A 4-D DISCRETE COSINE TRANSFORM-BASED LIGHT FIELD CODING SOLUTION

Gustavo de Oliveira e Alves

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica.

Orientadores: Eduardo Antônio Barros da Silva
Fernando Manuel Bernardo Pereira

Rio de Janeiro
Março de 2019

A 4-D DISCRETE COSINE TRANSFORM-BASED LIGHT FIELD CODING SOLUTION

Gustavo de Oliveira e Alves

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Examinada por:

Prof. Eduardo Antônio Barros da Silva, Ph.D.

Prof. José Gabriel Rodriguez Carneiro Gomes, Ph.D.

Prof. Lisandro Lovisolo, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2019

Alves, Gustavo de Oliveira e

A 4-D Discrete Cosine Transform-based Light Field Coding Solution / Gustavo de Oliveira e Alves. – Rio de Janeiro: UFRJ/COPPE, 2019.

XIV, 90 p.: il.; 29, 7cm.

Orientadores: Eduardo Antônio Barros da Silva
Fernando Manuel Bernardo Pereira

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2019.

Referências Bibliográficas: p. 87 – 90.

1. light field. 2. coding. 3. 4D-DCT. 4. JPEG Pleno. I. Silva, Eduardo Antônio Barros da *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

To my parents

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

UMA SOLUÇÃO DE CODIFICAÇÃO DE CAMPOS DE LUZ BASEADA NA TRANSFORMADA DISCRETA DE COSSENO 4-D

Gustavo de Oliveira e Alves

Março/2019

Orientadores: Eduardo Antônio Barros da Silva
Fernando Manuel Bernardo Pereira

Programa: Engenharia Elétrica

É bem conhecido que as formas convencionais de capturar a luz ao nosso redor são limitadas e, portanto, fornecem uma experiência limitada em termos de percepção de paralaxe. Como isso tem impedido que os sistemas 3D explodam no mercado, avanços significativos estão surgindo em termos de tecnologias de captura de luz, entre as quais é relevante destacar as chamadas câmeras de campos de luz que capturam uma representação mais rica da cena visual medindo a intensidade da luz para cada direção e para cada posição de pixel. Considerando a enorme quantidade de dados de intensidade de luz envolvidos, uma compressão eficiente torna-se uma obrigação. Neste contexto, a dissertação aborda o desafio da codificação do campo de luz propondo uma solução de codificação baseada em transformada de cosseno discreta quadridimensional. A DCT é uma candidata natural para este tipo de processamento, considerando a sua ampla adoção nos padrões convencionais de codificação de imagem e vídeo.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

A 4-D DISCRETE COSINE TRANSFORM-BASED LIGHT FIELD CODING SOLUTION

Gustavo de Oliveira e Alves

March/2019

Advisors: Eduardo Antônio Barros da Silva
Fernando Manuel Bernardo Pereira

Department: Electrical Engineering

It is well known that the conventional ways to capture the light around us are limited and thus provide a limited user experience, notably in terms of parallax capabilities. As this has been preventing 3D systems to explode in the market, significant advances are emerging in terms of light capturing technologies among which is relevant to highlight the so-called light field cameras which capture a richer representation of the visual scene by measuring the light intensity for each direction and for each pixel position. Considering the huge amount of light intensity data involved, efficient compression becomes a must. In this context, this dissertation addresses the light field coding challenge by proposing a 4D discrete cosine transform-based coding solution. The DCT is an natural candidate to this kind of processing considering its wide adoption in conventional image and video coding standards.

Contents

List of Figures

List of Tables

List of Abbreviations

CE	Cumulative Energy, p. 45
DCT	Discrete Cosine Transform, p. 14
GOB	Group Of Blocks, p. 58
GSVR	Geometric Space View Redundancy, p. 81
H.264/AVC	Advanced Video Coding, p. 30
HDCA	High Density Camera Array, p. 5
HEVC	High Efficiency Video Coding, p. 30
ICE	Integral Cumulative Energy, p. 45
JPEG	Joint Photographic Experts Group, p. 1
PSNR	Peak Signal-to-Noise Ratio, p. 11
RD	Rate-Distortion, p. 31

Chapter 1

Introduction

Light field photography is a new technology that adds outstanding features and functionalities to conventional (digital) photography trying to bring the users more realistic, immersive and interactive experiences. Among the main added functionalities are the possibilities of a posteriori varying the focus and changing the perspective of the rendered images after the light field image is captured; this added value for the user justifies the recent growth in popularity of this emerging imaging technology/modality.

Light field imaging captures not only information about the intensity of light on an image plane but also information about the direction of the light rays in space. These are acquired with a light field camera, typically consisting of an array of micro-lenses placed in front of an otherwise conventional image sensor; or acquired with a two-dimensional array of densely placed cameras. Both acquisition techniques generate large raw uncompressed files, since information about the direction of the light rays is also captured. The large amount of data corresponding to this richer form of visual representation needs to be efficiently compressed, notably when transmission and storage applications are targeted.

While light fields may be coded by adapting/extending available standard image or video coding solutions [1], these solutions are not able to exploit the new type of redundancy in the light field data. This drives the development of specific coding solutions which started to appear in the literature in the last few years, and are able to better exploit the redundancy of this new type of data.

As part of its standardization activities, the Joint Photographic Experts Group (JPEG) has initiated a new work item called JPEG Pleno, which aims at developing a next generation image coding standard to support light field imaging, among other plenoptic modalities. Recently, JPEG has issued a call for proposals on light field coding technologies [2], requesting contributions for coding solutions in the area of light fields encompassing, among other things, both lenslet-generated light field images, that are generated with light field cameras, and light field images obtained

using high density 2D camera arrays.

In this context, the work presented in this thesis intends to propose a novel coding solution for light fields. The proposed codec has a simple architecture, employing the 4D discrete cosine transform (4D-DCT), novel 4D scanning patterns and run-length coding, in an analogous way as it is done in 2D in the classic JPEG image coding standard [3].

The remaining part of this work is structured as follows: Chapter 2 presents an introduction to light field imaging, describing the acquisition techniques, representation models and existing coding approaches, together with a brief introduction to JPEG Pleno. As a prior step to the light field coding solution design, Chapter 3 presents a study which aims at characterizing the redundancy of available light field test material using as main tool the 4D-DCT. In Chapter 4 the design of scanning modes for the coefficients of 4D-DCT is addressed, while the complete proposed light field coding solution is described in Chapter 5. Finally, Chapter 6 presents the conclusions and suggestions for future work.

Chapter 2

Light fields basics

This chapter presents a brief introduction to light fields, the object of study in this work. Section 2.1 introduces the plenoptic function, a background concept essential to understanding the definition of a light field. Section 2.2 describes the light field acquisition methods, while Section 2.3 presents the main approaches to code light field data. Finally, Section 2.4 briefly describes the JPEG Pleno, a work item initiated by JPEG Committee which aims at developing a light field coding standard.

2.1 The plenoptic function

Adelson and Bergen pointed out what actually can be seen by an observer by describing the visual information contained in a single point of the space [4]. This visual information is composed by a set of infinite light rays of several wavelengths, coming at every direction, at any instant of time. The angle of incidence of a ray can be parameterized with two spherical coordinates (θ, ϕ). The color of the rays may be represented by the wavelength λ, while the time instant, by variable t. So, to fully describe the visual information on a point of the 3D space, four variables would be required. By adding the three cartesian coordinates (x, y, z), the function would be able to define such information for every point in 3D space, totalling seven variables. Therefore, a function intended to completely describe the visual information in the space, the so-called plenoptic function, has the form

$$P = P(\theta, \phi, \lambda, t, x, y, z). \tag{2.1}$$

The high dimensionality of the plenoptic function indicates that a huge amount of information is involved. Some simplifications are required in order to reduce the original information amount and achieve a handleable amount of data. By removing the variable t, $P = P(\theta, \phi, \lambda, x, y, z)$ represents the static visual information, which

is analogous to recording an "image" rather than a "video". By replacing λ by three color components as it is usually done in image capturing systems, the dimensionality is further reduced to five variables (θ, ϕ, x, y, z). Actually, the remaining five dimensions can be reduced to four, if the light rays are parameterized when using the two-plane parameterization, as shown in Figure 2.1 [5]. This approach takes advantage to the fact that light rays do not change direction when propagating in free space. Following this premise, the position and angle of a light ray can be fully described by the coordinates (u, v) and (s, t), representing the points of intersection of the light ray with two parallel planes. Such reduction the dimensionality of the plenoptic function to a 4D subset is know as light field [5].

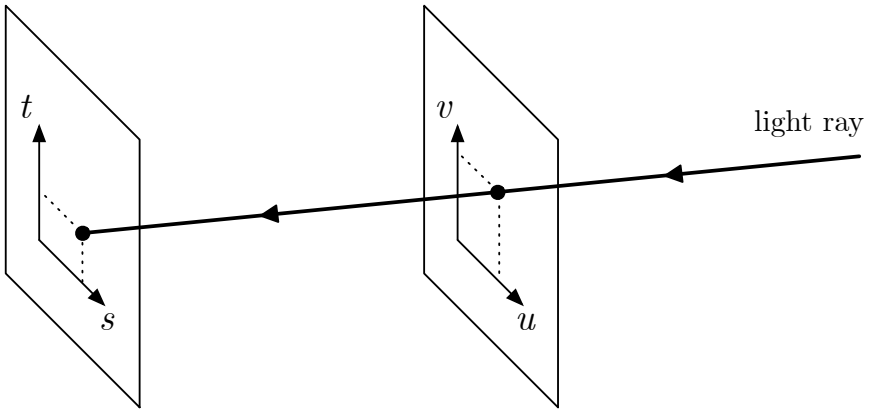


Figure 2.1: Two-plane parameterizations of light rays.

The (s, t) plane can be considered as a two dimensional array of cameras, collecting the light rays leaving the (u, v) plane, closer to the scene. Therefore, each point on the (s, t) plane collects a set of rays that constitute a viewpoint. When the four dimensions of the light field are organized as a two-dimensional (s, t) array of $u \times v$ slices, such slices are called sub-aperture images, or simply light field views.

2.2 Acquisition methods

This section describes the main light field imaging acquistion methods: cameras with micro-lenses arrays and or high density camera arrays.

2.2.1 Cameras with micro-lenses arrays

The first light field acquisition method presented consists in using plenoptic cameras, which are cameras equipped with an array of micro-lenses placed in front of the otherwise conventional image sensor. Such micro-lenses are usually referred as lenslets, and the light fields acquired using this method are called lenslet light fields.

Figure 2.2 depicts two plenoptic cameras available in the consumer market. The main advantages of this acquisition mode are the use of a single camera, which results in a reduced cost, and the high density of generated views, which is desirable for many applications. One drawback of this acquisition method is the reduced resolution of the acquired light filed, which is limited by the resolution of the single camera sensor involved.

(a) Lytro Illum (b) Raytrix R47

Figure 2.2: Examples of plenoptic cameras used for light field acquisition.

The views of a lenslet light field are subject to the *vignetting* effect, is caused by the darkened regions created by micro-lenses. This effect causes views to become darker according to the distance to the central view. Figure 2.3 depicts a mosaic made of a lenslet light field views showing the lighter central view and the darkened peripheral views.

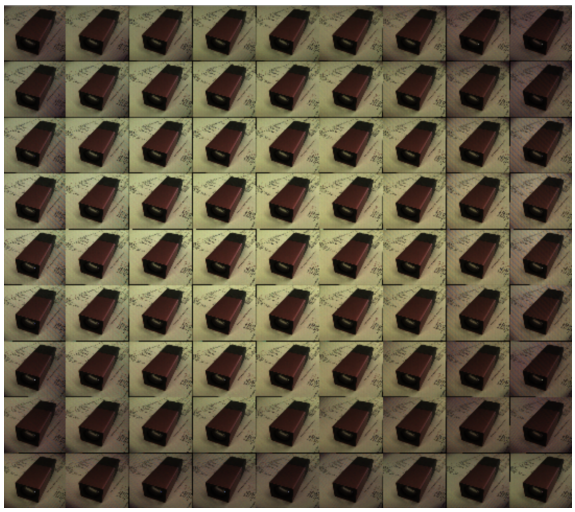


Figure 2.3: Mosaic of lenslet light field views revealing the *vignetting* effect.

2.2.2 High density camera arrays

Other method for light field acquisition consists in using a two-dimensional array of densely-placed cameras, as depicted in Figure 2.4. This acquisition method is usually referred as High Density Camera Array (HDCA). A drawback of this type

of acquisition is the high associated cost, since it requires a large number of cameras and sophisticated peripheral equipments. HDCAs are able to generate high resolution light fields, since the final resolution is the sum of individual camera sensors resolutions.

A characteristic of the light field content acquired with HDCAs is that views are not so dense as the ones acquired with a plenoptic camera, since the spacing between adjacent cameras is limited by the cameras own dimensions. In order to overcome this limitation, a single robotized camera can be used. This acquisition setup has the advantage of producing large number of high density views, and present the drawback of capturing only static scenes.

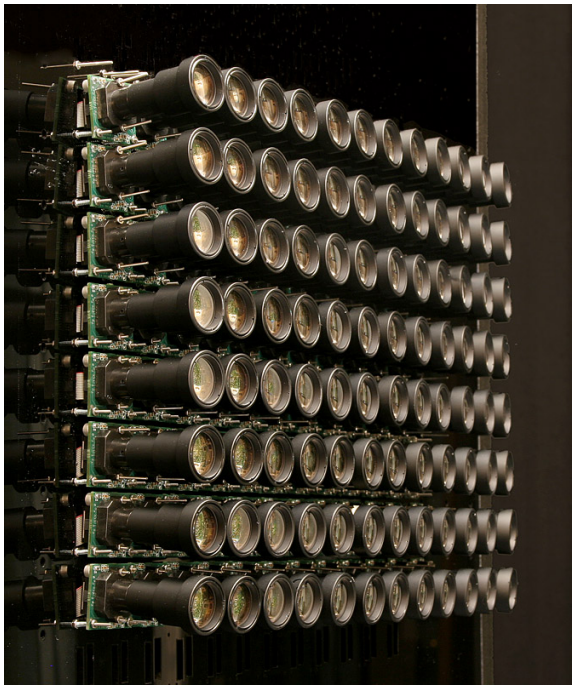


Figure 2.4: Camera array used for light field acquisition.

2.3 Coding

There are four main approaches to code light field data, notably:

- **Direct usage of available coding standards:** This is the simplest approach as it only implies directly applying the available image coding standards [6], thus also providing full compatibility; naturally, the redundancy among micro-images or sub-aperture images cannot be exploited.
- **Usage of available coding standards after some data restructuring:** This approach still implies using available image and vide coding standards, but the light field data is first rearranged to better exploit the available redundancy. For example, the multiple views may be first extracted to create a 2D

puzzle of views which are after coded as a video sequence using some scanning, e.g. top-down or spiral, to exploit the redundancy between the views as if it was some 'temporal' redundancy [7] [8].

- **Extension of available coding standards:** This approach involves extending the available coding standards, e.g. with additional coding modes specifically targeting the exploitation of the inter view redundancy which they are not ready to exploit [1].

- **Development of novel coding solutions:** This approach involves developing novel coding solutions [2][9][10] which go beyond the simple extension of available coding standard, more specifically adapted to the characteristics of light field imaging data.

2.4 JPEG Pleno

The JPEG Pleno is a new work item initiated by Joint Photographic Experts Group (JPEG) Committee which aims at developing a next image coding standard that moves beyond coding of 2D conventional image content by addressing plenoptic representations, including the emerging light fields technology. During the past decades, JPEG has offered a series of successful and widely adopted image coding solutions such as JPEG [3] and JPEG 2000 [11]. However, digital photography market now experiences a paradigm shift in the consumption of digital photographic content, moving from a planar world towards imaging in volumetric modalities such as omnidirectional, depth-enhanced, and light field imaging which will require specific coding tools and, consequently, new coding standards are required for this new kind of content.

JPEG Pleno intends to provide an efficient coding format that will guarantee the highest quality content representation with reasonable resource requirements in terms of data rates and computational complexity. In addition to features that are described next, supported functionalities will include some degree of backward compatibility with legacy JPEG formats, scalability, random access, error resilience, distributed processing and data sharing between displays or display elements. The associated file format will be compliant with JPEG Systems specifications.

Such need for efficient light field coding schemes is driving JPEG Pleno standardization activities, that has issued the JPEG Pleno Call for Proposals on Light Field Coding in January 2017 [2]. It requests contributions for coding solutions in the area of light fields encompassing both lenslet-generated and light fields obtained using high density 2D camera arrays.

2.4.1 Datasets

The light fields used in this study are described in the document *JPEG Pleno Light Field Coding Common Test Conditions* [12] and are the same used in JPEG Pleno standardization activities such as exploration studies and core experiments. The JPEG Pleno light fields include two datasets acquired by different devices and setups: lenslet-based plenoptic cameras and high-density array of conventional cameras. A third dataset comprises synthetically generated content.

Lenslet-based light field dataset

Figure 2.5 shows the central views of the light fields acquired with a lenslet based camera, Lytro Illum B01 (10-bit) [13]. Each light field consists of a 13×13 array of views, with spatial resolution of 626×434 pixels each. The images selected from this dataset are natural outdoor images presenting different levels of spatial information, with objects at different depths and repetitive patterns [14].

(a) *Bikes*

(b) *Danger de Mort*

(c) *Stone Pillars Outside*

(d) *Fountain & Vincent 2*

(e) *Friends 1*

Figure 2.5: lenslet light fields (central views) in JPEG Pleno dataset.

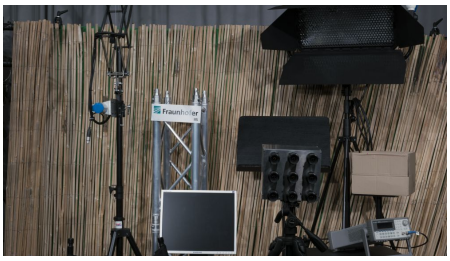
HDCA light field dataset

The central views of the HDCA light fields are depicted in Figure 2.6 They were acquired at Fraunhofer Institute using a high resolution camera attached to a moving robot [15]. All images portray rather similar indoor (studio) scenes and show different levels of detail, specularities, regular patterns and objects at different depths. Each light field has 101×21 views each with a 3840×2160 spatial resolution. This study also uses the view-subsampled version of the HDCA dataset, notably 33×11 views, with the same spatial resolution, as defined in the JPEG Pleno core experiments [16]. While this view-sampled version of the HDCA content has less inter-view redundancy, its usage within JPEG Pleno resulted from the need to limit the computational complexity associated to HDCA content coding.

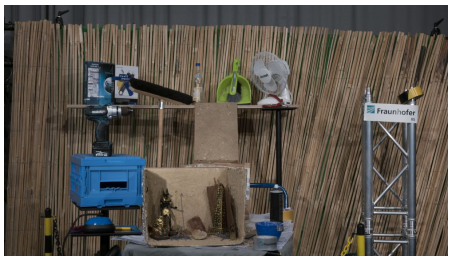
Currently, the JPEG Pleno test set includes only one light field image from the four initially provided, being *Set*2 the single one currently adopted for computational complexity reasons. Moreover, as even a single full set was very heavy to encode, the selected view-subsampled *Set*2 light field had each view cropped in terms of spatial resolution to 1920×1080; this light field is referred as *Set2 2K sub*.

(a) *Set 2*

(b) *Set 6*



(c) *Set 9*

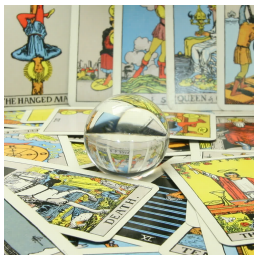


(d) *Set 10*

Figure 2.6: Fraunhofer HDCA light fields (central views) in JPEG Pleno dataset.

The JPEG Pleno test set also includes a light field from Poznan University of Technology called *Laboratory 1*, shown in Figure 2.7 (a) [17]. It has been acquired with a 2D parallel array of cameras. The light field presents 31×31 views with spatial resolution 1936×1288. JPEG Pleno test set includes one light field image in the Stanford HDCA data set, named *Tarot Cards*, shown in Figure 2.7 (b) [18]. This light field presents 17×17 views with spatial resolution 1024×1024.

(a) *Laboratory 1*



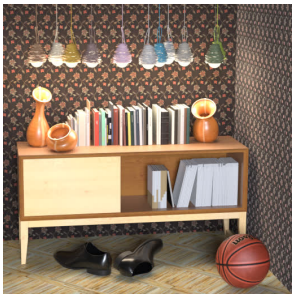
(b) *Tarot Cards*

Figure 2.7: Poznan and Stanford HDCA light fields (central views) in JPEG Pleno dataset.

Synthetically created dataset

Greek and *Sideboard* are the two synthetically created light fields included in the JPEG Pleno dataset, made available by the Heidelberg Collaboratory for Image Processing (HCI) [19]. The central views are depicted in Figure 2.8. The light fields have 9×9 views each with a 512×512 spatial resolution.

(a) *Greek*



(b) *Sideboard*

Figure 2.8: Synthetic light fields (central views) in JPEG Pleno dataset.

2.4.2 Custom light field dataset

Many of the experiments performed during this research require several coding/decoding operations and computation of quality metrics such as the peak signal-to-noise ratio (PSNR). The process is very time consuming, since some light fields from the JPEG Pleno dataset have high resolution views, such as *Set2 2k Sub* with 1920×1080 and *Laboratory 1* with 1936×1288. Aiming at the reduction of the computational complexity to encode the content, low-resolution versions of the some light fields were created, using only the luminance information. The new light fields were created by extracting 4D blocks of size $t \times s \times v \times u = 10 \times 10 \times 384 \times 512$ from the original light fields from the JPEG Pleno dataset and horizontally stacking them, creating a collage. The resulting resolution is $t \times s \times v \times u = 10 \times 10 \times 384 \times 2048$. Figure 2.9 shows one view for each mosaic light field created. The creation process for each dataset follows:

- ***Mosaic Lenslets***: The central region of resolution $u \times v = 512 \times 384$ of the 10×10 central views of the light fields *Bikes*, *Danger de Mort*, *Stone Pillars Outside* and *Fountain & Vincent* were extracted and horizontally stacked to create a light field of dimensions $t \times s \times v \times u = 10 \times 10 \times 384 \times 2048$. A view of the created light field is shown in Figure 2.9 (a).

- ***Mosaic Set2 2K Sub*** and ***Mosaic Laboratory1***: Rectangular regions of resolution $u \times v = 512 \times 384$ of the $s \times t = 10 \times 10$ central views were extracted from random positions and horizontally stacked to create a light field of dimensions $t \times s \times v \times u = 10 \times 10 \times 384 \times 2048$. Views of the light fields *Mosaic Set2 2K Sub* and *Laboratory1* can be seen in Figure 2.9 (b) and (c), respectively.

- ***Tarot Cards***: Regions of size $u \times v = 512 \times 384$ were extracted from positions (u, v) equal to $(0, 0)$, $(512, 0)$, $(0, 384)$ and $(512, 384)$ from the $s \times t = 10 \times 10$ central views. The 4D blocks were horizontally stacked to create a light field of dimensions $t \times s \times v \times u = 10 \times 10 \times 384 \times 2048$. The central view is depicted in Figure 2.9 (d).

(a) *Lenslets*

(b) *Set2 2k Sub*

(c) *Laboratory 1*

(d) *Tarot Cards*

Figure 2.9: Mosaic light fields used in experiments.

Chapter 3

Characterizing light fields redundancy

In image and video coding, several techniques exploit spatial and temporal redundancy targeting bitrate reduction. In this context, transform coding plays a fundamental role. Transforming the data consists in decomposing spatial image sample values into basis functions and respective coefficient values, a process in which no information is lost. An important property of the transforms is energy compaction, in which most of the information is retained in low frequency coefficients while almost no information is carried by the high frequency ones. Many image coding solutions developed in last decades, among which it is worthwhile to highlight the JPEG standard, apply transforms directly to image samples. Transforms are also employed in modern image coding solutions such as H.264/AVC Intra and HEVC Intra, in which a block intra prediction scheme uses previously encoded block samples from its causal spatial neighborhood blocks to predict its block samples. In both standards, transform coding is used as subsequent step aiming at further decorrelating the prediction residue. Due to the fact that, in general, the similarity (correlation) among samples within a small neighborhood tends to be high, image compression schemes use large blocks to exploit redundancy in regions presenting a low degree of texture (spatial complexity), whereas smaller blocks are employed in regions exhibiting high spatial complexity.

As shown in Chapter 2, the light field data can be organized as a two-dimensional array of sub-aperture images which tend to present a strong degree of correlation among neighboring views. Hence, besides the already mentioned redundancy within each view, additional redundancy associated with the geometry of lenslet-based cameras and HDCA systems can be exploited for image coding purposes. Among other ways, the light field data compression can be achieved by jointly exploiting the light field's intra-view and inter-view redundancies. A straightforward way to do this is by employing four-dimensional (4D) transforms, with the 4D discrete cosine transform

(DCT) being a natural candidate for this type of processing considering the wide adoption of its two-dimensional version in several image and video coding standards [3] [20] [21]. This chapter is inserted within this context, aiming at studying 4D light fields redundancy. This is achieved by using a 4D transform (the 4D-DCT), in order to investigate the 4D sparsity of the light fields. The sparsity is related to how much of the energy of the signal is concentrated, for a given s, in the $s\%$ transform coefficients with largest variances.

3.1 4D-DCT definition

In the context of image and video transform coding, the DCT is an important tool. It has been adopted by the most popular image and video coding standards such as JPEG [3], H.264/AVC [20] and HEVC [22]. Its name comes from the fact that its basis functions are formed by cosines. Like other transforms, the DCT attempts to decorrelate the image data and generate a better representation suited for quantization and entropy coding.

An important DCT property is its ability to compact the energy of the image samples into a few coefficients. In this aspect, this transform has a near optimal performance, as it is derived as an approximation to the Karhunen-Loève (KL) transform which is the optimal solution in terms of energy compactness. A common definition of the DCT of a 1D sequence $x(u)$ of length U is

$$X(k) = \sum_{u=0}^{U-1} x(u) \cos\left[\frac{k\pi}{2U}(2u+1)\right]. \tag{3.1}$$

In image and video coding, one uses the separable 2D DCT. For light field coding, the 4D-DCT of a signal $x(u,v,s,t)$ can thus be written as a direct extension of the 1D case:

$$\begin{aligned} X(k,l,m,n) = \sum_{u=0}^{U-1}\sum_{v=0}^{V-1}\sum_{s=0}^{S-1}\sum_{t=0}^{T-1} x(u,v,s,t) \cos\left[\frac{k\pi}{2U}(2u+1)\right] \\ \cos\left[\frac{l\pi}{2V}(2v+1)\right] \cos\left[\frac{m\pi}{2S}(2s+1)\right] \cos\left[\frac{n\pi}{2T}(2t+1)\right]. \end{aligned} \tag{3.2}$$

Equation 3.2 can also be written as

$$\begin{aligned} X(k,l,m,n) = \sum_{u=0}^{U-1} \cos\left[\frac{k\pi}{2U}(2u+1)\right] \sum_{v=0}^{V-1} \cos\left[\frac{l\pi}{2V}(2v+1)\right] \\ \sum_{s=0}^{S-1} \cos\left[\frac{m\pi}{2S}(2s+1)\right] \sum_{t=0}^{T-1} x(u,v,s,t) \cos\left[\frac{n\pi}{2T}(2t+1)\right]. \end{aligned} \tag{3.3}$$

Equation 3.3 has the advantage that $X(k, l, m, n)$ can be computed in four steps by successive 1D operations on each dimension of the light field. This idea is graphically illustrated in Figure 3.1. In this work the 4D-DCT is computed as depicted in the flow, applying the DCT successively to t, s, v and u dimensions. The argument presented for the direct transform can be identically applied for the inverse 4D-DCT computation, therefore the inverse transform is computed by successively applying the IDCT to u, v, s and t dimensions, respectively.

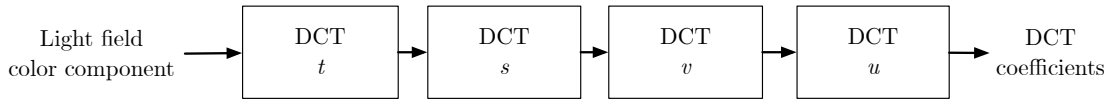


Figure 3.1: 4D-DCT processing pipeline.

3.2 Experimental framework

The light fields sparsity study is based on a simple experimental framework corresponding to the processing pipeline depicted in Figure 3.2. As a first step, a 4D data block is extracted from the luminance channel of the input light field, which is organized as a 2D array of views (e.g., for the lenslet light field images, these views are the sub-aperture images). In this work, the variables s and t represent the view coordinates while u and v represent the image coordinates within each (s, t) view. Next, a separable 4D-DCT (Figure 3.1) is applied to each 4D data block followed by a thresholding operation on the 4D-DCT coefficients, which basically performs a selection of the coefficients based on their energy. The 4D-DCT coefficients with values higher than a pre-defined threshold are retained, while the others are discarded. Naturally, the lower this threshold, the more coefficients are selected. The retained coefficients are used to reconstruct the 4D data blocks by using the inverse 4D-DCT. Finally, the full light field image is (lossy) recovered by appropriately combining the reconstructed blocks. The more coefficients are retained, the higher the quality of the recovered light field.

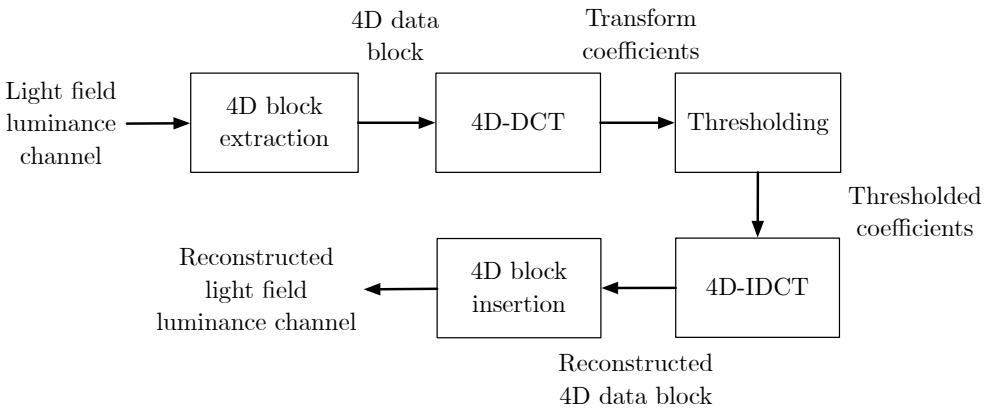


Figure 3.2: Experimental framework processing pipeline for the study of 4D sparsity of light fields.

Figure 3.1 illustrates the 4D-DCT processing pipeline. In this study, 4D-DCT sizes ($t \times s \times v \times u$) are $8 \times 8 \times 8 \times 8$ (4D), $1 \times 1 \times 8 \times 8$ (2D intra-view) and $8 \times 8 \times 1 \times 1$ (2D inter-view). While the $8 \times 8 \times 8 \times 8$ block size allows to jointly exploit the 4D redundancy, the same does not happen for the $1 \times 1 \times 8 \times 8$ and $8 \times 8 \times 1 \times 1$ block sizes which exploit only the intra or only the inter-view redundancies, respectively. In order to not mix different block sizes in one experiment, both inter-view and intra-view dimensions were truncated to the nearest multiple of 8. For lenslets light fields described in Section 2.4.1 only the central 624×432 portion is used out of original 625×434 resolution, for the central 8×8 sub-aperture views. This choice of sub-aperture views had the desirable consequence of avoiding to use their darkened views associated to the vignetting effect, explained in Section 2.2.1.

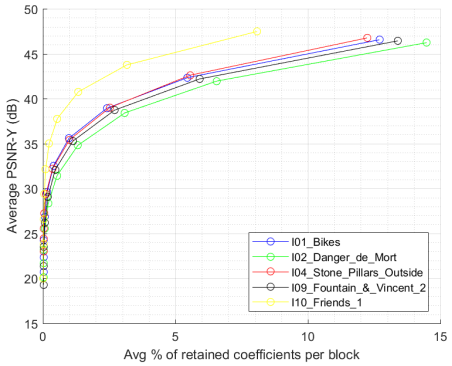
For the HDCA dataset described in Section 2.4.1, only *Set*2, *Set*6, *Set*9 and *Set*2 light fields and respective subsampled versions are analyzed. The use of both original and subsampled versions of HDCA light fields aims at understanding the impact of view subsampling on the light field redundancy. The same 4D-DCT sizes used for lenslets light fields are employed, with the addition of the $8 \times 8 \times 64 \times 64$ 4D-DCT (8×8 inter-view and 64×64 intra-view). The arrays of views are also truncated to the nearest multiple of 8, thus resulting in the choice of the central 96×16 views for the HDCA light fields out of the original 101×21 views and 32×8 out of the original 33×11views for the subsampled versions of the HDCA dataset. To further validate the experimental results and conclusions, the analysis includes a comparison with the Geometric Space-View Redundancy (GSVR) [23] descriptor results. The GSVR characterizes light fields in terms of space-view redundancy. Although it does not take into account the intra-view redundancy, it may bring interesting insights into the results of the experiment. The GSVR definition and results for the JPEG Pleno dataset are presented in Appendix C.

3.3 JPEG Pleno light field datasets sparsity: results and analysis

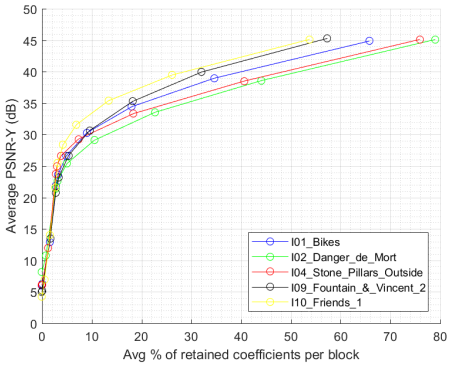
This section presents the results and conclusions of the sparsity study performed in both lenslets and HDCA JPEG Pleno datasets. The peak signal-to-noise ratio of the luminance information (PSNR-Y) between original and reconstructed views is averaged and plotted against the percentage of retained transform coefficients. The reconstruction quality expressed by the PSNR-Y represents the amount of energy carried out by the retained coefficients: high reconstruction qualities indicate that most of the energy is retained while low reconstruction qualities indicate that most of the energy was lost with the coefficients discarded in the thresholding process. By analyzing the relationship between energy concentration and the relative number of retained coefficients, it is possible to assess the light fields sparsity when a specific transform is applied.

3.3.1 Lenslet light field datasets

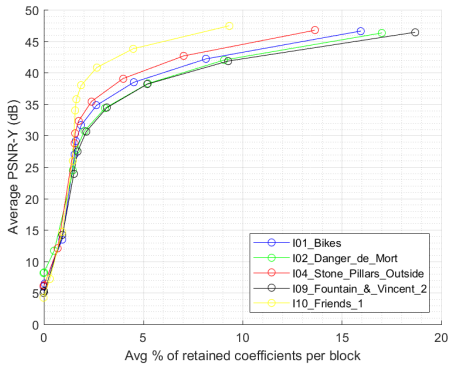
Figure 3.3 shows the sparsity results as expressed by the average PSNR-Y versus the average percentage of retained coefficients per block for the $8 \times 8 \times 8 \times 8$ 4D-DCT, 8×8 2D-DCT intra-view and 8×8 2D-DCT inter-view, for all the lenslets light fields selected for this experiment. The charts show that, for a same number of retained coefficients, the light field *Friends* has a higher reconstruction quality, which means that for this light field more energy is retained in the same number of coefficients. The conclusion is that *Friends* has higher 4D sparsity than the other ones.



(a) $8 \times 8 \times 8 \times 8$ 4D-DCT.



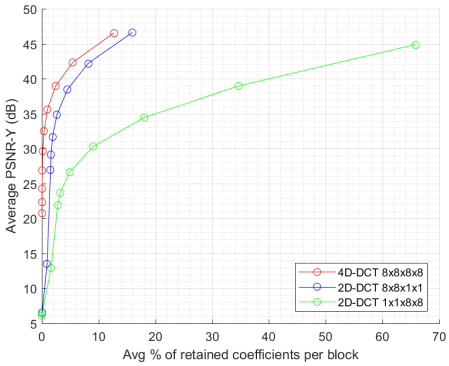
(b) 8×8 2D-DCT intra-view.



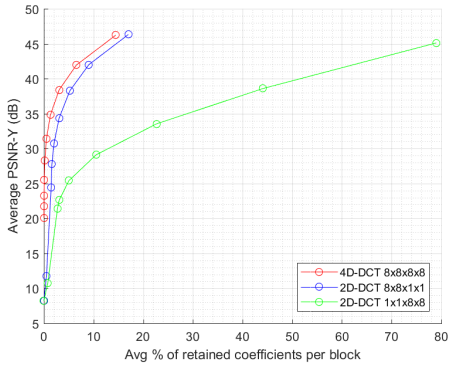
(c) 8×8 2D-DCT inter-view.

Figure 3.3: Average PSNR-Y versus average percentage of retained coefficients per block for 4D-DCT, 2D-DCT intra-view and 2D-DCT inter-view: comparison among lenslets light fields.

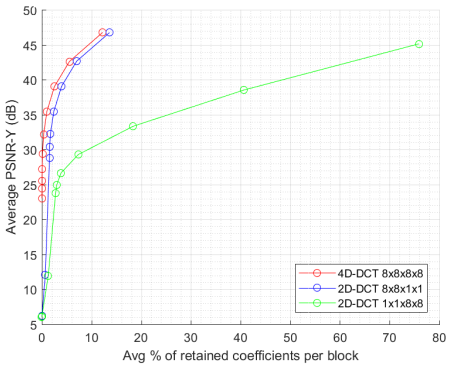
Figure 3.4 shows the same type of sparsity analysis but now comparing results for different block sizes for each lenslets light fields, thus able to exploit different types and amounts of redundancy. The 4D-DCT presents a higher reconstruction quality for a same number of average retained coefficients, leading to the conclusion that, for this transform, more energy is concentrated than for its 2D counterparts. This result implies that it is worthy to exploit the 4D redundancy in a DCT-based coding scheme for lenslets. In addition, the inter-view sparsity is larger than the intra-view one, indicating that for the lenslets it is more effective to use an inter-view transform than an intra-view one. As can be seen, the same behavior is observed for all analyzed lenslet light fields.



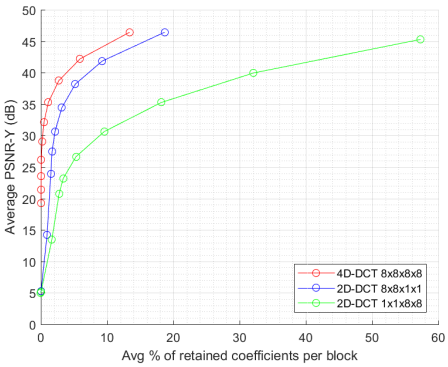
(a) *Bikes.*



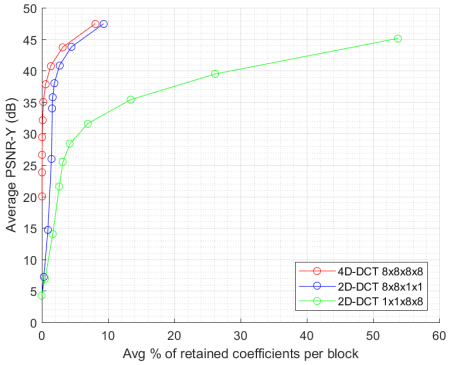
(b) *Danger de Mort.*



(c) *Stone Pillars Outside.*



(d) *Fountain & Vincent 2.*



(e) *Friends 1.*

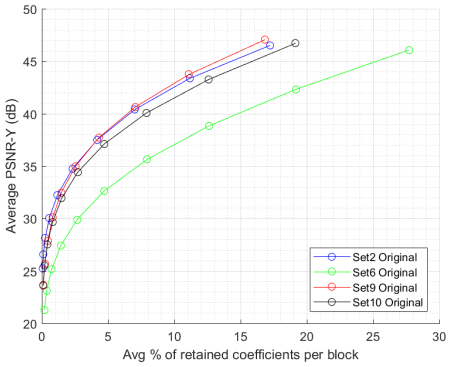
Figure 3.4: Average PSNR-Y versus average percentage of retained coefficients per block for lenslets datasets: comparison of $8 \times 8 \times 8 \times 8$ 4D-DCT, inter-view 8×8 2D-DCT and intra-view 8×8 2D-DCT.

The presented results are confirmed by the GSVR [23] descriptor results analysis. The GSVR is a quantitative descriptor which aims at characterizing the light fields defining the largest region in the 4D space that presents four-dimensional geometric redundancy worthy to be exploited for encoding purposes. It does so by measuring the probability that the image of a point in 3D space to remain in the space block in all views, for each space block size and range of views. The GSVR descriptor gives the relation between the intra-view and inter-view block dimensions guaranteeing, with a given probability, the existence of such 4D space-view redundancy. Lower values of GSVR(p) indicate that large amount of 4D redundancy is present on a light field. The derivation process and definition of the GSVR descriptor, as well as the corresponding analysis of the JPEG Pleno light field datasets may be found in Appendix C. The GSVR results in Figure C.5 (a) for the lenslets light fields show that for a permanence probability of 80%, GSVR(p) assumes values around 0.4, which means that the inter-view block-sizes of the 4D block should be 2.5 times the intra-view sizes. This confirms that the inter-view redundancy is indeed larger than

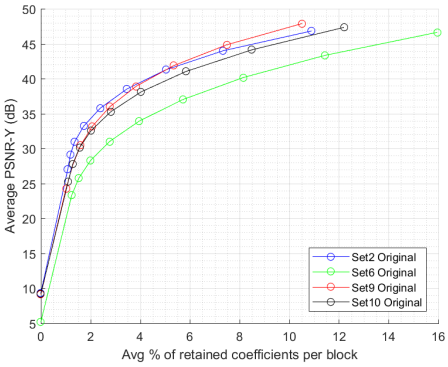
the intra-view one for the lenslet light fields since one should have a larger inter-view block dimension for the given permanence probability.

3.3.2 HDCA light field datasets

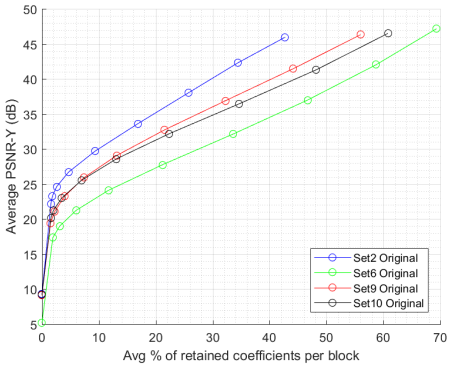
Figure 3.5 shows the sparsity results as expressed by the average PSNR-Y versus the average percentage of retained coefficients per block for the $8 \times 8 \times 8 \times 8$ 4D-DCT, 8×8 2D-DCT intra-view and 8×8 2D-DCT inter-view for all original (101×21) HDCA datasets. For a same number of retained coefficients, the reconstruction quality for *Set*6 is smaller than the obtained for other light fields and, therefore, less energy is retained. The results thus show that the *Set2, Set9* and *Set10* datasets are approximately equivalent regarding the 4D-DCT sparsity, while the sparsity of *Set6* is smaller.



(a) $8 \times 8 \times 8 \times 8$ 4D-DCT.



(b) 8×8 2D-DCT intra-view.

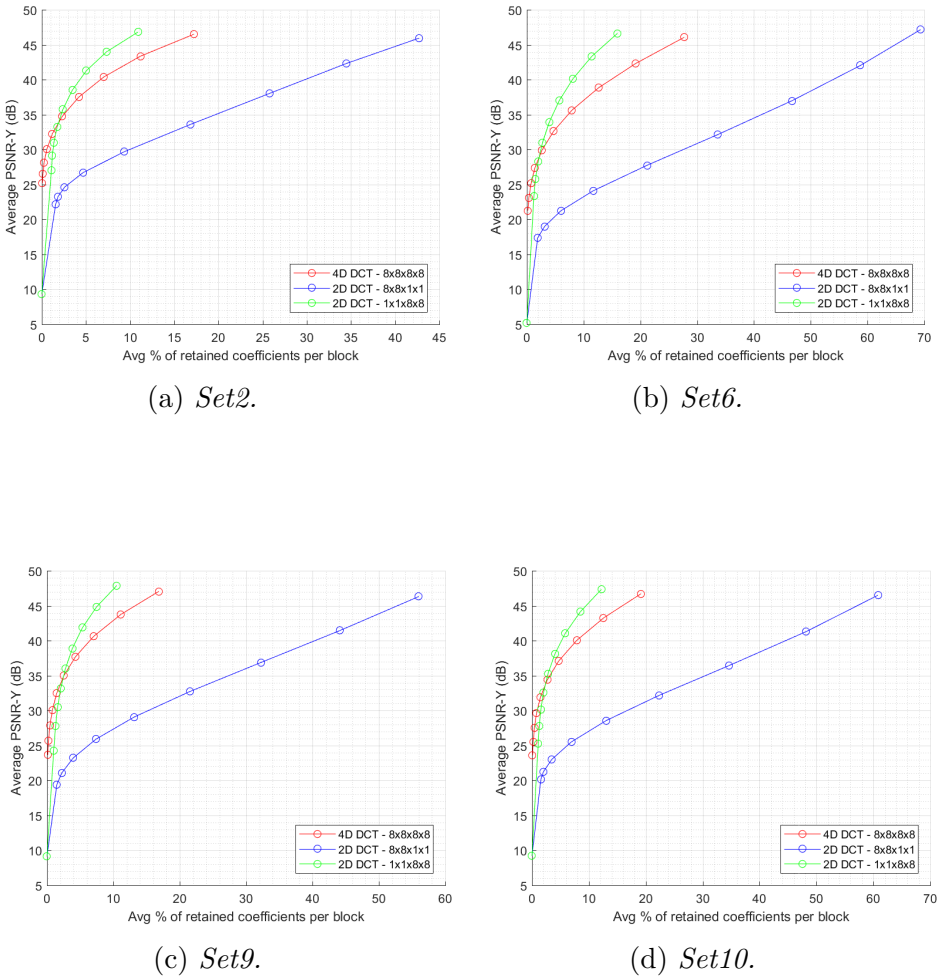


(c) 8×8 2D-DCT inter-view.

Figure 3.5: Average PSNR-Y versus average percentage of retained coefficients per block for 4D-DCT, 2D-DCT intra-view and 2D-DCT inter-view: comparison among original HDCA datasets.

Figure 3.6 compares the average PSNR-Y versus the average percentage of retained coefficients per block results for three different block sizes, for each selected

HDCA light field. For the majority of average percentages of retained coefficients tested, there is more energy concentration in the intra-view coefficients than in the other transforms. Thus, unlike the lenslets datasets where the 4D sparsity is significantly larger than the 2D sparsities, for the HDCA datasets the sparsity is dominated by the intra-view redundancy. As can be seen, the same behavior is observed for all analyzed HDCA light fields.



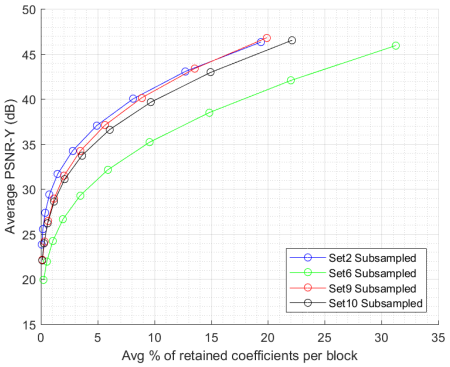
(a) *Set2.* (b) *Set6.*

(c) *Set9.* (d) *Set10.*

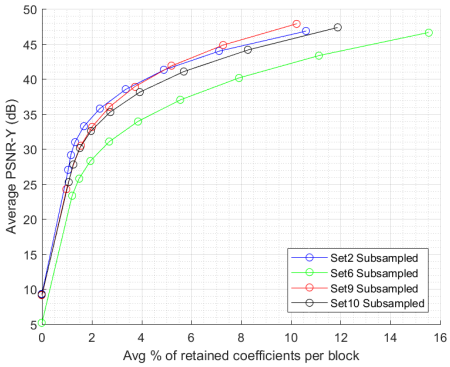
Figure 3.6: Average PSNR-Y versus average percentage of retained coefficients per block for light fields in HDCA dataset: comparison of $8 \times 8 \times 8 \times 8$ 4D-DCT, inter-view 8×8 2D-DCT and intra-view 8×8 2D-DCT.

Figure 3.7 shows the average PSNR-Y versus the average percentage of retained coefficients per block results for the $8 \times 8 \times 8 \times 8$ 4D-DCT, 8×8 2D-DCT intra-view and 8×8 2D-DCT inter-view transforms for selected subsampled light fields from the HDCA dataset, while Figure 3.8 compares the 4D-DCT sparsity to the 2D inter-view $(8 \times 8 \times 1 \times 1)$ and 2D intra-view $(1 \times 1 \times 8 \times 8)$ sparsities for the same test material. The behavior is very similar as for the corresponding original HDCA light fields (Figures 3.5 and 3.6) thus implying that, for the subsampled HDCA light

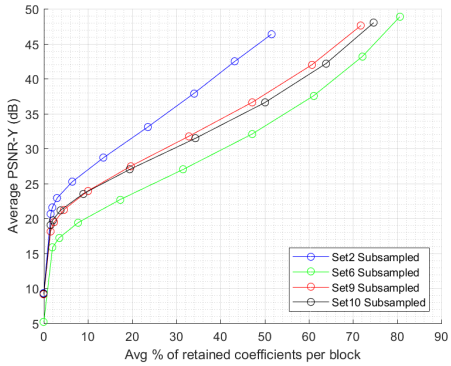
fields, the sparsity is also dominated by the intra-view redundancy, with a much smaller inter-view redundancy.



(a) $8 \times 8 \times 8 \times 8$ 4D-DCT.

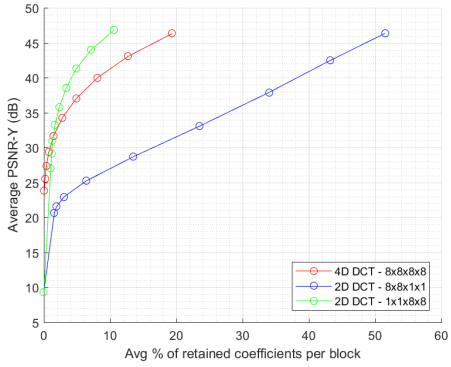


(b) 8×8 2D-DCT intra-view.

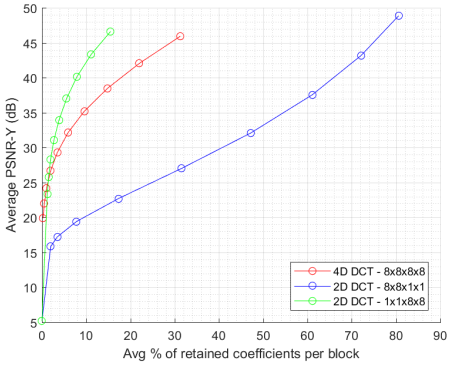


(c) 8×8 2D-DCT inter-view.

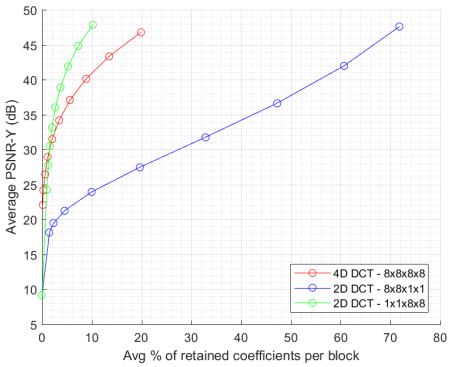
Figure 3.7: Average PSNR-Y versus average percentage of retained coefficients per block for 4D-DCT, 2D-DCT intra-view and 2D-DCT inter-view: comparison among subsampled HDCA light fields.



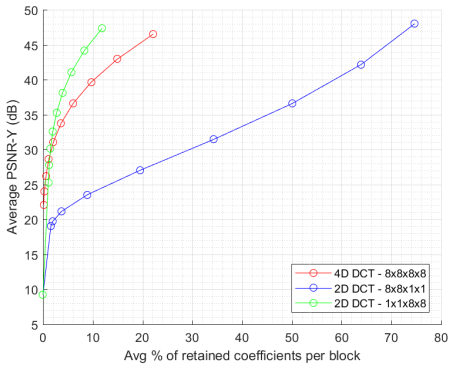
(a) *Set2.*



(b) *Set6.*



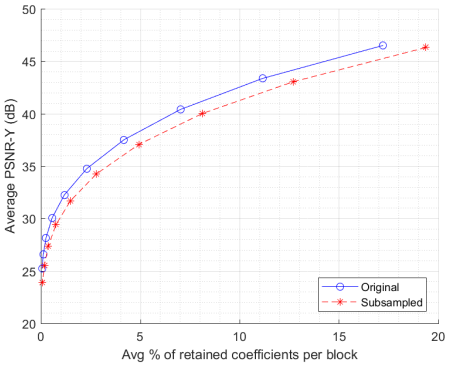
(c) *Set9*.



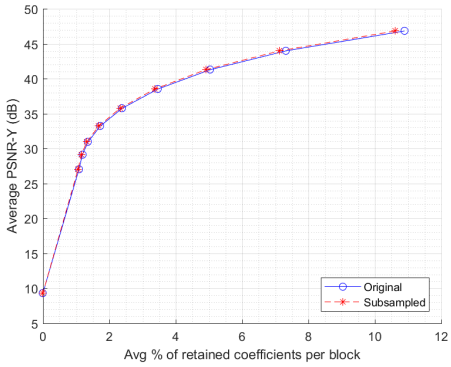
(d) *Set10*.

Figure 3.8: Average PSNR-Y versus average percentage of retained coefficients per block for subsampled HDCA light fields: comparison of $8 \times 8 \times 8 \times 8$ 4D-DCT, inter-view 8×8 2D-DCT and intra-view 8×8 2D-DCT.

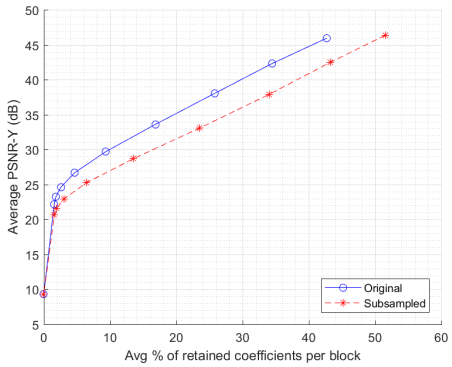
In the sequel, to understand the effects of subsampling the HDCA light fields, the results for both the original and subsampled datasets are compared. Figures 3.9 (a), (b) and (c) compare the 4D-DCT, 2D intra-view DCT and 2D inter-view DCT sparsity results, respectively, for the original and subsampled HDCA *Set2*. As expected, the intra-view sparsities of both datasets are nearly same, since the intra-view redundancy is loosely affected by the process of discarding views. The results in Figure 3.9 (b) comparing the inter-view sparsities are also expected, since the view subsampling process results into a sparser and less redundant set of views which results in a reduction of the sparsity. Since there is a decrease in the inter-view sparsity, and the intra-view sparsity remains unchanged, it is natural that the 4D sparsity also decreases for the subsampled HDCA dataset, as shown in Figure 3.9 (c).



(a) $8 \times 8 \times 8 \times 8$ 4D-DCT.



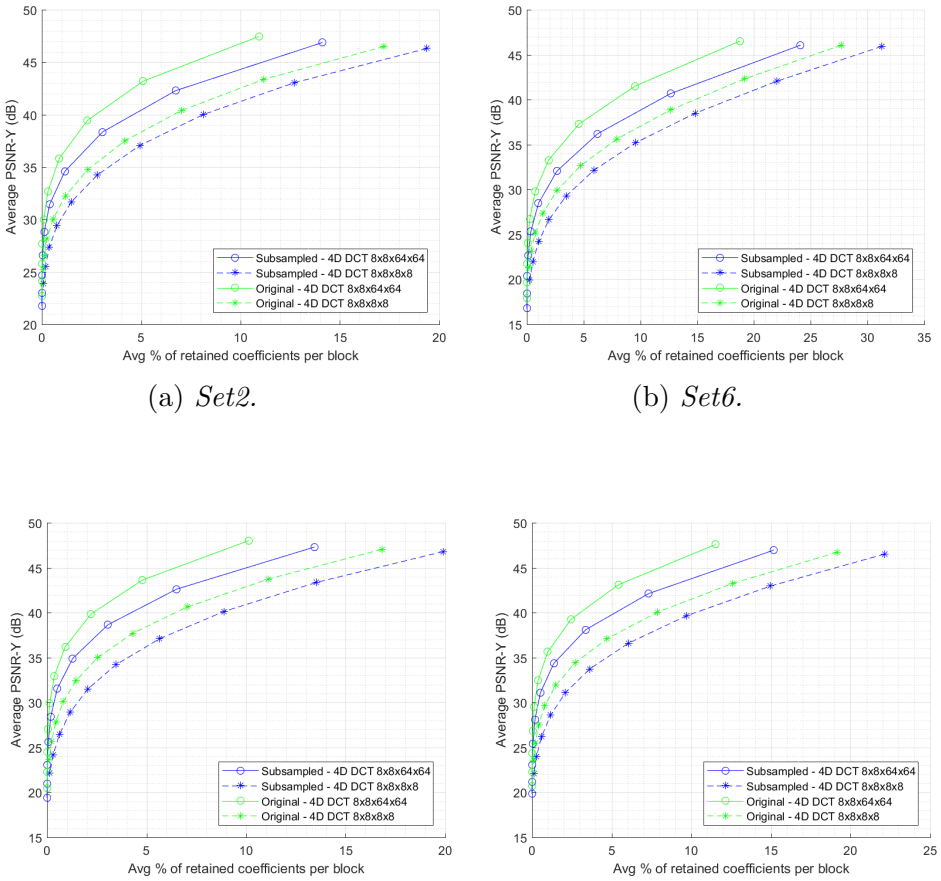
(b) 8×8 2D-DCT intra-view.



(c) 8×8 2D-DCT inter-view.

Figure 3.9: Average PSNR-Y versus average percentage of retained coefficients per block for HDCA *Set2*: comparison between original and subsampled datasets.

Finally, the last comparison between original and subsampled HDCA datasets is shown in Figure 3.10. It compares the sparsity of the $8 \times 8 \times 8 \times 8$ and the $8 \times 8 \times 64 \times 64$ 4D-DCT for both datasets. The difference between these two DCT sizes is only on the intra-view dimensions (8×8 and 64×64). It is possible to observe that, for the same average percentage of retained coefficients, the difference in PSNR values between the two 4D-DCT sizes is higher for the original dataset. Since the difference between the two datasets is only on the inter-view redundancy, this allows concluding that the intra-view block size impacts on the exploitation of the inter-view redundancy, with larger intra-view dimensions being better. This is in accordance with the results displayed in Figure C.5 (b) showing the GSVR curves for the original and subsampled HDCA datasets. For a permanence probability of 0.8, an intra-view dimension around 8 times larger than the inter-view's one should be used for the original HDCA dataset, which, for 8×8 inter-view dimensions requires intra-view dimensions of around 64×64. On the other hand, an intra-view dimension around 20 times larger than the inter-view dimension should be used for the subsampled HDCA dataset, which would require intra-view dimensions of around 160×160 for the 8×8 inter-view dimensions. Then, an $8 \times 8 \times 64 \times 64$ block would be adequate for exploring the 4D redundancy in the original HDCA dataset, while it would not be sufficient for the subsampled HDCA, that would require an $8 \times 8 \times 160 \times 160$ 4D-DCT size. This explains the fact that the sparsity increases from $8 \times 8 \times 8 \times 8$ to $8 \times 8 \times 64 \times 64$ is much larger for the original than for the subsampled HDCA dataset. In addition, one should note that, if the intra-view dimensions are too large, the intra-view redundancy across the block becomes smaller and cannot be well exploited.



(a) *Set2.* (b) *Set6.*

(c) *Set9.* (d) *Set10.*

Figure 3.10: Average PSNR-Y versus average percentage of retained coefficients per block for HDCA datasets, original and subsampled: comparison of $8 \times 8 \times 8 \times 8$ and $8 \times 8 \times 64 \times 64$ 4D-DCT.

3.4 Conclusions

From the above results, it is possible to conclude that lenslets dataset has a significant amount of 4D sparsity, the inter-view redundancy being significantly larger than the intra-view redundancy. For the HDCA dataset, the opposite is observed: the intra-view sparsity is much larger than the inter-view sparsity. In addition, the HDCA dataset has a much smaller amount of 4D redundancy than the lenslet dataset. Thus, it is likely that the most efficient coding solutions for the lenslet datasets are not the most efficient for the HDCA datasets.

The presented study of the 4D sparsity of the JPEG Pleno light field images shows that both the lenslet and HDCA datasets have a great amount of 4D redundancy that can be explored, notably for coding purposes. As a consequence, the results also suggest that not exploiting the 4D redundancy as a whole may be a severe limitation to the design of emerging light field codecs, notably the one currently under development in JPEG Pleno. The presented results also suggest that

the lenslet and HDCA datasets may require distinct coding solutions due to the different nature of their 4D redundancy. One should bear in mind that these conclusions are restricted to the JPEG Pleno datasets, whereas a more extensive study needs to be carried out to characterize the inter-view and intra-view redundancies of more general light field data.

Another important information towards the development of a 4D-DCT based codec is the energy distribution among transform coefficients. In the next chapter, an experiment aiming at characterizing the energy distribution over the 4D-DCT coefficients and defining efficient coefficients scanning patterns for coding purposes is performed.

Chapter 4

4D-DCT coefficients scanning modes

For 2D-DCT based image codecs, a key step consists in scanning the DCT block coefficients according to a certain pattern such that the coefficients with larger energy are scanned earlier, targeting maximum energy compaction. As transforms applied to redundant information tend to generate sparse representations [24], the resulting coefficients should be appropriately ordered so that those with larger energy are scanned before those with smaller energy enabling the obtained sparsity to be effectively exploited for compression purposes using run-length coding [25]. Many scanning solutions of 2D transform coefficients have been developed and are part of image and video coding standards. However, due to the richer 4D space structure of the light fields as compared to that of 2D conventional images, the extension of such scanning patterns from 2D to 4D is not trivial. In this context, this chapter investigates, proposes and assesses 4D scanning modes for light field 4D-DCT coefficients, targeting efficient coding using the run-length technique. The energy compactness power of the proposed 4D-DCT scanning modes is evaluated using a proposed metric called the integral of cumulative energy.

4.1 Run-length coding

Run-length encoding is a simple lossless data compression technique in which a sequence formed by the consecutive occurrence of a symbol is encoded as a pair (*value*, *run*) rather than as the original sequence [25], where the token *value* identifies the given symbol and *run* informs the number/length of its repetitions. The compression efficiency of this technique is improved by the increase of the length and frequency of occurrence of such runs.

Variations of the run-length coding method are usually employed in transform-

based lossy image and video codecs in order to save bits with the representation of frequent zero-valued quantized transform coefficients. One example is the JPEG standard, in which each nonzero coefficient (except the top-left DC coefficient) is represented in combination with the consecutive number of zero-valued coefficients which precede it in the scanning order [3]. Two symbols are used to represent each nonzero coefficient: the first symbol has the form (*run*, *size*), where *run* is the number of zero-valued coefficients preceding the nonzero coefficient and *size* is number of bits used to represent its amplitude. The second symbol contains the amplitude of the coefficient itself using *size* bits [3].

Figure 4.1 illustrates a run-length coding process similar to the one performed in JPEG standard. A 4×4 matrix of coefficients with the nonzero coefficients concentrated towards the top-left corner is scanned using three different patterns: (a) vertical, (b) horizontal and (c) diagonal. The tables below the scanning patterns include a line for each nonzero coefficient. The first column contains the coefficient amplitude while the second one contains the run of zeros before the respective coefficient. The symbols (*run*, *level*) resulting from the encoding using the horizontal scanning pattern (a) contain *run* values equal to zero or equal to two. For the vertical scanning pattern (b), four different *run* values of *run* are obtained. Finally, for the diagonal scanning pattern (c), just runs equal to zero are generated.

Among the tested patterns, the best result is obtained by the diagonal scanning, because the respective ordering, that tends to place low-frequency coefficients before high-frequency ones, facilitates entropy coding by concentrating the *run* values around zero.

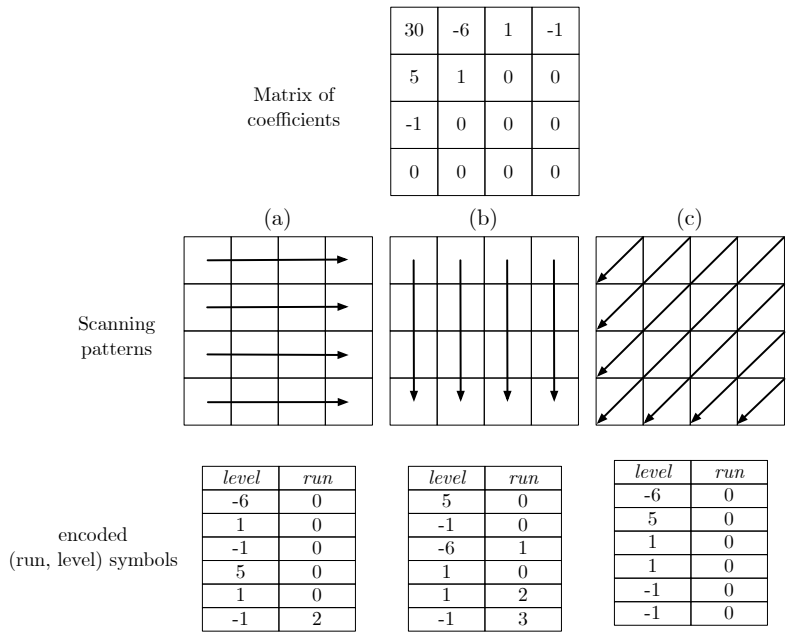


Figure 4.1: Example of run-length coding of a matrix of coefficients using three distinct scanning patterns.

The example shown in Figure 4.1 allows concluding that an appropriate ordering of the values before applying run-length coding impacts the final compression performance. The next section presents some of the scanning solutions used in several image and video coding standards.

4.2 DCT coefficients scanning in available standards

As far as the 2D case is concerned, effective scanning modes exist and have been largely used in several image and video coding standards. The JPEG image coding standard employs the classic zig-zag scanning of the 2D-DCT coefficients of 8×8 blocks [3]. This scanning pattern consists in scanning the coefficients in a direction perpendicular to the diagonal of the block starting from the lowest frequency coefficient at the top left and finishing at the highest frequency coefficient in the bottom right, alternating the orientation at every iteration. This so-called zig-zag scanning pattern is shown in Figure 4.2.

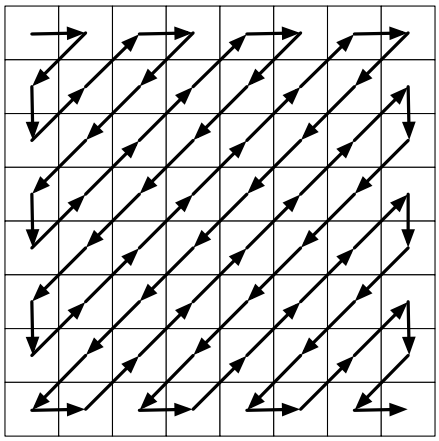


Figure 4.2: Classic zig-zag scanning pattern used in JPEG standard.

MPEG-2 Video [26] also employs 2D-DCT and uses the same zig-zag scanning pattern for progressive video frames. However, for the fields of interlaced frames or the fields of progressive frames converted from interlaced content, vertical frequencies tend to have higher energies than the horizontal ones, since the field is vertically sub-sampled from the original content. This means that non-zero coefficients tend to be concentrated at the top and towards the left side of the block. In order to visit the coefficients with larger energies first, such block should be scanned with a variation of the classic zig-zag pattern, depicted in Figure 4.3.

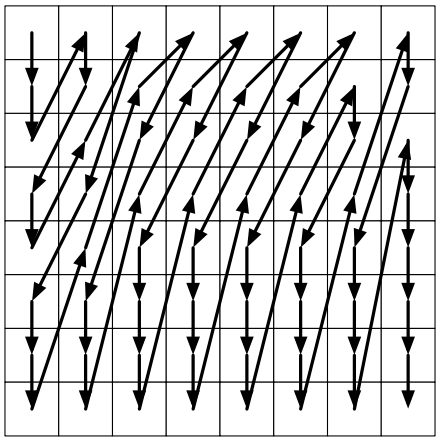


Figure 4.3: Zig-zag pattern variation for field scanning used in MPEG-2 Video coding standard.

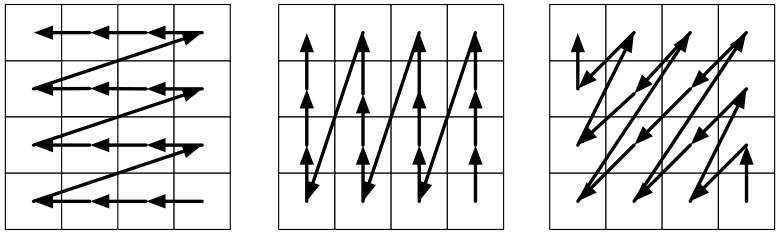
In H.264/AVC [20] standard, currently the most popular video coding solution, the same MPEG-2 Video standard scanning patterns are still used for the 8×8 DCT. However, H.264/AVC also employs 4×4 transforms, the scanning patterns shown in Figure 4.4 are also used.

(a) Progressive scan (b) Field scan

Figure 4.4: Zig-zag pattern variation for field scanning used in H.264/AVC standard.

HEVC [21] is the successor of H.264/AVC and one of the most efficient video codecs nowadays. HEVC employs transforms in square regions called transform blocks (TB) whose sizes ranging from 4×4 to 32×32, varying in powers of two. However, the coefficients scanning in larger TBs is performed in 4×4 by 4×4 transform sub-blocks (TSBs), originated from the non-overlapping decomposition of the TB [27]. Such a decomposition is possible since the TB sizes are always multiple of four. The three scanning patterns used are depicted in Figure 4.5. Differently from previous standards, the scanning starts at the bottom-right coefficient and ends in the top left coefficient. The first possibility is a horizontal pattern which scans the rows of the block from the bottom to the top, as depicted in Figure 4.5 (a). The second possibility is the vertical pattern shown in Figure 4.5 (b) which scans the columns of the block from the bottom to the top. The last possibility is the

diagonal scan illustrated in 4.5 (c), which scans the blocks coefficients in diagonal direction from the right to the left. For inter coded blocks, a diagonal scan is used for all block sizes. For intra coded blocks of size 4×4 or 8×8, the three scanning patterns diagonal, horizontal, and vertical are available. The applicable scanning pattern depends on the direction of the intra prediction mode. For blocks of size 16×16 and 32×32, only the diagonal scan is applied.



(a) Horizontal (b) Vertical (c) Diagonal

Figure 4.5: Scanning patterns used in HEVC standard.

The several scanning modes for 2D-DCT presented in this section should serve as inspiration for the design of novel scanning modes to be used with the 4D-DCT. However, the extension of 2D scanning patterns to 4D is far from obvious, since the 4D space allows for a much richer set of scanning patterns.

As discussed in this section, the definition of the scanning patterns should be driven by the distribution of energy among the transform coefficients. The typical 2D-DCT coefficients energy distributions for images and video are well known and have been widely discussed in the literature [28]. However, very few information can be found about the energy distribution of 4D-DCT coefficients when applied to light fields. In the next section, an analysis of the energy distribution of 4D-DCT coefficients for light fields from the JPEG Pleno dataset is performed, whose results will enable the future design and proposal of scanning patterns for the 4D-DCT coefficients.

4.3 4D-DCT coefficients energy distribution analysis

The objective of the transform coefficients scanning process is to generate an ordered list of coefficients such that the coefficients with higher energy are placed before those with lower energy as this helps improving the final rate-distortion (RD) performance. To make a proper design of the 4D scanning modes for light field 4D-DCT coefficients, it is important to first understand the nature of the light field

data to be 4D-DCT compressed. In this section, this goal is targeted by analyzing the energy distribution of the 4D-DCT coefficients of representative light fields.

The light fields in the JPEG Pleno dataset described in Section 2.4.1 may be divided in tree types: the lenslet dataset, the HDCA dataset and the synthetic dataset. As observed in Chapter 3, light fields acquired with lenslets light field cameras tend to have an inter-view redundancy much larger than the intra-view one, with the opposite happening for the HDCA light fields which tend to have an intra-view redundancy much larger than the inter-view one; on the other hand, the synthetic light fields from JPEG Pleno dataset tend to have equivalent intra and inter-view redundancies. Therefore, the 4D-DCT scanning patterns that are most efficient for lenslets light fields may be different from the ones most efficient for the HDCA and synthetic light fields.

In this section, the energy distribution of the 4D-DCT coefficients for one lenslet light field (*Bikes*), one HDCA light field (*Set2 2K Sub*) and one synthetic light field (*Greek*) are analyzed. Appendix B presents more extensive results of energy distribution for the remaining light fields in JPEG Pleno dataset. The energy distribution computation is carried out as the following steps, presented in Figure 4.6:

1. **Block extraction:** 4D blocks of size $8 \times 8 \times 8 \times 8$ are extracted from the input light field (while this size has been chosen because its DCT can be efficiently computed, the specific block size should not affect the overall conclusions);

2. **4D-DCT:** A separable 4D-DCT as described in Section 3.1 is applied to each 4D block of luminance samples;

3. **Coefficient energy estimation**: The energy of each 4D-DCT coefficient is estimated as the variance computed over all the blocks in a light field for each 4D-DCT coefficient position [28].

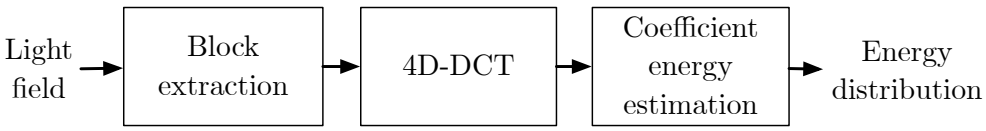
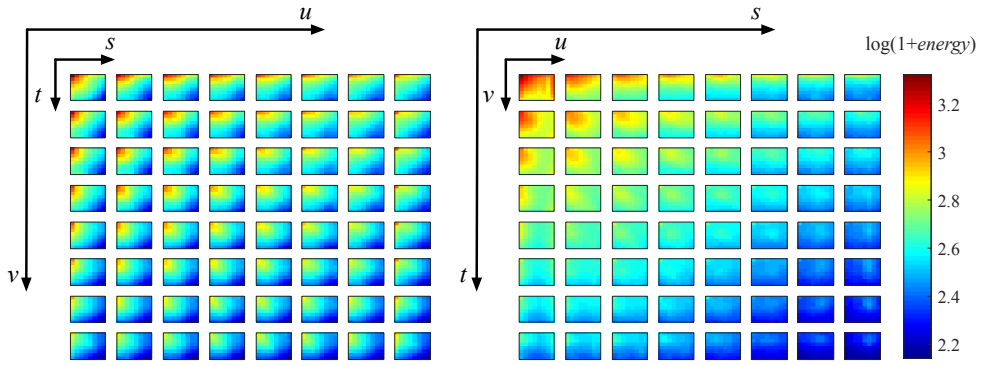


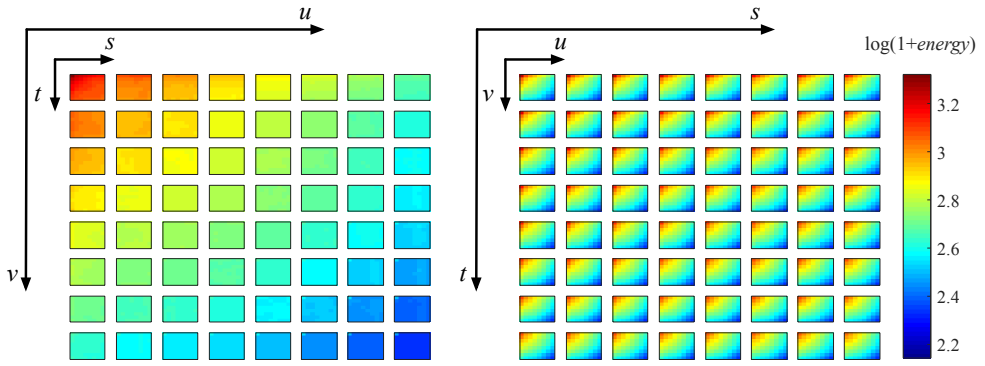
Figure 4.6: 4D-DCT coefficients energy distribution estimation pipeline.

Figure 4.7 shows the energy distribution of the 4D-DCT coefficients for the lenslet (*Bikes*), HDCA (*Set2 2K sub*) and synthetic (*Greek*) light fields. The coordinates (s, t) correspond, respectively, to the horizontal and vertical coordinates of a view in the array and the coordinates (u, v) to the horizontal and vertical coordinates of a sample within a view. In Figure 4.7 (left), the 4D structure is represented as a 2D array of $s \times t$ slices, each slice indexed by the coordinates (u, v), while in Figure

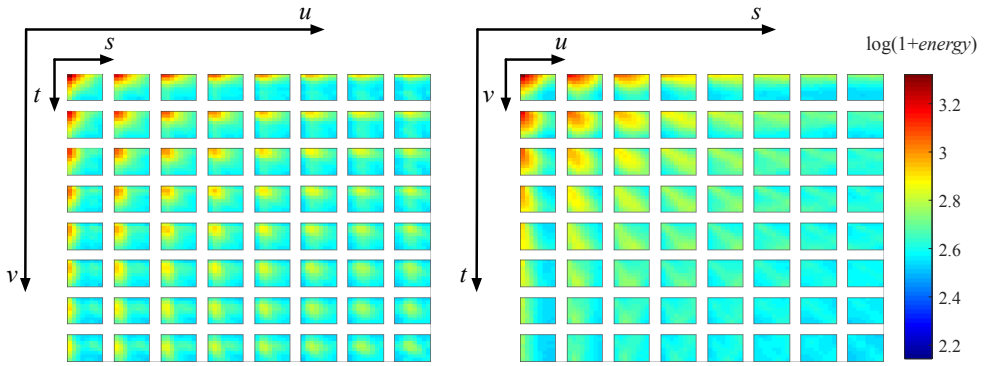
4.7 (right) it is represented as a 2D array of $u \times v$ slices, each slide indexed by the coordinates (s, t).



(a) Light field *Bikes*.



(b) Light field *Set2 2k Sub*.



(c) Light field *Greek*.

Figure 4.7: Energy distribution estimation for the 4D-DCT coefficients of three types of light fields.

The analysis of Figure 4.7 (a) leads to the conclusion that, for the lenslet light field *Bikes*, the energy decay is much slower along the (u, v) coordinates than along the (s, t) coordinates, which is coherent with the already mentioned fact that, for

the lenslet light fields, the inter-view redundancy is larger than the intra-view redundancy. Thus, in order to boost the probability of having the larger energy coefficients scanned before the lower energy ones, the scanning must be performed according to the representation in Figure 4.7 (a, right), that is, by scanning the coefficients (u, v) within each view in an inner loop, and scanning each view (s, t) in an outer loop; this is the consequence of a larger inter- than intra-view redundancy.

In Figure 4.7 (b), it is possible to observe that, for the HDCA light field *Set2 2K sub*, the roles of dimensions (s, t) and (u, v) are reversed, and thus the scanning should be performed differently and according to the configuration in Figure 4.7 (b, left), the inner scanning loop being on the dimensions (s, t); this is the consequence of a larger intra- than inter-view redundancy.

Finally, Figure 4.7(c) shows that, for the light field *Greek*, the 4D-DCT coefficients show equivalent energy decay patterns along the (u, v) and (s, t) coordinates, and thus a good scanning pattern for the light field *Greek*, i.e., one in which the larger energy coefficients are visited earlier, is one where the coefficients are scanned within hyperplanes perpendicular to the diagonal of the hyper-parallelepiped. The complete set of results, also including the other light fields in JPEG Pleno dataset, are available in Appendix B. In the sequel, different 4D-DCT scanning modes are proposed to be selected as appropriate to maximize the RD performance of 4D-DCT-based light field codecs using run-length coding.

4.4 4D-DCT coefficients scanning modes design

This section presents the proposed scanning modes for the 4D-DCT coefficients, which are extensions to 4D of the scanning modes used by image and video coding standards presented in Section 4.2. The energy distributions presented in Section 4.3 play a fundamental role in the design process, as the main scanning process objective is reordering the coefficients in a decreasing order according to their energy.

The design of the 4D-DCT scanning modes follows a hierarchical approach. The starting point is the 4D hyperparallelepiped, as shown in Figure 4.8. This 4D hyperparallelepiped can be scanned using a pattern called *4D diagonal* or it can be partitioned into lower dimensional structures. One possibility is the decomposition into a linear array of 3D parallelepipeds, as shown on the left of Figure 4.8. Other possibility is the decomposition into a two-dimensional array of 2D rectangles, as depicted on the right of Figure 4.8.

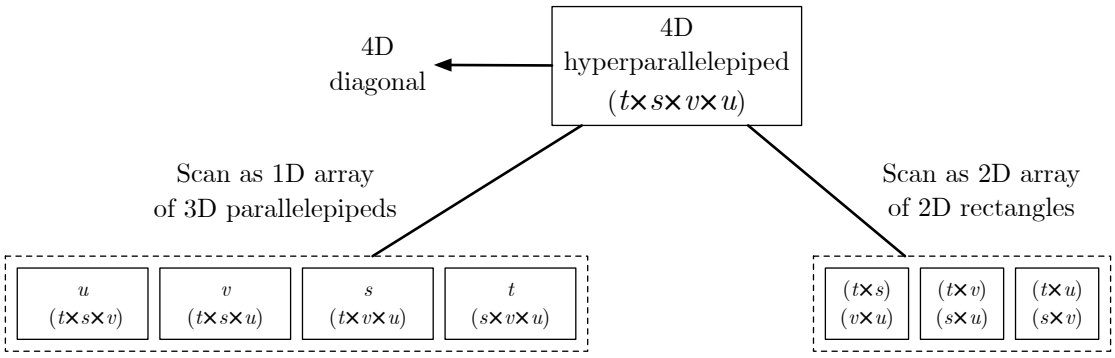


Figure 4.8: Three scanning possibilities for a 4D hyperparallelepiped: 4D diagonal, 1D array of 3D parallelepipeds or 2D array of 2D rectangles.

Next, the process proceeds to its second level. The 3D parallelepipeds generated in the previous level can be scanned using a pattern called *3D diagonal* or can be further partitioned into lower dimensional structures, as shown in Figure 4.9. The only possibility of decomposition is into a linear array of 2D rectangles. At the same level, each 2D rectangle in the two-dimensional array can be scanned using a pattern called *2D diagonal* or can be further partitioned into one-dimensional structures, corresponding to scanning its lines or columns. The patterns used in a rectangle are similar to the ones depicted in Figure 4.5, just changing the orientation of the scan to start in the top-left and end in the bottom-right coefficients.

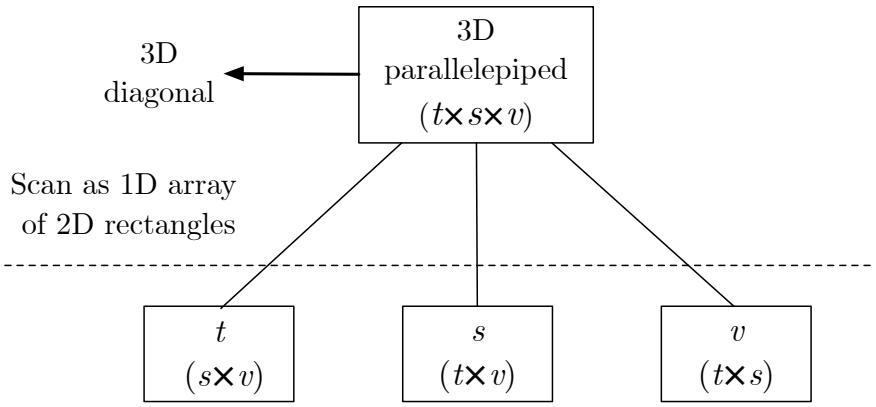


Figure 4.9: Two scanning possibilities for a 3D parallelepiped: 3D diagonal or linear array of rectangles.

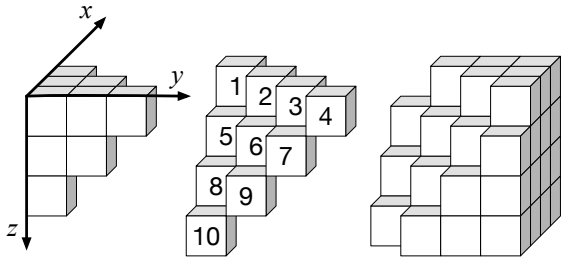
In the third and last level, the rectangles generated from the decomposition of a parallelepiped can be scanned using the same vertical, horizontal or diagonal patterns employed in the scanning of the rectangles generated in level 2. Thus, all possibilities of scanning the 4D hyperparallelepiped are covered. In the sequel, each scanning pattern is described and implementations using pseudocode are presented.

4.4.1 4D diagonal scanning mode

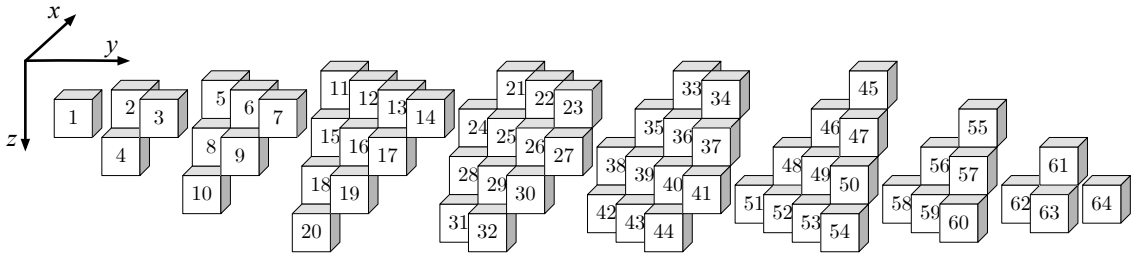
Following the previous demonstration that different scanning patterns are needed, a set of 4D-DCT scanning modes to be used for light field coding is here proposed, starting with the *4D diagonal* scanning, regarded as an extension to 4D of the *3D diagonal* scanning, which is itself an extension of the *2D diagonal/zig-zag* scanning. The non-diagonal 4D patterns are defined as combinations of the *3D diagonal*, *2D horizontal*, *2D vertical* and *2D diagonal* patterns.

In the *2D diagonal* scanning mode, the coefficients are scanned along a direction perpendicular to the main diagonal of the coefficients' matrix, that is, the direction along which the sum of the coordinates of each coefficient is equal to a constant, as depicted in Figure 4.5 (c) for a 4×4 block.

The *3D diagonal* scanning mode of a parallelepiped of coefficients is an extension of the *2D diagonal* scanning mode to 3D where the coefficients are scanned along planes perpendicular to the main diagonal of the 3D parallelepiped. Such planes are characterized by the constraint $x + y + z = c$ on the coefficients coordinates (x, y, z). Figure 4.10 (a) depicts a section of the *3D diagonal* scanning mode for a $4 \times 4 \times 4$ block of coefficients, where the highlighted blocks respect the constraint $x + y + z = 4$, while Figure 4.10 (b) shows the full scanning order for the block.



(a) Diagonal plane for $x + y + z = 4$ and respective scanning order within the plane.



(b) All diagonal planes and respective scanning order.

Figure 4.10: 3D diagonal scanning of a $4 \times 4 \times 4$ parallelepiped.

Likewise, the *4D diagonal* scanning mode may be derived from the 3D one by scanning the coefficients along hyperplanes orthogonal to the main diagonal of the 4D hyperparallelepiped. These hyperplanes are characterized by a $t + s + v + u = c$ constraint on the coefficients coordinates (t, s, v, u). Algorithm 1 generates the

ordered list of 4D-DCT coefficients corresponding to the 4D diagonal scanning mode. It does so by scanning each coordinate independently (in the order u, v, s, t) while adding to the list only the coordinates satisfying the restriction of belonging to the hyperplane characterized by $t + s + v + u = sum$ (line 9 of Algorithm 1). The value sum varies from 0 to $(size_t + size_s + size_v + size_u)$, where $size_{<k>}$ is the size of the block in the dimension k. Although each coefficient has to be visited $(size\_t + size\_s + size\_v + size\_u)$ times to create the list (line 2 of Algorithm 1), this does not impact the coding execution time adopting this scanning mode as the positions scanning sequence can be generated offline (this means in advance before the real coding). In addition, Algorithm 1 has the advantage to be easily adapted for scanning surfaces other than hyperplanes. For example, if scanning along a hyperspherical surface is desired, it suffices to change the conditional statement in line 9 of Algorithm 1 to $x^2 + y^2 + z^2 = sum^2$.

Algorithm 1 4D diagonal scanning mode

```
1:  procedure 4DDiagonalScanning(size_t, size_s, size_v, size_u)
2:      max_sum ← size_t + size_s + size_v + size_u
3:      scanning_order ← []
4:      for sum in [0, max_sum] do
5:          for t in [0, size_t] do
6:              for s in [0, size_s] do
7:                  for v in [0, size_v] do
8:                      for u in [0, size_u] do
9:                          if (t + s + v + u) = sum then
10:                             scanning_order.append( {t, s, v, u} )
11:                         end if
12:                     end for
13:                 end for
14:             end for
15:         end for
16:     end for
17:     return scanning_order
18: end procedure
```

4.4.2 Double 2D diagonal scanning modes

As pointed out previously, the analysis of Figure 4.7 (a) suggests that good scanning modes for the lenslet light fields are those where the inner loop is a diagonal scan on the (u, v) dimensions and the outer loop a diagonal scan on the (s, t) dimensions, with the reverse happening for HDCA light fields. Modes performing such type of scanning are referred here as *Double 2D diagonal* modes, with the dimensions of the inner and outer loops selected from the six possible dimension permutations. Figure

4.11 illustrates a *Double 2D diagonal* scanning mode with the inner diagonal in the (v, u) coordinates and the outer diagonal in the (t, s) coordinates for a hypercube of size $(t, s, v, u) = (3, 3, 3, 3)$. Algorithm 2 contains an implementation of the scanning mode with same choice of dimensions for the inner and outer diagonals. Besides the four loops used to scan the dimensions of the 4D block, the algorithm requires other two loops to perform the inner and outer diagonals. The conditional statement found in line 11 of Algorithm 2 verifies that the sum $(t + s)$ is equal to the outer sum and the sum $(v + u)$ is equal to the inner sum, ensuring the double diagonal scan.

There are six possibilities of arranging the four dimensions of the block into two groups of two, since the order of the dimensions in the diagonal scanning is not relevant, corresponding to $4!/[(4-2)! \cdot 2!] = 6$ possibilities. The other *Double 2D diagonal* modes are created by simply interchanging the dimensions in Algorithm 2. The *Double 2D diagonal* scanning modes are listed in Table 4.5.

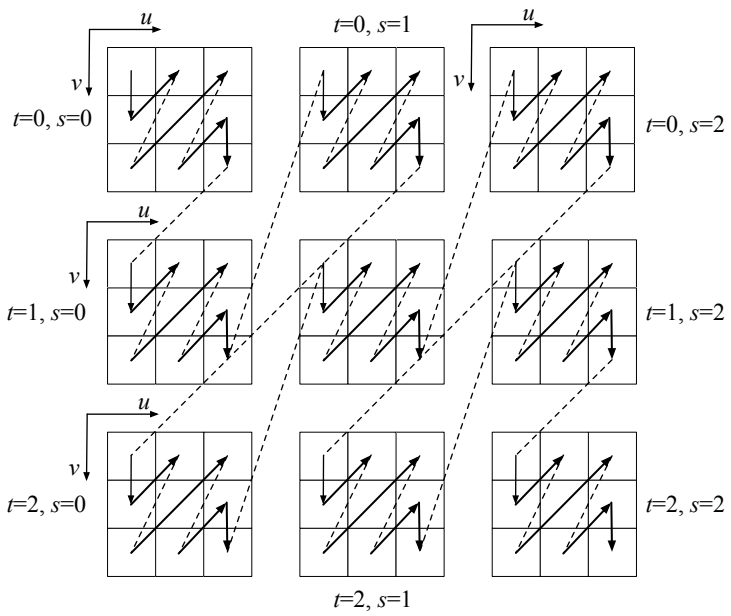


Figure 4.11: Example of double 2D diagonal scanning mode for a $(t, s, v, u) = (3, 3, 3, 3)$ hypercube: inner diagonal in (v, u) and outer diagonal in (t, s).

4.4.3 1D Directional with 3D diagonal scanning modes

In Figure 4.11, the 4D block is represented as a 2D array of rectangles. The four dimensions can be also represented as an one-dimensional array of 3D parallelepipeds. The scanning mode referred as 1D Directional with 3D diagonal consists in splitting the 4D block into parallelepipeds and performing a 3D diagonal scan into them. Figure 4.12 shows the 3D diagonal scan in s, u and v dimensions and a directional scan in the t dimension for a 4D block of size $(t, s, v, u) = (3, 4, 4, 4)$. For $t = 0$,

Algorithm 2 Double 2D diagonal scanning mode

```
1: procedure Double2DDiagonalScanning(size_t, size_s, size_v, size_u)
2:     outer_max_sum ← size_t + size_s
3:     inner_max_sum ← size_v + size_u
4:     scanning_order ← []
5:     for outer_sum in [0, outer_max_sum] do
6:         for inner_sum in [0, inner_max_sum] do
7:             for t in [0, size_t] do
8:                 for s in [0, size_s] do
9:                     for v in [0, size_v] do
10:                        for u in [0, size_u] do
11:                            if (t + s) = outer_sum and (v + u) = inner_sum then
12:                                scanning_order.append( {t, s, v, u} )
13:                            end if
14:                        end for
15:                    end for
16:                end for
17:            end for
18:        end for
19:    end for
20:    return scanning_order
21: end procedure
```

the cube is depicted sliced into planes that are perpendicular to its space diagonal and the numbers indicate the respective scanning order within the cube. There are four distinct ways to arrange the dimensions of a 4D hyperparallelepiped into sets of three diagonally scanned dimensions with one directionally scanned. Algorithm 3 contains an implementation in pseudocode of a scanning mode performing a 3D diagonal scan in (s, v, u) and a directional scan in t. The other three scanning modes are generated by interchanging any of s, u, or v dimensions and t. The scanning modes are listed in Table 4.3.

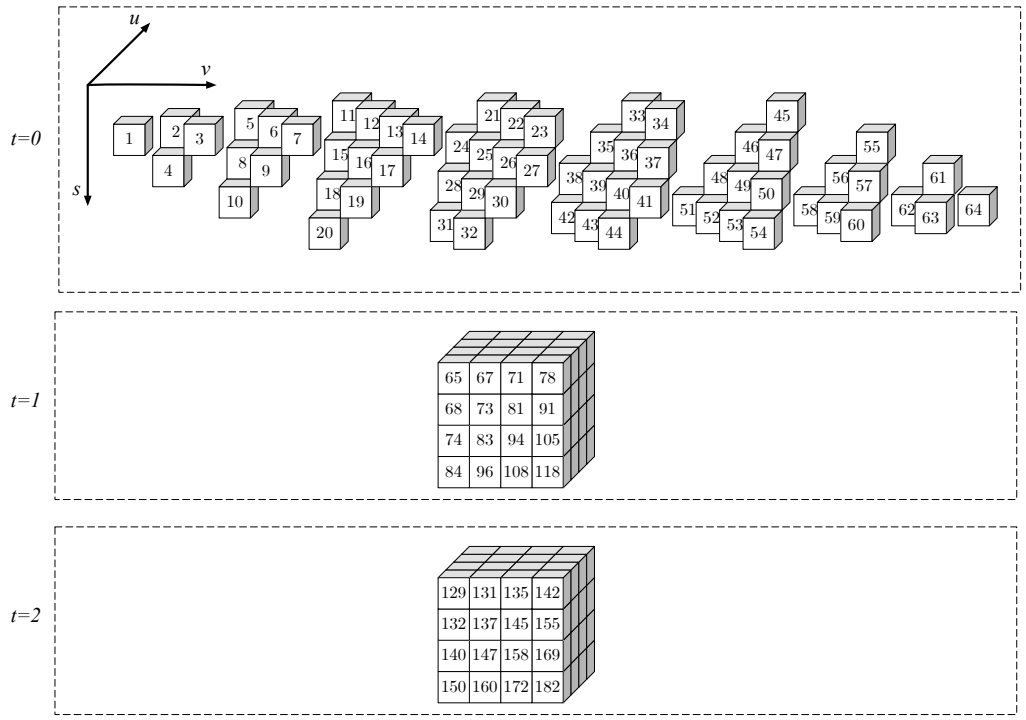


Figure 4.12: Example of diagonal 3D plus directional scanning mode for a 4D block of size $(t, s, v, u) = (3, 4, 4, 4)$: 3D diagonal in (s, v, u) and directional in t.

Algorithm 3 3D diagonal plus directional scanning mode

```
1:  procedure 3D DiagonalPlusDirectionalScanning(size_t, size_s, size_v, size_u)
2:      max_sum ← size_s + size_v + size_u
3:      scanning_order ← []
4:      for t in [0, size_t] do
5:          for sum in [0, max_sum] do
6:              for s in [0, size_s] do
7:                  for v in [0, size_v] do
8:                      for u in [0, size_u] do
9:                          if (s + v + u) = sum then
10:                             scanning_order.append( {t, s, v, u} )
11:                         end if
12:                     end for
13:                 end for
14:             end for
15:         end for
16:     end for
17:     return scanning_order
18: end procedure
```

4.4.4 2D Directional with 2D diagonal scanning modes

The *2D directional with 2D diagonal* scanning pattern consists in performing a 2D diagonal scan in two dimensions and a directional scan in the remaining two. Figure 4.13 shows an example of a 2D diagonal scan in (u, v) and the directional scan first in t and then in s. Algorithm 4 contains an implementation of such scanning mode in pseudocode. The 12 different ways to scan the 4D block following the *2D Directional with 2D diagonal* pattern are listed in Table 4.2.

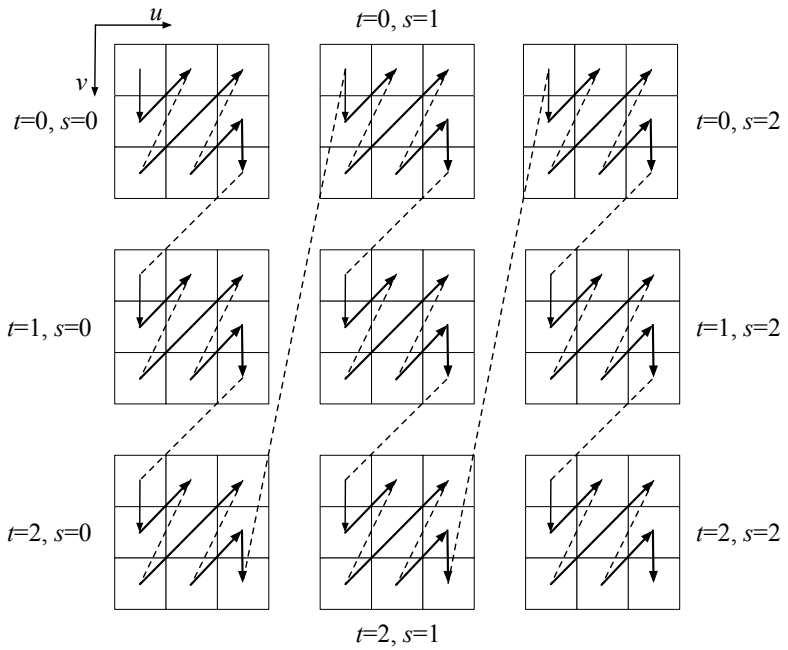


Figure 4.13: Example of *2D Diagonal 2D plus directional* scanning mode for a 4D block of size $(t, s, v, u) = (3, 3, 3, 3)$: diagonal in (u, v) dimensions and directional first in t and after in s.

Algorithm 4 2D diagonal plus directional scanning mode

```
1: procedure 2DDiagonal+DirectionalScanning(size_t, size_s, size_v, size_u)
2:     max_sum ← size_v + size_u
3:     scanning_order ← []
4:     for s in [0, size_s] do
5:         for t in [0, size_t] do
6:             for sum in [0, max_sum] do
7:                 for v in [0, size_v] do
8:                     for u in [0, size_u] do
9:                         if (v + u) = sum then
10:                            scanning_order.append( {t, s, v, u} )
11:                        end if
12:                    end for
13:                end for
14:            end for
15:        end for
16:    end for
17:    return scanning_order
18: end procedure
```

4.4.5 Directional scanning modes

The last scanning modes proposed are the *Directional*, in which the 4D block dimensions are scanned sequentially. Figure 4.14 shows the scanning order defined by the directional scanning $u \rightarrow s \rightarrow v \rightarrow t$ for a $3 \times 3 \times 3 \times 3$ block. First, the coefficients along the u dimension are scanned. When the end of a line is reached, the line at same position in the next (u, v) rectangle is scanned, incrementing the s dimension. When all first lines in (u, v) rectangles for $s = 0$ are scanned, the second line of the same views are scanned, which corresponds to incrementing the v dimension. At the point the first line of (u, v) rectangles are fully scanned, the process is repeated for the second line of (u, v) rectangles, corresponding to incrementing the t dimension. The process is similar for all directional modes derived from the directional pattern, only the scanning order of the dimensions is changed.

Algorithm 5 shows an implementation in pseudocode of the directional scanning mode $u \rightarrow s \rightarrow v \rightarrow t$. There are four nested **for** loops scanning each block's dimension. The inner loops must contain the dimensions that are scanned earlier. There are 24 different ways of scanning the dimensions a 4D block, corresponding to the permutation ${}^4P_4 = 4! = 24$. Table 4.1 lists all the directional scanning modes.

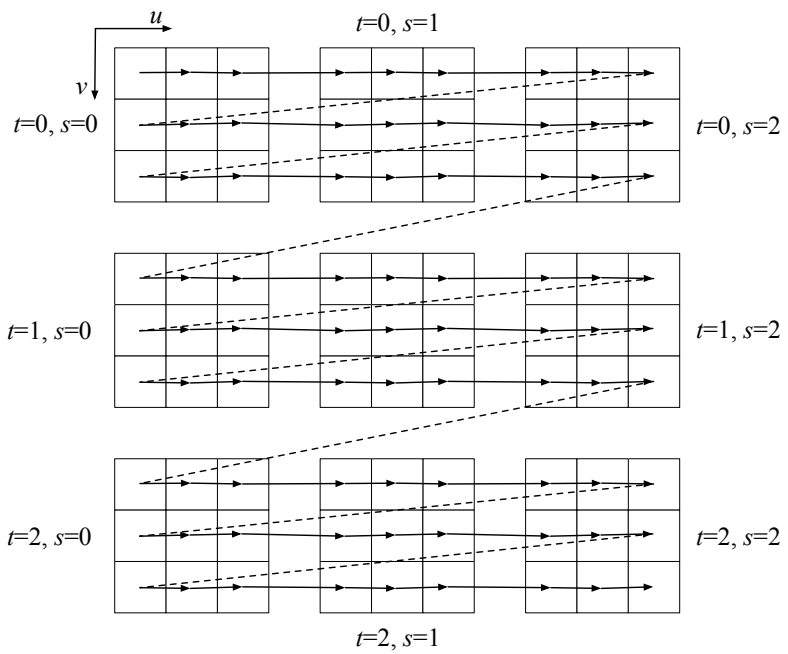


Figure 4.14: Example of directional scanning mode for a $(t, s, v, u) = (3, 3, 3, 3)$ 4D block: scanning order $u \rightarrow s \rightarrow v \rightarrow t$.

Algorithm 5 Directional scanning mode $u \rightarrow s \rightarrow v \rightarrow t$

```
1: procedure DirectionalScanning(size_t, size_s, size_v, size_u)
2:     scanning_order ← [ ]
3:     for t in [0, size_t] do
4:         for v in [0, size_v] do
5:             for s in [0, size_s] do
6:                 for u in [0, size_u] do
7:                     scanning_order.append( {t, s, v, u} )
8:                 end for
9:             end for
10:        end for
11:    end for
12:    return scanning_order
13: end procedure
```

4.4.6 Full set of scanning modes

The full set of proposed scanning modes is described in Tables 4.1 to 4.5. The 12 scanning modes composed by combinations of a 2D directional and a 2D diagonal scanning patterns, numbered from 24 to 35, as shown in Table 4.2. The four modes formed by combining a 1D directional and a 3D diagonal pattern are given numbers

from 36 to 39 and are shown in Table 4.3. The index 40 is reserved to the 4D diagonal scanning mode (Table 4.4) while the remaining indexes up to 46 are assigned to the double 2D diagonal modes, identified in Table 4.5. The 24 directional scanning modes and indexed from 0 to 23, as shown in Table 4.1.

Table 4.1: Directional scanning modes and respective assigned indexes.

Mode	Scanning Order			
–	1^{st}	2^{nd}	3^{rd}	4^{th}
0	T	S	V	U
1	T	S	U	V
2	T	V	S	U
3	T	U	S	V
4	T	V	U	S
5	T	U	V	S
6	S	T	V	U
7	S	T	U	V
8	V	T	S	U
9	U	T	S	V
10	V	T	U	S
11	U	T	V	S

Mode	Scanning Order			
–	1^{st}	2^{nd}	3^{rd}	4^{th}
12	S	V	T	U
13	S	U	T	V
14	V	S	T	U
15	U	S	T	V
16	V	U	T	S
17	U	V	T	S
18	S	V	U	T
19	S	U	V	T
20	V	S	U	T
21	U	S	V	T
22	V	U	S	T
23	U	V	S	T

Table 4.2: 2D Directional with 2D diagonal scanning modes and respective assigned indexes.

Mode	Scanning Order		
–	Diagonal 2D	3^{rd}	4^{th}
24	VU	T	S
25	SU	T	V
26	SV	T	U
27	TU	S	V
28	TV	S	U
29	TS	V	U

Mode	Scanning Order		
–	Diagonal 2D	3^{rd}	4^{th}
30	VU	S	T
31	SU	V	T
32	SV	U	T
33	TU	V	S
34	TV	U	S
35	TS	U	V

Table 4.3: 1D Directional with 3D diagonal scanning modes and respective assigned indexes.

Mode	Scanning Order	
–	Diagonal 3D	4^{th}
36	SVU	T
37	TVU	S
38	TSU	V
39	TSV	U

Table 4.4: Diagonal 4D scanning mode assigned index.

Mode	Scanning Order
–	Diagonal 4D
40	TSVU

Table 4.5: Double 2D diagonal scanning modes and respective assigned indexes.

Mode	Scanning Order	
–	Inner diagonal 2D	Outer diagonal 2D
41	TS	VU
42	TV	SU
43	SV	TU
44	TU	SV
45	SU	TV
46	VU	TS

4.5 4D-DCT scanning modes assessment

The proposed 4D-DCT scanning modes are assessed by comparing their energy compactness capabilities, this means how fast the energy accumulates as the scanning proceeds. Starting from the coefficients energy estimation procedure shown in Section 4.3, the assessment of each scanning mode is based on the following steps (Figure 4.15):

1. **Scanning:** The 4D-DCT coefficients are ordered according to each proposed scanning mode;

2. **Cumulative energy (CE) computation:** The ordered 4D-DCT coefficients are visited in scanning sequence, creating a curve of the cumulative energy. Energy values are expressed in percentage.

3. **Integral of cumulative energy (ICE) computation:** The percent cumulative energies along the scanning path are added, what is equivalent to computing the integral/area under the cumulative energy curve. Since the first cumulative energy has only the contribution of the first coefficient, the second contributions of the first and second coefficients, and so on, then the cumulative energy CE_k for the k-th scanned coefficient contributes $N-k+1$ times for the ICE computation, according to the expression

$$\mathrm{ICE} = \frac{\sum_{k=1}^{N}(N-k+1)\mathrm{CE}_k}{\sum_{k=1}^{N}\mathrm{CE}_k} \tag{4.1}$$

Therefore, the scanning modes with the larger energies CE_k being scanned earlier (smaller k), will create a CE curve growing faster and, finally, yielding larger ICE values.

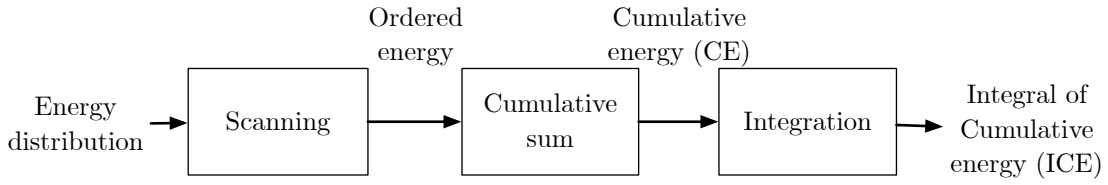
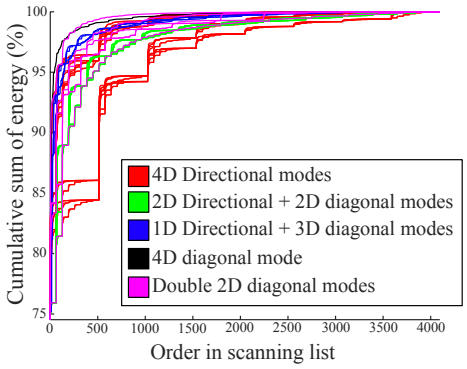


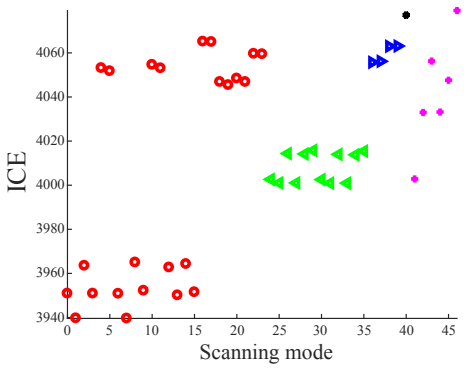
Figure 4.15: Energy compactness performance assessment pipeline.

Figure 4.16 (left) shows the CE curve (%) for all the proposed scanning modes and Figure 4.16(right) shows the corresponding ICE for the various scanning modes for the light fields *Bikes* (a), *Set2 2k Sub* (b) and *Greek* (c). More extensive CE results for the light fields in the JPEG Pleno dataset can be found in Appendix B. As expected from Figure 4.7 (a), the results for Bikes show that the directional scanning modes with inner loop in the (u, v) dimensions achieve higher ICE values than the modes with inner loop in the (s, t) dimensions. Indeed, for this light field, the best mode is the one with index 46, corresponding to the double 2D diagonal mode with inner loop in (u, v) and outer loop in (t, s). The 4D diagonal mode, identified by index 40 and shown as a black dot in the plot, presents a near-optimal performance. Among the 24 directional modes with indexes ranging from 0 to 23, those which scan first (u, v) dimensions presented better performance than the others. This confirms the results shown in Chapter 3: lenslets light fields have more inter-view than intra-view redundancy; consequently the energy will be more distributed along intra view

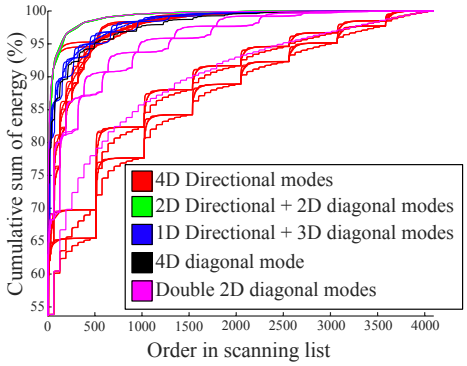
(u, v) dimensions. These dimensions should, therefore, be scanned before than the dimensions (s, t).



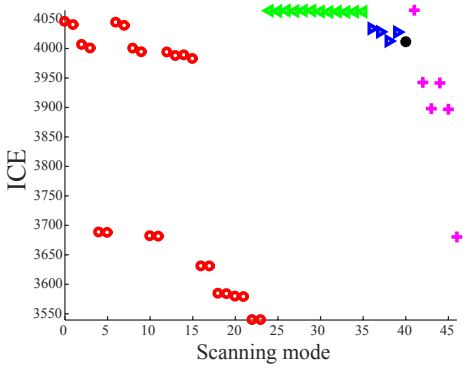
(a) CE curves for *Bikes*.



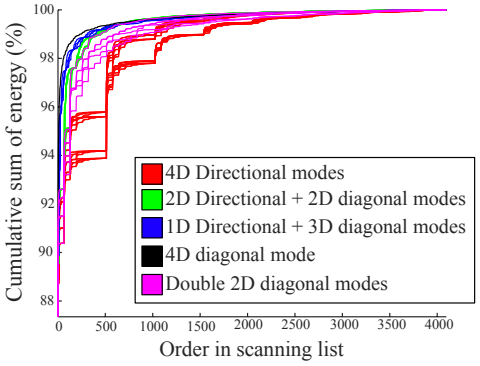
(b) ICE values for *Bikes*.



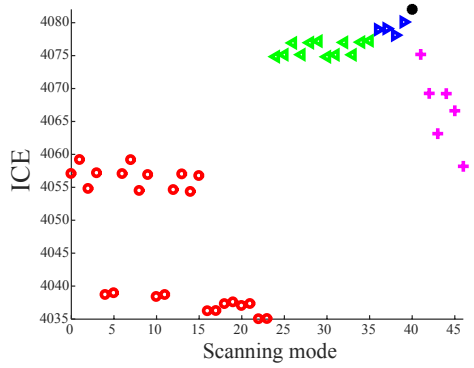
(c) CE curves for *Set2 2k Sub*.



(d) ICE values for *Set2 2k Sub*.



(e) CE curves for *Greek*.



(f) ICE values for *Greek*.

Figure 4.16: Left: Cumulative energy curve versus the order in the scanning list. Right: Integral of cumulative energy (ICE) for the 47 scanning modes – light fields *Bikes*, *Set2 2k Sub* and *Greek*.

The results for *Set2 2k Sub* are also those expected according to Figure 4.7 (b). The directional scanning modes with inner loop in the (s, t) dimensions achieve ICE

values larger than the modes with inner loop in the (u, v) dimensions. Indeed, for this light field, the best mode is the one with index 41, corresponding to the double 2D diagonal mode with inner loop in (s, t) and outer loop in (u, v). Among the 24 directional modes with indexes ranging from 0 to 23, those scanning firstthe (s, t) dimensions presented better performance than tho others. This also confirms the results shown in Chapter 3: HDCA light fields have more intra-view than inter-view redundancy; consequently the energy will be more distributed along inter view (s, t) dimensions. These dimensions should, therefore, be scanned before than (u, v).

Finally, Figure 4.16 (c) depicts the results for *Greek*. The best scanning mode according to ICE is the one with index 40, the 4D diagonal mode. As for Bikes and *Set2 2K Sub*, this result is consistent with the energy distribution shown in Figure 4.7. Since the energy decays along the (u, v) and (s, t) dimensions are equivalent, the 4D diagonal mode is able to visit the higher energy coefficients earlier than the lower energy ones.

Next chapter proposes a simple light field codec based on 4D-DCT, run-length coding and arithmetic coding. An experiment will be performed by integrating the designed scanning modes into the proposed codec. Results are expected to show that the RD performance improves when the scanning modes corresponding to larger ICE values are used.

Chapter 5

Proposing a 4D-DCT based light field coding solution

As discussed in Chapter 2, the light field coding can be performed in different ways such as using available image or video coding standards after restructuring the light field data or extending available coding solutions, adding the capability of exploiting the new kind of redundancy. However, the best light field coding performance is expected to be achieved by novel coding solutions, which are more specifically adapted to the unique characteristics of light field imaging data. Chapter 3 has shown that light fields present a significant amount of 4D redundancy, formed jointly by inter-view and intra-view portions which are likely to result in better coding performance if exploited as a whole rather than separately.

In this context, this chapter proposes a light field coding solution, which is called *Multidimensional Light field encoder with Transform and Diagonal scanning* (MuLE-TD). First, an overview of the proposed coding architecture is presented in Section 5.1 whereas Section 5.2 describes each framework coding tool in detail. In the sequel, Section 5.3 presents the RD-based assessment of the scanning modes proposed in Chapter 4. Finally, Section 5.4 presents MuLE-TD assessment conditions, results and analysis.

5.1 Architecture overview

This section presents an overview of the architecture of the proposed MuLE-TD light field codec which combines 4D-DCT computation, coefficient scanning and run-length encoding. The proposed coding solution has a pipeline similar to the classic JPEG image coding standard. In brief, JPEG extracts 8×8 blocks of image pixels and applies a 2D-DCT of same size. The AC transform coefficients are quantized using a quantization matrix and then are scanned using the zig-zag scanning depicted

if Figure 4.2. In the sequel, the non-zero AC coefficients are encoded using run-length encoding while DPCM is used in DC coefficients. The run-length symbols are coded using Huffman codes [3].

Likewise the JPEG standard, the proposed light field coding solution follows a block based approach, coding independently each channel of the input light field. In MuLE-TD, 4D blocks are extracted from the input light field and the 4D-DCT is applied. The transform coefficients are quantized using linear quantization and scanned with one of the modes presented in Section 4.4. In the sequel, transform coefficients are encoded using run-length coding and the resulting symbols are coded using arithmetic coding. A block diagram of the proposed light field coding solution is shown in Figure 5.1

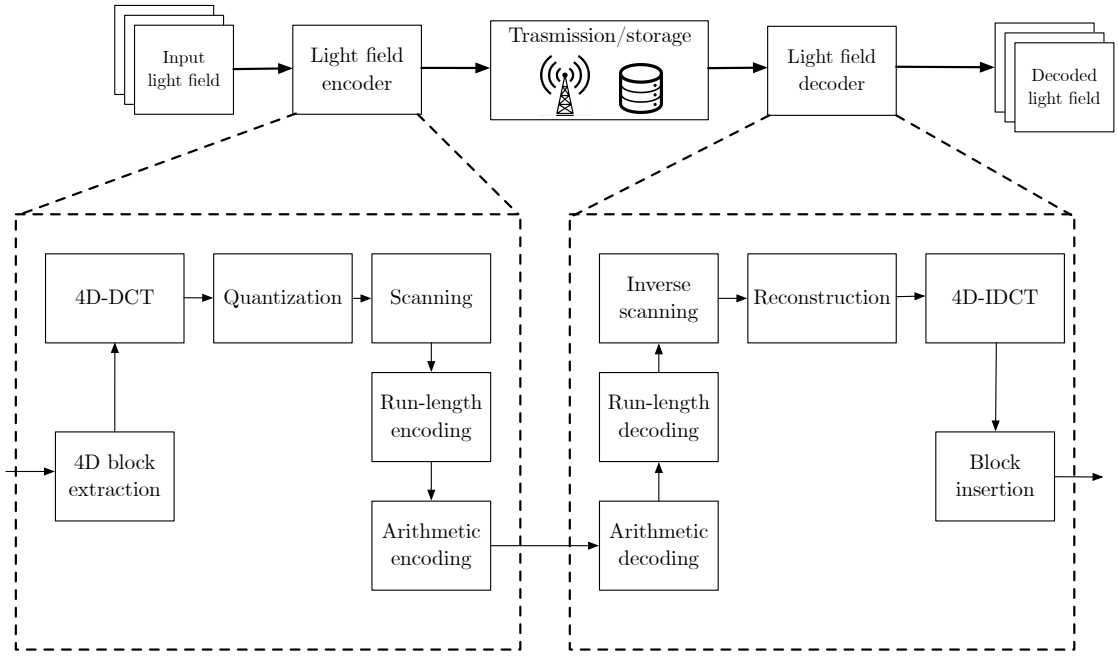


Figure 5.1: Block diagram of the proposed light field coding solution.

5.1.1 Encoding

- **4D block extraction:** As a first step, 4D blocks of fixed size are read from the input light field data. The codec input consists in several 10-bit PGM raw image files, corresponding to the Y, Cb and Cr channels of the sub-aperture views. MuLE-TD encodes each light field channel independently. The extraction of a 4D block at position (t, s, v, u) of size (s_t, s_s, s_v, s_u) requires extracting a rectangular region of size (s_v, s_u) from $s_t \times s_s$ views. This is the first challenging step, since the number of simultaneous open files may become greater than the limit stablished for most operational systems. Constantly opening, closing and performing *seek* operations within input files results in delays and

poor computing performance. The proposed encoder implements a sequential reading of the input files using a circular buffer scheme. Only the view files currently in use are maintained open and 4D block extraction module reads the images lines to the end and keep them in memory, for fast posterior access.

- **4D-DCT:** The 4D discrete cosine transform of the data is computed. A separable transform is used so the 4D transform is achieved by sequentially applying an one-dimensional transform in each dimension, as described in Section 3.1. This step is the most computationally complex of the framework. The 4D transform for several blocks is performed in parallel, to take advantage of the multi-core architecture present in most modern computer systems.

- **Quantization:** The transform coefficients are quantized using linear quantization. The process introduces irreversible losses as it maps values that have a wide dynamic range to a narrower one. As the transform applied in previous step has the property of energy compaction, most of the high frequency quantized coefficients have values equal to zero.

- **Scanning:** The quantized transform coefficients are scanned to give raise to a linear array. The scan order is intended to group together non-significant coefficients generating longs runs of zeros, which can be efficiently represented using run-length coding. The proposed coding solution implements novel scanning patterns for 4D-DCT coefficients described in Section 4.4.

- **Run-length encoding:** The quantized coefficients reordered according to the 4D scanning modes are encoded using run-length technique. Each symbol is formed by a pair $(run, length)$ in which run represents the number of zeros that precede the non-zero coefficient, and $length$ is the number of bits required to represent the quantized amplitude.

- **Arithmetic coding:** Finally, in the last step of the encoder pipeline, the entropy coding of the run-length symbols is performed using an adaptive arithmetic encoder. The arithmetic coding is a lossless compression process which targets to efficiently represent the symbols and code them into the bitstream, which can be stored in a file or transmitted.

5.1.2 Decoding

The steps of the decoding pipeline can be seen in Figure 5.1, in the right. The decoding process basically applies inverse operations used in the encoding pipeline, in reverse order. In brief, codewords are extracted from the bitstream and run-length symbols are recovered. The run-length decoding gives raise a linear array of

quantized transform coefficients. The inverse scanning reorganizes the coefficients in a 4D block and the inverse quantization is performed. In the sequel, the inverse 4D-DCT is applied, resulting in a 4D block of samples which is inserted into the output decoded light field. In next section, the MuLE-TD coding tools are described in more detail.

5.2 Coding tools

5.2.1 4D-DCT transform

The 4D-DCT transform is performed as described in Section 3.1, by applying separable one-dimensional transforms successively in t, s, v and u dimensions, following the flow depicted in Figure 3.1. No fixed-point DCT approximations or fast implementations are used, since the choice of transform size is completely free. Rather, the transforms are computed in double precision floating arithmetic, from the definition shown in Equation 3.1. The four DCT matrices to be used in t, s, v and u dimensions are computed once at the beginning of the encoding process, to avoid recurrent and complex double precision cosine computations. Thus, transforming operations come down to matrix multiplications.

5.2.2 Transform coefficients Scanning

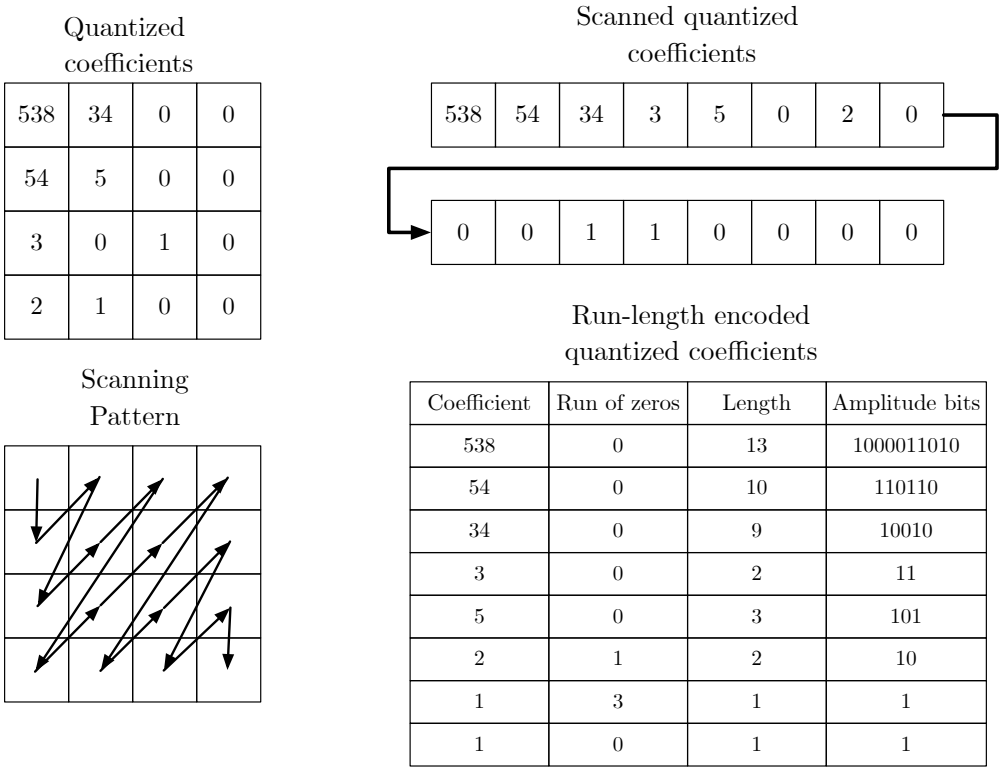
The transform coefficients are scanned using one of the 47 modes described in Chapter 4 and identified in Tables 4.1 to 4.5. As for DCT matrices, the scanning order is computed once at the beginning of the codec execution, prior to encoding process, aiming at reducing complexity. The codec receives the index of the scanning mode to be used via configuration file. The default value is 40, corresponding to *4D diagonal* scanning mode, as shown in Table 4.4.

5.2.3 Run-length coding

The linear array of quantized coefficients are submitted to the next step in codec pipeline which is the run-length encoding. To facilitate the graphic representation, Figure 5.2 illustrates the run-length encoding process for a 2D block of quantized transform coefficients. The process for a 4D block is analogous. The input of the run-length encoder is the quantized transform coefficients, previously rearranged in a one dimensional array according to a given scanning mode. The run-length encoder scans this linear array and counts the number of zeros before each nonzero coefficient, generating a pair $(run, length)$ in which the run represents the count of zeros before the nonzero coefficient and $length$ is the number of bits necessary to

represent the amplitude of respective quantized coefficient, as shown in the table in the bottom os Figure 5.2. Each symbol is composed by two tokens: the first is the $(run, length)$ pair and the second is the amplitude of the coefficient [3]:

token-1 **token-2**
$(run, length)$ $(amplitude)$



Coefficient	Run of zeros	Length	Amplitude bits
538	0	13	1000011010
54	0	10	110110
34	0	9	10010
3	0	2	11
5	0	3	101
2	1	2	10
1	3	1	1
1	0	1	1

Figure 5.2: Example of run-length encoding.

The number of possible run-length pairs will be the product of the maximum possible length $length_{max}$ and the maximum possible run of zeros run_{max}. The run_{max} is limited by the total number of coefficients in the 4D block, e.g. for a block of dimensions (S_t, S_s, S_v, S_u) an upper limit for the run of zeros is $run_{max} = S_t \cdot S_s \cdot S_v \cdot S_u$. In MuLE-TD the run_{max} is specified as an input parameter of the encoder. By limiting the possible run of zeros, the final number of possible symbols is consequently reduced which improves the performance of the arithmetic coder, the next step in the encoding pipeline.

The maximum value of $length$ depends on the quantization step q_{step} and the dynamic range of the transform coefficients which, in turn, depends on the dynamic range of the pixels in the input light field and the size of the transform. If the bit depth of the input light field is B_d, the maximum value of a pixel is given by:

$$p_{max} = 2^{B_d} - 1. \tag{5.1}$$

The transform coefficients, in absolute value, are not expected to exceed the value of the DC coefficient of a transformed matrix resulted from the application of the 4D-DCT in the 4D matrix

$$\mathbf{M}_{max} = p_{max}\ \mathbf{J}_{t\times s\times v\times u} \tag{5.2}$$

where $\mathbf{J}_{t\times s\times v\times u}$ is the 4D all-ones matrix. So the maximum absolute value of a transform coefficient is DC_{max} given by

$$\mathrm{DC}_{max} = \mathrm{DC}(\mathbf{T}_{4D}(\mathbf{M}_{max})) \tag{5.3}$$

where $\mathbf{T}_{4D}$ operator corresponds to applying the separable 4D-DCT and the operator DC means selecting the coefficient at position $(0, 0, 0, 0)$ of the 4D matrix. So the maximum *length*, in bits, of a quantized transform coefficient is given by:

$$\mathrm{length}_{max} = \log_2(1 + \mathrm{DC}_{max}/q_{step})\ \ \text{bits.} \tag{5.4}$$

Observing the third column of the table in the bottom of Figure 5.2 which corresponds to the amplitude of quantized coefficient in bits, one should note that all values start with bit "1". This is a rule, since if a value v is represented with n bits, the $(n-1)th$ digit must be "1", otherwise v can be represented with less than n bits. Since the last bit is always equal to "1" it does not need to be transmitted. MuLE-TD uses this bit to store the signal of the quantized coefficient. The bit "1" is used for positive values and "0" otherwise.

An extra symbol used in the run-length encoding is the *EOB*, which means "End Of Block". The symbol is inserted after the last nonzero coefficient encoded, indicating to the decoder that there is no more significant coefficients and the remaining values are all equal to zero. In the case the last coefficient in scanning order is nonzero-valued, the *EOB* is not included since the decoder can infer that it has reached the end of the block by counting the decoded coefficients. The total number of run-length symbols is therefore

$$n_{sym} = \mathrm{length}_{max}(run_{max} + 1) + 1, \tag{5.5}$$

since possible values for the run of zeros rely in the interval $[0, run_{max}]$ and one extra symbol *EOB* is added.

The choice of run_{max} does not interfere in the quality of the decoded light field, but affects the final rate obtained by the codec. In order to reduce the number of free input parameters of the codec an experiment was conduced aiming at defining the optimal run_{max} as a function of transform size and q_{step}. The following configuration was used in the experiment:

- **Light field data**: The mosaic light fields of size $t \times s \times v \times u = 10 \times 10 \times 384 \times 2048$ described in section 2.4.1.

- **Quantization steps**: The values of q_{step} used are [0.02, 0.035, 0.08, 0.1, 0.5, 1.3]. These values are known to produce reconstruction quality in the range 25-45 dB.

- **Transform lengths**: The quantity *transform length* is defined as the product of the four dimensions of the transform size. The transform sizes used are obtained by combining the three $t \times s$ inter-view block sizes shown in Table 5.1 with six $v \times u$ intra-view sizes shown in same table. The resulting 18 transform sizes $t \times s \times v \times u$ range from $2 \times 2 \times 4 \times 4$ to $10 \times 10 \times 128 \times 128$ and the respective *transform lengths* range from $2 \cdot 2 \cdot 4 \cdot 4 = 64$ to $10 \cdot 10 \cdot 128 \cdot 128 = 1,638,400$.

Table 5.1: Transform sizes used in the run_{max} determination experiment.

Inter-view transform size ($t \times s$)	**Intra-view transform size** ($v \times u$)
2×2	4×4
5×5	8×8
10×10	16×16
–	32×32
–	64×64
–	128×128

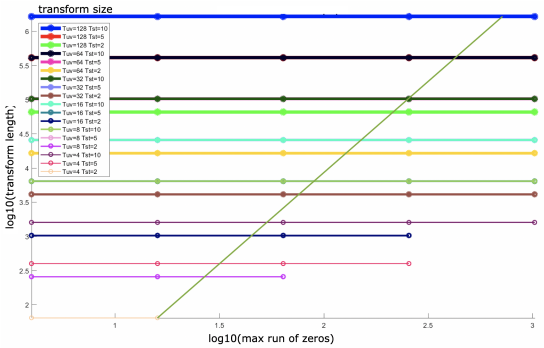
- **Maximum run of zeros** (run_{max}): The values are powers of four between 4 and 16384: [4, 16, 64, 256, 1024, 4096, 16384]. The verification $run_{max} <$ *transform length* is made, since does not make sense to use a maximum run of zeros greater than the *transform length*.

- **Arithmetic encoder model**: The adaptive model is used to encode run-length symbols (see Section 5.2.4).

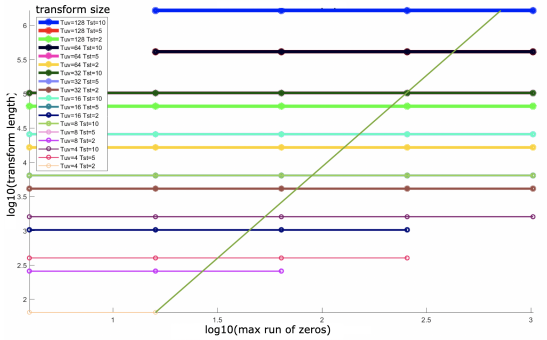
The first step of the experiment consists in coding the selected light field data using all combinations of transform lengths, q_{step} and maximum run of zeros and finding which maximum run of zeros result in the minimal rate for a given *transform length* and q_{step} . The second step consists in finding a heuristic to determine the values of run_{max} which will result in the minimum rate with a certain tolerance, chosen as being of 5%. This experiment is conduced for HDCA and lenslets datasets.

For both datasets, it is possible to define the desired run_{max} as a function *transform length*, valid for all q_{step} tested.

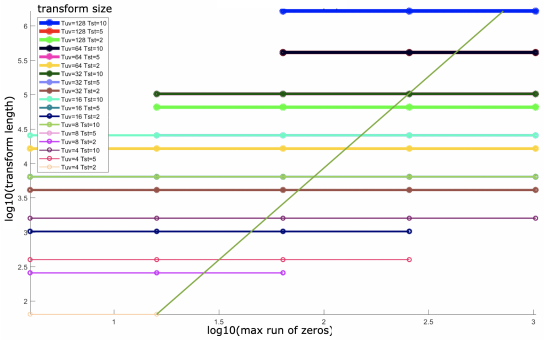
Figure 5.3 shows plots $\log_{10}(run_{max})$ vs $\log_{10}$(transform length) for each q_{step} tested, for *Set2 2k Sub* mosaic light field. The plots contain several colored horizontal lines, one for each transform length, representing the interval of run_{max} values which result in the minimal rate with a tolerance of 5%. In the graph legend box, T_{uv} and T_{st} refer, respectively, to the transform size used in (u, v) and (s, t) dimensions. The desired curve $run_{max} = f(q_{step}, \textit{transform length})$ which defines the optimal run_{max} as a function of q_{step} and *transform length* should cross all horizontal lines in all plots. The slanted lines depicted in Figure 5.3 represent an example of linear function which creates a relationship between the transform length and the maximum run of zeros so that the codec achieves the minimum rate with the desired tolerance. The same function is used for all q_{step} values, indicating that the heuristic can be a function of *transform length* exclusively.



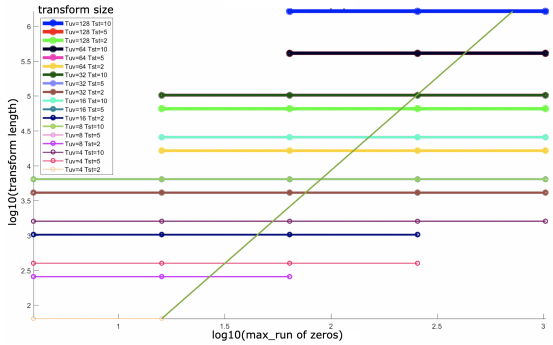
(a) q_{step} *0.02*



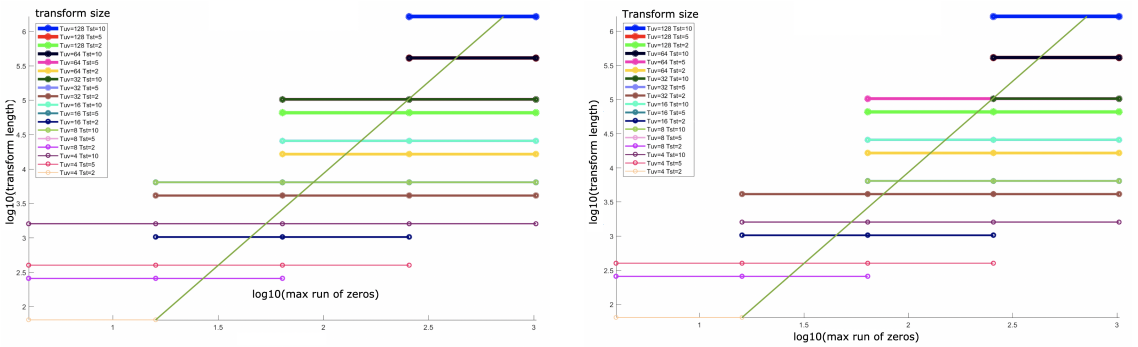
(b) q_{step} *0.035*



(c) q_{step} *0.08*



(d) q_{step} *0.1*



(e) q_{step} *0.5* (f) q_{step} *1.3*

Figure 5.3: Graphic representation showing run_{max} values for light field *Set2 2K Sub*.

For the HDCA dataset, the function

$$\log_{10}(run_{max}) = 0.47 \cdot \log_{10}(transform\ length) + 0.34 \tag{5.6}$$

is used to define run_{max} as a function of *transform length* in order to obtain the minimum achievable rate with a tolerance of 5%. The same experiment is performed for lenslets dataset, resulting in the following the relationship between run_{max} and *transform length*:

$$\log_{10}(run_{max}) = 0.41 \cdot \log_{10}(transform\ length) + 0.46. \tag{5.7}$$

5.2.4 Arithmetic coding

The entropy coding in MuLE-TD is performed using an arithmetic encoder [29]. The algorithm implemented is based on the mechanism presented in [30]. The arithmetic encoder can operate in two modes regarding the probability model: fixed or adaptive. In the adaptive mode, the model containing the symbol probabilities is updated after encoding a symbol. The adaptive approach is able to learn the probability distribution and reduce the overall rate by assigning shorter codewords to more frequent symbols. The other possibility is the fixed mode, in which the probabilities in the model remain the same during the entire encoding process.

MuLE-TD uses two probability models, one for each token generated by the run-length encoding. The first model is used to code the $(run, length)$ token and the second is used to code the $(amplitude)$ token. The pair $(run, length)$ is indexed jointly, using

$$\text{index} = run + length \cdot run_{max}. \tag{5.8}$$

The option of using an adaptive model is only valid for encoding $(run, length)$

tokens. The model used to encode the bits of (*amplitude*) token is binary with fixed equiprobable distribution. The MuLE-TD has the following features regarding its arithmetic coding module:

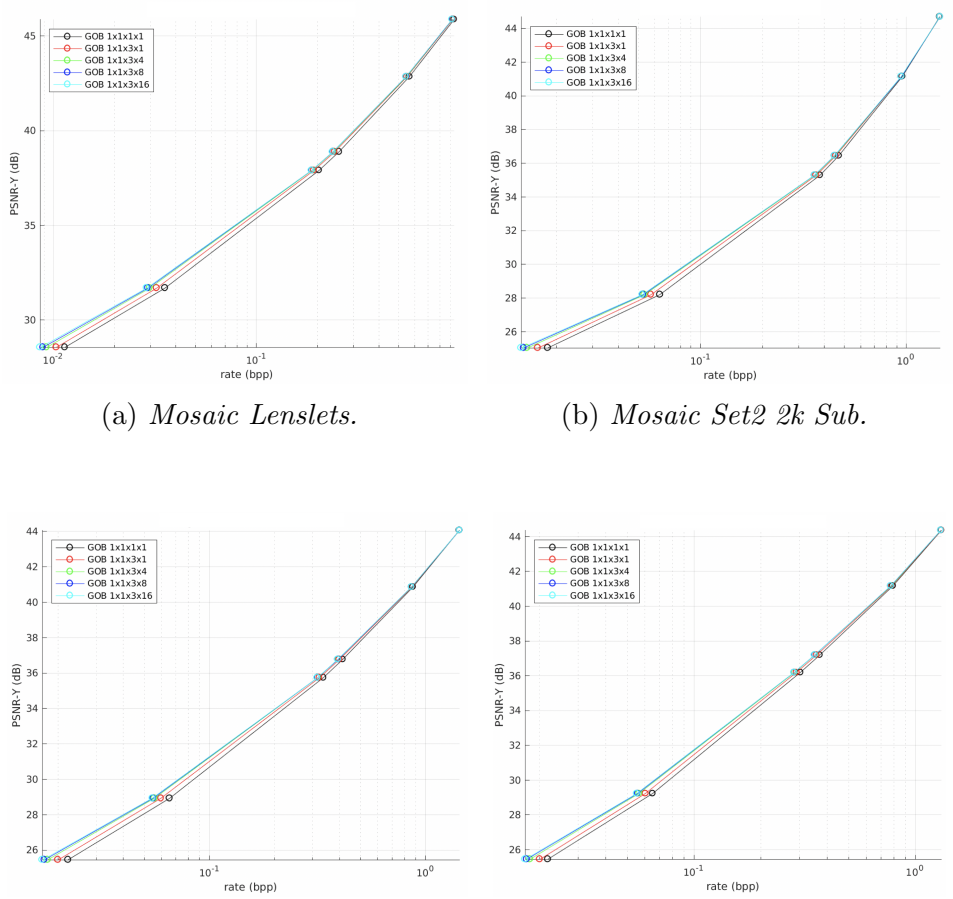
- ***Choice of adaptive arithmetic coding***: Defines if the codec will adapt the probability model used to code $(run, length)$ tokens. Using an adaptive model results in significative savings in bitrate.
- ***Saving arithmetic encoder model to a file***: The codec has the option of saving the final state of the probability model used to code $(run, length)$ tokens in a file. Saved probability models can be used for future encoding processes.
- ***Use of initial arithmetic encoder model***: The codec can use an initial probability model read from an input file to code $(run, length)$ tokens.

The choice of using a fixed or adaptive model in the arithmetic coding impacts the random access capability of the codec. If an adaptive model is used to encode run-length symbols for all 4D blocks, it is not possible to decode blocks in a random order in the decoder side, since the probability distribution used to encode such block is unknown and depends on the blocks previously encoded. If a fixed model is used, such random access becomes feasible with the addition of some extra information such as the position of the encoded block in the bitstream. One possibility to obtain some degree of random access using adaptive arithmetic coding is reseting the probability model to an initial state after encoding a group of blocks (GOB). Hence, GOBs becomes independent to each other and the random access becomes possible. MuLE-TD has the feature of grouping symbols from sets of 4D blocks and defining GOBs that will be encoded using the initial model configured via configuration file. If no GOB size configuration is informed, the entire light field is considered one single GOB and no random access is possible if adaptive arithmetic coding is used.

Table 5.2: MuLE-TD settings used in GOBs performance assessment experiment.

Parameters	**Value**
Light fields	Mosaic Bikes Mosaic Set2 2k Sub, Mosaic Tarot Cards Mosaic Laboratory1
Transform size	$10 \times 10 \times 128 \times 128$
Arithmetic encoder model	Adaptive
Scanning mode	40 (*4D diagonal*)
Quantization steps	0.02, 0.035, 0.08, 0.1, 0.5, 1.3
GOB sizes	$1 \times 1 \times 1 \times 1$ $1 \times 1 \times 3 \times 1$ $1 \times 1 \times 3 \times 4$ $1 \times 1 \times 3 \times 16$

Figure 5.4 contains RD performance rate (bpp) vs PSNR-Y (dB) for the mosaic light fields *Lenslets*, *Set2 2k Sub*, *Tarot Cards* and *Laboratory1* of size $t \times s \times v \times u = 10 \times 10 \times 384 \times 2048$ each, as described in Section 2.4.2. The test conditions are shown in Table 5.2. Since the transform size is $10 \times 10 \times 128 \times 128$, during the encoding process the input light field is segmented in a $1 \times 1 \times 3 \times 16$ array of blocks. The GOB sizes tested $1 \times 1 \times 1 \times 1$, $1 \times 1 \times 3 \times 1$, $1 \times 1 \times 3 \times 4$ and $1 \times 1 \times 3 \times 16$ result in 48, 16, 4 and 1 GOBs, respectively. As expected, same reconstruction quality is obtained for every point obtained by varying just the GOB sizes. Regarding the rate, the compression becomes worse as the size of the GOBs decreases. The best compression is obtained for the GOB size $1 \times 1 \times 3 \times 16$, when all light field blocks are grouped together into a single GOB and the arithmetic encoder model is never reset to initial state during the encoding process. For higher rates, the compression performance is nearly the same but differences increase for lower rates. The results are similar for all light fields tested.



(a) *Mosaic Lenslets.* (b) *Mosaic Set2 2k Sub.*

(c) *Mosaic Laboratory1.* (d) *Mosaic Tarot Cards.*

Figure 5.4: RD performance for several GOB sizes.

Some additional experiments have been performed regarding the arithmetic encoder, using fixed models. One of these experiments consisted in encoding a given light field using a fixed model obtained from encoding other light field, using the adaptive approach. In such experiment, the mosaic dataset described in Section 2.4.2 is split into two sets labeled as *development* and *testing* sets. The light fields in *development* set are encoded using adaptive arithmetic encoder in a sort of "training" process and the resulting model is used to encode the light fields in *testing* set. The green curve in Figure 5.5 shows the results for mosaic light field *Lenslets*, using a model trained by coding the mosaic light fields *Laboratory1, Tarot Cards* and *Set2 2K Sub.* The performance obtained when using the pre-trained model is a little worse than the performance obtained using an adaptive model. The conclusion is that a fixed pre-trained model could be used without significant penalties in the resulting rate. The little impact in rate can be justified by the random access capability which is acquired when a fixed arithmetic encoder model is used. A major drawback of this approach is the need of several pre-trained models, since the number of symbols in a model depends on the transform size used, which is a free input parameter of MuLE-TD.

In order to overcome the need of several pre-trained models to account several transform size choices, other experiment was conduced. It consisted in obtaining an arithmetic encoder model m_1 with N_1 symbols from a pre-trained model m_2 with N_2 symbols, where $N_2 > N_1$, by discarding the extra unused symbols. The model m_1 is obtained by encoding the light fields in the *development* set using a large transform size, and the encoding of light fields in the *testing* set is performed using a smaller transform size, which requires a model with fewer symbols. The codec RD performance achieved in this experiment is worse than the one obtained when using a pre-trained model for respective transform size, as shown in Figure 5.5, in red. From this result, we conclude that the approach using GOBs is preferable since it provides reasonable degree of random access with less penalty in the rate.

The final experiment regarding the arithmetic encoder was the encoding of a given light field using a fixed equiprobable model. This experiment resulted in RD performance far worse than previous experiments, suggesting that there is a significant difference in run-length symbols probabilities that should be exploited for reducing the average bitrate. The results are shown in Figure 5.5 for mosaic light field *Lenslets*.

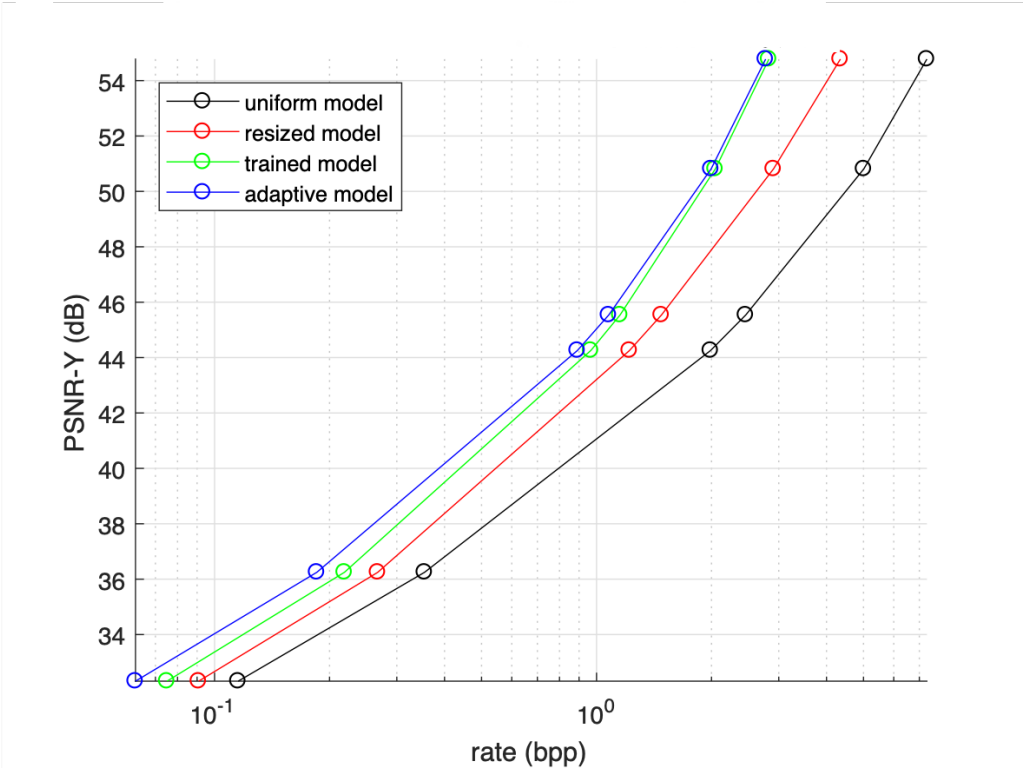


Figure 5.5: MuLE-TD RD performance using several arithmetic encoder models for mosaic light field *Lenslets*.

5.3 RD performance-based scanning modes assessment

In Chapter 4, the scanning modes are assessed according to their capability of scanning the 4D coefficients in decreasing order of energy. Now, with a complete codec implementing such scanning modes, they can be assessed according to their capability of reducing the final rate in a real coding scenario. An experiment aiming at assessing the efficiency of the scanning modes based on RD performance consists in encoding light fields employing the scanning modes and comparing the resulting bitrate. The codec settings used in the experiment are shown in table 5.3:

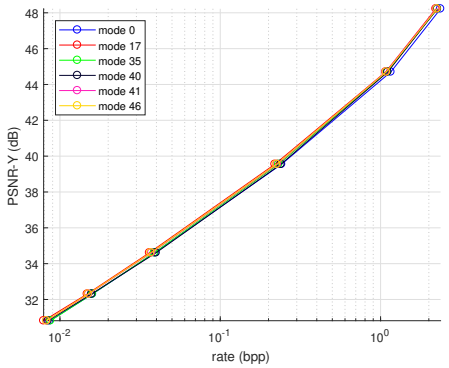
Table 5.3: MuLE-TD settings used in scanning modes performance assessment experiment.

	Light fields		
Parameters	*Bikes*	*Set2 2k Sub*	*Greek*
Transform size	$13 \times 13 \times 21 \times 25$	$11 \times 11 \times 108 \times 128$	$9 \times 9 \times 128 \times 128$
Arithmetic encoder model	Adaptive	Adaptive	Adaptive
Scanning modes	0, 17, 35, 40, 41, 46	0, 23, 30, 39, 40, 41, 46	0, 23, 30, 39, 40, 41, 46
Quantization steps	0.01, 0.02, 0.04, 0.13, 0.3, 1.4, 5	0.01, 0.02, 0.04, 0.13, 0.3, 1.4, 5	0.01, 0.02, 0.04, 0.13, 0.3, 1.4, 5
GOB size	$1 \times 1 \times 14 \times 25$	$1 \times 1 \times 10 \times 15$	$1 \times 1 \times 4 \times 4$

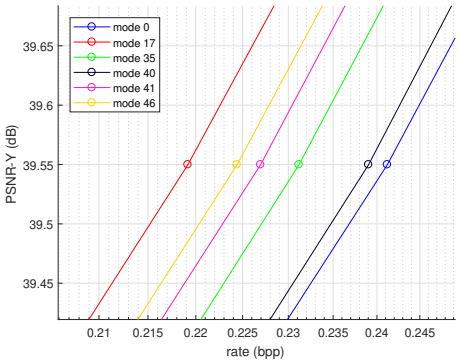
Figure 5.6 shows the RD results expressed as PSNR-Y (dB) versus rate (bpp) for selected scanning modes for light fields *Bikes*, *Set2 2k Sub* and *Greek*. Figure 5.6 (a) shows the RD performance for selected scanning patterns: mode 0 (directional $t \rightarrow s \rightarrow v \rightarrow u$), mode 17 (directional $u \rightarrow v \rightarrow t \rightarrow s$), mode 35 (2D directional in $u \rightarrow v$ with diagonal in (t, s)), mode 40 (4D diagonal), mode 41 (double 2D diagonal with inner diagonal in (t, s) and outer diagonal in(v, u)) and mode 46 (double 2D diagonal with inner diagonal in (v, u) and outer diagonal in (t, s)) for the light field *Bikes*. Figure 5.6 (b) depicts the same results in detail for PSNR values around 39.5 dB and highlights the differences in the codec performance applying the selected scanning modes. The results show that the codec tends to perform better when the scanning patterns corresponding to higher integral values, as depicted in Figure 4.16 (b), are used, which is also consistent with the energy distribution shown on Figure 4.16 (a). As expected from results presented in Section 4.3, the worst performance is obtained for directional scanning mode 0, which scans the block dimensions in the order $t \rightarrow s \rightarrow v \rightarrow u$. The best result is obtained for mode 46 which performs the double 2D diagonal scanning with inner loop in (v, u) and outer loop in (t, s) the

best performances.

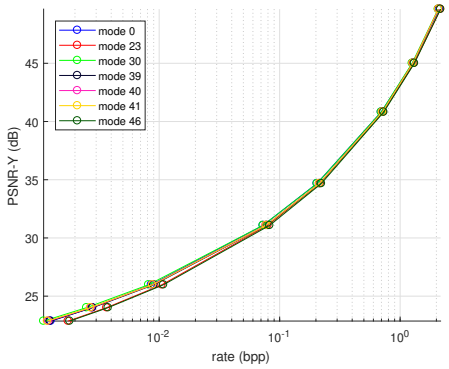
Figure 5.6 (c) shows the RD performance for selected scanning patterns: mode 0 (directional $t \to s \to v \to u$), mode 23 (directional $u \to v \to s \to t$), mode 39 (3D diagonal in $(t, s.v)$ with directional in u), mode 40 (4D diagonal), mode 41 (double diagonal 2D with inner diagonal in (t, s) and outer diagonal in(v, u)) and mode 46 (double diagonal 2D with inner diagonal in (v, u) and outer diagonal in (t, s)) for the light field *Set2 2k Sub*, while Figure 5.6 (d) depicts the same results in detail for PSNR values around 31 dB. The results also agree with the ones shown in Figure 5.6 (d). As expected, modes 0 and 30 have the best performances, while the worst are achieved by modes 23 and 46.



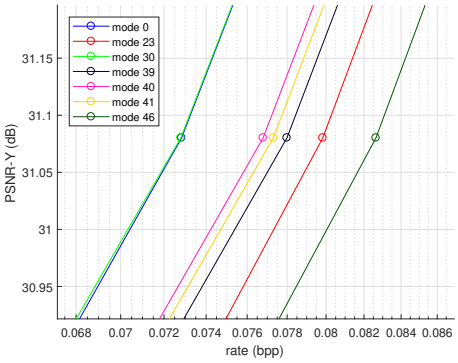
(a) *Bikes*



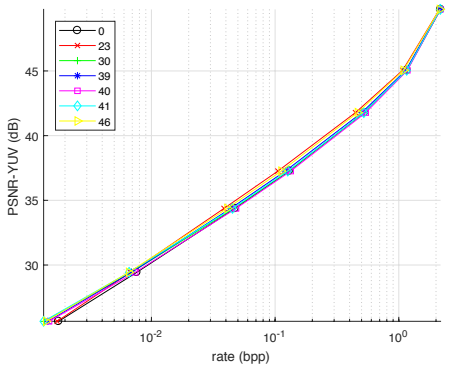
(b) *Bikes*: zoom around 39.5 dB.



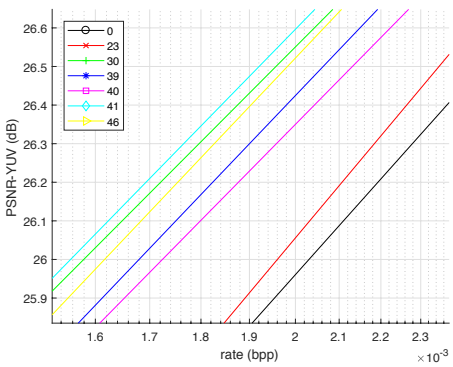
(c) *Set2*



(d) *Set2*: zoom around 31 dB.



(e) *Greek*



(f) *Greek*: zoom around 26 dB.

Figure 5.6: MuLE-TD RD performance using selected scanning modes for light fields *Bikes*, *Set2 2k Sub* and *Greek*.

Figure 5.6 (e) shows the RD performance *Greek*, using same scanning patterns as for *Set2 2k Sub*. Figure 5.6 (f) depicts the same results in detail for PSNR values around 26 dB. The results also agree with the ones shown in Figure 5.6 (f). As expected, the directional modes 0 and 23 have the worst performances, while the remaining modes produce smaller bitrates.

5.4 Coding solution performance assessment

In this section, the overall RD performance of MuLE-TD is assessed. Subsection 5.4.1 describe performance metrics and benchmarks while subsection 5.4.4 describes the test material and coding conditions used in the assessment of the proposed coding solution. This information is made public through the document *JPEG Pleno: light field coding common test conditions* [12].

5.4.1 Performance metrics and benchmarks

Rate metrics

The rate used for coding light fields is defined as the number of bits used to represent the compressed light field by the total number of pixels in the light field [12]:

$$\text{bpp} = \frac{\text{N total bits}}{\text{N total pixels}} \tag{5.9}$$

For example, for HDCA *Set 2 2k Sub* which has 33×11 views with resolution 1920×1080 each, the total number pixels is $33 \times 11 \times 1920 \times 1080 = 752,716,800$.

PSNR quality metric

A common definition for the PSNR is

$$\text{PSNR} = 10 \cdot \log_{10}\left(\frac{2^n - 1}{\text{MSE}}\right) \tag{5.10}$$

where MSE is the mean square error, defined between two images I and I' of resolution $M \times N$ as

$$\text{MSE} = \frac{1}{M \cdot N}\sum_{i=0}^{M}\sum_{j=0}^{N}(I(i,j) - I'(i,j))^2. \tag{5.11}$$

Once the PSNR is calculated for each one of the channels Y, Cb and Cr, the overall PSNR_{YCbCr} is computed for a single view as follows [12]:

$$\text{PSNR}_{YCbCr} = \frac{6 \cdot \text{PSNR}_Y + \text{PSNR}_{Cb} + \text{PSNR}_{Cr}}{8}. \tag{5.12}$$

The PSNRs for the whole light field are computed by averaging the PSNRs of the individual views.

SSIM quality metric

The SSIM quality metric is computed according according to [31], using

$$\text{SSIM\_Y}(I_1, I_2) = \frac{(2\mu_1\mu_2 + c_1)(2\sigma_{1,2} + c_2)}{(\mu_1^2 + \mu_2^2 + c_1)(\sigma_1^2 + \sigma_2^2 + c_2)} \tag{5.13}$$

in various windows of the images, where $\mu_{<X>}$ is the average of image X, $\sigma^2_{<X>}$ is the variance of image X and $c_{<X>}$ is a variable which depends on image X dynamic range. The SSIM for the whole light field is computed by averaging the SSIM values of the individual views.

5.4.2 Benchmarks

In JPEG Pleno activities, the coding solutions candidates to integrate the novel standard have their RD performance compared with an anchor. The anchor selected by JPEG Pleno is the HEVC standard. Next, the anchor generation process is described.

HEVC anchor

The HEVC anchor generation process is summarized in Figure 5.7. [12]:

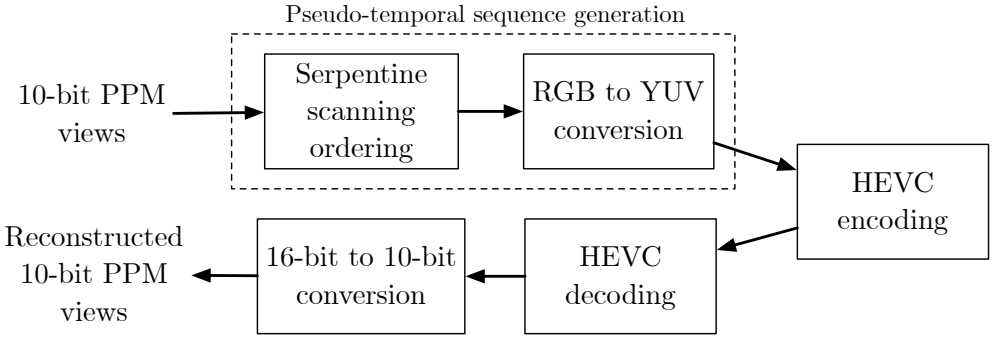


Figure 5.7: HEVC anchor generation encoding/decoding pipeline.

The light field represented as multiple 10-bit PPM files are scanned according to a pattern defined as *Serpentine*, depicted in Figure 5.8. Then, the conversion from RGB to YUV is preformed and the pseudo-temporal sequence is encoded and decoded using *x265* implementation of HEVC, configured to use profile MAIN (10-bit). The conversion between color spaces uses the ITU-R Recommendation BT. 709-6 [32]. After the decoding process, the light field views are converted back to 10-bit PPM files and are ready to be evaluated using PSNR-YUV and SSIM-Y quality metrics.

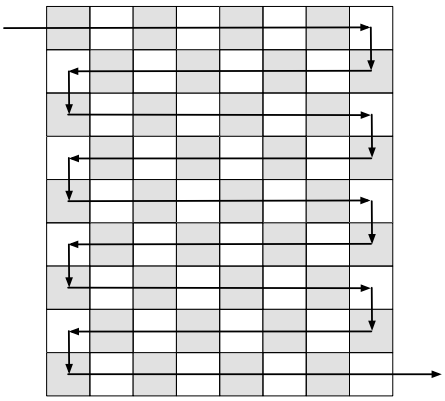


Figure 5.8: Serpentine scanning used for pseudo temporal sequence generation from light field views.

5.4.3 Test material and conditions

Test material

The light fields specified by JPEG Pleno are the ones described in Section 2.4.1. Table 5.4 summarizes the light fields used in JPEG Pleno core and exploration experiments [12].

Table 5.4: Summary of JPEG Pleno test material.

Type	Name	Number of views	Spatial resolution (pixels)
Lenslets	Bikes Danger de Mort Stone Pillars Outside Fountain & Vincent	13×13	625×434
HDCA	Set2 2K Sub	33×11	1920×1080
HDCA	Laboratory1	31×31	1936×1288
HDCA	Tarot Cards	17×17	1024×1024
Synthetic	Greek Sideboard	13×13	625×434

Coding conditions

Table 5.5 contains the target bitrates for the experiments performed with the light fields in JPEG Pleno dataset [12].

Table 5.5: Target bitrates for the JPEG Pleno light field datasets.

Light field	Target bitrate (bpp)					
All lenslets	-	0.001	0.005	0.021	0.1	0.75
Greek and *Sideboard*	-	0.001	0.005	0.02	0.1	0.75
Tarot	-	0.001	0.005	0.021	0.1	0.75
Set2 2k Sub	0.0005	0.001	0.005	0.01	0.05	0.1
Laboratory1	0.0005	0.001	0.005	0.01	0.05	0.1

5.4.4 Overall MuLE-TD RD performance

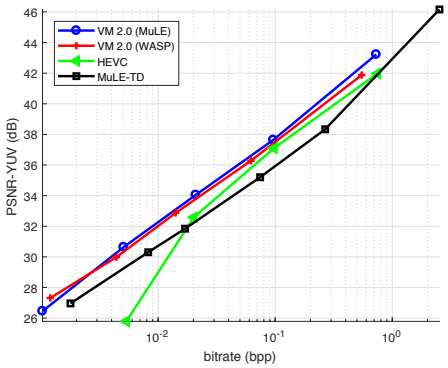
Figure 5.9 compares the MuLE-TD performance with HEVC anchor and other two coding solutions which are participating to the standardization process, MuLE-MTH and WaSP [9]. The RD performance expressed as PSNR-YUV (dB) versus bitrate (bpp) and is obtained for selected lenslet light fields of JPEG Pleno dataset. The PSNR and bitrate values are calculated following the specification found in Sections 5.4.1 and 5.4.1, respectively. The MuLE-TD codec configurations used to generate such results are:

- **Transform size:** The 4D-DCT size used is $(t, s, v, u) = (13, 13, 25, 31)$. If the input light field size in a given dimension is not multiple of the transform size

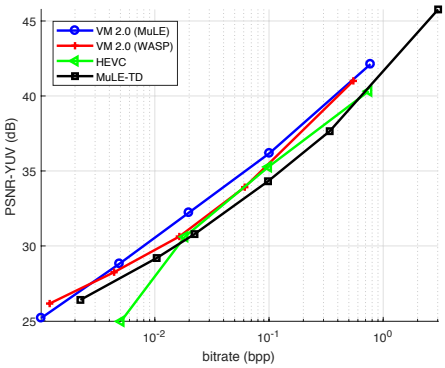
in the respective dimension, the light field should be extended to fulfil this requirement. Since lenslet light field size is $(t, s, v, u) = (13, 13, 625, 434)$, by choosing the given transform size no rate is spent by extending the light field.

- **Arithmetic encoder model:** The adaptive encoder model is chosen for encoding (*run*, *length*) symbols, which is known to be more efficient choice, as shown in results presented in Section 5.2.4;
- **GOB size:** The GOB size configured is $(t, s, v, u) = (1, 1, 25, 14)$, which is the result of the division of lenslet light field size by the transform size. With this choice of GOB size, the adaptive model of the arithmetic encoder is never reset, which results in the reduction of the final rate, as shown in Section 5.2.4;
- **Quantization steps:** The quantization steps used are [0.02, 0.035, 0.08, 0.1, 0.5, 1.3], in order to produce bitrates in the range specified by the JPEG Pleno coding conditions, described in Section ;
- **Maximum run of zeros:** The run_{max} values are obtained using *transform length* $= 13 \times 13 \times 25 \times 31 = 130,975$ in Equation 5.7, resulting in $run_{max} = 362$.

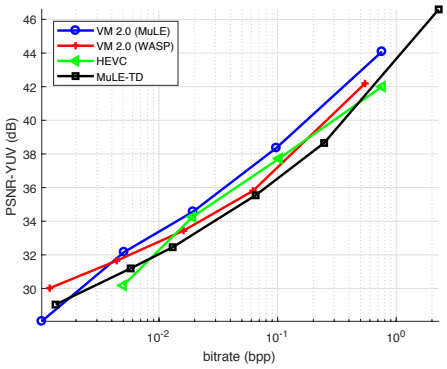
The results for light field *Bikes* shown in 5.9 (a) show MuLE-TD performance around 2 dB below MuLE-MTH and WaSP, and around 1 dB below HEVC anchor. The PSNR and bitrate values are calculated following the specification found in Sections 5.4.1 and 5.4.1, respectively. For bitrates lower than 0.01 bpp, MuLE-TD outperforms HEVC anchor. For light fields *Danger de Mort* and *Stone Pillars Outside* (Figures 5.9 (b) and (c), respectively) MuLE-TD curve is around 1 dB below WaSP curve for a wide range of target bitrates.



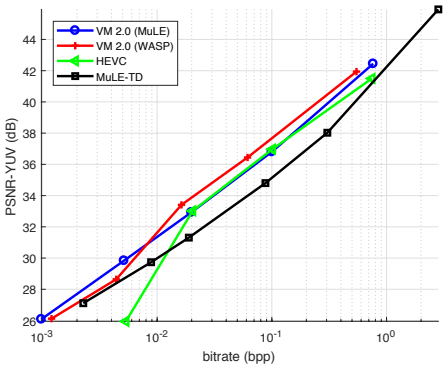
(a) *Bikes*



(b) *Danger de Mort*



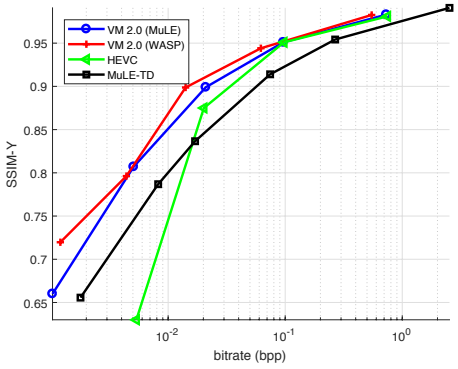
(c) *Stone Pillars Outside*



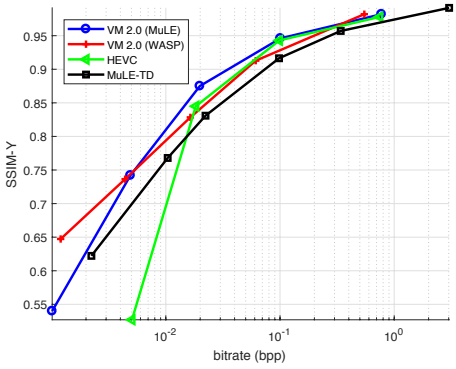
(d) *Fountain & Vincent*

Figure 5.9: RD performance expressed as PSNR-YUV (dB) versus rate (bpp) for lenslets light fields

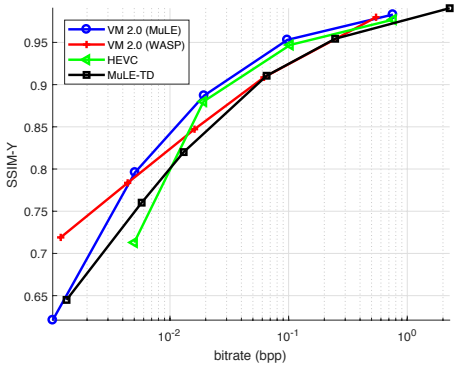
Similar results are obtained when analyzing the RD performance expressed as SSIM-Y versus bitrate (bpp), as show in Figure 5.9, for the same codecs and light fields.



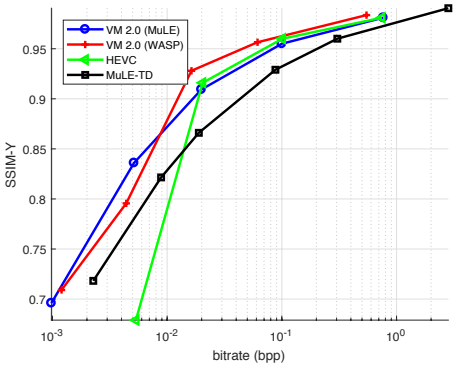
(a) *Bikes*



(b) *Danger de Mort*



(c) *Stone Pillars Outside*



(d) *Fountain & Vincent*

Figure 5.10: RD performance expressed as SSIM-Y versus rate (bpp) for lenslets light fields

As seen in RD results for both PSNR-YUV and SSIM-Y quality metrics, MuLE-TD is not able to achieve same performance level as the coding solutions currently under evaluation in JPEG Pleno and only outperforms HEVC anchor for bitrates lower than 0.01 bpp. The overall conclusion is that a simple and naive coding solution such as MuLE-TD is not competitive when compared with more sophisticated coding solutions.

The MuLE-MTH is light field coding solution similar to MuLE-TD. MuLE-MTH employs the same 4D-DCT transform described in this work, but with variable block sizes. Instead of using scanning and run-length coding, in MuLE-MTH the transform coefficients are grouped using hexadeca-trees and the subsequent encoding is performed in a bitplane-by-bitplane basis [10]. The generated stream is encoded using an adaptive arithmetic encoder in a similar way as it is done in MuLE-TD. Currently, MuLE-MTH coding solution is part of JPEG Pleno Verification Model (VM) 2.0, and it is likely to become part of future JPEG Pleno light field coding standard.

The codec described in [10] is called MuLE-TH and the only difference between it and its near homonymous MuLE-MTH is the variable transform size. MuLE-TH employs fixed block sizes as it is done in MuLE-TD. The RD performances obtained for MuLE-TH and MuLE-MTH are practically the same, thus we conclude that MuLE-TH outperforms MuLE-TD as shown in Figures 5.9 and 5.10. The overall conclusion is that the hexadeca-tree bitplane decomposition is more efficient than the combination of diagonal scanning with run-length coding.

This section presented results only for lenslets light fields in JPEG Pleno dataset. As shown in Chapter 3, the HDCA datasets present much smaller inter-view redundancy when compared to lenslets datasets. The 4D-DCT is not able to efficiently exploit the low degree of 4D redundancy present in HDCA light fields, concluding that 4D-DCT based light field codecs are competitive only for lenslet light field coding.

Chapter 6

Conclusion and future work

As discussed in Chapter 1, the conventional ways to capture the light around us are limited and thus provide a limited user experience, notably in terms of parallax capabilities. Recently, significant advances are emerging in terms of light capturing technologies among which is relevant to highlight the light field imaging which captures a richer representation of the visual scene by measuring the light intensity for each direction and for each pixel position. Such richer representation corresponds to a large amount of data that needs to be efficiently compressed, notably when transmission and storage applications are targeted. Among other ways, this can be achieved by jointly exploiting the light field's intra-view and inter-view redundancies. A straightforward way to do this is by employing 4D transforms, with the 4D-DCT being a natural candidate for this type of processing. In this context, this dissertation has presented a study of light field redundancy and has proposed a light field coding solution based on the 4D DCT.

Chapter 2 started by presenting an introduction to light field imaging, the acquisition techniques and representation models. An introduction to light field coding approaches was addressed, by directly applying or extending available solutions to light fields, or developing novel coding solutions. It also presented the JPEG Pleno, the recent work item initiated by JPEG committee ainming at developing a next generation image coding standard for emerging plenoptic modalities, such as light fields. Emphasis was placed in the JPEG Pleno light field dataset, used in this work as test material.

As a first step towards the design of a light field codec, Chapter 3 has presented a study which aims at characterizing the redundancy of the light fields in the JPEG Pleno dataset using as main tool the 4D-DCT. The main conclusion of the study is that both the lenslet and HDCA JPEG Pleno light field datasets have a great amount of 4D redundancy that can be explored for coding purposes, even though these two types of data may require distinct coding solutions due to the different nature of their 4D redundancy. For the lenslets light fields, the inter-view redundancy is

significantly larger than the intra-view redundancy and for the HDCA dataset the opposite was observed: the intra-view redundancy is much larger than the inter-view sparsity one.

After better understanding the characteristics of light field redundancy and the potential of the 4D-DCT in exploiting it, Chapter 4 presented scanning modes for the 4D-DCT coefficients. The design process started by listing the scanning patterns used by image and video coding standards in the scanning of 2D-DCT coefficients. Several patterns were identified and served as inspiration for the design of the 4D scanning modes. The literature review has shown that the design of efficient scanning modes depends on previous knowledge of the coefficients energy distribution, since the main purpose of such scanning operation is reordering the transform coefficients according to their energies. A study of the 4D-DCT coefficients energy distribution revealed that lenslet, HDCA and synthetic light fields in JPEG Pleno dataset present distinct energy distribution, thus require distinct scanning solutions. A total of 47 modes were designed as extension to 4D of two-dimensional scanning modes used by image and video coding standards. The proposed modes were assessed by their capability of scanning 4D-DCT coefficients in decreasing order of energy.

The designed scanning modes are just one coding tool among others used by MuLE-TD, the light field coding solution proposed in Chapter 5. MuLE-TD has a pipeline similar to the classic JPEG image coding standard, employing transforms, coefficients scanning and run-length coding, in a block based approach. The analysis of the results has shown that this simple codec can achieve a performance close to the coding solutions under evaluation by JPEG Pleno.

One of the major drawbacks of the coding solution is the lack of a prediction step. A suggestion for future work is introducing a 4D prediction of light field samples, thus applying the 4D-DCT in residual data in a similar way as it is done in the state-of-the-art image compression standard HEVC Intra. Hence, the choice among the several designed scanning modes would be function of the selected prediction mode. This approach is likely to improve the codec RD performance.

Appendix A

Published and submitted papers

This appendix presents the list of published and submitted papers resulted from the research work.

A.1 Published papers

C.1 G. Alves, F. Pereira, E. A. B. da Silva. "Light field imaging coding: Performance assessment methodology and standards benchmarking". In: *IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, Seattle, USA, July 2016.

C.2 M. P. Pereira, G. Alves, C. L. Pagliari, M. B. de Carvalho E. A. B. da Silva, F. Pereira, "A geometric space-view redundancy descriptor for light fields: Predicting the compression potential of the JPEG Pleno light field datasets". In: *IEEE 19th International Workshop on Multimedia Signal Processing (MMSP)*, Luton, UK.October 2017.

C.3 G. Alves, M. P. Pereira, M. B. de Carvalho, F. Pereira, C. L. Pagliari, V. Testoni and E. A. B. da Silva. "A Study on the 4D Sparsity of JPEG Pleno Light Fields Using the Discrete Cosine Transform". In: *25th IEEE International Conference on Image Processing (ICIP)*, Athens, Greece. October 2018.

C.4 M. B. de Carvalho, M. P. Pereira, G. Alves, E. A. B. da Silva, C. L. Pagliari, F. Pereira, V. Testoni. "A 4D DCT-Based Lenslet Light Field Codec". In: *25th IEEE International Conference on Image Processing (ICIP)*, Athens, Greece. October 2018.

A.2 Submitted papers

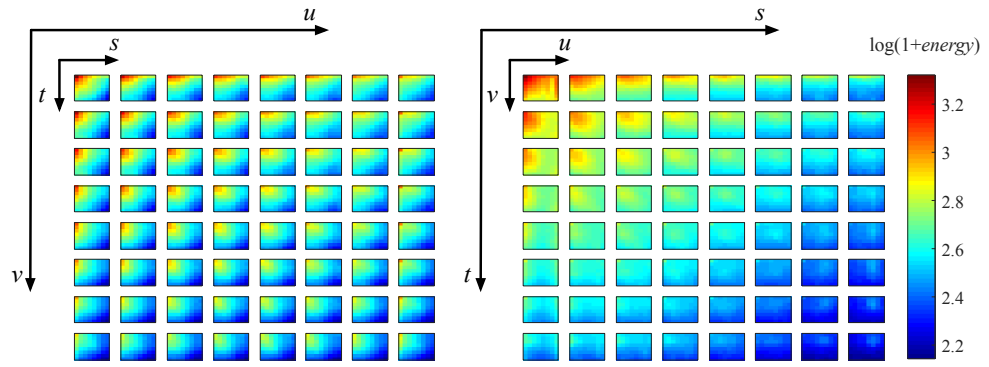
C.1 G. Alves, F. Pereira, C. L. Pagliari, M. B. de Carvalho, M. P. Pereira, V. Testoni, P. Garcia and E. A. B. da Silva. "Efficient 4D-DCT Scanning Modes For Light Field Coding". In: *26th IEEE International Conference on Image Processing (ICIP)* Taipei, Taiwan. October 2019.

Appendix B

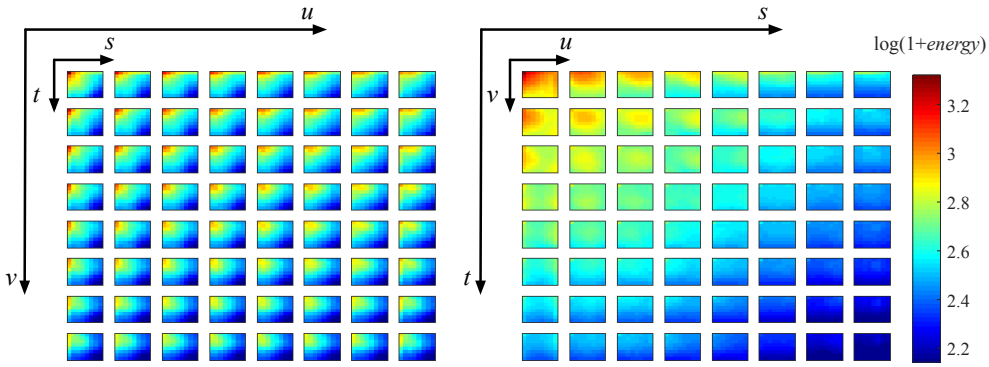
Additional results

B.1 4D-DCT coefficients energy distributions

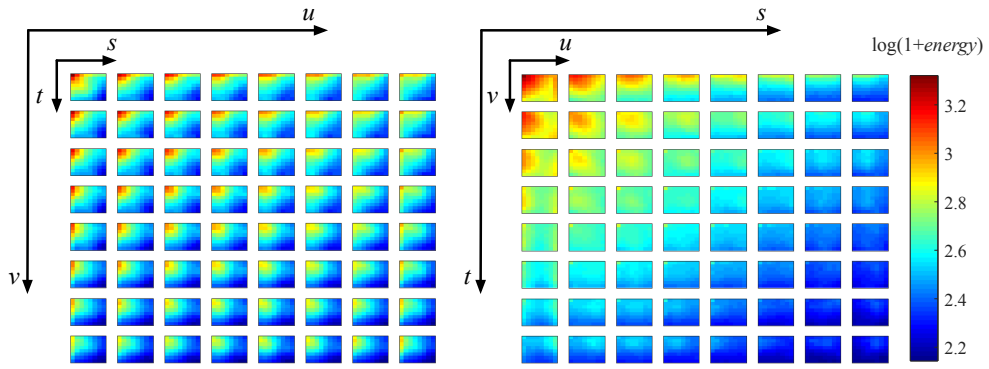
Figures B.1, B.2 and B.3 show additional results to the ones presented in Section 4.3, for all light fields in the JPEG Pleno dataset.



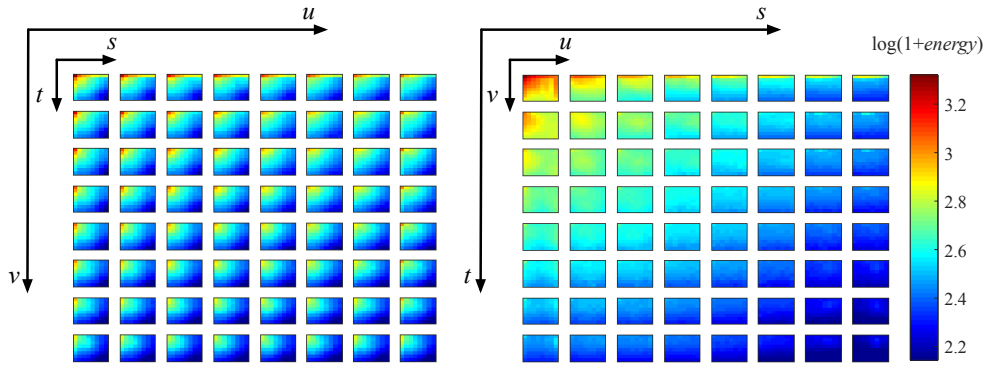
(a) *Bikes.*



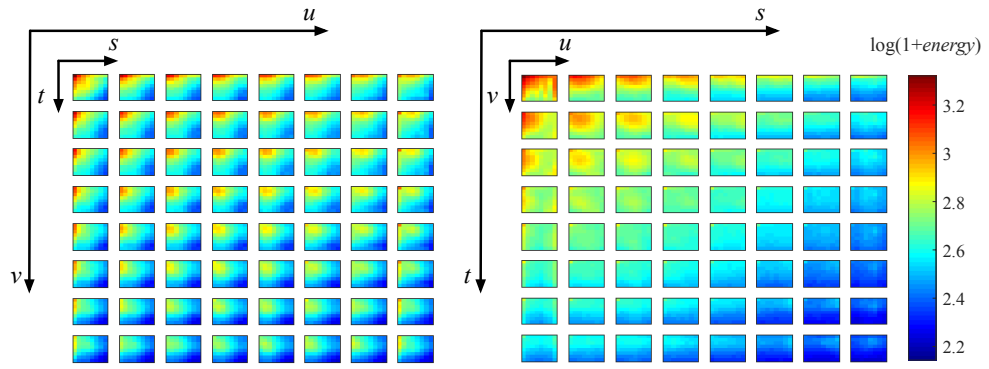
(b) *Stone Pillars Outside.*



(c) *Danger de Mort.*

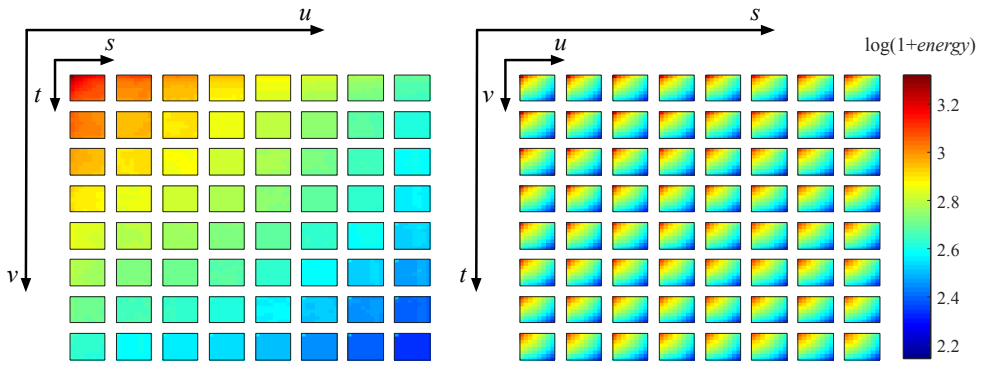


(d) *Friends.*

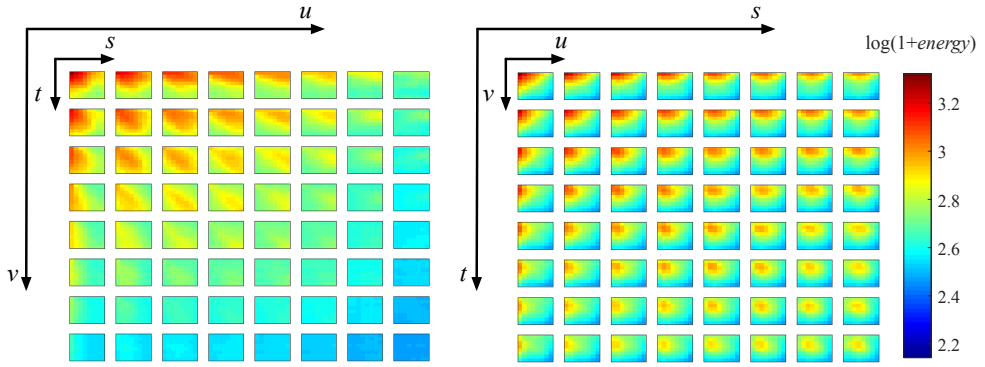


(e) *Fountain & Vincent.*

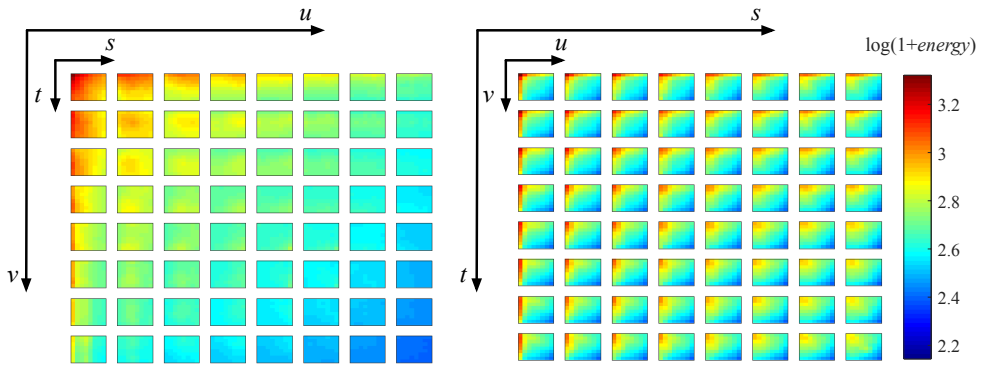
Figure B.1: Energy distribution estimation for the 4D-DCT coefficients for the lenslet light fields of JPEG Pleno dataset.



(a) *Set 2 2K Sub.*

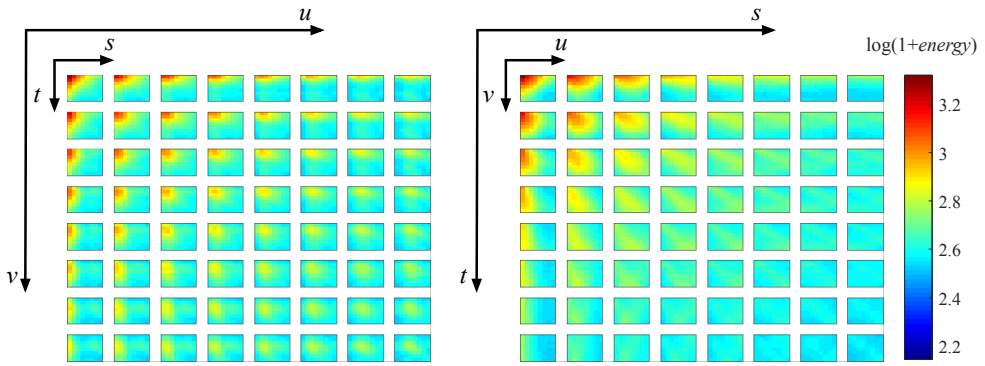


(b) *Tarot Cards.*

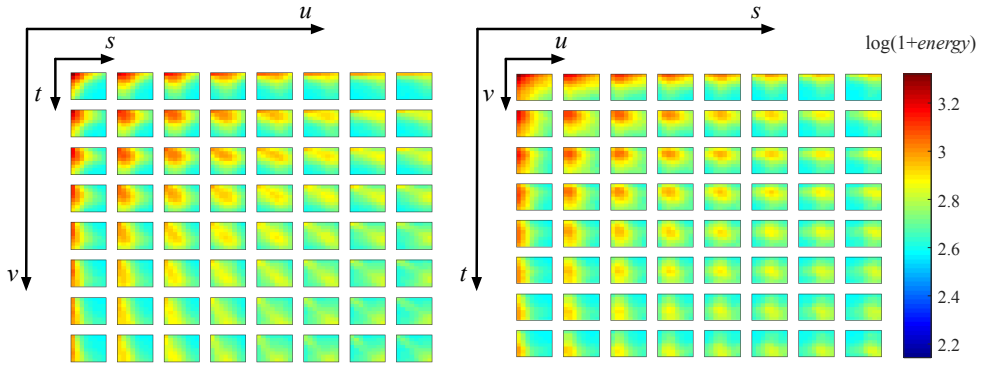


(c) *Laboratory 1.*

Figure B.2: Energy distribution estimation for the 4D-DCT coefficients for the HDCA light fields of JPEG Pleno dataset.



(a) *Greek*.

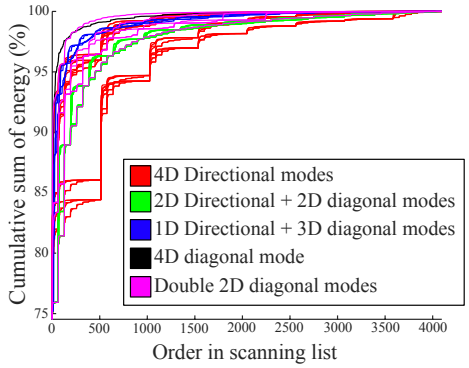


(b) *Sideboard*.

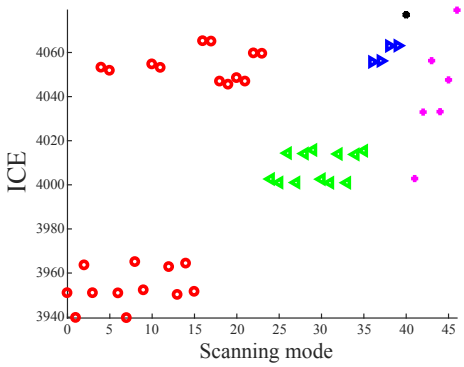
Figure B.3: Energy distribution estimation for the 4D-DCT coefficients for the synthetic light fields of JPEG Pleno dataset.

B.2 Integral of cumulative energy

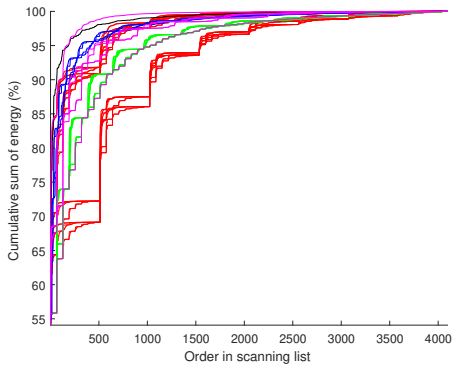
Figures B.4, B.5 and B.6 show additional results to the ones presented in Section 4.5, for all light fields in the JPEG Pleno dataset.



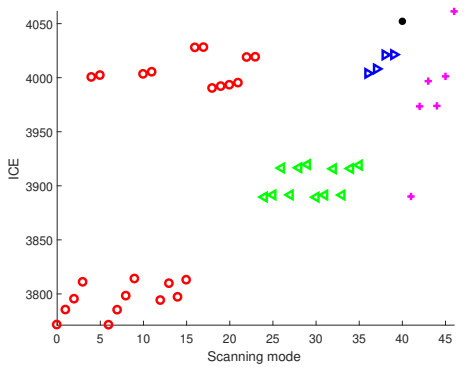
(a) CE curves for *Bikes*.



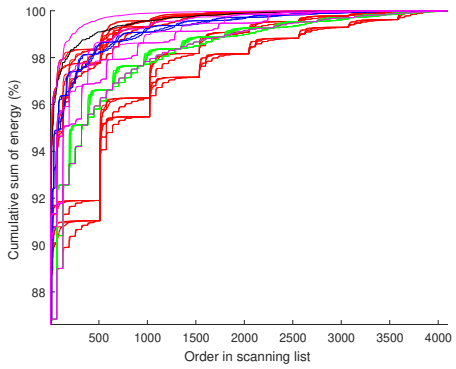
(b) ICE values for *Bikes*.



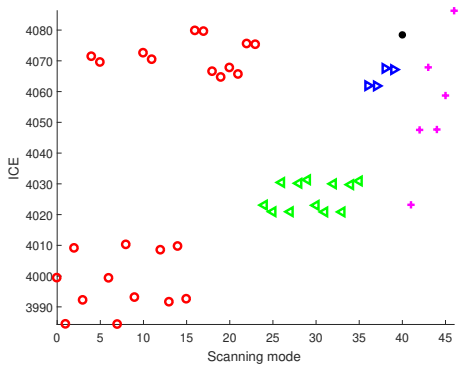
(c) CE curves for *Danger de Mort*.



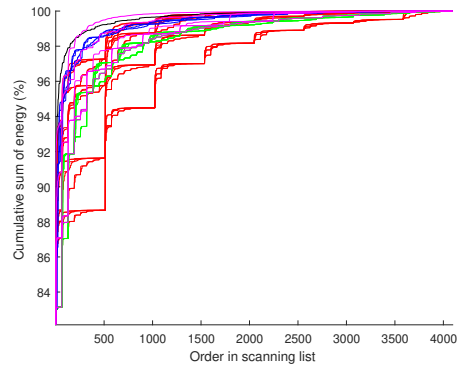
(d) ICE values for *Danger de Mort*.



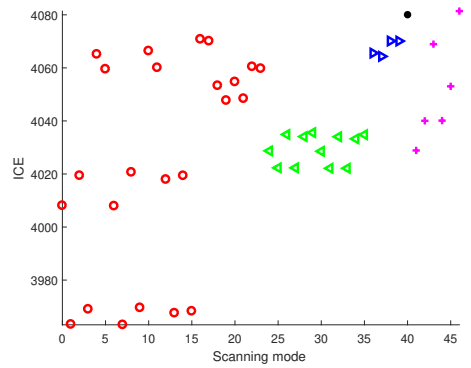
(e) CE curves for *Stone Pillars Outside*.



(f) ICE values for *Stone Pillars Outside*.

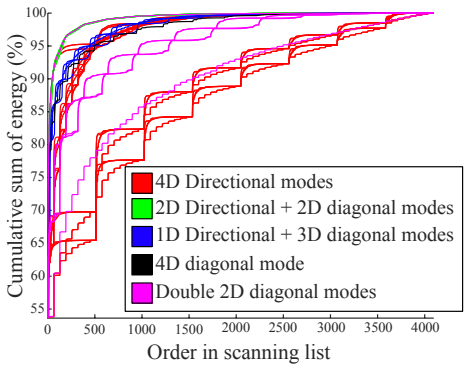


(g) CE curves for *Fountain & Vincent*.

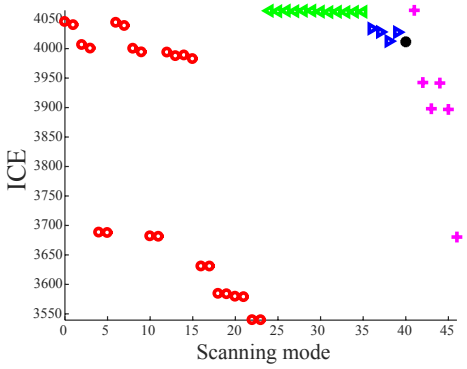


(h) ICE values for *Fountain & Vincent*.

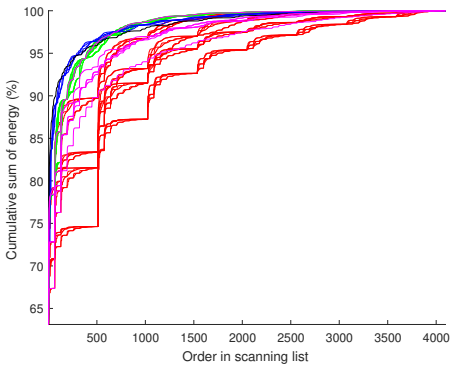
Figure B.4: Left: Cumulative energy curve versus the order in the scanning list. Right: Integral of cumulative energy (ICE) for the 47 scanning modes – lenslets light fields *Bikes*, *Danger de Mort*, *Stone Pillars Outside* and *Fountain & Vincent*.



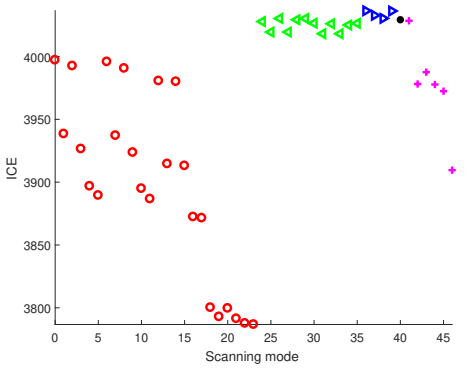
(a) CE curves for *Set2 2k Sub*.



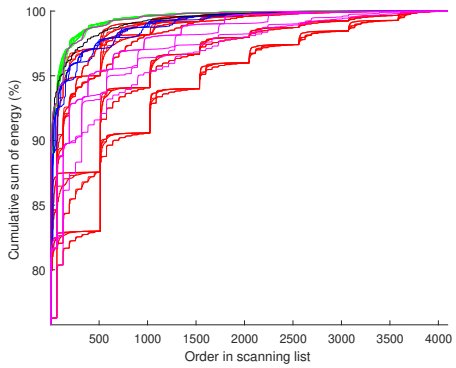
(b) ICE values for *Set2 2k Sub*.



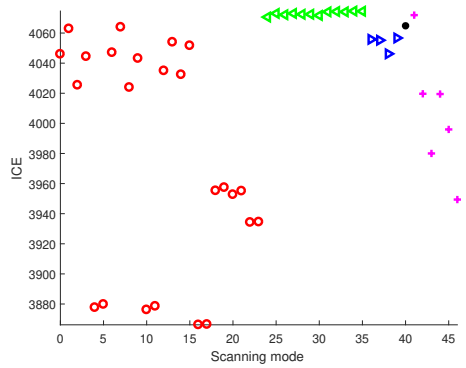
(c) CE curves for *Tarot Cards*.



(d) ICE values for *Tarot Cards*.

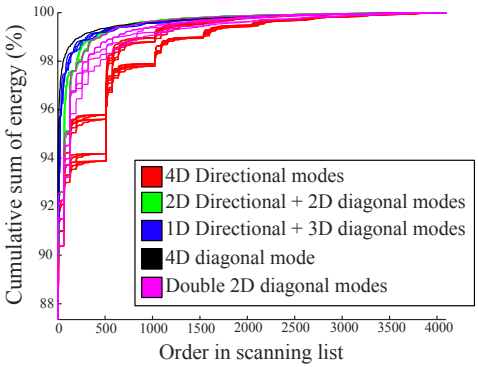


(e) CE curves for *Laboratory1*.

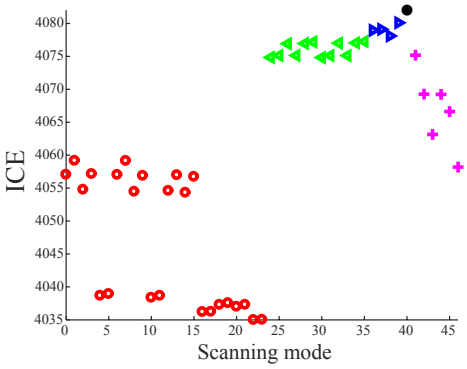


(f) ICE values for *Laboratory1*.

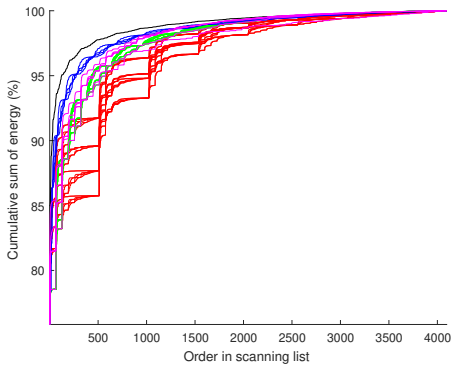
Figure B.5: Left: Cumulative energy curve versus the order in the scanning list. Right: Integral of cumulative energy (ICE) for the 47 scanning modes – HDCA light fields *Set2 2k Sub*, *Tarot Cards* and *Laboratory1*.



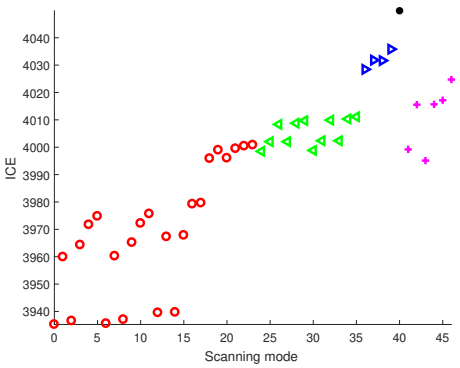
(a) CE curves for *Greek*.



(b) ICE values for *Greek*.



(c) CE curves for *Sideboard*.



(d) ICE values for *Sideboard*.

Figure B.6: Left: Cumulative energy curve versus the order in the scanning list. Right: Integral of cumulative energy (ICE) for the 47 scanning modes – Synthetic light fields *Greek* and *Sideboard*.

Appendix C

Geometric space-view redundancy descriptor

This Appendix briefly describes the Geometric Space-View Redundancy (GSVR) descriptor [23]. The results of the descriptor when applied to JPEG Pleno dataset are used in the analysis of the sparsity of the four-dimensional discrete cosine transform coefficients performed in Chapter 3.

C.1 Space-view redundancy descriptor definition

In block-based image and video coding, the block size defines a region containing a redundancy level worthy to be exploited. In the adaptive block size approach, the choice of the block size is driven by the statistical characteristics of the content to be coded. It is natural that larger block sizes will be chosen for smooth and near constant regions and smaller blocks will be used in heavily texturized ones. Nevertheless, if a block based light field coding is considered, the choice of the dimensions of the four-dimensional block would be far more complicated, due to the presence of a novel and more complex kind of redundancy. The appropriate choice of block inter-view dimensions requires a way to characterize the geometric redundancy of the light fields. In this section, a descriptor devoted to this purpose is presented.

The GSVR [23] is a descriptor which aims at characterizing the light fields defining the largest region in the 4D space that presents four-dimensional geometric redundancy worthy to be exploited for encoding purposes. It does so by measuring the probability that the image of a point in 3D space to remain in the space block in all views, for each space block size and range of views. The GSVR descriptor gives the relation between the intra-view and inter-view block dimensions guaranteeing, with a given probability, the existence of such 4D space-view redundancy.

In order to estimate the largest region that presents four-dimensional geometric redundancy, the method takes 4D light field blocks of $L \times L \times K \times K$ dimensions, with $L \times L$ being the spatial (intra-view) block size and with $K \times K$ being the number of horizontal and vertical views in the region. Let $UV\ (s,t)$ be an $L \times L$ square region in the view of coordinates (s,t). For each 4D block, the method evaluates the probability that a point in 3D space, with a projection onto $UV\ (s_0,t_0)$ will also have a projection onto another view $UV\ (s_1,t_1)$ of the same 4D block.

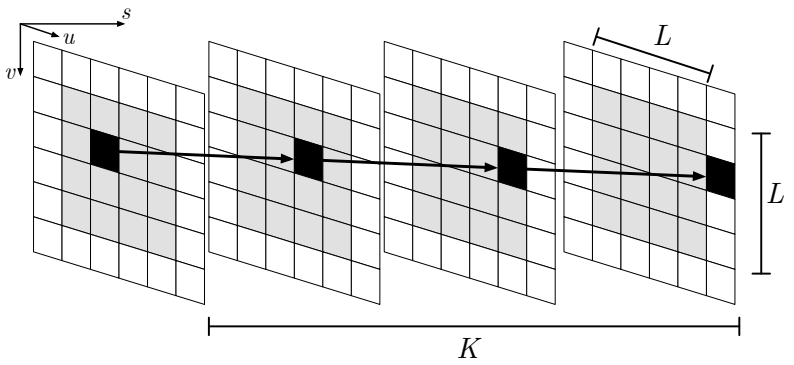


Figure C.1: A point moving across four 4×4 blocks (gray area) in four horizontal views.

Figure C.1 illustrates the idea in 3D which means considering a light field with only horizontal views (i.e., fixing $t = 0$), for better visualization. There, a point represented as a black pixel moves across four blocks, of size 4×4, depicted in gray, belonging to four different horizontal views ((u,v) planes) along the s axis. In this case, $L = 4$, which corresponds to the gray block size, and $K = 4$, which is the number of horizontal views . The first 4×4 block belongs to view of coordinates ($s = 0$, $t = 0$), the second to view ($s = 1$, $t = 0$), the third to view ($s = 2$, $t = 0$) and the fourth to view ($s = 3$, $t = 0$). The set of gray blocks correspond to the $4 \times 4 \times 4$ space-view 3D block. Note that the black pixel belongs to the 4×4 spatial block in only the first three views from the left to the right, falling outside it in the fourth view. The desired probability is estimated by computing the number of views in which a chosen point is still inside the gray area, repeating the process for each pixel in the block. The position of a chosen point in several views can be determined by computing the displacements using disparity estimation techniques. The correspondence method provides the displacement (disparity) values for each pixel inside a block in the (u,v) plane, belonging to a view in the (s,t) plane. Therefore, the geometric space-view redundancy (GSVR) descriptor is obtained by following the steps depicted in Figure C.2.

Using the disparities between adjacent views, calculated for vertical and horizontal directions, probabilities of a pixel stay or leaving a block of size $L \times L$, in a

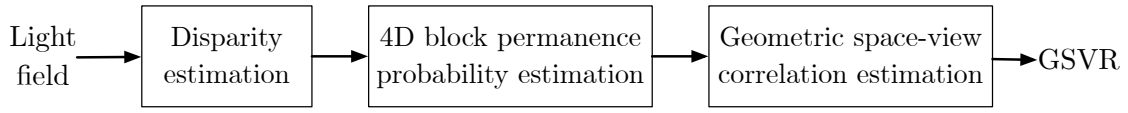


Figure C.2: GSVR descriptor computation pipeline.

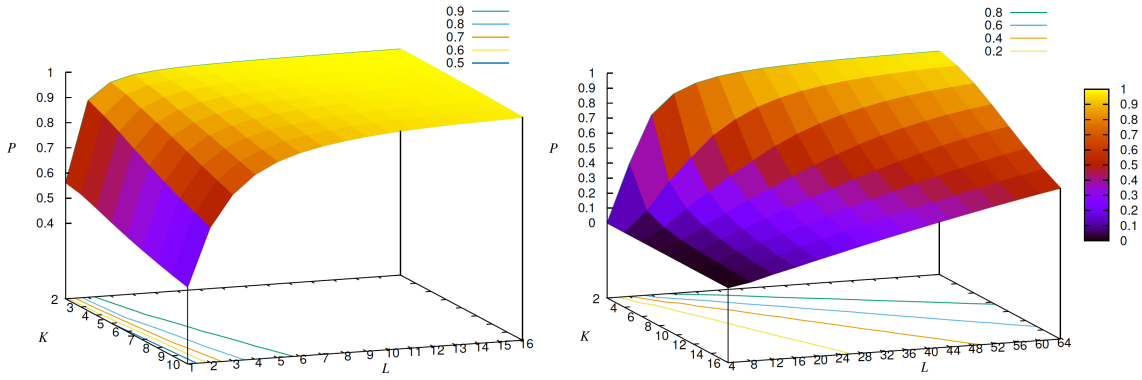
views block of size $K \times K$, are calculated. For example, the test is done for each pixel in a block (L^2 pixels) for the horizontal direction, using horizontal disparities (relative to $K-1$ views), for each line of views (K lines of views). The total number of tests is $L \cdot L \cdot K \cdot (K-1)$, quantity used as normalization factor, for the number of positive tests for permanence of a pixel inside a block. The same concept is used for vertical disparities. A formalization follows: Let $d\ (u, v, s, t)$ be the disparity values at spatial position (u, v) for the s-th horizontal view and the t-th vertical view. The vertical and horizontal probabilities, respectively P_v and P_h, that a point at any position $(u^{'}, v^{'}, s^{'}, t^{'})$ inside a block of size $L \times L \times K \times K$ will have a geometrically related counterpart inside the same block can be estimated by:

$$P_h(L,K) = \frac{1}{L^2K} \sum_{u=0}^{L-1} \sum_{v=0}^{L-1} \sum_{t=0}^{K-1} \frac{1}{k-1} \sum_{S=0}^{K-2} \phi_L \left(u + \sum_{s=0}^{S} d_h(u,v,s,t)) \right) \tag{C.1}$$

$$P_v(L,K) = \frac{1}{L^2K} \sum_{u=0}^{L-1} \sum_{v=0}^{L-1} \sum_{s=0}^{K-1} \frac{1}{k-1} \sum_{T=0}^{K-2} \phi_L \left(v + \sum_{t=0}^{T} d_v(u,v,s,t)) \right) \tag{C.2}$$

$$P(L,K) = P_h + P_v - P_h P_v \tag{C.3}$$

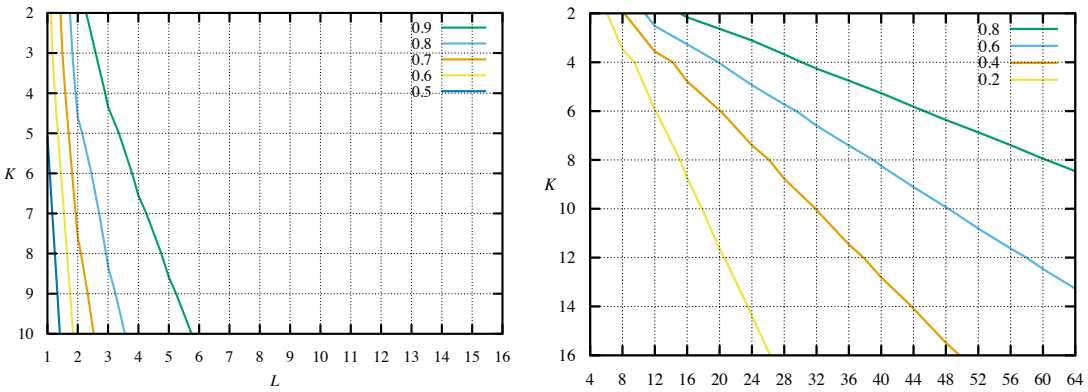
where $\phi_L(x)$ is an indicator function that equals 1 if $(0 \leq x \leq L-1)$, and zero otherwise. For example, in Equation C.1, the summation from $S = 0$ to $K-2$ increments a counter if a disparity-shifted pixel position is still inside the window. The probability of a pixel to stay inside a block is the probability of the union of horizontal and vertical permanence of a pixel inside a block, calculated in Equation C.3. Figure C.3 shows the probability surfaces for the lenslets light field *Bikes* (a) and the HDCA light field *Set 2* (b), computed for several values of K and L.



(a) *Bikes.* (b) *Set 2.*

Figure C.3: 4D block permanence probability for selected light fields.

Figure C.3 carries a great amount of information, which can be heavy to process. However, when looking at the iso probability contour plots in the same figure, that are also depicted in Figure C.4, it can be noticed that these curves may be approximated by straight lines passing through the origin. In addition, it can be seen that the way that the slopes of these curves (space-view correlation) vary with the probability, is different for different types of datasets. This suggests a compact way to represent the geometric space-view redundancy using the variation of slope of those straight lines with the change of probability. A geometric space-view redundancy descriptor (GSVR), for the space-view correlation, based on those slopes is proposed and will be formally described in what follows.



(a) Iso probability contours for *Bikes.* (b) Iso probability contours for *Set 2.*

Figure C.4: Iso probability contours for selected light fields from JPEG Pleno dataset.

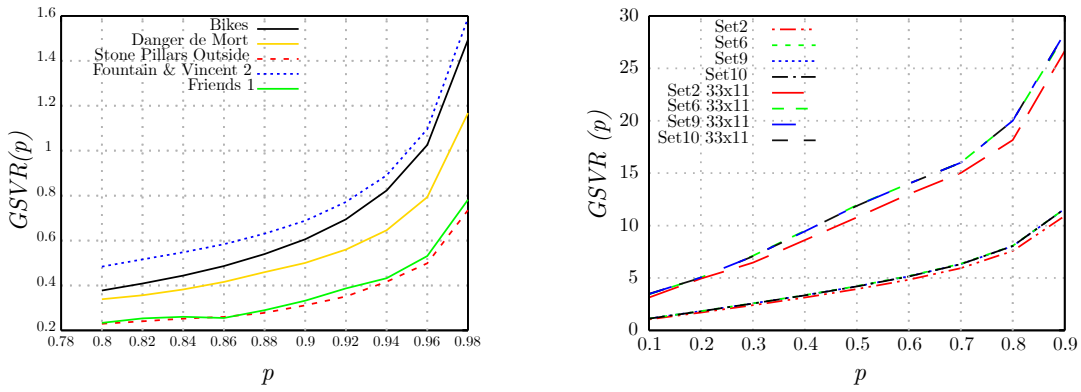
Let $C(L, K, p) = \{(L, K)|P(L, K) = p\}$ be the set of points of the contour curve at probability p. Let $L = \alpha(C(L, K, p))K$ be the best fitting line obtained by a linear regression on the points of $C(L, K, p)$. The geometric space-view redundancy descriptor, GSVR(p), is defined as the angular coefficient (α) of this line, given by Equation C.4:

$$\mathrm{GSVR}(p) = \alpha(C(L, K, p)). \tag{C.4}$$

Such angular coefficient corresponds to the slope of the best fitting line of an iso probability contour of the 4D block permanence probability surface. It expresses the correlation between the spatial and view coordinates for a given permanence probability. Therefore, it measures the space-view correlation within the light field.

C.2 Analyzing JPEG Pleno light field datasets

Figure C.5 displays the GSVR(p) curves for lenslets (a) with p varying from 0.8 to 1 and for HDCA (b) with p varying from 0.1 to 0.9. All curves monotonically increase, indicating that 3D points with large disparity values move faster across the different views and need larger spatial block sizes to remain inside a block across the views. Likewise, smaller disparity values move slower across the different views and need smaller spatial block sizes to remain inside a block across the views. In addition, a smaller spatial block size tends to be more stationary, therefore with easier redundancy exploitation. By observing the plots, it is noticed that, for the lenslets, the inter-view redundancy of a 4D block can be well exploited using spatial blocks of a much smaller size than in the case of HDCA. Moreover, it is possible to observe that the HDCA light fields present very close GSVR(p) values, leading to a conclusion that such light fields may present very little diversity of scene geometries.



(a) GSVR(p) plot for the lenslets light fields.

(b) GSVR(p) plot for the HDCA light fields.

Figure C.5: GSVR(p) plots for JPEG Pleno dataset.

Bibliography

[1] CONTI, C., NUNES, P., SOARES, L. D. "New HEVC prediction modes for 3D holoscopic video coding". In: *2012 19th IEEE International Conference on Image Processing*, pp. 1325–1328, Sep. 2012. doi: 10.1109/ICIP.2012.6467112.

[2] Doc. ISO/IEC JTC 1/SC 29/WG1 N74014. "JPEG Pleno Call for Proposals on Light Field Coding". January 2017.

[3] WALLACE, G. K. "The JPEG still picture compression standard", *IEEE Transactions on Consumer Electronics*, v. 38, n. 1, pp. xviii–xxxiv, Feb 1992. ISSN: 0098-3063. doi: 10.1109/30.125072.

[4] ADELSON, E. H., BERGEN, J. R. "The Plenoptic Function and the Elements of Early Vision". In: *Computational Models of Visual Processing*, pp. 3–20. MIT Press, 1991.

[5] LEVOY, M., HANRAHAN, P. "Light Field Rendering". In: *23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pp. 31–42, New York, NY, USA, 1996. ACM. ISBN: 0-89791-746-4. doi: 10.1145/237170.237199. Available at: <`http://doi.acm.org/10.1145/237170.237199`>.

[6] ALVES, G., PEREIRA, F., DA SILVA, E. A. B. "Light field imaging coding: Performance assessment methodology and standards benchmarking". In: *2016 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, pp. 1–6, July 2016. doi: 10.1109/ICMEW.2016.7574774.

[7] VIEIRA, A., DUARTE, H., PERRA, C., et al. "Data formats for high efficiency coding of Lytro-Illum light fields". In: *2015 International Conference on Image Processing Theory, Tools and Applications (IPTA)*, pp. 494–497, Nov 2015. doi: 10.1109/IPTA.2015.7367195.

[8] CONCEIÇÃO, R., PORTO, M., ZATT, B., et al. "LF-CAE: Context-Adaptive Encoding for Lenslet Light Fields Using HEVC". In: *2018 25th IEEE*

International Conference on Image Processing (ICIP), pp. 3174–3178, Oct 2018. doi: 10.1109/ICIP.2018.8451345.

[9] ASTOLA, P., TABUS, I. "WaSP: Hierarchical Warping, Merging, and Sparse Prediction for Light Field Image Compression". In: *2018 7th European Workshop on Visual Information Processing (EUVIP)*. IEEE, 11 2018. doi: 10.1109/EUVIP.2018.8611756.

[10] DE CARVALHO, M. B., PEREIRA, M. P., ALVES, G., et al. "A 4D DCT-Based Lenslet Light Field Codec". In: *2018 25th IEEE International Conference on Image Processing (ICIP)*, pp. 435–439, Oct 2018. doi: 10.1109/ICIP.2018.8451684.

[11] ITU-T. "JPEG 2000 image coding system: Extensions". Recommendation ITU-T T.801, August 2002.

[12] Doc. ISO/IEC JTC 1/SC 29/WG1 N81053. *JPEG Pleno Light Field Coding Common Test Conditions*. Relatório técnico, October 2018.

[13] "Lytro - Wikipedia". `https://en.wikipedia.org/wiki/Lytro`, 2019. Accessed: 2019-02-01.

[14] RERABEK, M., EBRAHIMI, T. "New Light Field Image Dataset", 2016. Available at: <`http://infoscience.epfl.ch/record/218363`>.

[15] Doc. ISO/IEC JTC 1/SC 29/WG1 M75008. "Specifications for High-density Camera Array (HDCA) Data Sets". March 2017.

[16] Doc. ISO/IEC JTC 1/SC 29/WG1 N77016. "Core Experiments Set #2 for JPEG Pleno". October 2017.

[17] Doc. ISO/IEC JTC 1/SC 29/WG1 M36566. "Super-multi-view light-field images from Poznan University of Technology". July 2015.

[18] "The (New) Stanford Light Field Archive". `http://lightfield.stanford.edu/`. Accessed: 2019-02-01.

[19] "HCI 4D Light Field Dataset". `http://hci-lightfield.iwr.uni-heidelberg.de/`. Accessed: 2019-02-01.

[20] ITU-T. "Advanced Video Coding for Generic Audiovisual Services". Recommendation ITU-T H.264, March 2010.

[21] SULLIVAN, G. J., OHM, J.-R., HAN, W.-J., et al. "Overview of the High Efficiency Video Coding (HEVC) Standard", *IEEE Trans. Cir. and Sys.*

for Video Technol., v. 22, n. 12, pp. 1649–1668, dez. 2012. ISSN: 1051-8215. doi: 10.1109/TCSVT.2012.2221191. Available at: <http://dx.doi.org/10.1109/TCSVT.2012.2221191>.

[22] ITU-T, ISO/IEC. "High Efficiency Video Coding". Recommendation ITU-T H.265, February 2013.

[23] PEREIRA, M. P., ALVES, G., PAGLIARI, C. L., et al. "A geometric space-view redundancy descriptor for light fields: Predicting the compression potential of the JPEG Pleno light field datasets". In: *2017 IEEE 19th International Workshop on Multimedia Signal Processing (MMSP)*, pp. 1–6, Oct 2017. doi: 10.1109/MMSP.2017.8122292.

[24] JAIN, A. K. *Fundamentals of Digital Image Processing*. Upper Saddle River, NJ, USA, Prentice-Hall, Inc., 1989. ISBN: 0-13-336165-9.

[25] GOLOMB, S. "Run-length encodings (Corresp.)", *IEEE Transactions on Information Theory*, v. 12, n. 3, pp. 399–401, July 1966. ISSN: 0018-9448. doi: 10.1109/TIT.1966.1053907.

[26] ITU-T, ISO/IEC. "Information Technology – Generic Coding of moving Pictures and Associated Audio Information: Video". Recommendation H.262 and ISO/IEC 13818-2, 1995.

[27] SOLE, J., JOSHI, R., NGUYEN, N., et al. "Transform Coefficient Coding in HEVC", *IEEE Transactions on Circuits and Systems for Video Technology*, v. 22, n. 12, pp. 1765–1777, Dec 2012. ISSN: 1051-8215. doi: 10.1109/TCSVT.2012.2223055.

[28] LAM, E. Y., GOODMAN, J. W. "A mathematical analysis of the DCT coefficient distributions for images", *IEEE Transactions on Image Processing*, v. 9, n. 10, pp. 1661–1666, Oct 2000. ISSN: 1057-7149. doi: 10.1109/83.869177.

[29] BELL, T. C., CLEARY, J. G., WITTEN, I. H. *Text Compression*. Upper Saddle River, NJ, USA, Prentice-Hall, Inc., 1990. ISBN: 0-13-911991-4.

[30] WITTEN, I. H., NEAL, R. M., CLEARY, J. G. "Arithmetic Coding for Data Compression", *Commun. ACM*, v. 30, n. 6, pp. 520–540, jun. 1987. ISSN: 0001-0782. doi: 10.1145/214762.214771. Available at: <http://doi.acm.org/10.1145/214762.214771>.

[31] WANG, Z., SIMONCELLI, E. P., BOVIK, A. C. "Multiscale structural similarity for image quality assessment". In: *The 37th Asilomar Conference on Signals, Systems Computers, 2003*, v. 2, Nov 2003. doi: 10.1109/ACSSC.2003.1292216.

[32] TU-R BT.709-6. "Parameter values for the HDTV standards for production and international programme exchange". June 2015.