbf{\Phi}^{(1)}\mathbf{U}^{(1,k)}\right)\mathbf{s}\right\|_2^2, \tag{19}$$

where $\mathbf{y} = \text{vec}\left(\boldsymbol{\mathcal{P}} \times_1 \mathbf{\Phi}^{(1)} \times_2 \mathbf{\Phi}^{(2)} \cdots \times_n \mathbf{\Phi}^{(n)}\right)$, and $\mathbf{\Phi}^{(j)} \in \mathbb{R}^{s_j \times m_j}$ is a sampling matrix for mode $j \in \{1, 2, \dots, n\}$. Therefore we can have a variable number of samples, s_j, for each mode. Equation (19) is particularly useful for designing new compressive light field cameras. Since each mode is sampled independently, this produces new possibilities for designing multi-sensor and single-sensor coded-aperture cameras.

In this paper we consider point sampling operators. The sensing matrix corresponding to a point sampling operator can be defined as follows. Given an index set $\{1, \dots, m\}$, associate with each index a vector $\mathbf{e}^i \in \mathbb{R}^m$, where $(\mathbf{e}^i)_j = 0, \forall i \neq j$, and $(\mathbf{e}^i)_j = 1$, if $i = j$. Let $\{k_1, \dots, k_m\}$ be a uniform random permutation of the set $\{1, \dots, m\}$.

Definition 1. *The sensing matrix* $\mathbf{I}_{\Lambda,.} \in \mathbb{R}^{s \times m}$, $\Lambda = \{k_1, \dots, k_s\}$, *called a spike ensemble, is constructed by stacking* $\{\mathbf{e}_i\}_{i=k_1}^{k_s}$ *as rows of a matrix.*

4.1 Uniqueness Analysis

In this section, we will present theoretical results for exact recovery of a sparse data point $\boldsymbol{\mathcal{P}} \in \mathbb{R}^{m_1 \times \cdots \times m_n}$ under an AMDE and using a point sampling operator $\mathbf{I}_{\Lambda,.}$. To define exact recovery, we need the following definition

Definition 2. *The support of a sparse data point* $\mathbf{p}$ *under a dictionary* $\mathbf{D}$, *denoted* $\text{supp}(\mathbf{p})_{\mathbf{D}}$, *is an index set holding the location of nonzero values in* $\mathbf{x}$, *where* $\mathbf{p} = \mathbf{D}\mathbf{x}$.

By exact recovery, we mean the correct identification of the support. For noiseless data points, once we have the support, we can exactly recover the nonzero values. Moreover, when the data point is contaminated by noise but we know the location of the nonzero values, the error in the reconstructed data point is proportional to the noise power [Ben-Haim et al. 2010; Miandji et al. 2017].

In Theorem 1 we show that given an AMDE and a spike ensemble, it is possible to exactly recover a sufficiently sparse light field data point with high probability. Before presenting the theorem, let us describe the tools used here. The function $\text{orth}(\mathbf{X})$ defines orthogonalization of the columns of $\mathbf{X}$, using e.g. QR decomposition [Horn and Johnson 2012]. Moreover,

Definition 3. *The mutual coherence of a matrix* $\mathbf{X} \in \mathbb{R}^{m \times n}$ *is*

$$\mu(\mathbf{X}) = \max_{1 \leq i \neq j \leq n} \frac{|\mathbf{X}_{.,i}^T \mathbf{X}_{.,j}|}{\|\mathbf{X}_{.,i}\|_2 \|\mathbf{X}_{.,j}\|_2}. \tag{20}$$

We can now state the main result.

Theorem 1. *Define* $m = \prod_{j=1}^{n} m_j$. *Assume that the data point* $\mathbf{p} = \text{vec}\left(\boldsymbol{\mathcal{P}} \in \mathbb{R}^{m_1 \times \cdots \times m_n}\right)$ *is at most* τ*-sparse under all dictionaries in* $\mathbf{\Psi} = \left\{(\mathbf{U}^{(n,k)} \otimes \cdots \otimes \mathbf{U}^{(1,k)})\right\}_{k=1}^{CK}$, *where*

$$\tau \leq \frac{1}{2}\left(1 + \frac{1}{\mu\left(\mathbf{I}_{\Lambda,.}\mathbf{A}\right)}\right), \ \forall \mathbf{A} \in \mathbf{\Psi} \tag{21}$$

is satisfied for a fixed spike ensemble $\mathbf{I}_{\Lambda,.} \in \mathbb{R}^{s \times m}$. *If*

$$s \geq -m\alpha \log\left(\frac{1}{r}\right), \tag{22}$$

where

$$\alpha = \max_{j=1,\dots,m} \left\| \mathrm{orth}\left([\mathbf{A}_{.,I}, \mathbf{B}_{.,J}]\right)_{j,.} \right\|_2^2, \ \forall \mathbf{A}, \forall \mathbf{B} \in \Psi, \ \mathbf{A} \neq \mathbf{B}, \quad (23)$$

$$r = \mathrm{rank}\left([\mathbf{A}_{.,I}, \mathbf{B}_{.,J}]\right), \quad (24)$$

$$I = \mathrm{supp}(\mathbf{p})_{\mathbf{A}}, \ \mathbf{A} \in \Psi, \quad (25)$$

$$J = \mathrm{supp}(\mathbf{p})_{\mathbf{B}}, \ \mathbf{B} \in \Psi, \quad (26)$$

then with probability at least

$$1 - r\,(0.3679)^{\frac{s}{m\alpha}}, \quad (27)$$

the sparsest solution of (18)*, among CK solutions, is unique, i.e. it is the global minimizer of* (18) *for all* $k = 1, \dots, CK$.

The proof is presented in the supplementary document accompanying this paper. Note that (21) ensures there exists a unique solution for each dictionary [Gribonval and Nielsen 2003]. Moreover, equation (21) can be replaced with similar conditions on the dictionary, see e.g. [Candès et al. 2006; Donoho and Elad 2003; Tropp 2004]. Hence (27) describes the probability that among CK solutions of (18), the sparsest solution is the unique global minimizer.

Since we can rewrite (19) as

$$\min_{\mathbf{s}} \|\mathbf{s}\|_0 \quad \text{s.t.} \quad \left\| \mathbf{y} - \left(\mathbf{\Phi}^{(n)} \otimes \mathbf{\Phi}^{(n-1)} \cdots \otimes \mathbf{\Phi}^{(1)}\right) \left(\mathbf{U}^{(n,k)} \otimes \mathbf{U}^{(n-1,k)} \cdots \otimes \mathbf{U}^{(1,k)}\right) \mathbf{s} \right\|_2^2, \quad (28)$$

Theorem 1 can also be applied to evaluate the uniqueness of the solution of (19).

To demonstrate the usefulness of Theorem 1, we report the probability of success, i.e. equation (27), given a randomly generated data point and an AMDE with $C = 4$ and $K = 8$ trained on the *Dragon and Bunnies* data set [Marwah et al. 2013]. The data point dimensionality was set to $m_1 = 9$, $m_2 = 9$ (spatial), $m_3 = 5$, $m_4 = 5$ (angular), and $m_5 = 3$ (spectral). Since Theorem 1 only requires the support of the data point to be known, we construct the two support sets I and J uniformly at random. We perform 10^5 trials, where in each trial we create a new random spike ensemble, as well as the random support sets. The effect of the number of samples and sparsity on the probability of success is shown in Fig. 2. We do not take into account (21) and assume that the data point is sparse in each dictionary.

Theorem 1 can be used to guide the design of compressive light field cameras. For instance, single sensor light field cameras have been proposed by placing a translucent random mask in front of the sensor [Marwah et al. 2013; Miandji et al. 2018]. The compressive light field camera is modeled using a measurement matrix $\mathbf{\Phi}$, as in (16) for 1D light fields or (18) for nD light fields. Changing the design of the camera, e.g. by changing the placement of the mask, leads to a different measurement matrix. Indeed if we know the properties of a "*good*" measurement matrix, as defined in Theorem 1, hardware implementations can be more cost effective and efficient. We have left the theoretical analysis of existing light field capture systems using Theorem 1 for future work. Moreover, design of new multidimensional compressive light field cameras can be realized using the proposed nD CS framework and its theoretical analysis.

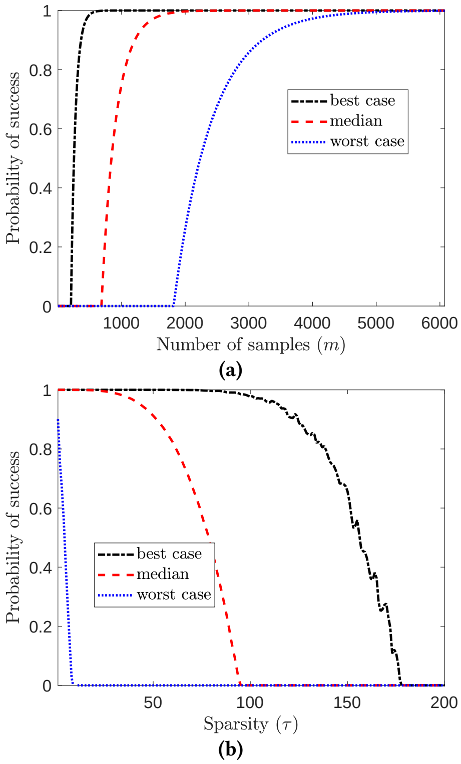


Fig. 2. Simulation results for Theorem 1. We use an AMDE trained on the *Dragon and Bunnies* synthetic light field data set.

5 PARAMETER ANALYSIS

In this section we analyze the parameters used in our compression algorithm described in Section 3. For the training set, the *Dragon and Bunnies* synthetic light field data set of [Marwah et al. 2013] was used. This data set is a collection of 25 images taken on a 5 × 5 grid. We create non-overlapping data points of size $m_1 = 5$, $m_2 = 5$, $m_3 = 5$, $m_4 = 5$, and $m_5 = 3$.

The results of parameter analysis are summarized in Fig. 3. For each parameter, we fix other parameters and plot the reconstruction quality using PSNR. Table 3 lists all the values for fixed parameters in each test scenario. These values are set to emphasize the effect of each parameter on the reconstruction quality. For instance, N_r and p are related to pre-clustering, and therefore we use a small number of dictionaries, $K = 2$, and a large number of pre-clusters $C = 8$. Moreover, the PSNR is calculated as the average of 8 trials of our compression algorithm. This is due to random initializations in K-Means, CP, and the training algorithm.

Figures 3a and 3b show that increasing C and K will improve quality. We see a sharp increase in quality for small values of C and K. Thereafter, the increase in quality is gradual (almost linear). Since these two parameters only affect the training (in terms of computational complexity), we set them to the largest number possible, as long as training can be done in a reasonable time. To fine-tune the choice of these parameters, one can produce plots such as 3a and 3b for a small training set chosen at random from the main training set.

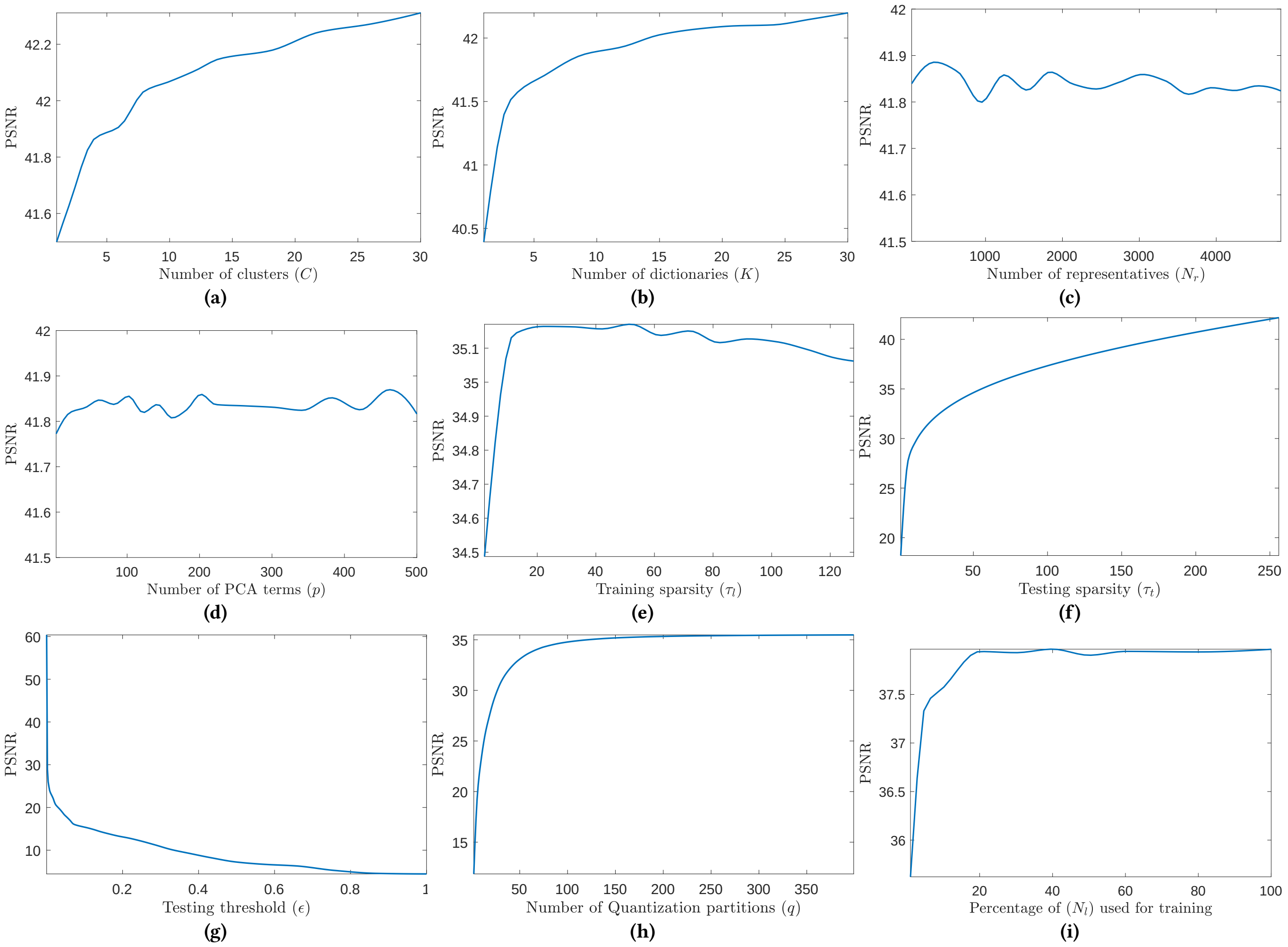


Fig. 3. Results for parameter analysis. Table 4 lists all the configurations of parameters used here.

Table 4. Values for parameters used in various plots of Figure 3.

parameter	C	K	N_r	p	τ_l	τ_t	q	ϵ	N_l
Fig. 3a	-	2	3000	64	20	256	-	10^{-6}	100%
Fig. 3b	1	-	3000	32	20	256	-	10^{-6}	100%
Fig. 3c	8	2	-	128	20	256	-	10^{-6}	100%
Fig. 3d	8	2	3000	-	20	256	-	10^{-6}	100%
Fig. 3e	1	4	3000	32	-	64	-	10^{-6}	100%
Fig. 3f	4	8	3000	32	64	-	-	10^{-6}	100%
Fig. 3g	4	8	3000	32	20	1875	-	-	100%
Fig. 3h	4	8	3000	32	20	64	-	10^{-6}	100%
Fig. 3i	2	4	3000	32	20	128	-	10^{-6}	-

In this way, we can set these parameters to the smallest value such that the image quality or the training time becomes acceptable.

As it can be seen in figures 3c and 3d, the parameters N_r and p have negligible effect on reconstruction quality. This shows the effectiveness of the pre-clustering algorithm since both of these parameters reduce computation time drastically. Moreover, we see that as we increase N_r, the PSNR stabilizes more. This is because the dictionary training stage of Algorithm 1 has more data points as the training set. The PSNR variations in Fig. 3d are due to random initialization in our implementation of CP and K-Means.

In Fig. 3e, we plot reconstruction quality with respect to training sparsity τ_l. The testing sparsity was fixed, $\tau_t = 64$. An interesting property of this plot is that it shows the intrinsic sparsity of the data set used for training. The rapid increase in PSNR continues until we have about $\tau_l = 10$, and linearly afterwards. This implies that the light field data set used here is approximately 10-sparse. Note that the sparsity of a data set is indeed dependent on the dictionary. Therefore it is not possible to estimate this value prior to training. Moreover, we see that PSNR declines when $\tau_l > \tau_t = 64$. Hence, the training sparsity should be set to the smallest value possible.

In Fig. 3f, we analyze the effect of testing sparsity. This figure reveals another interesting property. Even though $\tau_l = 64$, we see that the sharp increase in quality stops at around $\tau_t = 10$, which is the actual sparsity of the data set as discussed earlier. Intuitively one expects the reconstruction quality to increase until $\tau_t = 64$ since this is the sparsity value we used for training. This shows that even if we set the training sparsity to a value that is different from the actual sparsity of the data set, we will still see results that obey the true sparsity value. Figure 3g shows a similar trend as 3f, which is expected since both parameters work together to define the number of coefficients. The effect of the number of quantization partitions used for Huffman coding is shown in Figure 3h. Indeed increasing this value beyond a threshold does not contribute to quality, but rather reduces the compression ratio. Finally, we also plot the effect of training set size on the quality of reconstruction in Figure 3i. It can be seen that the reconstruction quality does not improve with more than 20% of the data points in the training set. The subset of the data points used for training were selected uniformly at random.

The following remark will present guidelines for setting parameter values in order to obtain a desired compression ratio or quality.

Remark 1. *Most of the parameters are for the training. For τ_l, we produce a plot of τ_l such as the one in Fig. 3e on a subset of the training set. The plot clearly shows a suitable value for τ_l (e.g. $\tau_l \approx 10$ in Fig. 3e). Moreover, the parameters C and K only affect the training time and reconstruction quality. Because the training is performed once, we use largest values possible, as long as the training time is acceptable. N_r and p do not have a significant effect on the quality of the ensemble. Given the dictionary ensemble, the main parameter used for encoding is ϵ, see Eq.* (13). *Since τ_t is an upper bound for sparsity, we can ignore this parameter by setting $\tau_t = \prod_{j=1}^{n} m_j$. However, in practice we set τ_t to a smaller value since some data points may not be compressible.*

6 RESULTS

In this section, we describe our results for compression of natural light fields (Section 6.1) and natural light field videos (Section 6.2). Results for compression of synthetic data sets are included in the supplementary document accompanying this paper. Our method (AMDE) is compared with K-SVD [Aharon et al. 2006], multidimensional Discrete Cosine Transform (*n*D DCT), higher order singular value decomposition (HOSVD), and CDF 9/7 wavelets [Cohen et al. 1992]. The K-SVD algorithm has been used previously for compression and compressed sensing of light fields [Marwah et al. 2013] and BTF compression [Roland and Reinhard 2009]. The *n*D DCT algorithm has been used as a sparsifying dictionary in [Kamal et al. 2016] for compressive light field photography. Moreover, HOSVD, and other variants of tensor decomposition, have widely been used in graphics for compression [Ballester-Ripoll and Pajarola 2016; Bilgili et al. 2011; Pajarola et al. 2013; Wang et al. 2005]. The JPEG2000 standard [Taubman and Marcellin 2013] uses CDF 9/7 wavelets for sparse representation of images.

We use half-precision for storing input data sets, coefficients, and the dictionaries used in all the methods above. After encoding a data set using our method, we calculate the storage cost and set the parameters for other methods so that the same storage cost is achieved. We then measure the visual quality after decoding. The coefficients are stored as value-location pairs, where we only store nonzero coefficients. The storage cost is calculated according to this model. Since we are interested in comparing the effectiveness of different dictionaries, the results reported here do not utilize quantization of the coefficients and Huffman coding, as described in Section 3.4. Given that these two steps do not have a substantial effect on quality (see Section (5) of the supplementary document), the compression ratio can be significantly improved. Note that if Huffman coding is performed, there is no need to store the location of nonzero coefficients. For an overview of techniques for light field coding see [Wu et al. 2017a].

The K-SVD algorithm trains a dictionary $\mathbf{D} \in \mathbb{R}^{(\prod_{j=1}^{n} m_j)\times\kappa(\prod_{j=1}^{n} m_j)}$ on one dimensional data points, where $\kappa > 1$ defines the overcompleteness. The encoding stage calculates sparse coefficients using Orthogonal Matching Pursuit (OMP) [Pati et al. 1993] by solving

$$\hat{\mathbf{x}}^{(i)} = \min_{\mathbf{x}} \|\mathbf{t}^{(i)} - \mathbf{D}\mathbf{x}\|_2^2 \quad \text{s.t.} \quad \|\mathbf{x}\|_0 \le \tau, \tag{29}$$

for all data points $\mathbf{t}^{(i)} = \mathrm{vec}(\boldsymbol{\mathcal{T}}^{(i)})$. Since the size of vectorized data points is very large, utilizing a greedy and fast algorithm such as OMP is essential. Moreover, OMP has been shown to have strong theoretical guarantees [Chang and Wu 2014; Miandji et al. 2017; Tropp 2004]. Given the sparse coefficient vectors, decoding is achieved by calculating $\hat{\mathbf{t}}^{(i)} = \mathbf{D}\hat{\mathbf{x}}^{(i)}$. For HOSVD, we calculate the coefficient tensor and the multidimensional dictionary for each patch, see (1). The elements of the obtained coefficient tensor with smallest absolute value are nullified until the target storage cost is achieved. Note that in this method a multidimensional dictionary have to be stored for each data point. Hence the storage cost for each data point is $\tau + \prod_{j=1}^{n}(m_j)^2$. The dictionary for CDF 9/7 is analytical, hence there is no need to store it. However, this dictionary can only be used for 2D data points. Therefore we create patches from light field views. Each patch is projected onto CDF 9/7 and the coefficients are nullified to achieve the target storage cost. The patch size and the number of levels for CDF 9/7 is set for best image quality. The *n*D DCT algorithm does not have any parameters and we use the same nullification procedure that is used for other methods.

To measure reconstruction quality, we use Peak Signal to Noise Ratio (PSNR), and SNR, defined respectively as

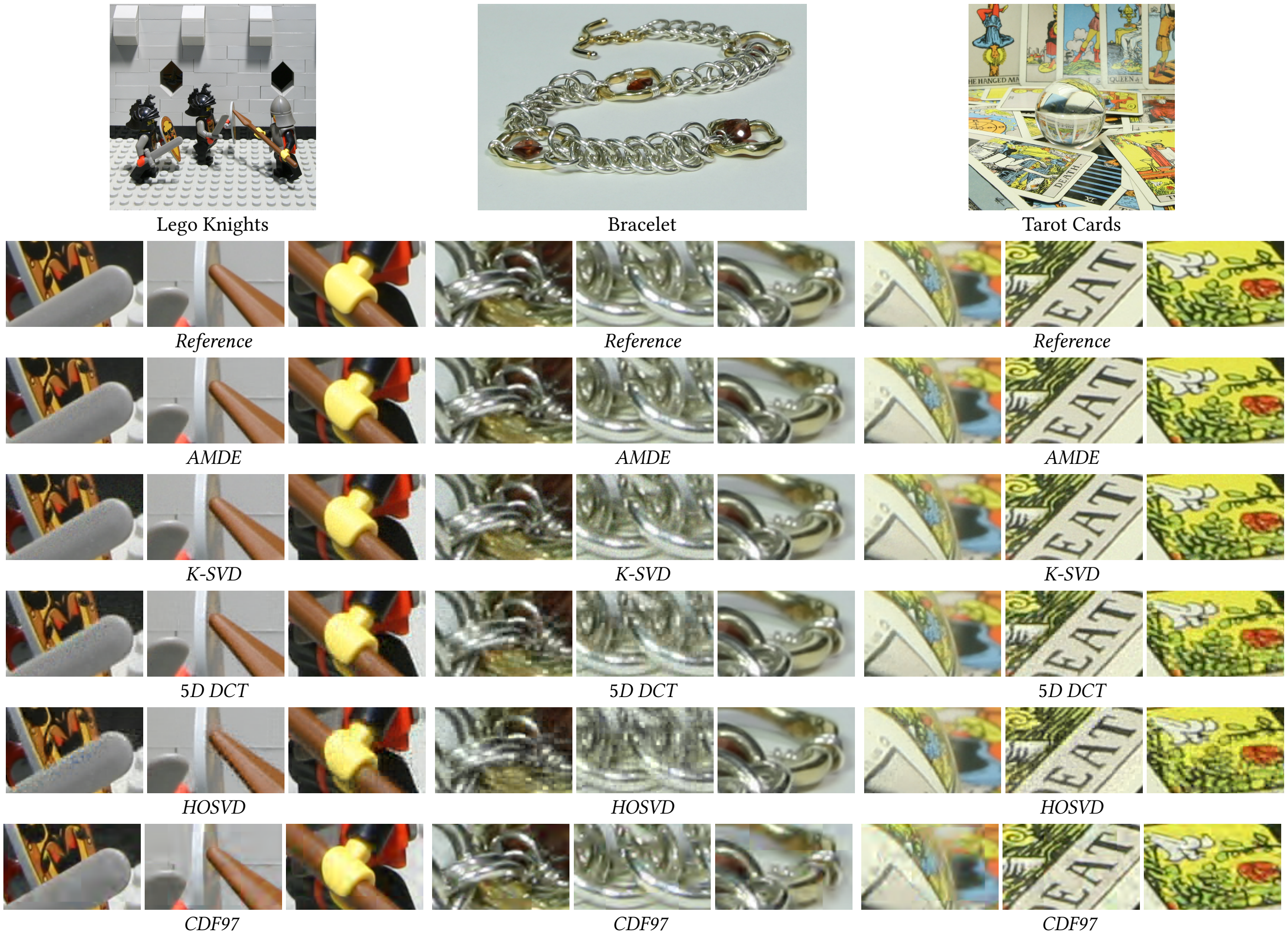
$$\mathrm{PSNR} = 10\,\log_{10}\left(\frac{l^2}{\eta}\right), \quad \mathrm{SNR} = 10\,\log_{10}\left(\frac{1}{N_t\eta}\sum_{i}^{N_t}\left\|\boldsymbol{\mathcal{T}}^{(i)}\right\|_F^2\right).$$

where η is the mean square error. We also report results using two perceptual image quality metrics. In particular, we use SSIM [Wang et al. 2004] and LPIPS [Zhang et al. 2018], where the latter is a newly proposed image quality metric using a deep neural network. Note that LPIPS measures the distance between two images; hence, a smaller value corresponds to a higher reconstruction quality. PSNR and SNR are calculated over the entire data set, while SSIM and LPIPS are calculated by taking the average of obtained values from each angular image. Timing results were obtained using a machine with four Xeon E7-4870, i.e. a total of 40 cores operating at 2.4GHz. Encoding and decoding times are denoted t_e and t_d, respectively, and measured in seconds.

6.1 Light Field

To evaluate our method for natural light fields, we use the Stanford Lego Gantry data set[2]. The results are summarized in Fig. 4 and Table 5, as well as the Section 6 of the supplementary document

[2] http://lightfield.stanford.edu/



Lego Knights Bracelet Tarot Cards

Reference *Reference* *Reference*

AMDE *AMDE* *AMDE*

K-SVD *K-SVD* *K-SVD*

5D DCT *5D DCT* *5D DCT*

HOSVD *HOSVD* *HOSVD*

CDF97 *CDF97* *CDF97*

Fig. 4. Visual quality comparison for a natural light field data set. Visual quality metrics and timing results are reported in Table 5 below. Moreover, Fig. 5 contains false color insets to facilitate image quality comparisons.

Table 5. Compression results for the natural light field data set. The storage cost is measured in megabytes (MB). The size of the K-SVD dictionary is 88MB, while AMDE only requires 0.046MB. Encoding and decoding times are denoted t_e and t_d, respectively, and measured in seconds. Note that a smaller value for LPIPS corresponds to a higher reconstruction quality.

	Lego Knights (size: 384MB)						*Bracelet* (size: 240MB)						*Tarot Cards* (size: 384MB)					
	size	PSNR	SSIM	LPIPS	t_e	t_d	size	PSNR	SSIM	LPIPS	t_e	t_d	size	PSNR	SSIM	LPIPS	t_e	t_d
AMDE	29.3	40.90	0.9725	0.012	124	1.2	18.1	39.90	0.9801	0.014	83	0.7	44.2	38.54	0.9731	0.004	122	1.2
K-SVD	29.3	38.39	0.9592	0.025	882	18	18.1	36.73	0.9734	0.018	556	12	44.3	38.81	0.9803	0.003	1651	18
5D DCT	29.3	37.24	0.9580	0.030	10.6	10.6	18.0	33.98	0.9624	0.034	6.7	6.4	44.3	34.53	0.9659	0.012	10.1	9.8
HOSVD	29.4	37.29	0.9547	0.032	2.3	1.2	18.1	32.31	0.9525	0.040	1.4	0.6	44.2	33.03	0.9600	0.013	2.3	1.1
CDF 9/7	29.0	33.71	0.9138	0.051	8	10	18.2	31.98	0.9387	0.040	5	6	44.3	29.17	0.8645	0.066	7	9

where we include false color insets. The insets were taken from the first angular image of each light field. Among the available light fields, we used the following for training: *Chess*, *Lego Bulldozer*, *Lego Truck*, *Eucalyptus Flowers*, *Amethyst*, *Bunny*, *Jelly Beans*, and *Treasure Chest*. The test set includes *Lego Knights*, *Bracelet*, and *Tarot Cards*. The central 8×8 views of the light field were used to create 5D data points, where $m_1 = 5$, $m_2 = 5$, $m_3 = 8$, $m_4 = 8$, and $m_5 = 3$. Training of AMDE was done using $C = 16$, $K = 8$, $\tau_l = 20$, $p = 128$, and $N_r = 4000$. During encoding we used $\epsilon = 5 \times 10^{-5}$, $\epsilon = 5 \times 10^{-5}$, and $\epsilon = 7 \times 10^{-5}$, respectively for the three light fields. Moreover, the sparsity was set to 300, 390, and 412, respectively. The dictionary for K-SVD was trained using $\tau = 20$ and $\kappa = 1.5$. We did not observe improvements in image quality by increasing κ. In addition, the patch size for CDF 9/7 was set to 64×64 and we used 4 wavelet levels.

Our method significantly outperforms other approaches except for *Tarot Cards*, for which K-SVD achieves a marginally higher PSNR. However, the size of the K-SVD dictionary is more than three orders

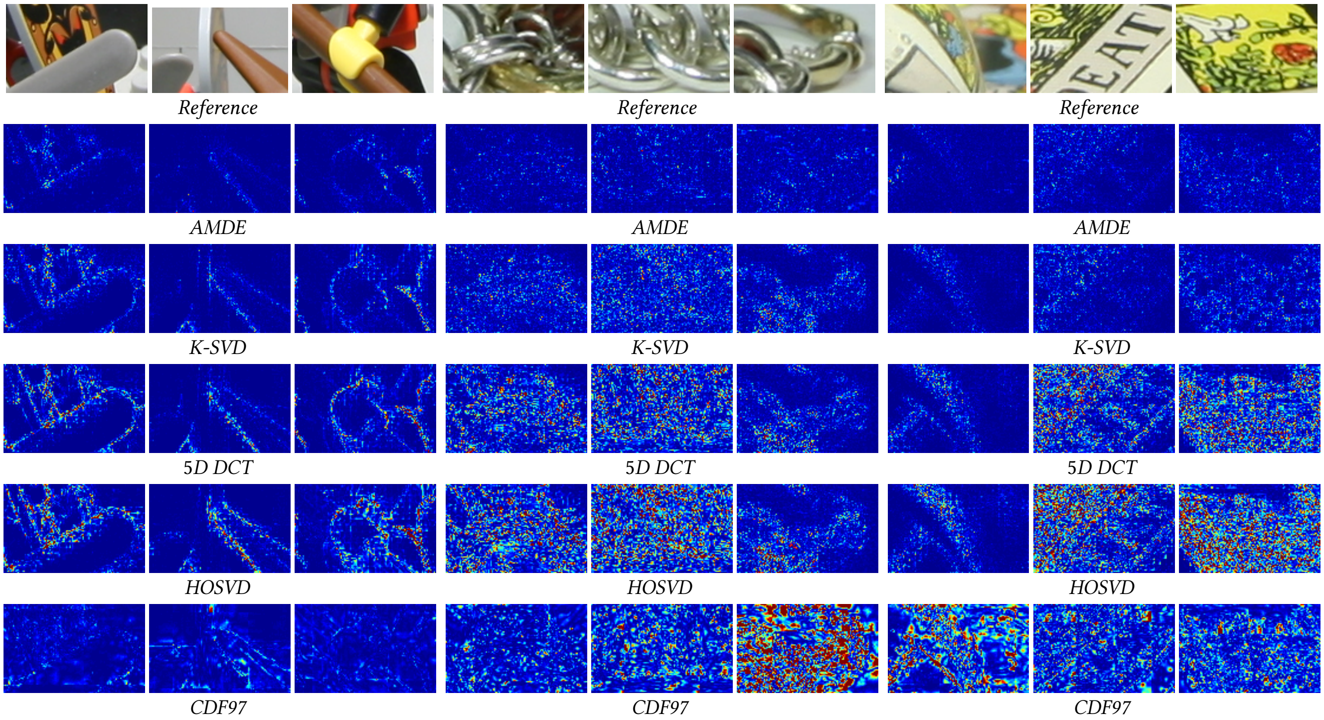


Fig. 5. False color insets for Fig. 4 above.

of magnitude larger than AMDE. In fact the K-SVD dictionary is two times larger than the resulting compressed data. This is a well-known problem of K-SVD with regards to large data points and has been addressed previously [Rubinstein et al. 2010], however, at the cost of reducing the representative power of the dictionary. Apart from dictionary size, encoding and decoding time is about 13 times higher for K-SVD. Note that we have ignored the dictionary size of K-SVD in computing the storage cost of compressed data in Table 5. For *Tarot Cards*, the storage cost of our method for dictionary and coefficient (combined) is about 33% of what is required by K-SVD.

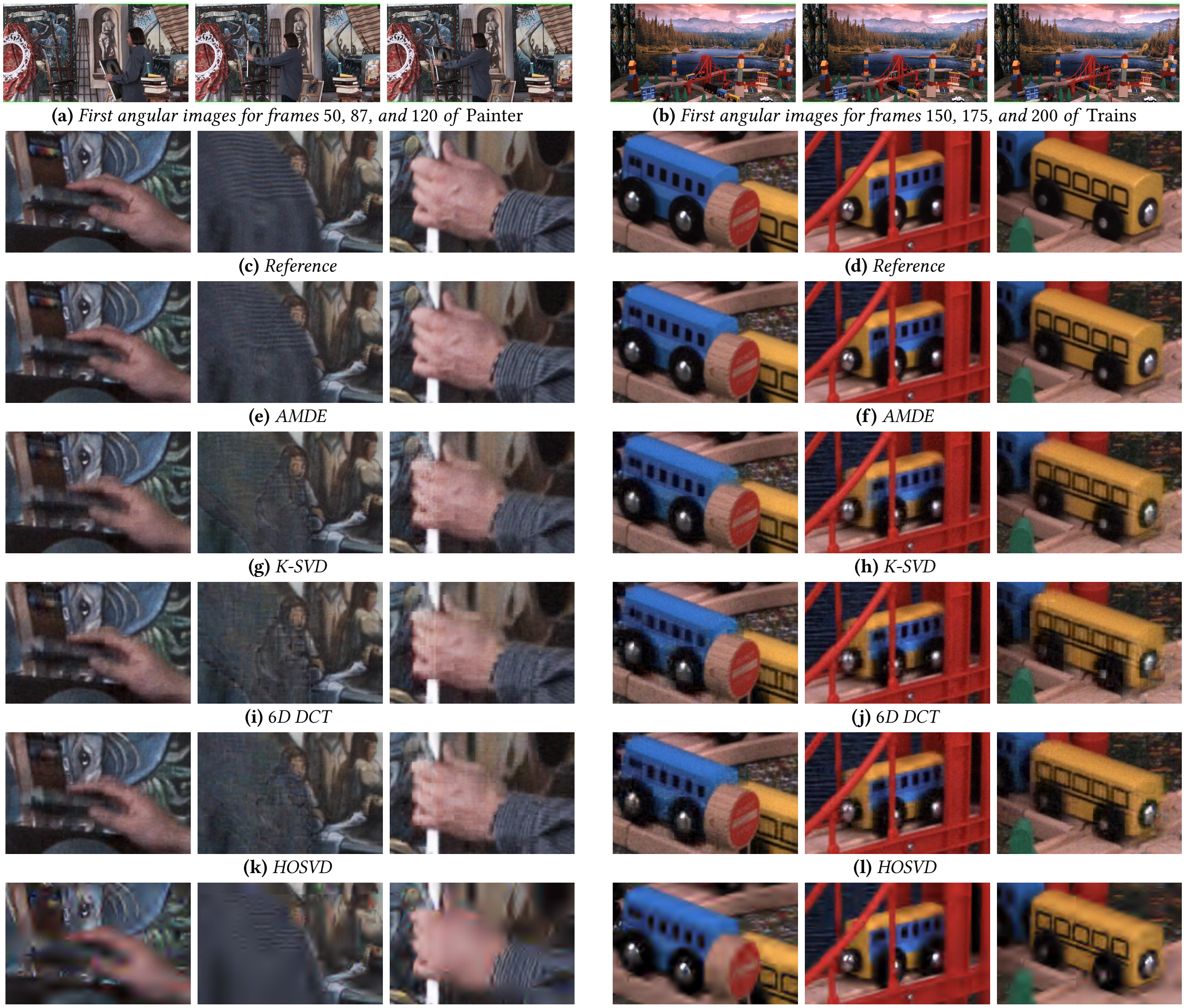
6.2 Light Field Video

We used the Technicolor natural light field video data set [Sabater et al. 2017] to evaluate our method, see Fig. 6. The data set consists of multiple light field videos with an angular resolution of 4×4. We chose frames 50 to 150 of the *Painter* scene and frames 150 to 200 of the *Trains* scene in our experiments. Frames 50 to 100 of *Painter* were used for training. The trained AMDE was used to compress all the frames of both light field videos. The data set is very challenging for compression since the disparity between adjacent views in each frame is very large. In Fig. 9 we plot the disparity for the *Painter* and *Trains* with respect to two adjacent horizontal views. As it can be seen, the disparity can be up to 150 pixels. Since each data point includes all the angular information, there exists high frequency variations within each data point.

The results for light field video compression are demonstrated in Fig. 6 and Table 6. The insets were taken from moving objects, which are more challenging to reconstruct. False color insets corresponding to Fig. 6 are included in Section 6 of the supplementary document. The data point dimensionality was set to $m_1 = 10$, $m_2 = 10$, $m_3 = 4$, $m_4 = 4$, $m_5 = 3$, and $m_6 = 4$. To train the AMDE, we set $C = 4$, $K = 32$, $\tau_l = 100$, $p = 400$, and $N_r = 6000$. The encoding for both scenes was done using $\tau = 1024$ and $\epsilon = 0.00015$. The K-SVD dictionary was trained on the same training set and with $\tau = 20$ and $\kappa = 1.5$. Moreover, due to the large dictionary size and long encoding time, we use a smaller spatial patch size of 8×8. For CDF 9/7 we used a patch size of 256×256 and 4 wavelet levels for *Painter*, and a patch size of 512×512 and 3 wavelet levels for *Trains*.

While the PSNR and SSIM of K-SVD for *Painter* are competitive with our method, the visual quality comparison shows that our approach is more efficient in reconstructing animated objects. This is due to the fact that the K-SVD dictionary is trained on vectorized data points. Given that the data points contain temporal information, the linear combination of dictionary atoms leads to ghosting artifacts. A severe case of this effect can be seen in the middle inset of Fig. 6g, where what is behind the painter is visible. We would like to emphasize that here we have also ignored the dictionary size of K-SVD in computing the storage cost; the size of the compressed data for K-SVD in Table 6 does not include the dictionary size. Yet our method shows higher image quality even though the combined storage cost of coefficients and dictionary is about 68% of K-SVD.

Our method shows significantly higher image quality for *Trains* compared to K-SVD. This is important because both methods were trained on a subset of the *Painter* scene to compress a completely different data set. Hence, AMDE shows more representative power for sparse representation. HOSVD and 6D DCT suffer from ghosting



(a) *First angular images for frames* 50, 87, *and* 120 *of* Painter

(b) *First angular images for frames* 150, 175, *and* 200 *of* Trains

(c) *Reference*

(d) *Reference*

(e) *AMDE*

(f) *AMDE*

(g) *K-SVD*

(h) *K-SVD*

(i) *6D DCT*

(j) *6D DCT*

(k) *HOSVD*

(l) *HOSVD*

(m) *CDF 9/7*

(n) *CDF 9/7*

Fig. 6. Visual quality comparison for natural light field videos. See Table 6 below for timing and image quality metric results. Moreover, Fig. 7 contains false color insets to facilitate image quality comparisons. See the supplementary video for real-time playback of these two data sets, as well as synthetic light fields.

Table 6. Results for the natural light field video data sets. The storage cost is measured in megabytes (MB). Encoding and decoding times are denoted t_e and t_d, respectively, and measured in seconds. Note that a smaller value for LPIPS corresponds to a higher reconstruction quality.

	Painter (size: 20604MB)							*Trains* (size: 10302MB)							
	size	PSNR	SNR	SSIM	LPIPS	t_e	t_d	size	PSNR	SNR	SSIM	LPIPS	t_e	t_d	dict. size (MB)
AMDE	941	38.25	27.40	0.9286	0.051	12436	572	809MB	37.00	29.39	0.9461	0.014	6688	301	0.063MB
K-SVD	942	38.12	27.27	0.9281	0.051	67235	1465	807MB	35.06	27.47	0.9283	0.018	93548	730	432MB
6D DCT	942	36.91	26.06	0.9189	0.080	489	532	807MB	35.29	27.71	0.9372	0.019	247	252	-
HOSVD	941	36.79	25.92	0.9147	0.068	792	555	807MB	35.20	27.60	0.9337	0.020	402	294	-
CDF 9/7	941	31.69	23.25	0.8222	0.210	974	371	1116MB	29.80	24.08	0.7461	0.190	486	223	-

and block artifacts. The results for CDF 9/7 are severely blurred. It should be noted that only AMDE and HOSVD admit real-time playback. AMDE achieves 40 frames per second on an NVidia Titan Xp when rendered at the full spatial light field resolution. See the supplementary video for the rendering performance of AMDE.

Finally, results for compressed sensing of *Knights* and *Painter* data sets are shown in Fig. 8. The samples were taken uniformly

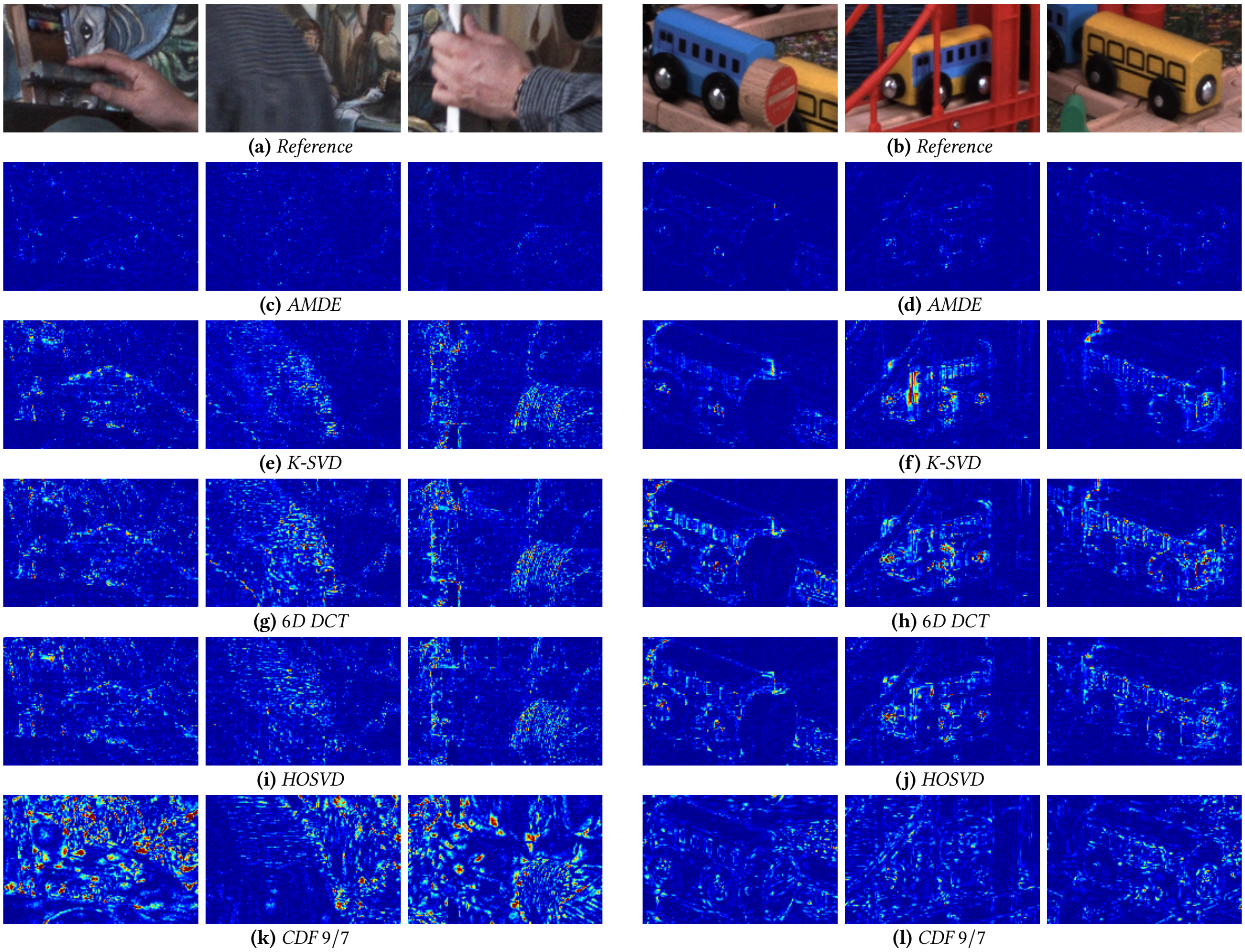


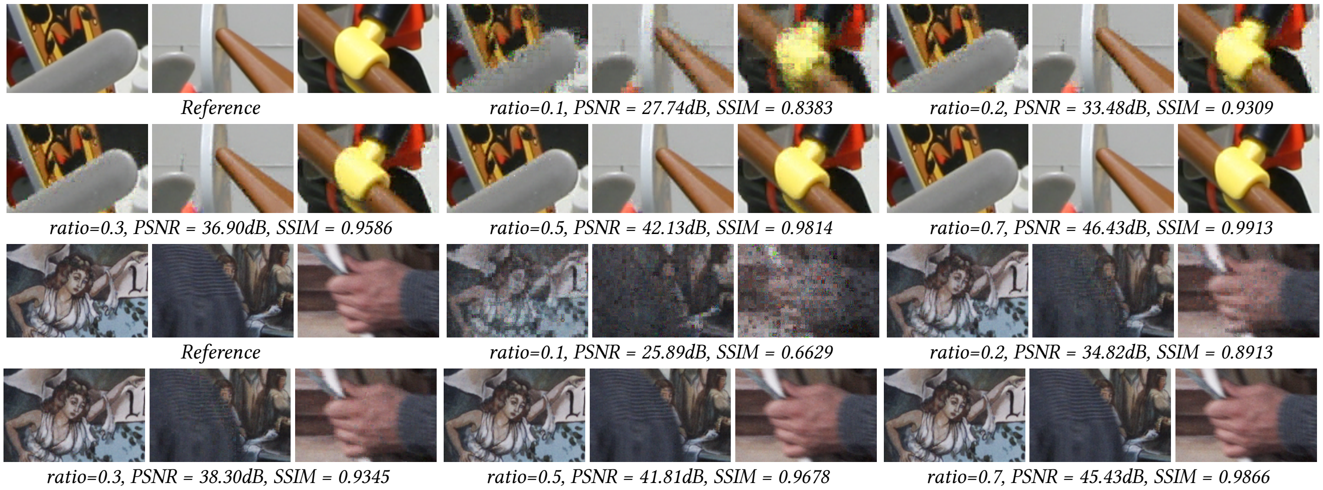
Fig. 7. False color insets for the *Painter* (left) and the *Train* (right) scenes corresponding to Fig. 6 above.

at random to construct a spike ensemble for each data point, as described in Section 4. Moreover, the SL0 algorithm [Mohimani et al. 2009] was used for solving (18). We can see that using a sampling ratio of at least 0.3 one can achieve acceptable visual quality. Note that all the reported results are with non-overlapping patches. As shown in [Marwah et al. 2013; Miandji et al. 2015], using overlapping patches significantly improves image quality.

7 CONCLUSIONS AND FUTURE WORK

We presented a unified framework for compression and compressed sensing of light fields. The proposed training based dictionary ensemble is negligible in size, yet shows superior effectiveness for sparse representation of light fields and light field videos. Additionally, the pre-clustering method proposed here enables training on large data sets. We showed that this increase in efficiency is accompanied with improved reconstruction quality. An important property of our framework is the representation of the light field data in its intrinsic dimensionality. Our empirical results show that this approach leads to sparser coefficients. AMDE was also used for compressed sensing. The theoretical results presented here provide guidelines for improving AMDE for compressed sensing, as well as designing effective consumer-level light field video cameras.

The empirical results obtained from a wide range of data sets show the clear advantage of our method over the state-of-the-art. In particular, our method overwhelmingly outperforms *n*D DCT, HOSVD, and CDF 9/7 in image quality. In terms of computational complexity, our method under performs during encoding. However, considering the large gap in image quality, we believe that our method can be utilized in variety of applications. Moreover, the encoding can be significantly accelerated using a GPU-based implementation to achieve interactive encoding of light field videos. Our method outperforms K-SVD in image quality for an overwhelming majority of data sets. In addition, encoding and decoding times are significantly lower for AMDE. Furthermore, the size of AMDE is orders of magnitude smaller than the dictionary trained by K-SVD. The results reported in the supplementary document of this paper shows the advantages of our method compared to deep learning and convolutional sparse coding methods for compression and



Reference

ratio=0.1, PSNR = 27.74dB, SSIM = 0.8383

ratio=0.2, PSNR = 33.48dB, SSIM = 0.9309

ratio=0.3, PSNR = 36.90dB, SSIM = 0.9586

ratio=0.5, PSNR = 42.13dB, SSIM = 0.9814

ratio=0.7, PSNR = 46.43dB, SSIM = 0.9913

Reference

ratio=0.1, PSNR = 25.89dB, SSIM = 0.6629

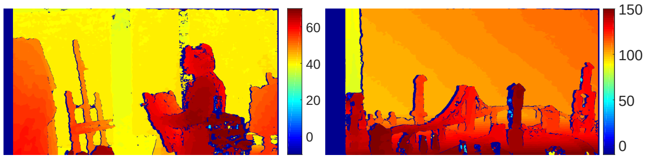
ratio=0.2, PSNR = 34.82dB, SSIM = 0.8913

ratio=0.3, PSNR = 38.30dB, SSIM = 0.9345

ratio=0.5, PSNR = 41.81dB, SSIM = 0.9678

ratio=0.7, PSNR = 45.43dB, SSIM = 0.9866

Fig. 8. Visual results for compressed sensing of a natural light field (*Lego Knights*) and light field video (*Painter*).



(a) Painter, *disparity range is:* $[-10, 70]$

(b) Train, *disparity range is:* $[-10, 150]$

Fig. 9. Disparity maps for the data sets used in Fig. 6.

compressed sensing. In addition, due to the independence of the proposed framework from data dimensionality, our method can be readily applied to a variety of data sets used in computer graphics, as well as new data sets to appear in the future. Preliminary results of utilizing our framework for the compression of scene appearance data sets (i.e. spherical light fields), as well as compressive BRDF measurements for accelerating the capturing process of a gonioreflectometer are reported in [Miandji 2018].

With respect to future work, we believe that our theoretical results can be significantly improved by replacing the mutual coherence with another metric to tighten the bound on sparsity. For instance, Restricted Isometry Property (RIP) has been previously used for compressive spectral imaging [Arguello and Arce 2014]. In addition, utilization of the theoretical results for the evaluation of existing light field imaging systems, as well as deriving new designs for efficient coded-aperture light field video cameras are left for future work. Moreover, as suggested by a reviewer of this paper, an iterative approach based on consecutive pre-clustering and AMDE training is an interesting direction for research.

REFERENCES

Edward H. Adelson and James R. Bergen. 1991. The Plenoptic Function and the Elements of Early Vision. In *Computational Models of Visual Processing*. MIT Press, 3–20.

M. Aharon, M. Elad, and A. Bruckstein. 2006. K -SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation. *Signal Processing, IEEE Transactions on* 54, 11 (Nov. 2006), 4311–4322. DOI:https://doi.org/10.1109/TSP.2006.881199

H. Arguello and G. R. Arce. 2014. Colored Coded Aperture Design by Concentration of Measure in Compressive Spectral Imaging. *IEEE Transactions on Image Processing* 23, 4 (April 2014), 1896–1908. DOI:https://doi.org/10.1109/TIP.2014.2310125

Amit Ashok and Mark A. Neifeld. 2010. Compressive Light Field Imaging. *Proc. SPIE* 7690 (2010), 1–12. DOI:https://doi.org/10.1117/12.852738

S.D. Babacan, R. Ansorge, M. Luessi, P. Ruiz Mataran, R. Molina, and A. K. Katsaggelos. 2012. Compressive Light Field Sensing. *IEEE Trans. on Image Processing* 21, 12 (Dec. 2012), 4746–4757.

R. Ballester-Ripoll and R. Pajarola. 2016. Compressing Bidirectional Texture Functions via Tensor Train Decomposition. In *Proceedings of the 24th Pacific Conference on Computer Graphics and Applications: Short Papers*. Eurographics Association, Goslar Germany, Germany, 19–22. DOI:https://doi.org/10.2312/pg.20161329

Zvika Ben-Haim, Yonina C. Eldar, and Michael Elad. 2010. Coherence-based Performance Guarantees for Estimating a Sparse Vector Under Random Noise. *Trans. Sig. Proc.* 58, 10 (Oct. 2010), 5030–5043. DOI:https://doi.org/10.1109/TSP.2010.2052460

Ahmet Bilgili, Aydn Öztürk, and Murat Kurt. 2011. A General BRDF Representation Based on Tensor Decomposition. *Computer Graphics Forum* 30, 8 (2011), 2427–2439. DOI:https://doi.org/10.1111/j.1467-8659.2011.02072.x

Ori Bryt and Michael Elad. 2008. Compression of facial images using the K-SVD algorithm. *J. Vis. Comun. Image Represent.* 19 (2008), 270–282. Issue 4. DOI:https://doi.org/10.1016/j.jvcir.2008.03.001

E. J. Candes, J. Romberg, and T. Tao. 2006. Robust Uncertainty Principles: Exact Signal Reconstruction From Highly Incomplete Frequency Information. *IEEE Transactions on Information Theory* 52, 2 (Feb. 2006), 489–509. DOI:https://doi.org/10.1109/TIT.2005.862083

E. J. Candès, J. K. Romberg, and T. Tao. 2006. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics* 59, 8 (2006), 1207–1223. DOI:https://doi.org/10.1002/cpa.20124

E. J. Candes and T. Tao. 2005. Decoding by Linear Programming. *IEEE Transactions on Information Theory* 51, 12 (Dec. 2005), 4203–4215. DOI:https://doi.org/10.1109/TIT.2005.858979

E. J. Candes and T. Tao. 2006. Near-Optimal Signal Recovery From Random Projections: Universal Encoding Strategies? *IEEE Transactions on Information Theory* 52, 12 (Dec. 2006), 5406–5425. DOI:https://doi.org/10.1109/TIT.2006.885507

E. J. Candes and M. B. Wakin. 2008. An Introduction To Compressive Sampling. *IEEE Signal Processing Magazine* 25, 2 (March 2008), 21–30. DOI:https://doi.org/10.1109/MSP.2007.914731

L. H. Chang and J. Y. Wu. 2014. An Improved RIP-Based Performance Guarantee for Sparse Signal Recovery via Orthogonal Matching Pursuit. *IEEE Transactions on Information Theory* 60, 9 (Sept. 2014), 5702–5715. DOI:https://doi.org/10.1109/TIT.2014.2338314

B. Choudhury, R. Swanson, F. Heide, G. Wetzstein, and W. Heidrich. 2017. Consensus Convolutional Sparse Coding. In *2017 IEEE International Conference on Computer Vision (ICCV)*. 4290–4298. DOI:https://doi.org/10.1109/ICCV.2017.459

A. Cohen, Ingrid Daubechies, and J.-C. Feauveau. 1992. Biorthogonal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics* 45, 5 (June 1992), 485–560. DOI:https://doi.org/10.1002/cpa.3160450502

Kristin J. Dana, Bram van Ginneken, Shree K. Nayar, and Jan J. Koenderink. 1999. Reflectance and Texture of Real-world Surfaces. *ACM Transactions on Graphics* 18, 1 (Jan 1999), 1–34. DOI:https://doi.org/10.1145/300776.300778

D. L. Donoho. 2006. Compressed sensing. *IEEE Transactions on Information Theory* 52, 4 (April 2006), 1289–1306. DOI: https://doi.org/10.1109/TIT.2006.871582

David L. Donoho and Michael Elad. 2003. Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ_1 minimization. *Proceedings of the National Academy of Sciences* 100, 5 (2003), 2197–2202. DOI: https://doi.org/10.1073/pnas.0437847100

P. L. Dragotti and Y. M. Lu. 2014. On Sparse Representation in Fourier and Local Bases. *IEEE Transactions on Information Theory* 60, 12 (Dec. 2014), 7888–7899. DOI: https://doi.org/10.1109/TIT.2014.2361858

Michael Elad. 2010. *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing* (1st ed.). Springer-Verlag New York. DOI: https://doi.org/10.1007/978-1-4419-7011-4

M. Elad and M. Aharon. 2006. Image Denoising Via Sparse and Redundant Representations Over Learned Dictionaries. *IEEE Transactions on Image Processing* 15, 12 (Dec. 2006), 3736–3745. DOI: https://doi.org/10.1109/TIP.2006.881969

Y.C. Eldar, P. Kuppinger, and H. Bolcskei. 2010. Block-Sparse Signals: Uncertainty Relations and Efficient Recovery. *IEEE Transactions on Signal Processing* 58, 6 (2010), 3042–3054. DOI: https://doi.org/10.1109/TSP.2010.2044837

E. Elhamifar and R. Vidal. 2013. Sparse Subspace Clustering: Algorithm, Theory, and Applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 11 (Nov. 2013), 2765–2781. DOI: https://doi.org/10.1109/TPAMI.2013.57

Walter D. Fisher. 1958. On Grouping for Maximum Homogeneity. *J. Amer. Statist. Assoc.* 53, 284 (1958), 789–798.

Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. 1996. The Lumigraph. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*. ACM, New York, NY, USA, 43–54. DOI: https://doi.org/10.1145/237170.237200

R. Gribonval and M. Nielsen. 2003. Sparse representations in unions of bases. *Information Theory, IEEE Transactions on* 49, 12 (Dec. 2003), 3320–3325. DOI: https://doi.org/10.1109/TIT.2003.820031

K.S. Gurumoorthy, A. Rajwade, A. Banerjee, and A. Rangarajan. 2010. A Method for Compact Image Representation Using Sparse Matrix and Tensor Projections Onto Exemplar Orthonormal Bases. *IEEE Transactions on Image Processing* 19, 2 (2010), 322–334. DOI: https://doi.org/10.1109/TIP.2009.2034991

Michael Guthe, Gero Müller, Martin Schneider, and Reinhard Klein. 2009. BTF-CIELab: A Perceptual Difference Measure for Quality Assessment and Compression of BTFs. *Computer Graphics Forum* 28, 1 (2009), 101–113. DOI: https://doi.org/10.1111/j.1467-8659.2008.01299.x

R.A. Horn and C.R. Johnson. 2012. *Matrix Analysis*. Cambridge University Press.

David A. Huffman. 1952. A Method for the Construction of Minimum-Redundancy Codes. *Proceedings of the IRE* 40, 9 (Sept. 1952), 1098–1101. DOI: https://doi.org/10.1109/JRPROC.1952.273898

Adrian Jarabo, Belen Masia, Adrien Bousseau, Fabio Pellacini, and Diego Gutierrez. 2014. How Do People Edit Light Fields? *ACM Trans. Graph.* 33, 4, Article 146 (July 2014), 10 pages. DOI: https://doi.org/10.1145/2601097.2601125

A. Jones, K. Nagano, J. Busch, X. Yu, H. Y. Peng, J. Barreto, O. Alexander, M. Bolas, P. Debevec, and J. Unger. 2016. Time-Offset Conversations on a Life-Sized Automultiscopic Projector Array. In *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, 927–935. DOI: https://doi.org/10.1109/CVPRW.2016.120

Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. 2016. Learning-based View Synthesis for Light Field Cameras. *ACM Trans. Graph.* 35, 6, Article 193 (Nov. 2016), 10 pages. DOI: https://doi.org/10.1145/2980179.2980251

Mahdad Hosseini Kamal, Barmak Heshmat, Ramesh Raskar, Pierre Vandergheynst, and Gordon Wetzstein. 2016. Tensor low-rank and sparse light field photography. *Computer Vision and Image Understanding* 145 (2016), 172–181. DOI: https://doi.org/10.1016/j.cviu.2015.11.004 Light Field for Computer Vision.

David Kittle, Kerkil Choi, Ashwin Wagadarikar, and David J. Brady. 2010. Multiframe image estimation for coded aperture snapshot spectral imagers. *Appl. Opt.* 49, 36 (Dec. 2010), 6824–6833. DOI: https://doi.org/10.1364/AO.49.006824

Seungjae Lee, Changwon Jang, Seokil Moon, Jaebum Cho, and Byoungho Lee. 2016. Additive Light Field Displays: Realization of Augmented Reality with Holographic Optical Elements. *ACM Trans. Graph.* 35, 4, Article 60 (July 2016), 13 pages. DOI: https://doi.org/10.1145/2897824.2925971

Marc Levoy and Pat Hanrahan. 1996. Light Field Rendering. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*. ACM, New York, NY, USA, 31–42. DOI: https://doi.org/10.1145/237170.237199

Xinguo Liu, Peter-Pike Sloan, Heung-Yeung Shum, and John Snyder. 2004. All-frequency Precomputed Radiance Transfer for Glossy Objects. In *Proceedings of Eurographics Conference on Rendering Techniques (EGSR'04)*. Eurographics Association, 337–344. DOI: https://doi.org/10.2312/EGWR/EGSR04/337-344

S. Lloyd. 1982. Least squares quantization in PCM. *IEEE Transactions on Information Theory* 28, 2 (1982), 129–137. DOI: https://doi.org/10.1109/TIT.1982.1056489

Dhruv Mahajan, Ira Kemelmacher Shlizerman, Ravi Ramamoorthi, and Peter Belhumeur. 2007. A Theory of Locally Low Dimensional Light Transport. *ACM Trans. Graph.* 26, 3 (July 2007). DOI: https://doi.org/10.1145/1276377.1276454

Andrew Maimone, Gordon Wetzstein, Matthew Hirsch, Douglas Lanman, Ramesh Raskar, and Henry Fuchs. 2013. Focus 3D: Compressive Accommodation Display. *ACM Trans. Graph.* 32, 5, Article 153 (Oct. 2013), 13 pages. DOI: https://doi.org/10.1145/2503144

J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. 2009. Non-local sparse models for image restoration. In *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2272–2279. DOI: https://doi.org/10.1109/ICCV.2009.5459452

S. Mallat. 2008. *A Wavelet Tour of Signal Processing: The Sparse Way*. Elsevier Science.

Kshitij Marwah, Gordon Wetzstein, Yosuke Bando, and Ramesh Raskar. 2013. Compressive Light Field Photography Using Overcomplete Dictionaries and Optimized Projections. *ACM Trans. Graph.* 32, 4, Article 46 (July 2013), 12 pages. DOI: https://doi.org/10.1145/2461912.2461914

Ehsan Miandji. 2018. *Sparse Representation of Visual Data for Compression and Compressed Sensing*. Ph.D. Dissertation. Department of Science and Technology, Linköping University, Sweden.

Ehsan Miandji, Mohammad Emadi, Jonas Unger, and Ehsan Afshari. 2017. On Probability of Support Recovery for Orthogonal Matching Pursuit Using Mutual Coherence. *IEEE Signal Processing Letters* 24, 11 (Nov. 2017), 1646–1650. DOI: https://doi.org/10.1109/LSP.2017.2753939

Ehsan Miandji, Joel Kronander, and Jonas Unger. 2013. Learning Based Compression of Surface Light Fields for Real-time Rendering of Global Illumination Scenes. In *SIGGRAPH Asia 2013 Technical Briefs (SA '13)*. ACM, New York, NY, USA, Article 24, 4 pages. DOI: https://doi.org/10.1145/2542355.2542385

Ehsan Miandji, Joel Kronander, and Jonas Unger. 2015. Compressive Image Reconstruction in Reduced Union of Subspaces. *Comput. Graph. Forum* 34, 2 (May 2015), 33–44. DOI: https://doi.org/10.1111/cgf.12539

Ehsan Miandji, Jonas Unger, and Christine Guillemot. 2018. Multi-Shot Single Sensor Light Field Camera Using a Color Coded Mask. In *EUSIPCO 2018 - 26th European Signal Processing Conference*. Roma, Italy, 1–5.

H. Mohimani, Massoud Babaie-Zadeh, and C. Jutten. 2009. A Fast Approach for Overcomplete Sparse Decomposition Based on Smoothed ℓ_0 Norm. *IEEE Transactions on Signal Processing* 57, 1 (2009), 289–301. DOI: https://doi.org/10.1109/TSP.2008.2007606

Gero Müller, Jan Meseth, Mirko Sattler, Ralf Sarlette, and Reinhard Klein. 2005. Acquisition, Synthesis and Rendering of Bidirectional Texture Functions. *Computer Graphics Forum* 24, 1 (March 2005), 83–109.

Ren Ng, Marc Levoy, Mathieu Brédif, Gene Duval, Mark Horowitz, and Pat Hanrahan. 2005. Light field photography with a hand-held plenoptic camera. *Computer Science Technical Report CSTR* 2, 11 (2005), 1–11.

H. Nyquist. 1928. Certain Topics in Telegraph Transmission Theory. *Transactions of the American Institute of Electrical Engineers* 47, 2 (April 1928), 617–644. DOI: https://doi.org/10.1109/T-AIEE.1928.5055024

Renato Pajarola, Susanne K. Suter, and Roland Ruiters. 2013. Tensor Approximation in Visualization and Computer Graphics. In *Eurographics 2013 - Tutorials*. Eurographics Association, Girona, Spain. DOI: https://doi.org/10.2312/conf/eg2013/tutorials/t6

Y.C. Pati, R. Rezaiifar, and P. S. Krishnaprasad. 1993. Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. In *Conference Record of The Twenty-Seventh Asilomar Conference on Signals, Systems and Computers*, Vol. 1. IEEE, 40–44. DOI: https://doi.org/10.1109/ACSSC.1993.342465

M. Rahmani and G. K. Atia. 2017. Coherence Pursuit: Fast, Simple, and Robust Principal Component Analysis. *IEEE Transactions on Signal Processing* 65, 23 (Dec. 2017), 6260–6275. DOI: https://doi.org/10.1109/TSP.2017.2749215

Ruiters Roland and Klein Reinhard. 2009. BTF Compression via Sparse Tensor Decomposition. *Computer Graphics Forum* 28, 4 (2009), 1181–1188. DOI: https://doi.org/10.1111/j.1467-8659.2009.01495.x

R. Rubinstein, M. Zibulevsky, and M. Elad. 2010. Double Sparsity: Learning Sparse Dictionaries for Sparse Signal Approximation. *IEEE Transactions on Signal Processing* 58, 3 (2010), 1553–1564. DOI: https://doi.org/10.1109/TSP.2009.2036477

Neus Sabater, Guillaume Boisson, Benoit Vandame, Paul Kerbiriou, Frederic Babon, Matthieu Hog, Tristan Langlois, Remy Gendrot, Olivier Bureller, Arno Schubert, and Valerie Allie. 2017. Dataset and Pipeline for Multi-View Light-Field Video. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, 1743–1753.

PeterH. Schönemann. 1966. A generalized solution of the orthogonal procrustes problem. *Psychometrika* 31, 1 (1966), 1–10. DOI: https://doi.org/10.1007/BF02289451

Kai Schröder, Reinhard Klein, and Arno Zinke. 2013. Non-Local Image Reconstruction for Efficient Computation of Synthetic Bidirectional Texture Functions. *Computer Graphics Forum* 32 (2013), 61–71. DOI: https://doi.org/10.1111/cgf.12144

C. E. Shannon. 1949. Communication in the Presence of Noise. *Proceedings of the IRE* 37, 1 (Jan. 1949), 10–21. DOI: https://doi.org/10.1109/JRPROC.1949.232969

Peter-Pike Sloan, Jesse Hall, John Hart, and John Snyder. 2003. Clustered principal components for precomputed radiance transfer. *ACM Trans. Graph.* 22 (2003), 382–391. Issue 3. DOI: https://doi.org/10.1145/882262.882281

Peter-Pike Sloan, Jan Kautz, and John Snyder. 2002. Precomputed Radiance Transfer for Real-time Rendering in Dynamic, Low-frequency Lighting Environments. *ACM Trans. Graph.* 21, 3 (July 2002), 527–536. DOI: https://doi.org/10.1145/566654.566612

Mahdi Soltanolkotabi, Emmanuel J Candes, and others. 2012. A geometric analysis of subspace clustering with outliers. *The Annals of Statistics* 40, 4 (2012), 2195–2238.

David Taubman and Michael Marcellin. 2013. *JPEG2000 Image Compression Fundamentals, Standards and Practice.* Springer Publishing Company, Incorporated.

J.A Tropp. 2004. Greed is good: algorithmic results for sparse approximation. *Information Theory, IEEE Transactions on* 50, 10 (Oct. 2004), 2231–2242. DOI: https://doi.org/10.1109/TIT.2004.834793

Yu-Ting Tsai. 2015. Multiway K-Clustered Tensor Approximation: Toward High-Performance Photorealistic Data-Driven Rendering. *ACM Trans. Graph.* 34, 5, Article 157 (Nov. 2015), 15 pages. DOI: https://doi.org/10.1145/2753756

Yu-Ting Tsai and Zen-Chung Shih. 2012. K-clustered Tensor Approximation: A Sparse Multilinear Model for Real-time Rendering. *ACM Trans. Graph.* 31, 3, Article 19 (June 2012), 17 pages. DOI: https://doi.org/10.1145/2167076.2167077

Vaibhav Vaish, Marc Levoy, Richard Szeliski, C. L. Zitnick, and Sing Bing Kang. 2006. Reconstructing Occluded Surfaces Using Synthetic Apertures: Stereo, Focus and Robust Measures. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition.* IEEE, 2331–2338. DOI: https://doi.org/10.1109/CVPR.2006.244

M. Alex O. Vasilescu and Demetri Terzopoulos. 2004. TensorTextures: Multilinear Image-based Rendering. *ACM Trans. Graph.* 23, 3 (Aug. 2004), 336–342. DOI: https://doi.org/10.1145/1015706.1015725

Hongcheng Wang, Qing Wu, Lin Shi, Yizhou Yu, and Narendra Ahuja. 2005. Out-of-core Tensor Approximation of Multi-dimensional Matrices of Visual Data. *ACM Trans. Graph.* 24, 3 (July 2005), 527–535. DOI: https://doi.org/10.1145/1073204.1073224

Y. Wang, F. Liu, K. Zhang, G. Hou, Z. Sun, and T. Tan. 2018. LFNet: A Novel Bidirectional Recurrent Convolutional Neural Network for Light-Field Image Super-Resolution. *IEEE Transactions on Image Processing* 27, 9 (Sept. 2018), 4274–4286. DOI: https://doi.org/10.1109/TIP.2018.2834819

Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing* 13, 4 (April 2004), 600–612. DOI: https://doi.org/10.1109/TIP.2003.819861

Gordon Wetzstein, Douglas Lanman, Matthew Hirsch, Wolfgang Heidrich, and Ramesh Raskar. 2012b. Compressive light field displays. *IEEE computer graphics and applications* 32, 5 (2012), 6–11.

Gordon Wetzstein, Douglas Lanman, Matthew Hirsch, and Ramesh Raskar. 2012a. Tensor Displays: Compressive Light Field Synthesis Using Multilayer Displays with Directional Backlighting. *ACM Trans. Graph.* 31, 4, Article 80 (July 2012), 11 pages. DOI: https://doi.org/10.1145/2185520.2185576

G. Wu, B. Masia, A. Jarabo, Y. Zhang, L. Wang, Q. Dai, T. Chai, and Y. Liu. 2017a. Light Field Image Processing: An Overview. *IEEE Journal of Selected Topics in Signal Processing* 11, 7 (Oct. 2017), 926–954. DOI: https://doi.org/10.1109/JSTSP.2017.2747126

G. Wu, M. Zhao, L. Wang, Q. Dai, T. Chai, and Y. Liu. 2017b. Light Field Reconstruction Using Deep Convolutional Network on EPI. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 1638–1646. DOI: https://doi.org/10.1109/CVPR.2017.178

H. Xu, C. Caramanis, and S. Mannor. 2013. Outlier-Robust PCA: The High-Dimensional Case. *IEEE Transactions on Information Theory* 59, 1 (Jan. 2013), 546–572. DOI: https://doi.org/10.1109/TIT.2012.2212415

Y. Yoon, H. Jeon, D. Yoo, J. Lee, and I. S. Kweon. 2015. Learning a Deep Convolutional Network for Light-Field Image Super-Resolution. In *2015 IEEE International Conference on Computer Vision Workshop (ICCVW).* 57–65. DOI: https://doi.org/10.1109/ICCVW.2015.17

Y. Yoon, H. Jeon, D. Yoo, J. Lee, and I. S. Kweon. 2017. Light-Field Image Super-Resolution Using Convolutional Neural Network. *IEEE Signal Processing Letters* 24, 6 (June 2017), 848–852. DOI: https://doi.org/10.1109/LSP.2017.2669333

J. Zepeda, C. Guillemot, and E. Kijak. 2011. Image Compression Using Sparse Representations and the Iteration-Tuned and Aligned Dictionary. *IEEE Journal of Selected Topics in Signal Processing* 5, 5 (Sept. 2011), 1061–1073. DOI: https://doi.org/10.1109/JSTSP.2011.2135332

Richard Zhang, Phillip Isola, Alexei Efros, Eli Shechtman, and Oliver Wang. 2018. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 586–595. DOI: https://doi.org/10.1109/CVPR.2018.00068

Zhiming Zhou, Guojun Chen, Yue Dong, David Wipf, Yong Yu, John Snyder, and Xin Tong. 2016. Sparse-as-possible SVBRDF Acquisition. *ACM Trans. Graph.* 35, 6, Article 189 (Nov. 2016), 12 pages. DOI: https://doi.org/10.1145/2980179.2980247