

Light Field Streaming Compression Algorithms Based On Fourier Disparity Layer Transmission Using Hybrid Tensor Decomposition via Sketching

An M.Tech project report

submitted by

HAYAT ALI

in partial fulfilment of the requirements

for the award of the degree of

MASTER OF TECHNOLOGY

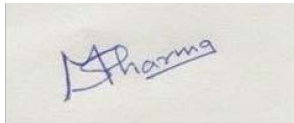


**DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS.**

JUNE 2021

THESIS CERTIFICATE

This is to certify that the thesis entitled **Light Field Streaming Compression Algorithms Based On Fourier Disparity Layer Transmission Using Hybrid Tensor Decomposition via Sketching** , submitted by **Hayat Ali (EE19M019)**, to the Indian Institute of Technology Madras, in partial fulfillment of the requirements for the award of the degree of **Master of Technology**, is a bona fide record of the research work done by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.



Dr. Mansi Sharma
Research Guide
INSPIRE FACULTY
Dept. of Electrical Engineering
IIT-Madras, 600 036

Place: Chennai

Date: 1st June 2021

ACKNOWLEDGEMENTS

First of all, I owe my debt of gratitude to the Indian Institute of Technology Madras for giving me the opportunity to come here and work with some of the brightest minds in the world. I am extremely grateful to my guide, Dr. Mansi Sharma without whose constant support and guidance, this work would not have been possible. I am also grateful to her for her incredible patience and for granting me the opportunity to learn multiple things all of which have been instrumental in shaping my research interest

ABSTRACT

KEYWORDS: Fourier Disparity Layer; Tucker TS; Hybrid Tensor Decomposition;

Light field imaging has emerged as a very promising technology in the field of computational photography. Many acquisition device have been recently designed to capture LF, from array of cameras to single camera mounted on moving gantries and plenoptic cameras.

Compared to classical 2D imaging, light field capture the intensity values of light rays in the form of large volumes of data retaining both spatial and angular information of a scene, which enables a variety of post capturing processing like re-focusing, extended focus, different view point rendering and depth estimation from a single exposure

However given large volumes of data of high dimensionality, the design of efficient compression scheme of light field is a key challenge for practical use of technology. The thesis proposed novel scheme that helps to compress streaming light field data with low rank approximation using Hybrid Tucker TS via sketching based on Fourier Disparity layer. Streamed LF views are approximated via Hybrid Tucker TS and then approximated views are encoded using HEVC with different layer configurations. Proposed scheme shows significant gain in bitrates and PSNR compared to Dib et al. and HEVC for some rank and K values.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	i
ABSTRACT	ii
LIST OF TABLES	v
LIST OF FIGURES	vi
ABBREVIATIONS	vii
1 INTRODUCTION	1
1.1 Introduction	1
1.2 Background and Motivation	2
1.3 Thesis Organization	3
2 RELATED WORK	4
2.1 Pseudo Sequence Based Image Coding	4
2.2 Randomize Hierarchical MV-HEVC	5
2.3 HEVC Based LF Coding with Bi-SS Compensation	6
2.4 Compression using Homography Based low Rank approximation . .	7
3 PROPOSED SCHEME	8
3.1 TENSOR	8
3.1.1 What is Tensor	8
3.1.2 Why Tensor?	8
3.1.3 Tensor Terminology	9
3.1.4 TENSOR DECOMPOSITION	9
3.2 Matrix Decomposition	11
3.2.1 Singular Value Decomposition	11
3.3 Karhunen-Loeve Tranform	11
3.4 Proposed Scheme	12

3.5	Light Field Compression based on FDL using Hybrid Tensor decomposition	15
3.5.1	Layer configuration	15
3.5.2	Encoding and Decoding using FDL	16
3.6	Experiments Results	16
3.6.1	Data set & Implementation Details	16
3.6.2	Results & Comparative Analysis	17
4	CONCLUSION AND FUTURE WORK	26
4.1	Conclusion	26
4.2	Future Work	26

LIST OF TABLES

3.1	The total number of bytes written for each subset of the Circular 2 scheme using our proposed scheme, HEVC and scheme by Dib et al.	19
3.2	The total number of bytes written for each subset of the Hierarchical 2 scheme using our proposed scheme, HEVC and scheme by Dib et al.	19
3.3	The total number of bytes written for each subset of the Circular 4 scheme using our proposed scheme, HEVC and scheme by Dib et al.	20
3.4	The total number of bytes written for each subset of the Circular 4 scheme using our proposed scheme, HEVC and scheme by Dib et al.	20
3.5	The total number of bytes written for each subset of the Hierarchical 4 scheme using our proposed scheme, HEVC and scheme by Dib et al.	21
3.6	The total number of bytes written for each subset of the Hierarchical 4 scheme using our proposed scheme, HEVC and scheme by Dib et al.	21
3.7	Bjontegaard percentage bitrate savings for the proposed compression scheme with respect to Dib et al. on <i>Bikes</i> data. (positive values represent gain)	23
3.8	Bjontegaard DSNR savings for the proposed compression scheme with respect to Dib et al. on <i>Bikes</i> data. (negative values represent gain)	24
3.9	Bjontegaard percentage bitrate savings for the proposed compression scheme with respect to HEVC on <i>Bikes</i> data. (positive values represent gain)	24
3.10	Bjontegaard DSNR savings for the proposed compression scheme with respect to HEVC on <i>Bikes</i> data. (negative values represent gain)	25

LIST OF FIGURES

2.1	Coding order and prediction structure with colour indicating its layer	5
2.2	Workflow of RH MV-HEVC	5
2.3	Random encoding scheme within MV-HEVC architecture syntax and adaptive hierarchical prediction structure	6
3.1	Visualization of different types of tensor	9
3.2	Different modes of Tensor	10
3.3	CP decomposition	10
3.4	Tucker decomposition	11
3.5	Overview block diagram of proposed scheme encoding	14
3.6	Overview block diagram of proposed scheme decoding	14
3.7	4 different layer configuration	15
3.8	Bitrate vs PSNR graphs for Circular 2 and Hierarchical 2 layers. . .	22
3.9	Bitrate vs PSNR graphs for Circular 4	22
3.10	Bitrate vs PSNR graphs for Hierarchical 4	23

ABBREVIATIONS

IITM	Indian Institute of Technology, Madras
TS	Tensor Sketch
QP	Quantization Parameter
KLT	Karhunen-Loève Transform
SVD	Singular Value Decomposition
HEVC	High Efficiency Video Coding
LF	Light Field
FDL	Fourier Disparity Layer
PSNR	Peak Signal to Noise Ratio
DSNR	Difference in Signal to Noise Ratio
C2	Circular 2
C4	Circular 4
H2	Hierarchical 2
H4	Hierarchical 4

CHAPTER 1

INTRODUCTION

1.1 Introduction

From past few years light field imaging is gaining popularity for variety of vision application, due to the emergence of light field capturing devices and commercially available cameras. In 2012 an American company “Lytro Inc.” launched its first generation Light filed camera in 8GB and 16 GB version later on after two years second generation camera was launched for commercial and experimental purpose.

Now the things which give light field camera an edge over other camera are its features like variable depth of field or refocusing because we collect a lot of information about object and using software manipulation we can alter focus, speed because there is less need to focus the lens before taking a picture, a light field camera can capture images more quickly than conventional point-and-shoot digital camera, low light sensitivity, the ability of light field to adjust focus post processing allows the use of larger apertures compared to the one feasible for conventional camera thus it helps in photographing even in low light environments, 3D images to record depth information we use plenoptic camera so stereo images can be constructed in software from a single plenoptic image capture. All these features of light field has attracted attention of scientist and researchers towards the use, application and analysis of light field data. Light field technology being technology for future also has some shortcomings. On the one hand, this higher-dimensional representation of visual data offers powerful capabilities for scene understanding, and substantially improves the performance of traditional computer vision problems such as depth sensing, post-capture refocusing, segmentation, video stabilization, material classification, etc. On the other hand, the high-dimensionality of light fields also brings up new challenges in terms of data capture, data compression, content editing and display. With regard to compression, a light field involves a large amount of data, but also records a scene with a set of images from different viewpoints, thus exhibiting data redundancy in both the spatial and angular

dimensions examples of this redundancy are the smooth regions in each sub-aperture image and light field subview. Therefore, the compression approaches for light fields offer much opportunity and are essential for light field storage, transmission and display. For compression we have lossy and lossless compression techniques. Lossy compression, transform coding approach typically rely on discrete cosine transform (DCT) or the discrete wavelet transform (DWT), to compress a light field. We use classical coding schemes like JPEG 2000(using DWT) or JPEG (using DCT) to compress light field raw images. But these two schemes are not specifically designed to be used for raw light field images, thus we do not get optimal compression results. In this thesis our work is focused on efficient compression of light field data. In past several efforts are done to improve the efficiency of light field data compression. Aggoun (6) proposed a 3D - DCT to exploit the spatial redundancy within the light field views. They first selected some subview and their neighbouring views and arranged them into a 3 D bricks, then they applied 3D-DCT to get the decorrelated group of subviews. Magnor et al. (7) presented 4D-DWT to directly compress the sub-aperture images without any arrangement. Some predictive coding approaches are also developed in which first a set of images from the light field array to be coded as *intra* also known as I-images. Then these I-images are used as a reference to code the remaining light field images also called P-images. Magnor and Girod (8) separated each sub-aperture image into blocks and predicted blocks of P-images using blocked disparity and blocks in the I-images. Same way Conti et al. (13) used blocked disparity to code 3D light field.

In this thesis we are presenting work for lossy light field compression based on Fourier disparity layer transmission using hybrid tensor decomposition via sketching.

1.2 Background and Motivation

The most obvious challenge in light field photography is immense data, it becomes hard to manage efficiently such large dimension data. The two additional dimension compared to traditional image technology has raise visual data information from two to four order of magnitude and as a result the scalability of light field system are challenge by capturing, rendering and displaying this vast amount of rays. Besides the price, weight and physical size of the many computing components required with their own

resource footprint regarding power requirements, heat generation, bandwidth usage and storage space.

Compression of light field data is a hot research topic and would help reducing the bandwidth and storage requirements of light field systems. There are some existing method like 2D video compression methods are efficient and can be used to compress but they does not exploit the fact the neighbouring views of light field data are correlated.

Many solution which are proposed so far adapt standardized image and video compression technique for encoding light field data. Jiang et al. (11) and Dib et al. (12) investigated the application of homography based low rank approximation for light field compression and super ray light field compression for reducing the angular displacement, while Verhack et al. (2) used local Gaussian mixture model in the 4D ray space are considered, while depth based segmentation of light field into 4D spatio-angular blocks is used in Tabus et al. (14) for prediction and the prediction residue is encoded using JPEG 2000.

In this thesis a novel lossy compression algorithm is described which uses Fourier disparity layer transmission using Hybrid Tensor Decomposition via Sketching.

1.3 Thesis Organization

The thesis is organised in the order in which research is carried out.

- In Chapter 1, we wrote about light field data problems, background and motivation. In this chapter we also summarize the work done related to light field compression.
- In Chapter 2, we have discussed in brief about some of the work done in the field of light field compression in the past.
- In Chapter 3, we have described our proposed algorithm Light Field Streaming Compression Algorithm Based On Fourier Disparity Layer using Hybrid Tensor Decomposition via sketching in detail, and also done comparative analysis of proposed scheme bitrate and PSNR using Bjontegaard metric.
- In Chapter 4, we have concluded our work by briefly discussing our scheme and its contribution in research and we have also discussed about the future work that could be possible using our research.

CHAPTER 2

RELATED WORK

2.1 Pseudo Sequence Based Image Coding

In pseudo sequence based image coding all the views are organized as a pseudo sequence like video, one view is coded first (I-frame) and the other views are coded by referring to the reconstruction of already coded views (P or B-frames). The raw light data is encoded as pseudo sequence of images exploiting the fact of higher correlation among views. Correlation between the adjacent views both horizontally and vertically is more, hence inter-view prediction from adjacent view is used. Moreover higher similarity is observed between views around centre compared with between the views near the border, so the centre view rather than border view are used for prediction structure of views. Centre view is used as an I-frame, the remaining views are compared as P and B frames in a 2D hierarchical structure. Each view is assigned a layer, views at higher level layer are encoded after views at lower level layer and thus can be predicted from the latter. For each view four reference are chosen one at top, one at bottom, one at right and one at left direction, in all four direction the view at nearest distance is chosen.

For rate allocation, a general guideline which is followed in video coding is that the frames that are used as reference must have higher quality than the frames not used for reference due to error propagation in prediction. Therefore a higher quantization parameter QP value is given to views of layer of higher level and vice versa. Therefore the I-frames which are used as reference for most number of other views has lowest QP values. Pseudo sequence based scheme often outperforms image based scheme but there are certain exception and there is necessity of rate allocation among views for better compression.

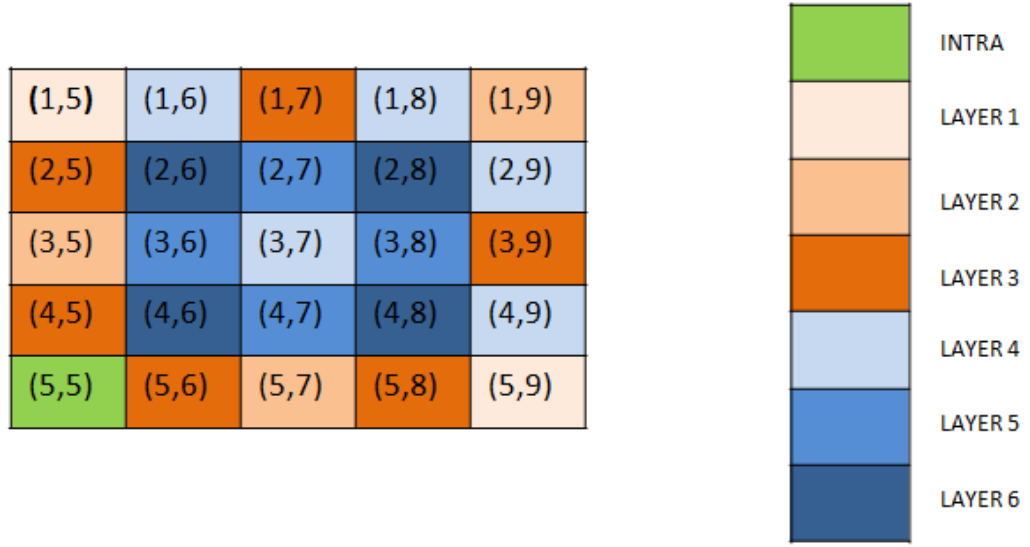


Figure 2.1: Coding order and prediction structure with colour indicating its layer

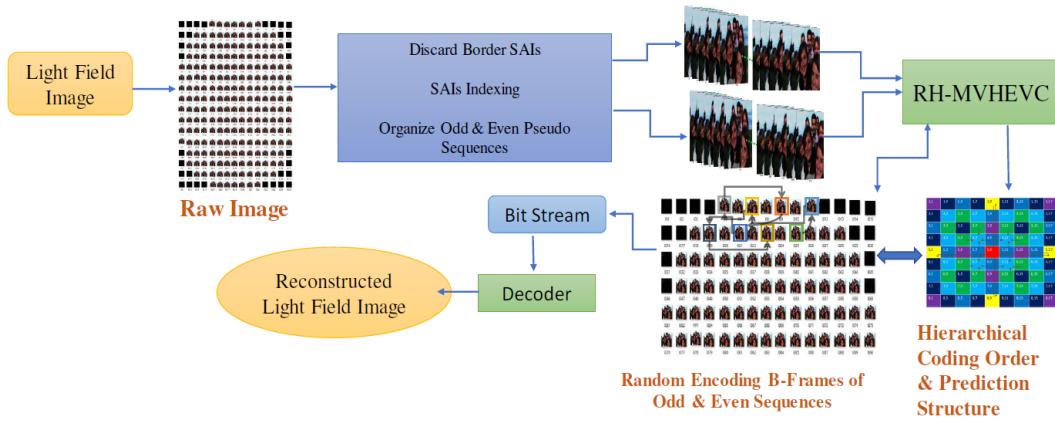


Figure 2.2: Workflow of RH MV-HEVC

2.2 Randomize Hierarchical MV-HEVC

This is a Randomized Hierarchical Extension of multi view HEVC for improved compression of light field data. Main features of this technique was it exploit the temporal, inter view and non linear redundancy of adjacent sub aperture images. This RH-MVHEVC takes advantage of random encoding in Hierarchical prediction. This scheme integrates the advantage of both hierarchical prediction and random encoding techniques. But this technique lags behind HEVC based encoder which is severely constrained by 1-D coding structure. But this scheme takes benefits of random encoding and guarantee to maintain the quality for all reconstructed sub aperture images.

The workflow for RH MV-HEVC is shown in figure2.2 as given in (10). First light

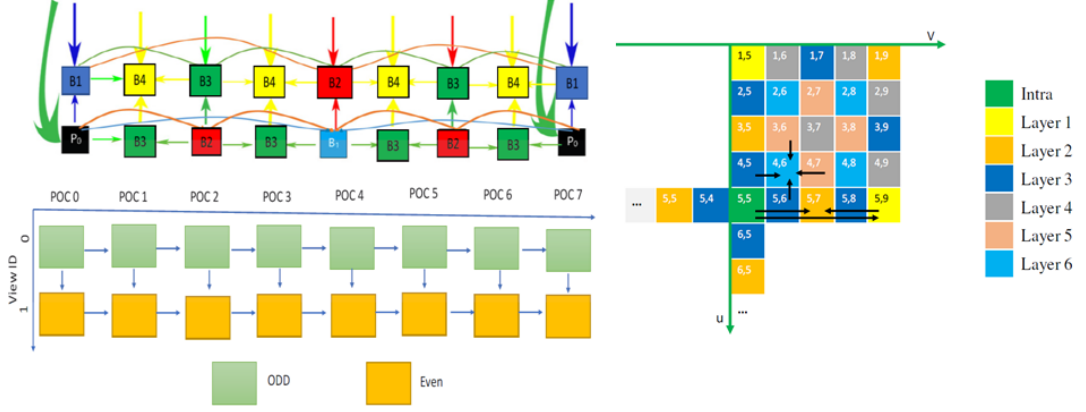


Figure 2.3: Random encoding scheme within MV-HEVC architecture syntax and adaptive hierarchical prediction structure

field data is decomposed into grid 15x15 SAIs later some SAIs from border are discarded and only 13x13 are selected. The extracted SAIs are indexed in ascending order starting from top left to bottom right. To best avail the temporal and inter-view redundancy SAIs images are partitioned into two sequence odd and even indices. These pseudo sequence is given as an input to MV-HEVC encode and enables a random coding structure within features and characteristic of MV-HEVC. Random encoding scheme used for encoding B-frame is shown in figure2.3. Furthermore a hierarchical prediction structure is employed to carefully manage the relationship between adjacent SAIs in horizontal as well as vertical direction by assigning P-frames and B-frames to layers of two input pseudo sequence as shown in figure 2.3. Views in odd and even pseudo sequence are compressed as P and B frames while views at centre is compressed I-frames. Each view is associated with a layer as shown in figure2.3. The view at higher level layer are coded preceding to the views at lower level layer. Each view chooses at most four reference frames. Specifically nearest neighbour frames that must be at lower level layers are chosen to predict the middle current frames, as shown in figure 2.3

2.3 HEVC Based LF Coding with Bi-SS Compensation

The proposed Light Field coding make use of the self similarity compensated prediction concept to efficiently explore the inherent correlation of this type of content. To further improve the coding performance, a bi-predicted SS estimation and SS compensation is proposed. Regarding light field coding approaches, High Efficiency Video Coding

(HEVC) provides significant gain as compared to image coding technologies like JPEG 2000 and JPEG standards. Previous work which has been done shows that there is possibility of further improvement to exploit the inherent correlation. SS estimation uses block based matching between previously coded and current frames as similar to motion compensation. And hence selected block becomes the predictor candidate and the relative position between the candidate predictor and current block is define as SS vector. To further improve the SS compensation technique, A Bi predicted SS estimation and SS compensation technique is developed in which the prediction candidate is defined as a linear combination of two blocks it the same vicinity of SS reference.

2.4 Compression using Homography Based low Rank approximation

In this scheme Low rank approximation exploits scene and data geometry. In this scheme first a reference view is selected, preferably centre view is selected as reference, then a homography projection is searched for all other views in order to obtain best low rank approximation for a given target rank k , where k is less than number of views, later those views are align as a column vector in a matrix. The k matrix is represented as a product of two matrices B and C where B is matrix which contains k basis vector an C contains weighting coefficient. Low rank optimization problem is then formulated as.

$$\operatorname{argmin}_{h,B,C} = \|Ioh - BC\|_F^2$$

Where $\|\cdot\|$ is Frobenius norm , $B \in \mathbb{R}^{m \times k}$ and $C \in \mathbb{R}^{k \times n}$ $k < n$ where n are total views and m is total pixels per views and $Ioh = [vec(I_1oh_1); vec(I_2oh_2); \dots; vec(I_noh_n)]$. B can be found by, first find SVD of $Ioh = U \sum V^T$, then B is first k column vectors of U and C is k rows of V^T . Where K is rank.

After low rank approximation views are coded with HEVC-intra along with coefficient and homography parameter. This propose scheme is also extended for images of multi layer depth, for such case homography for each layer is calculated for a view.

CHAPTER 3

PROPOSED SCHEME

3.1 TENSOR

3.1.1 What is Tensor

Tensor can be thought of as mathematical objects that is used to define physical properties same as scalar and vector. Even we can state that tensor are merely a generalisation of scalars and vectors. A zero rank tensor is called scalar and first rank tensor is called vector.

Rank of tensor : We can define rank of tensor by number of dimension needed to define a tensor. For ex. a 3×1 vector containing three elements needs only one direction to define tensor and is called one rank tensor, and a 3×3 matrix which contains 9 elements requires 2 direction to define and is called two rank tensor. Simply stating a N-way array or multi dimension array is called Tensor. Tensor of order of three or more are generally called high order Tensor. Figure 3.1 shows a visualization of different types low order and high order tensors.

3.1.2 Why Tensor?

First Tensor gives a way to represents an image and video in a easy way. An RGB image is represented by a combination of three matrix stacked together to produce a RGB image or an RGB image is a tensor of three order in same way a video can be represented by a tensor of four order. In medical imaging the multi modality images of patients captured under different conditions produce high order tensor. Second large number of vectors and matrices stacked together in tensor can be used for efficient data analysis and compression through tensor decomposition.

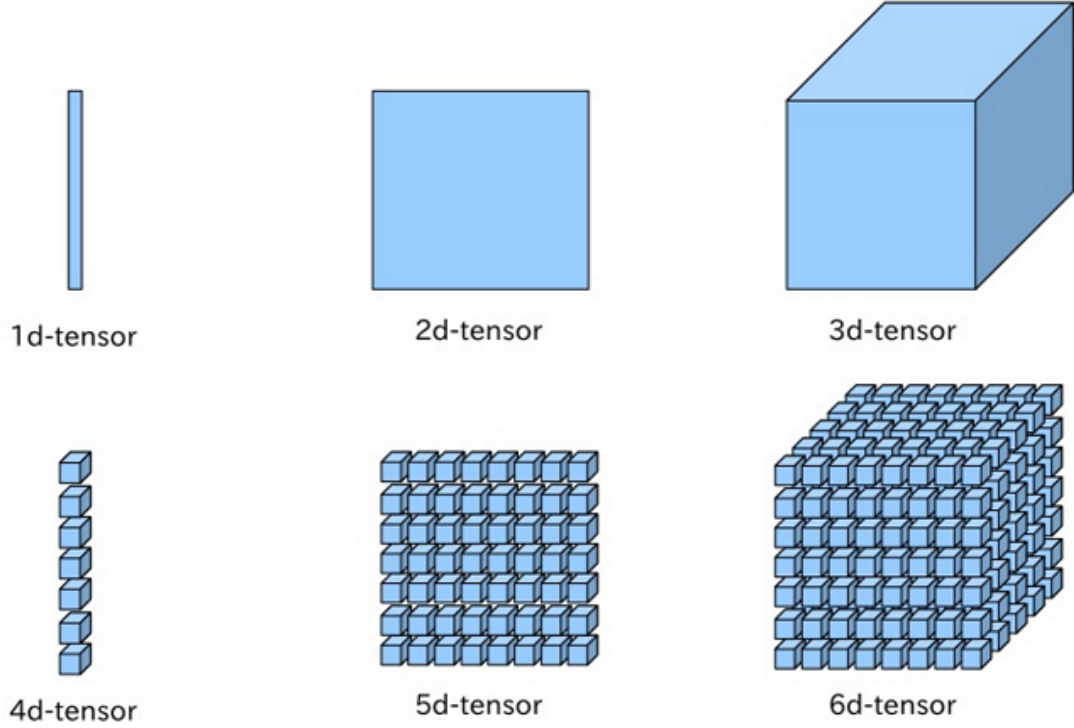


Figure 3.1: Visualization of different types of tensor

3.1.3 Tensor Terminology

Let say we have a Tensor \mathcal{A} which is of order K ie. $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_K}$, we can access elements of Tensor using K indices as in $a_{i_1 i_2 \dots i_K}$. Sub-array of tensor can be accessed by keeping of subset of indices fixed. The term *fiber* is used to refer a sub array of tensor with all but one fixed indices which is same as row or column in matrix. If we talk about three mode tensor, we can have three different fibers. We name them as column (mode-1), row (mode-2) and tube (mode-3) fibre as shown in Figure 3.2 . Another term that is generally encountered with tensor is *unfolding* or *flattening*. It just means to arrange a tensor as a matrix or we can say unfolding a tensor in form of matrix. If we unfold a tensor along its one of the modes, say mode- m , then it is to be said we are doing mode- m flattening. It is simply formed by arranging the mode- m fibres as its column.

3.1.4 TENSOR DECOMPOSITION

As for a matrix we can decompose it in several ways like Singular Value Decomposition (SVD) and Principle Component Analysis (PCA) to reduce its dimensionality .Similarly we have several ways to decompose a tensor

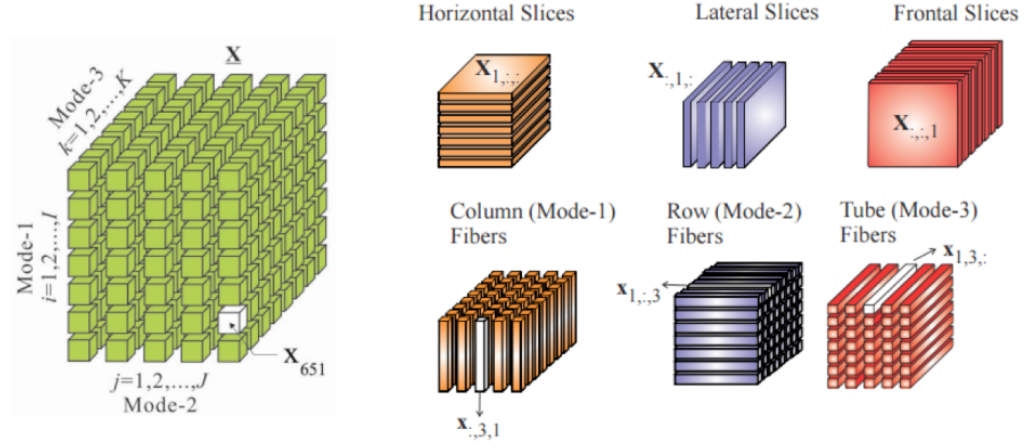


Figure 3.2: Different modes of Tensor

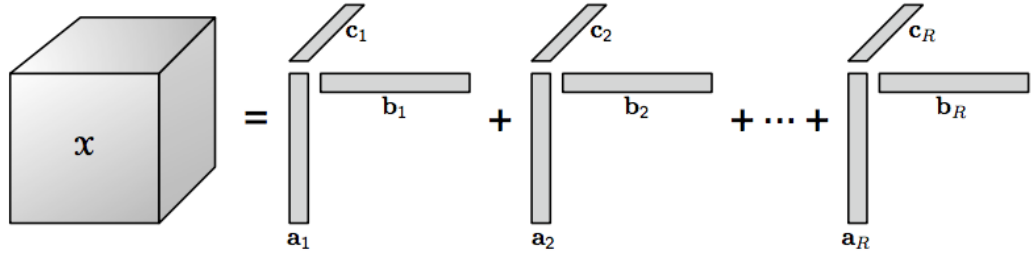


Figure 3.3: CP decomposition

CP decomposition : The CP decomposition factorizes a tensor into linear combination of rank one tensor. For ex. a tensor of order 3 can be decomposed as a sum of K one rank tensor

$\mathcal{X} = \sum_{n=1}^K a^n \circ b^n \circ c^n$ where 'o' represents outer product. It is illustrated in Figure 3.3. This decomposition is performed by a minimization algorithm known as Alternating least square.

Tucker decomposition : In Tucker decomposition we express a tensor in terms of a *core tensor* and n-mode products through *factor* matrices. Thus for a given tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ its Tucker decomposition is given as

$$\mathcal{X} = \mathcal{G} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} = \sum_{p=1}^P \sum_{q=1}^Q \sum_{r=1}^R g_{pqr} a^p \circ b^q \circ c^r$$

where \mathcal{G} is a core tensor of size $P \times Q \times R$ and factor matrices \mathbf{A} , \mathbf{B} and \mathbf{C} are of size $I \times P$, $J \times Q$ and $K \times R$ respectively. The column vectors of factor matrices span the corresponding mode space. The factor matrices are usually orthogonal and represent the principal component in each mode, and the elements of the core tensor represent the interaction of different modes. The rank for mode generally is kept low as compared to

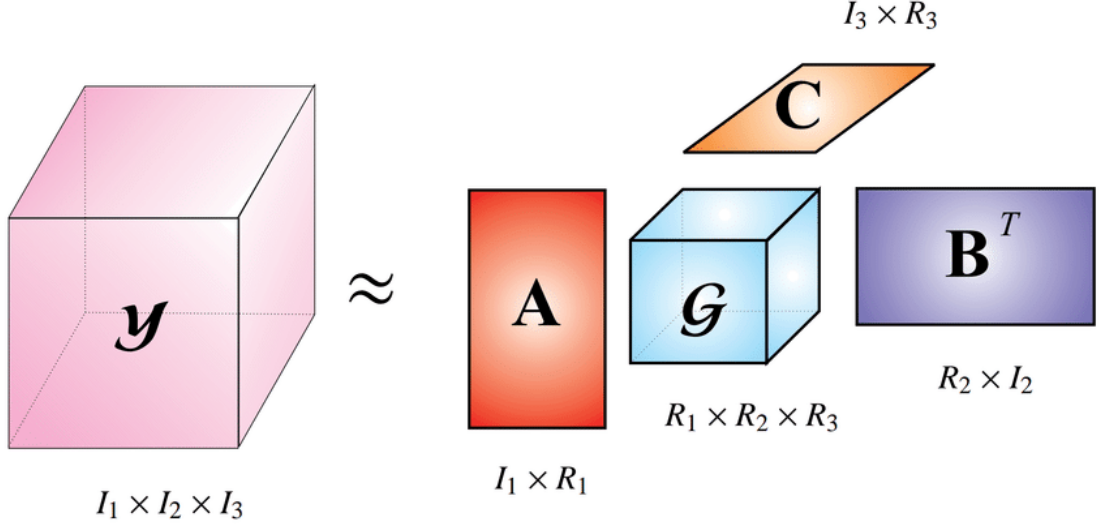


Figure 3.4: Tucker decomposition

mode rank of original tensor. Thus the core tensor is viewed as compressed representation of the original tensor. Pictorial representation of tucker decomposition is shown in figure 3.4

3.2 Matrix Decomposition

3.2.1 Singular Value Decomposition

Singular Value Decomposition (SVD) is a one of the matrix decomposition technique by which a matrix is decomposed into three sub-matrices namely $\mathbf{U}, \mathbf{S}, \mathbf{V}$ where \mathbf{U} is the left eigen vectors, \mathbf{S} is the diagonal matrix containing singular values and \mathbf{V} is called the right eigen vector. SVD can be used for low rank approximation, for this we truncate some of the largest values from the singular matrix and other singular values are replaced by zeros and using those largest singular values we reconstruct the approximated matrix known as Eckart-Young theorem.

3.3 Karhunen-Loeve Transform

The KL Transform is also known as the Hotelling transform or the Eigen Vector transform. The KL Transform is based on the statistical properties of the image and has several importance properties that makes it useful for image processing particularly for

image compression. The main purpose of image compression is to store the image in fewer bits as compared to original image, the neighbouring pixels data in an image is highly correlated, an excellent image compression can be achieved by de-correlating the data. Here comes the KL Transform which does the work of de-correlating the data thus facilitates higher degree of compression.

For KLT first we divide our zero mean image into sub blocks, then we find co-variance matrix for each sub blocks, after that eigen vectors and eigen values of co-variance matrix is found. Then we create a transform matrix whose column are eigen vectors of co-variance matrix, first column of transform matrix is eigen vector corresponds to largest eigen value. Due to the idea of using eigen vector corresponds to largest eigen value, this is also known as principle components transform.

3.4 Proposed Scheme

KLT is considered to be optimal transform if we use a single transform for all sub blocks of an image. But if we talk about Singular Value Decomposition (SVD) its use in image compression is motivated by its energy compaction property . SVD is known to be the deterministically optimal separable transform for energy compaction. Suppose an image is divided in blocks of $N \times N$, and if we use k_1 singular values and $2k_1$ vectors will produce the optimal least square approximation using separable basis function in k_1 components of this block. For KLT the coefficient and $2k_1$ vectors if used to approximate the same block will produce an optimal approximation in terms of mean square given that KLT basis function are obtained from vertical and horizontal co variance matrices of image.

Here in this proposed scheme we have combine SVD-KLT as described in (18).

In this algorithm for computing Tucker Decomposition of a tensor which incorporates random sketching. A key challenge of incorporating sketching in Tucker decomposition is Kronecker products of factor matrices. This makes them too large to store in RAM and process. Earlier work has led to a new technique called Tensor Sketch which is suited for sketching Kronecker product. This algorithm can handle streamed data ie. a tensor can be decomposed even when one element of tensor is revealed at a time and then discarded, no matter in what way it is done. In application of scientific compression

of high fidelity simulation the data tensor is of high rank. Thus an algorithm which is one pass can handle such large data without the need of storing whole data at once in RAM. This algorithm is intended for low rank decomposition.

There are multiple tensor decomposition technique as discussed in section 3.1.4. Tucker decomposition is considered here. A Tucker decomposition of a tensor $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$ The core tensor and factor matrices initially, are initialized randomly with each element independent and identically distributed Uniformly between -1 and 1. The factor matrices are subsequently orthogonalized. Tensor Sketch operator is defined of appropriate size. Symbol " \otimes " denotes Kronecker product and mode-n matricization of a tensor $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$ is denoted by $X_{(n)} \in \mathbb{R}^{I_n \times \prod_{i \neq n} I_i}$. Count sketch operator can be defined as $S : \mathbb{R}^I \rightarrow \mathbb{R}^J$ is as $\mathbf{S} = \mathbf{P}\mathbf{D}$, where $\mathbf{P} \in \mathbb{R}^{J \times I}$ is a matrix with $p_{h(i),i} = 1$ and all other entries equals 0. $h : [I] \rightarrow [J]$ is a random map such that $(\forall i \in [I]) (\forall j \in [J]) (\mathbb{P}(h(i) = j) = 1/J)$ and $D \in \mathbb{R}^{I \times I}$ is a diagonal matrix, with each diagonal entry equal to +1 and -1 with equal probability.

Then each operator $T^{(n)}$, for $n \in [N]$, is defined as in (15) but based on $S_1^{(n)}$ $n \in [N]$ and with the nth term excluded in the Kronecker and Khatri-Rao products. $T^{(N+1)}$ is defined similarly, but based on $S_2^{(n)}$ $n \in [N]$ without excluding any term in the Kronecker and Khatri-Rao products. Two different sketch dimension J1 and J2 are used for $S_1^{(n)}$ and $S_2^{(n)}$ respectively where $J2 > J1$.

In this proposed algorithm Hybrid Tucker Ts via sketching, we have taken an RGB image of light field data as a 3 way tensor as an input \mathcal{Y} , for decomposition we have chosen certain Rank values in multiples of five ie. 5,10,15,20 and 25 to check for both relative low and high rank. We have named these rank value as target rank, which is also dimension of core tensor, ie for target rank 5, target rank vector would be [5,5,3] and our core tensor will also be of size [5,5,3] of rank 5, same way our rank value 10, 15, 20 and 25 will have target rank vector, third element of target rank we have taken as three because our input image has three matrix R,G and B, and the target rank dimension has to be less than or equal to input dimension.

So along with rank we have taken different values of K in multiple of five 5,10,15,20 and 25. Hence we have different combination of Rank and K value to decompose an image. With different values of Rank and K values we have core tensor \mathcal{G} and factor matrices $A^{(1)}, A^{(2)}, \dots, A^{(N)}$ after decomposition. In next step we have hybridized the

factor matrices.

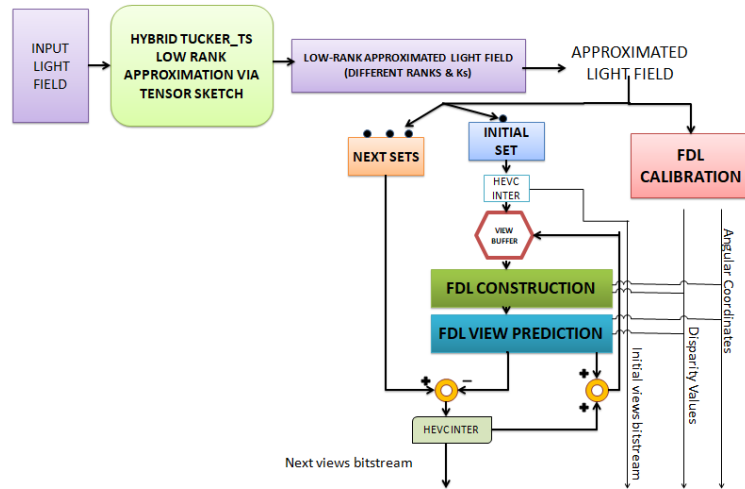


Figure 3.5: Overview block diagram of proposed scheme encoding

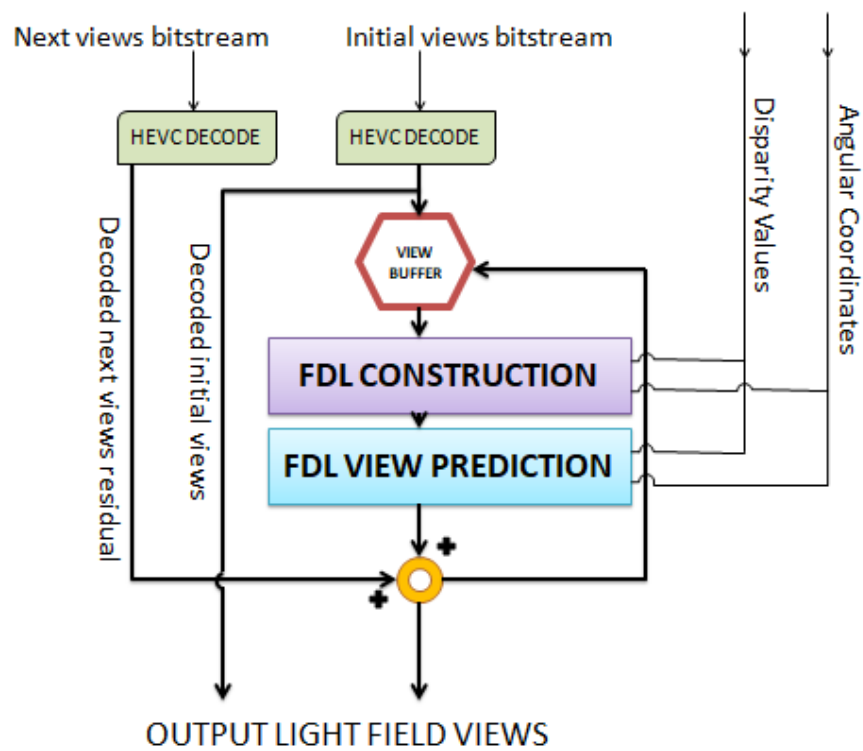


Figure 3.6: Overview block diagram of proposed scheme decoding

3.5 Light Field Compression based on FDL using Hybrid Tensor decomposition

Data-set which is used as an input is grid of 9×9 light field data images. In first step the input light field data is streamed into the block Hybrid Tucker-Ts low rank approximation as shown in figure 3.5. Decomposition and low rank approximation is done. This part is mainly for reducing redundancy in input light field data and finally we have approximated light field data with different values of rank and K as mentioned in section 3.4 .

3.5.1 Layer configuration

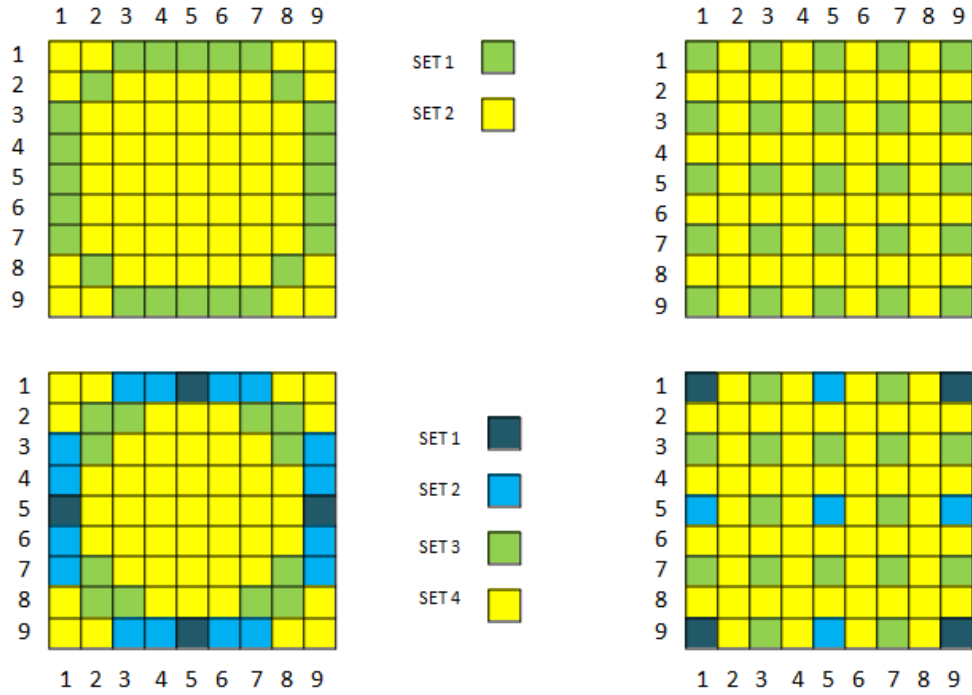


Figure 3.7: 4 different layer configuration

After approximating input light field input images, those approximated images is divided into different sets, Circular-2, Circular-4, Hierarchical-2 and Hierarchical-4 as shown in figure 3.7 where different colour represent group of images to be put in different sets. Shown layer configuration is for 9×9 data set images. Note that in deciding subset circle formation is preferred not a square in order to avoid use of corner view to

predict inner view, as in real light field data captured by plenoptic cameras corner views are of low quality than inner views.

3.5.2 Encoding and Decoding using FDL

4-D representation of light field describing radiance along ray by a function $f(u, v, x, y)$. This representation is based on parameterization of light ray with two parallel plane with (x, y) and (u, v) representing spatial and angular co-ordinates of light rays. As shown in figure 3.5 first Fourier disparity layer calibration described in (1) determine set of disparity value and angular co-ordinates, which are used for Fourier disparity layer construction and view prediction step, also these values are transferred as a meta data to decoder. After partitioning the data into different subset, first set is directly encoded by HEVC as a group of picture with 8-Bit with inter coding. The first set is used to construct Fourier disparity layer representation, which allows to predict view of subsequent subset. After its prediction residue is encoded using HEVC with 10 bit in order to avoid precision loss, decoded and added to predicted view to construct the corresponding view. Then both set are used to construct FDL and the next set is predicted, this iteration goes on until all the sets are coded. In order to optimize the bit-rate allocation, we use different Quality Parameter values in HEVC coding. If QP_t is the QP value for initial set the for t set index $QP_t = QP_{t-1} + 1$ is used.

3.6 Experiments Results

3.6.1 Data set & Implementation Details

The proposed coding scheme is implemented on a single high-end HP OMEN 342 X 15-DG0018TX Gaming laptop with 9th Gen i7-9750H, 16 GB RAM, RTX 2080 8 GB Graphics, and Windows 10 operating system. Proposed scheme performance was evaluated using real light field data captured by plenoptic cameras. We have used *Bikes* light field images from the EPFL Lighfield JPEG Pleno database (17). The raw plenoptic images were extracted into sub aperture images of 15 X 15 using Matlab Light field toolbox (3), each image has resolution of 430 X 620 pixels. The border sub-aperture images suffer from geometric distortion and blurring, have no use in recovering

light field so for a fair compression we have considered only central 9 X 9 views for our experiment. In Dib et al. (5) and HEVC , only central 9 X 9 views are considered.

Proposed scheme first decompose each image into core tensor and factor matrices via random sketching, then factor matrices are hybridized, from these core tensor and hybridized factor matrices an approximated image is reconstructed. We have used different combination of rank and sketch dimension for decomposition and reconstruction of approximated image in multiples of five (5,10,15,20 and 25). We have tested compression for our scheme with six quantization parameter values. (QPs = 2,6,10,14, 20 and 26). These QP values corresponds to first set.

In Dib et al. (5) part, light field data is compressed without low rank approximation, we have chosen six quantization parameters, QP = 2,6,10,14,20 and 26 for compression, corresponds to first set.

In Direct HEVC , encoding is done using 8 bits and YUV444 color space file is used to compress for each subset using HEVC-inter, in particular the HM 16.10. We have chosen six quantization parameters, QP = 2,6,10,14,20 and 26 for experiment.

3.6.2 Results & Comparative Analysis

The performance of our proposed scheme has been compared with Dib et al. (5) and direct HEVC (9) encoding. I have subjected all the mentioned schemes to same test conditions and quantization parameter, (QP = 2,6,10,14,20 and 26). I have experimented with *Bikes* data using four different layer configuration, Circular 2, Circular 4, Hierarchical 2 and Hierarchical 4. Each layer configuration is run for different combination of rank and sketch dimension values, for each combination of rank and sketch dimension I have run our scheme for six QPs values. Total number of bytes written to file for each layer configuration and different combination of rank , sketch dimension and QPs along with Dib et al. (5) and HEVC is shown in Table 3.1-3.6. The bit rate and PSNR graphs for each subset of layer configurations of our proposed scheme along with Dib et al. and HEVC are shown in Figure 3.8-3.10. Furthermore an objective assessment is performed Bjontegaard (4) metric. This metric can compare performance of two coding techniques. The average percentage difference in bit-rate change and PSNR is estimated over a range of six QP values. Comparison of percentage of bit rate savings for pro-

posed scheme with respect to Dib et al. for different layer configuration is summarized in Table 3.7. Comparison of PSNR gain for proposed scheme with respect to Dib et al. for different layer configuration is summarized in Table 3.8. Bjontegaard percentage bitrate savings for proposed scheme with respect to HEVC for different layer configuration is summarized in Table 3.9. Comparison of PSNR gain for proposed scheme with respect to HEVC for different layer configuration is summarized in Table 3.10.

Our proposed scheme out performed the HEVC and Dib et al.(5) for some rank and K value in terms of bitrate as shown in figure 3.8-3.10. Total number of bytes written to the file for each layer configuration is comparatively less than other schemes for some ranks and K value, as it is clearly evident from Table 3.1-3.6. In terms of Bjontegaard rate saving, our proposed scheme saves rate for some rank and K values . Same way if we look at the PSNR gain in differential PSNR with respect to Dib et al. then there is gain for some rank and K values. Similarly Bjontegaard rate saving of our proposed scheme with respect to HEVC is mentioned in Table 3.9

If we talk about the analysis with respect to HEVC then proposed scheme performs better more frequently, our proposed scheme on an average over different rank and k values saves bitrate for set2 C2, set 2 H2, set2, 3 and 4 for C4 and H4. as shown in table 3.9. If we see in terms of PSNR proposed scheme on an average provide gain in PSNR for set2 in C2 and H2, set3 and 4 in C4 and set2,3 and 4 in H4. as shown in table 3.10

Table 3.1: The total number of bytes written for each subset of the Circular 2 scheme using our proposed scheme, HEVC and scheme by Dib et al.

	QP = 2		QP = 6		QP = 10		QP = 14		QP = 20		QP = 26	
Scheme	Subset 1	Subset 2	Subset 1	Subset 2	Subset 1	Subset 2	Subset 1	Subset 2	Subset 1	Subset 2	Subset 1	Subset 2
HEVC	7725765	17886297	5645842	12969691	3721540	8447324	2168041	4873883	807340	1793197	228218	502167
Dib et al	5349708	10033607	5349708	6181713	3498472	3418623	2043542	1722241	760957	549257	216655	132715
Our Rank5 K5	8484411	15058973	6504274	10987196	4814181	7574338	3415201	4975744	1859310	2484743	826059	1010410
Our Rank5 K10	6757464	11216122	4918893	7841828	3514576	5264475	2438271	3417050	1320726	1716736	596513	706321
Our Rank5 K15	6102512	9654260	4350673	6654995	3074168	4428743	2117251	2856848	1144292	1419905	519488	585541
Our Rank5 K20	5731440	9114765	4050946	6278643	2850209	4178408	1958266	2700681	1061394	1349060	488101	561374
Our Rank5 K25	5575430	8397086	3927518	5746237	2764237	3801678	1903135	2446805	1029530	1217536	471386	512493
Our Rank10 K5	8305885	14674962	6350606	10728531	4709480	7399264	3349913	4836346	1832808	2373444	815473	930268
Our Rank10 K10	7492198	12585482	5611143	8988728	4095059	6088229	2877706	3916961	1559942	1880395	690390	722918
Our Rank10 K15	7134264	11942961	5284105	8495249	3829913	5743321	2680657	3692576	1451427	1778525	641672	685958
Our Rank10 K20	6942916	11677086	5126238	8300603	3718787	5615295	2607708	3621551	1414226	1753597	628262	684032
Our Rank10 K25	6780912	11287581	5015802	7995456	3639528	5382108	2558221	3445512	1396843	1644560	626613	629244
Our Rank15 K5	9053038	16231250	7073087	12114020	5352601	8479228	3868789	5557322	2129648	2655500	917896	969762
Our Rank15 K10	8392000	14729506	6444900	10837573	4811194	7504896	3442025	4881494	1875129	2310744	805442	848125
Our Rank15 K15	8204899	14631016	6277874	10725588	4683662	7393491	3356624	4783316	1841443	2250326	795642	820561
Our Rank15 K20	8193805	14490324	6276905	10654266	4688145	7386921	3366996	4818585	1857368	2289495	810482	859590
Our Rank15 K25	8184506	14426766	6278231	10518114	4682294	7227582	3350456	4675195	1832222	2200326	796815	813921
Our Rank20 K5	9743736	17915696	7751630	13682761	5966525	9793510	4378146	6527916	2431771	3133281	1025198	1123045
Our Rank20 K10	9129783	16489253	7160080	12405159	5435813	8761918	3949395	5772540	2178488	2717394	920555	960577
Our Rank20 K15	9163290	16498300	7194832	12356660	5478479	8678001	3981620	5692760	2207022	2671163	944788	947108
Our Rank20 K20	9173682	16382568	7210924	12243747	5494143	8569145	4006198	5596676	2230359	2626701	954291	924121
Our Rank20 K25	8882908	16348139	6928151	12249455	5246966	8600612	3800058	5626868	2091550	2628269	884097	927485
Our Rank25 K5	10064692	18899001	8080313	14599205	6267602	10542365	4631446	7040741	2589628	3317913	1079620	1132578
Our Rank25 K10	9790731	18218234	7808405	13964793	6020542	10018784	4423916	6673878	2463092	3149304	1030220	1091037
Our Rank25 K15	9715093	18102733	7742409	13860099	5966117	9918383	4384823	6578737	2449166	3087885	1030508	1057695
Our Rank25 K20	9385412	17776762	7403741	13558117	5655656	9684974	4128159	6422476	2285002	3012314	955728	1039288
Our Rank25 K25	9627573	17867121	7651253	13635062	5875672	9740309	4303367	6476652	2381568	3066338	993222	1067951

Table 3.2: The total number of bytes written for each subset of the Hierarchical 2 scheme using our proposed scheme, HEVC and scheme by Dib et al.

	QP = 2		QP = 6		QP = 10		QP = 14		QP = 20		QP = 26	
Scheme	Subset 1	Subset 2	Subset 1	Subset 2	Subset 1	Subset 2	Subset 1	Subset 2	Subset 1	Subset 2	Subset 1	Subset 2
HEVC	8015620	17568888	5877326	12691883	3874255	8262698	2272244	4777113	852829	1760507	249596	490331
Dib et al	7931704	9565827	5773077	5799099	3780108	3148351	2190108	1572613	810166	494546	232805	115399
Our Rank5 K5	8920538	14604779	6850520	10622163	5084016	7312692	3625365	4805448	1989098	2405491	894189	975689
Our Rank5 K10	7184018	10907436	5241041	7615046	3751462	5094836	2607997	3304764	1412786	1656136	642485	680114
Our Rank5 K15	6359840	9406459	4529918	6503582	3199426	4337984	2210017	2806095	1199093	1399890	551455	577405
Our Rank5 K20	5983685	8868890	4220744	6100396	2969625	4058586	2049394	2622153	1116021	1310171	518528	573727
Our Rank5 K25	5895915	8107219	4133318	5550462	2904470	3678166	1996689	2372412	1083853	1183140	504401	489424
Our Rank10 K5	8744209	14391182	6681133	10496882	4952700	7213874	3530616	4698657	1942552	2311346	869405	900627
Our Rank10 K10	7815227	12252668	5841907	8751290	4265315	5921811	3005996	3804926	1636706	1830283	732721	700436
Our Rank10 K15	7545843	11606965	5603042	8238930	4077643	5557892	2872821	3567715	1570165	1714507	708215	659171
Our Rank10 K20	7352497	11328409	5423700	8040078	3928237	5436686	2758849	3502170	1502645	1690068	674895	653852
Our Rank10 K25	7306234	11034753	5382154	7795404	3901595	5237785	2740620	3348913	1494255	1594968	666877	609645
Our Rank15 K5	9527371	15745638	7458529	11721439	5642420	8177866	4086897	5345139	2261873	2541760	986507	918958
Our Rank15 K10	8776418	14212515	6731056	10417539	5015736	7193245	3589313	4667410	1966710	2199555	855846	804167
Our Rank15 K15	8643337	14155805	6619625	10351922	4932439	7113393	3530795	4597461	1938006	2155842	846505	779737
Our Rank15 K20	8677908	13974204	6649774	10230150	4957828	7064304	3554850	4592156	1947051	2179653	845827	804341
Our Rank15 K25	8519057	13912387	6497992	10168471	4829447	6998289	3453546	4521517	1890830	2114902	827295	774750
Our Rank20 K5	10263562	17373047	8188442	13213580	6313121	9396355	4652692	6207346	2612091	2939633	1126941	1029359
Our Rank20 K10	9854238	16565580	7784810	12446243	5955793	8750878	4356871	5735763	2430283	2698409	1048081	947006
Our Rank20 K15	9506994	16110222	7445987	12064228	5654952	8455503	4114261	5527726	2278744	2576432	979136	902293
Our Rank20 K20	9616894	15825867	7555657	11798827	5743225	8237127	4177570	5359379	2306417	2491654	986573	869433
Our Rank20 K25	9389949	15997152	7343021	11966859	5566816	8396919	4042525	5502226	2234548	2586534	955863	916072
Our Rank25 K5	10625514	18610930	8558994	14359172	6659697	10347295	4953883	6886130	2811068	3243059	1205710	1095289
Our Rank25 K10	10348481	17686389	8282101	13509799	6406047	9654965	4736247	6395264	2658637	2992652	1125215	1027395
Our Rank25 K15	9539346	15932768	7485341	11919492	5693771	8353537	4149910	5469407	2310041	2558353	997105	900158
Our Rank25 K20	10048983	17342828	7992423	13202776	6150762	9407436	4535614	6239666	2543619	2935586	1089553	1007543
Our Rank25 K25	9993470	17414704	7928249	13263656	6084294	9464885	4460823	6288785	2483015	2970588	1052081	1029442

Table 3.3: The total number of bytes written for each subset of the Circular 4 scheme using our proposed scheme, HEVC and scheme by Dib et al.

	QP = 2		QP = 6		QP = 10		QP = 14		QP = 20		QP = 26	
Scheme	Subset 1	Subset 2	Subset 1	Subset 2	Subset 1	Subset 2	Subset 1	Subset 2	Subset 1	Subset 2	Subset 1	Subset 2
HEVC	1378368	4910101	1055620	3502869	728140	2260753	460776	1298608	197691	486425	71359	144593
Dib et al	1370767	3604956	1047434	2396820	720530	1443437	453541	797714	195256	296109	71889	80630
Our Rank5 K5	1449622	4933538	1132801	3708151	840245	2605754	600205	1724526	332078	874714	153362	365773
Our Rank5 K10	1159357	3817710	862727	2732632	612968	1856015	426274	1205077	229491	601999	106038	247914
Our Rank5 K15	1034156	3295300	765433	2329939	542184	1584104	378047	1042783	205998	533226	95209	224463
Our Rank5 K20	973828	3051468	706419	2134144	491198	1439350	338642	940971	185382	483070	88161	205610
Our Rank5 K25	971785	3033863	703008	2102328	487201	1404313	334510	905074	181470	447998	85154	185944
Our Rank10 K5	1386172	4563561	1078972	3411124	798712	2406307	576228	1604331	325259	803800	151687	323515
Our Rank10 K10	1262549	4056153	964921	2963507	704244	2046235	498856	1341875	275501	665143	127968	265955
Our Rank10 K15	1190971	3751718	901066	2714968	647109	1868676	457355	1218665	252324	595759	116810	233650
Our Rank10 K20	1188549	3753696	903210	2722071	655630	1877749	464871	1232336	257146	610026	118936	244446
Our Rank10 K25	1162140	3557287	879834	2561074	637443	1755285	453605	1140882	252444	550884	117931	212252
Our Rank15 K5	1526972	5049904	1218417	3860125	921115	2773053	673084	1864388	376956	932112	167786	360026
Our Rank15 K10	1416127	4497745	1113470	3362398	829385	2368757	597397	1569816	331309	771406	151495	296433
Our Rank15 K15	1389174	4341358	1084674	3231812	802859	2274083	576600	1502714	320043	724601	143563	269489
Our Rank15 K20	1410892	4409559	1107033	3298032	826411	2325820	600992	1544108	337053	755313	155365	288130
Our Rank15 K25	1348775	4349536	1047512	3239316	772362	2279232	552343	1509618	306052	737763	136536	284737
Our Rank20 K5	1642901	5384922	1329563	4177785	1020339	3037514	754682	2056107	426447	1011982	190555	372721
Our Rank20 K10	1561885	5111681	1252534	3924292	951703	2832509	698657	1907339	395638	937239	176553	346004
Our Rank20 K15	1500278	4972125	1190671	3801659	895404	2738763	650737	1844214	363830	903772	160654	336389
Our Rank20 K20	1606475	5186099	1297181	3992867	992159	2888181	738196	1949036	426325	958216	196872	354591
Our Rank20 K25	1478485	4789080	1179300	3630073	888900	2584882	652403	1721155	368287	826213	167094	299325
Our Rank25 K5	1675977	5509486	1369664	4301804	1058025	3154279	787623	2149804	449526	1058306	199151	378977
Our Rank25 K10	1638221	5380940	1330174	4175458	1022343	3043818	758517	2062230	432418	1010312	193419	364035
Our Rank25 K15	1601083	5366700	1293413	4170043	989432	3048925	729682	2079424	412561	1027224	183964	371103
Our Rank25 K20	1564934	5056586	1259003	3871162	957332	2787831	703866	1880065	396805	924961	177952	343966
Our Rank25 K25	1586269	5264134	1278152	4064432	972441	2943301	713252	1983304	400571	963495	178578	340812

Table 3.4: The total number of bytes written for each subset of the Circular 4 scheme using our proposed scheme, HEVC and scheme by Dib et al.

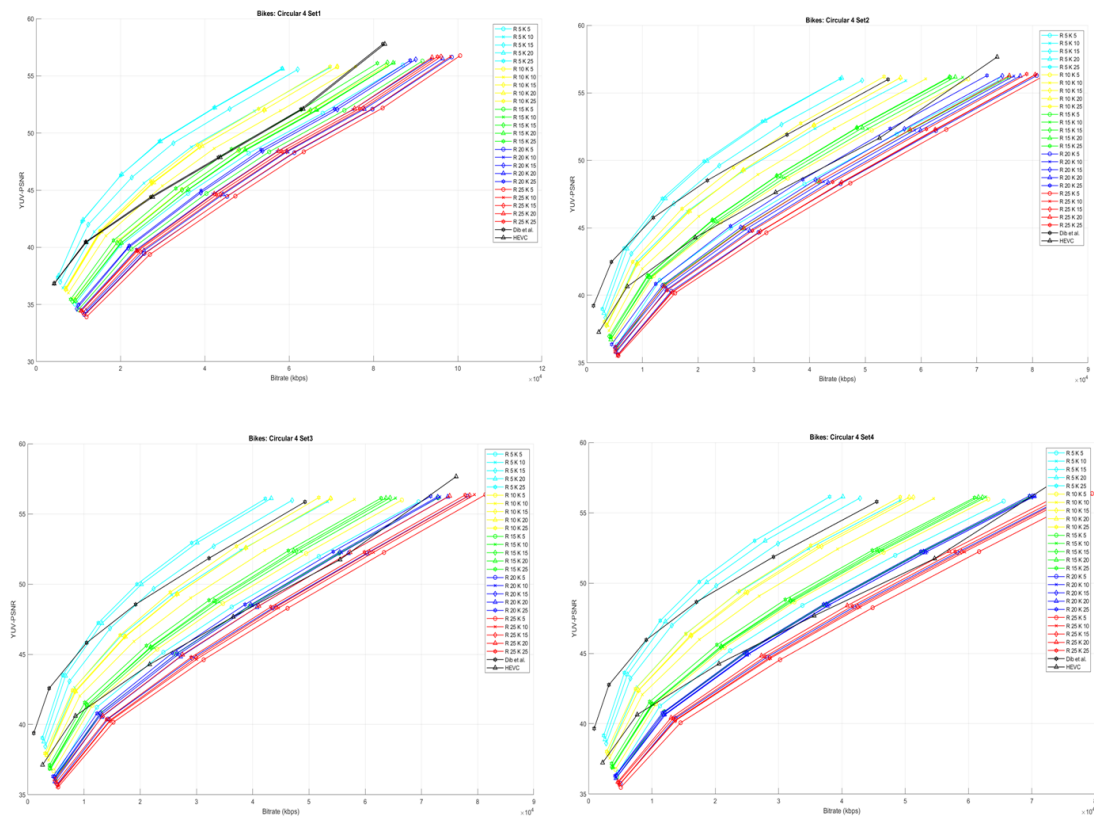
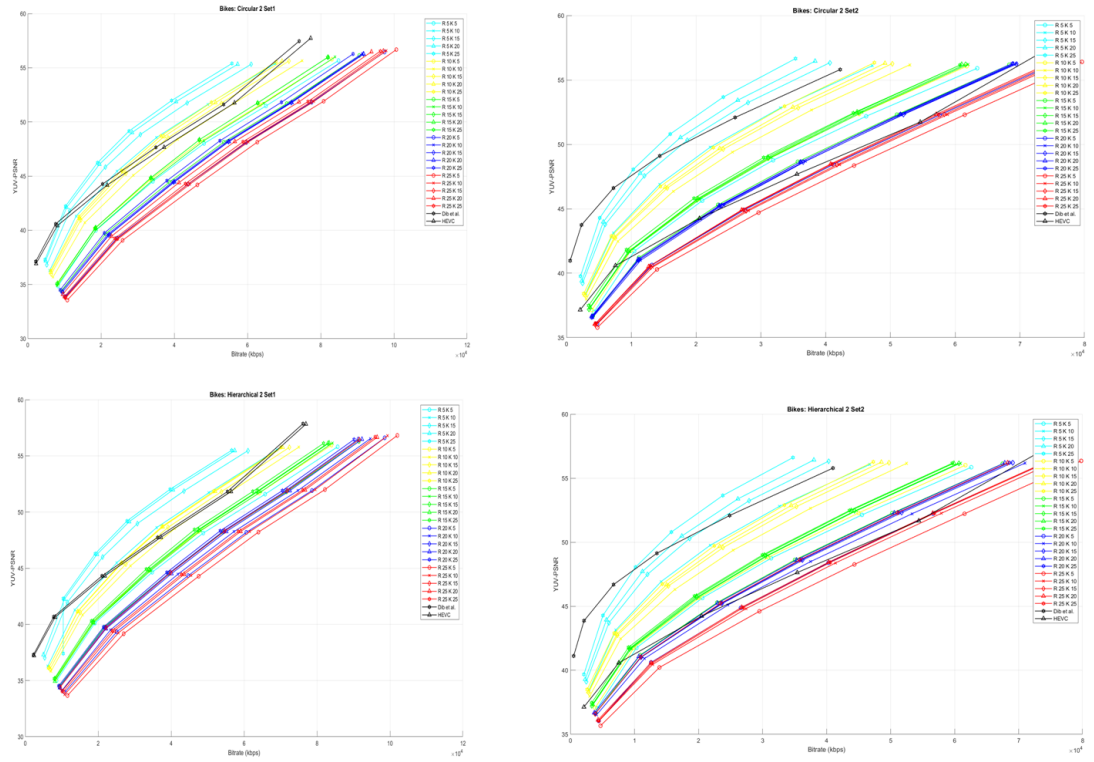
	QP = 2		QP = 6		QP = 10		QP = 14		QP = 20		QP = 26	
Scheme	Subset 3	Subset 4	Subset 3	Subset 4	Subset 3	Subset 4	Subset 3	Subset 4	Subset 3	Subset 4	Subset 3	Subset 4
HEVC	3808612	15397445	2776314	11158441	1826343	7273555	1083722	4202517	423932	1568183	131838	451220
Dib et al	5750563	14604140	3760641	9366615	2237900	5460885	1222652	2922288	444786	1028935	124533	279879
Our Rank5 K5	8105398	21029835	6037318	15541192	4230974	10835424	2813842	7177225	1433956	3615164	600490	1501596
Our Rank5 K10	6219770	15791054	4418079	11155711	2988249	7541943	1941552	4914366	972413	2467664	401291	1026241
Our Rank5 K15	5481825	13744474	3856047	9598805	2609029	6463807	1709364	4213172	866981	2117381	366432	891172
Our Rank5 K20	5051069	12886454	3515584	8942885	2358314	5987860	1537776	3891720	783112	1963819	334187	834293
Our Rank5 K25	4927043	12213350	3401513	8413953	2266839	5610028	1463325	3631085	725062	1813153	302388	761439
Our Rank10 K5	7759840	20248321	5774737	14951147	4050120	10426645	2687133	6888448	1339410	3415912	534609	1354215
Our Rank10 K10	6775374	17497333	4916454	12626343	3376646	8639864	2203106	5613367	1081870	2728847	430561	1072493
Our Rank10 K15	6284791	16461655	4521501	11800250	3091773	8045805	2008089	5215771	978478	2534337	384266	991063
Our Rank10 K20	6281031	16243033	4528501	11639639	3105125	7946160	2026630	5161911	996990	2525736	396773	996636
Our Rank10 K25	6037465	15792549	4329556	11253235	2956359	7636649	1916977	4930918	928239	2376644	358128	913343
Our Rank15 K5	8513265	22359910	6459015	16834675	4603534	11919637	3073697	7904472	1518054	3843273	579878	1447025
Our Rank15 K10	7627029	20132642	5672016	14899982	3974675	10397443	2619480	6815416	1270660	3273121	481122	1223213
Our Rank15 K15	7518683	19958540	5575778	14732593	3896585	10247814	2557152	6696619	1222119	3189477	451982	1176433
Our Rank15 K20	7430123	19751177	5516426	14617456	3864162	10214634	2546818	6719883	1232519	3233948	467713	1229357
Our Rank15 K25	7328701	19558053	5404953	14372333	3760135	9965696	2461929	6504203	1183817	3098559	450369	1164444
Our Rank20 K5	9128588	24360207	7036204	18689306	5080569	13465212	3413224	9044841	1663416	4406622	609708	1617888
Our Rank20 K10	8546983	22633989	6509899	17154453	4657422	12224264	3112047	8144087	1506800	3904525	550535	1412784
Our Rank20 K15	8503466	22489924	6469636	16971909	4626046	12044228	3096246	8004507	1503892	3839042	558163	1398083
Our Rank20 K20	8711564	22574981	6648250	17054898	4753511	12098226	3169986	8012910	1535184	3825382	564123	1386729
Our Rank20 K25	8356838	22345886	6335545	16842483	4507351	11913558	3000093	7875361	1437773	3734001	524984	1340284
Our Rank25 K5	9492695	25513948	7390206	19790061	5389715	14393634	3648952	9704145	1777380	4660380	631096	1631610
Our Rank25 K10	9268425	24762191	7171926	19059566	5201222	13755376	3504223	9219299	1695893	4403727	605380	1549482
Our Rank25 K15	9166157	24576673	7091226	18905096	5148418	13644104	3493997	9167418	1710243	4402221	619625	1558994
Our Rank25 K20	8751773	23892700	6686023	18272425	4794975	13108445	3212660	8755923	1556375	4178564	565292	1481585
Our Rank25 K25	9071818	24229652	6983993	18574306	5042729	13361808	3389909	8944760	1635265	4281955	582568	1509231

Table 3.5: The total number of bytes written for each subset of the Hierarchical 4 scheme using our proposed scheme, HEVC and scheme by Dib et al.

Scheme	QP = 2		QP = 6		QP = 10		QP = 14		QP = 20		QP = 26	
	Subset 1	Subset 2	Subset 1	Subset 2	Subset 1	Subset 2	Subset 1	Subset 2	Subset 1	Subset 2	Subset 1	Subset 2
HEVC	1395673	1730543	1072261	1322951	746449	914577	478947	574545	212148	242859	79349	85341
Dib et al	1334132	1332418	1003076	940407	685523	605122	431203	358628	185854	148936	67543	51893
Our Rank5 K5	1474313	1663254	1158274	1284956	865099	938176	624609	641761	348678	326964	158837	138183
Our Rank5 K10	1154267	1280224	860974	935516	610213	651275	425757	435016	232896	224667	107857	98949
Our Rank5 K15	1023594	1015004	747695	718389	523066	488892	364187	320528	200265	161352	94114	68499
Our Rank5 K20	1037813	1022396	760109	721812	532410	491717	369450	321374	201992	164140	94942	72090
Our Rank5 K25	967976	1037270	699320	725648	486732	490287	336442	321573	185365	164393	88680	71800
Our Rank10 K5	1402018	1522606	1094896	1152114	808329	823274	579198	551696	320160	279012	146666	114176
Our Rank10 K10	1244963	1278801	945328	938010	682323	654901	482097	434074	265279	217055	123265	90644
Our Rank10 K15	1240208	1216176	941491	894364	678426	629074	478859	422490	264269	217326	122022	91450
Our Rank10 K20	1190404	1185916	905048	866843	653016	605214	462327	403120	255196	203219	118864	84171
Our Rank10 K25	1201218	1199244	915528	879863	662966	617629	469789	412611	258928	207301	117722	84935
Our Rank15 K5	1541665	1548463	1230132	1183206	930016	855910	678366	581038	381887	295588	172903	119125
Our Rank15 K10	1414774	1416423	1108691	1064884	823716	761884	592377	515193	328978	262803	146778	110365
Our Rank15 K15	1374535	1386956	1072453	1042446	790427	740449	565714	498192	309966	251771	137938	104009
Our Rank15 K20	1352552	1397385	1055913	1053587	780493	753802	559666	506865	310987	254331	140061	103037
Our Rank15 K25	1393010	1354247	1091982	1011443	810632	720863	586468	487569	329104	250161	151760	104769
Our Rank20 K5	1640582	1695966	1330762	1319614	1020212	972849	754213	670747	427822	341962	190332	137255
Our Rank20 K10	1605097	1629312	1291810	1255689	983604	915635	723443	622366	405715	308075	176727	114503
Our Rank20 K15	1547922	1592219	1240704	1224589	937202	890257	684027	606670	381466	304695	168240	121537
Our Rank20 K20	1506494	1595805	1199433	1228035	905302	891166	662088	604367	372970	301843	169781	120239
Our Rank20 K25	1494331	1545639	1191166	1178876	898661	852560	656705	577819	371068	292226	165990	117215
Our Rank25 K5	1697518	1778287	1389071	1398505	1073571	1037877	799238	718782	454597	363614	200910	139349
Our Rank25 K10	1652566	1708893	1343408	1333040	1036027	979613	768932	671403	440733	334764	196485	127885
Our Rank25 K15	1739194	1762697	1434121	1382986	1116196	1024056	836583	707666	478795	358876	212284	138406
Our Rank25 K20	1642541	1712131	1335671	1333211	1027690	982907	761925	676056	429945	340628	186072	129556
Our Rank25 K25	1614743	1620700	1303336	1249180	995580	909137	732011	621577	412480	314587	182947	125681

Table 3.6: The total number of bytes written for each subset of the Hierarchical 4 scheme using our proposed scheme, HEVC and scheme by Dib et al.

Scheme	QP = 2		QP = 6		QP = 10		QP = 14		QP = 20		QP = 26	
	Subset 3	Subset 4	Subset 3	Subset 4	Subset 3	Subset 4	Subset 3	Subset 4	Subset 3	Subset 4	Subset 3	Subset 4
HEVC	5074139	17568888	3688802	12691883	2401244	8262698	1375933	4777113	508537	1760507	147145	490331
Dib et al	5042974	14831641	3414815	9560398	2081030	5596573	1163015	3024635	432932	1069623	125007	301923
Our Rank5 K5	6328796	21158633	4762496	15676914	3367260	10951133	2247227	7267580	1133304	3652036	472849	1504499
Our Rank5 K10	4874536	16026505	3493286	11330477	2384029	7666285	1565201	5010472	795432	2531124	338334	1061446
Our Rank5 K15	4021062	13651740	2809638	9508722	1890286	6382618	1229894	4150849	615670	2078712	253866	863254
Our Rank5 K20	3956047	13198892	2755805	9161869	1851722	6140659	1202594	3986544	606176	2003123	257041	842099
Our Rank5 K25	3830132	12246324	2651641	8432364	1773109	5620739	1149338	3640718	579815	1827412	248270	772672
Our Rank10 K5	5983511	20558589	4476495	15223717	3156873	10635782	2091304	7008486	1047224	3475063	423252	1384644
Our Rank10 K10	5162542	17647471	3753660	12749289	2583047	8735951	1684255	5673977	821024	2755334	321215	1076681
Our Rank10 K15	4982455	16824995	3611294	12086230	2485910	8247878	1623002	5351596	798880	2607411	317115	1027632
Our Rank10 K20	4785760	16339768	3460783	11711653	2376383	7985062	1550786	5177876	760661	2522601	302054	994505
Our Rank10 K25	4807326	16126895	3472135	11535988	2383858	7853522	1551367	5077764	753215	2451019	295667	952722
Our Rank15 K5	6508503	22403958	4956828	16869693	3543099	11941793	2357269	7905629	1152390	3827771	431216	1417280
Our Rank15 K10	5856376	20252549	4366374	14999383	3070235	10479950	2024663	6873499	984698	3294804	376203	1233587
Our Rank15 K15	5709967	20036318	4260838	14835612	2996095	10355929	1984664	6804163	968697	3266965	373511	1234118
Our Rank15 K20	5760641	19894052	4302590	14707131	3024827	10259453	1993237	6723531	962470	3225840	365876	1206354
Our Rank15 K25	5658344	19762984	4198303	14549943	2942753	10090439	1938785	6581715	942792	3139617	361617	1177805
Our Rank20 K5	7005926	24511992	5425730	18829446	3942348	13567847	2663352	9098794	1303652	4397284	479878	1612508
Our Rank20 K10	6566251	22735548	5019951	17235390	3599572	12280614	2402023	8170520	1161692	3910318	414405	1400913
Our Rank20 K15	6583396	22667489	5035945	17155931	3616105	12207902	2418440	8115231	1170015	3877934	431125	1403647
Our Rank20 K20	6694305	22659421	5124645	17119989	3671157	12134600	2444666	8028034	1181326	3821643	428262	1381034
Our Rank20 K25	6388544	22518363	4855839	16974037	3465632	11999983	2303033	7904892	1110323	3754447	406720	1354477
Our Rank25 K5	7341553	26036196	5755922	20250757	4229912	14751997	2883450	9941561	1409262	4774389	504428	1672822
Our Rank25 K10	7156392	24951854	5581264	19253851	4082506	13920606	2767199	9331306	1344107	4455365	477825	1557609
Our Rank25 K15	7191993	24869442	5614190	19209492	4101074	13915245	2772409	9369908	1338415	4496282	471044	1579174
Our Rank25 K20	6979542	24424960	5406376	18766299	3924808	13513199	2646116	9047842	1278618	4318392	454311	1509920
Our Rank25 K25	6851023	24362330	5278738	18685717	3807430	13423897	2539784	8954069	1215033	4267035	433300	1502645



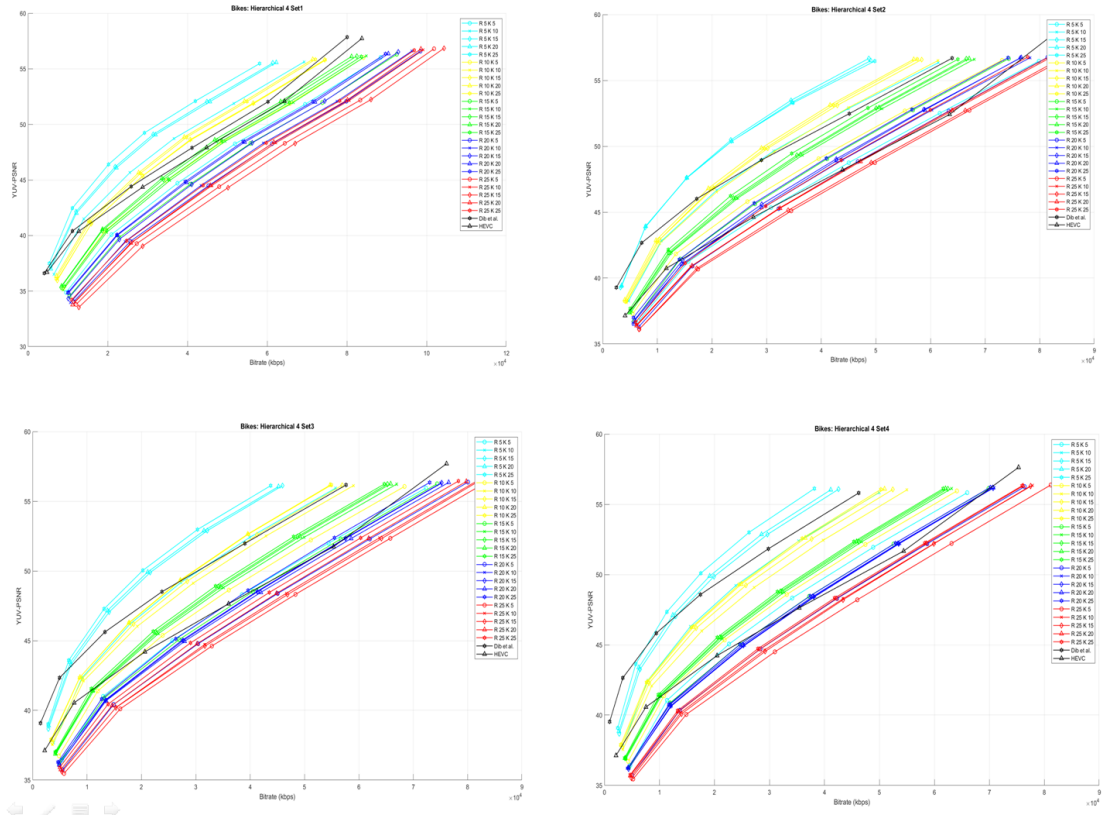


Figure 3.10: Bitrate vs PSNR graphs for Hierarchical 4

Table 3.7: Bjontegaard percentage bitrate savings for the proposed compression scheme with respect to Dib et al. on *Bikes* data. (positive values represent gain)

	C2		H2		C4				H4			
	Subset 1	Subset 2	Subset 1	Subset 2	Subset 1	Subset 2	Subset 3	Subset 4	Subset 1	Subset 2	Subset 3	Subset 4
Our Rank5 K5	-42.328	-68.401	-41.188	-70.239	-26.494	-57.548	-59.46	-62.236	-32.304	-47.441	-51.968	-61.404
Our Rank5 K10	-13.332	-47.453	-13.461	-50.549	11.468	-31.449	-33.829	-37.831	6.617	-13.952	-23.308	-37.064
Our Rank5 K15	3.6757	-31.365	7.5535	-36.985	32.695	-15.078	-20.081	-21.751	33.426	32.452	9.717	-16.029
Our Rank5 K20	15.61	-25.118	18.902	-30.438	53.943	-1.5659	-7.0801	-12.188	29.536	31.575	13.259	-11.691
Our Rank5 K25	21.727	-13.593	13.302	-19.05	53.707	3.4877	-1.0608	-4.4626	48.377	30.88	20.365	-0.23366
Our Rank10 K5	-40.024	-66.934	-38.718	-69.423	-21.023	-51.702	-56.287	-59.518	-25.722	-36.784	-46.212	-58.619
Our Rank10 K10	-28.251	-56.609	-25.789	-59.583	-6.563	-40.204	-43.8	-47.436	-7.7481	-13.929	-28.44	-45.401
Our Rank10 K15	-21.698	-52.47	-21.289	-55.522	4.0008	-31.157	-35.761	-41.4	-7.4935	-8.4548	-24.397	-40.853
Our Rank10 K20	-18.375	-51.093	-17.463	-54.018	3.2534	-31.649	-36.342	-40.718	-1.0342	-4.033	-19.81	-38.202
Our Rank10 K25	-15.933	-48.151	-16.886	-51.621	6.6973	-24.257	-31.61	-37.12	-3.2095	-6.7609	-19.914	-37.102
Our Rank15 K5	-48.075	-72.078	-47.093	-73.708	-33.395	-59.818	-62.666	-65.62	-37.469	-40.498	-52.906	-63.986
Our Rank15 K10	-40.898	-67.042	-39.001	-68.749	-23.793	-50.508	-54.497	-58.537	-26.986	-30.491	-43.153	-57.079
Our Rank15 K15	-38.867	-66.654	-37.632	-68.471	-20.853	-47.351	-53.246	-58	-22.81	-28.049	-41.294	-56.317
Our Rank15 K20	-38.954	-66.349	-38.337	-67.976	-23.727	-49.333	-52.912	-57.777	-21.458	-29.101	-42.007	-55.815
Our Rank15 K25	-38.884	-65.524	-36.189	-67.515	-16.608	-47.717	-50.896	-56.102	-25.506	-25.316	-39.818	-54.954
Our Rank20 K5	-53.935	-76.565	-53.261	-77.797	-40.824	-64.196	-66.91	-70.38	-43.71	-49.46	-58.911	-69.229
Our Rank20 K10	-48.686	-72.992	-50.075	-75.796	-35.695	-60.783	-63.116	-66.577	-41.376	-44.736	-53.839	-65.196
Our Rank20 K15	-49.247	-72.804	-46.831	-74.637	-30.761	-58.872	-62.507	-65.818	-37.798	-42.742	-54.185	-65.006
Our Rank20 K20	-49.457	-72.436	-47.837	-73.869	-39.071	-62.015	-64.278	-66.425	-34.935	-43.916	-54.916	-64.596
Our Rank20 K25	-46.36	-72.523	-45.808	-74.445	-29.787	-55.789	-61.322	-65.393	-34.179	-39.853	-51.524	-64.045
Our Rank25 K5	-56.152	-78.471	-55.75	-80.165	-42.914	-65.582	-69.084	-72.594	-46.667	-53.119	-62.249	-72.08
Our Rank25 K10	-54.245	-77.158	-53.727	-78.455	-40.758	-64.257	-67.834	-71.069	-44.41	-49.303	-60.448	-70.099
Our Rank25 K15	-53.778	-76.91	-47.293	-74.301	-38.18	-64.219	-67.365	-70.777	-49.163	-52.368	-60.853	-70.427
Our Rank25 K20	-50.804	-76.237	-51.662	-77.752	-35.857	-59.905	-64.203	-69.188	-43.853	-49.629	-58.649	-69.14
Our Rank25 K25	-52.955	-76.393	-50.917	-77.956	-36.868	-62.477	-66.45	-69.949	-41.754	-44.884	-56.866	-68.739
Average	-34.409	-62.052	-33.457	-64.360	-15.096	-46.157	-50.103	-53.954	-20.465	-26.396	-38.493	-52.532

Table 3.8: Bjontegaard DSNR savings for the proposed compression scheme with respect to Dib et al. on *Bikes* data. (negative values represent gain)

	C2		H2		C4				H4			
	Subset 1	Subset 2	Subset 1	Subset 2	Subset 1	Subset 2	Subset 3	Subset 4	Subset 1	Subset 2	Subset 3	Subset 4
Our Rank5 K5	3.6305	5.9173	3.4976	6.1254	2.3393	4.976	5.1662	5.4174	3.0056	4.2803	4.3503	5.4147
Our Rank5 K10	0.52032	2.7861	0.54409	3.0133	-0.85308	1.8105	1.9605	2.2111	-0.60182	0.79869	1.2634	2.2143
Our Rank5 K15	-0.63425	1.3397	-0.88349	1.6942	-2.0134	0.58137	0.87135	0.91578	-2.0364	-1.6229	-0.66602	0.59055
Our Rank5 K20	-1.3008	0.91551	-1.5021	1.2293	-2.9045	-0.21922	0.059176	0.30895	-1.8645	-1.6032	-0.8714	0.32237
Our Rank5 K25	-1.5676	0.23803	-2.5091	0.51614	-2.8784	-0.43158	-0.21875	-0.082684	-2.6909	-1.5887	-1.1881	-0.28591
Our Rank10 K5	3.3173	5.6245	3.179	5.9643	1.6871	4.1277	4.6636	4.9616	2.1917	2.9109	3.5746	4.8927
Our Rank10 K10	1.8923	3.9272	1.6506	4.2181	0.37197	2.7081	2.9959	3.2584	0.43606	0.86216	1.7335	3.0939
Our Rank10 K15	1.255	3.39	1.204	3.6528	-0.41182	1.8514	2.1852	2.6	0.42784	0.4753	1.3972	2.608
Our Rank10 K20	0.94997	3.2316	0.87786	3.4681	-0.37313	1.8755	2.2358	2.537	-0.1057	0.23116	1.0685	2.3513
Our Rank10 K25	0.71739	2.9163	0.84233	3.1789	-0.62128	1.3026	1.836	2.2039	0.094927	0.42046	1.0848	2.2574
Our Rank15 K5	4.672	6.745	4.5531	6.9707	3.2956	5.4913	5.8013	6.1033	3.7799	3.4664	4.5611	5.8624
Our Rank15 K10	3.5127	5.64	3.277	5.8259	2.047	3.987	4.3937	4.7843	2.3797	2.2981	3.2424	4.6578
Our Rank15 K15	3.2267	5.5792	3.0872	5.7687	1.7478	3.5918	4.2194	4.7051	1.9102	2.0791	3.0332	4.5335
Our Rank15 K20	3.2216	5.5185	3.2181	5.6745	2.0197	3.868	4.1938	4.6952	1.7445	2.2002	3.13	4.4702
Our Rank15 K25	3.221	5.3024	2.9078	5.5445	1.3148	3.6312	3.8797	4.3827	2.1538	1.799	2.8723	4.3326
Our Rank20 K5	5.9105	8.0545	5.8625	8.1955	4.4874	6.3885	6.6975	7.2208	4.8727	4.7494	5.612	7.0505
Our Rank20 K10	4.8388	7.0229	5.2145	7.4989	3.6352	5.6889	5.9023	6.3259	4.5154	4.0085	4.7165	6.1284
Our Rank20 K15	4.9483	6.919	4.5859	7.1933	2.9643	5.3557	5.7706	6.1218	3.9079	3.7281	4.7873	6.084
Our Rank20 K20	4.9772	6.8251	4.7654	6.9905	4.1569	5.9602	6.1413	6.2796	3.4023	3.9505	4.8771	5.9633
Our Rank20 K25	4.4364	6.8984	4.4108	7.1645	2.7841	4.8102	5.5484	6.0392	3.3083	3.381	4.3904	5.8804
Our Rank25 K5	6.5061	8.6563	6.5135	9.0222	4.8968	6.7128	7.2255	7.8017	5.486	5.3271	6.289	7.7445
Our Rank25 K10	6.0488	8.2379	6.0528	8.4084	4.4963	6.4545	6.9463	7.3741	5.0335	4.6857	5.9213	7.2469
Our Rank25 K15	5.9373	8.1536	4.665	7.1414	4.0669	6.4587	6.8556	7.3189	6.052	5.2055	5.9661	7.3834
Our Rank25 K20	5.2943	7.9726	5.5702	8.1851	3.698	5.5219	6.1015	6.9072	5.0054	4.7263	5.566	7.0297
Our Rank25 K25	5.7586	7.9967	5.4099	8.2794	3.8397	6.0235	6.5967	7.0843	4.5457	4.0528	5.231	6.9139
Average	3.251	5.432	3.079	5.636	1.751	3.941	4.321	4.699	2.278	2.432	3.277	4.589

Table 3.9: Bjontegaard percentage bitrate savings for the proposed compression scheme with respect to HEVC on *Bikes* data. (positive values represent gain)

	C2		H2		C4				H4			
	Subset 1	Subset 2	Subset 1	Subset 2	Subset 1	Subset 2	Subset 3	Subset 4	Subset 1	Subset 2	Subset 3	Subset 4
Our Rank5 K5	-38.714	6.9763	-39.274	9.0254	-25.757	-21.919	-6.6881	-4.2367	-25.146	-7.3921	-16.292	-7.1594
Our Rank5 K10	-7.706	89.941	-10.566	94.073	12.631	31.258	59.498	65.422	17.916	57.45	38.399	58.274
Our Rank5 K15	10.472	152.53	11.185	150.35	34.224	65.054	94.325	111.49	47.082	143.07	104.21	117.07
Our Rank5 K20	24.367	175.31	23.364	176.66	55.737	92.493	126.83	138.45	42.832	141.16	112.11	129.31
Our Rank5 K25	30.881	217.42	17.508	222.1	55.489	104.31	142.94	161.16	63.482	140.77	125.54	159.73
Our Rank10 K5	-36.274	12.952	-36.732	12.875	-20.228	-10.787	1.3428	3.3502	-17.879	12.992	-5.8908	0.37402
Our Rank10 K10	-23.681	55.808	-23.344	57.228	-5.6059	12.15	33.404	37.92	2.0056	57.207	28.954	36.691
Our Rank10 K15	-16.68	72.209	-18.69	74.812	5.0729	31.399	54.586	55.771	2.2889	66.773	36.529	49.053
Our Rank10 K20	-13.134	77.048	-14.731	80.744	4.3176	30.277	52.761	57.264	9.4073	75.344	45.955	56.429
Our Rank10 K25	-10.525	88.981	-14.135	92.002	7.7947	46.225	65.019	68.181	7.0092	70.032	46.101	59.862
Our Rank15 K5	-44.907	-4.9897	-45.414	-3.2663	-32.744	-26.309	-13.937	-12.604	-30.89	4.6824	-17.973	-12.762
Our Rank15 K10	-37.232	14.423	-37.037	17.609	-23.034	-8.0515	6.3954	6.9048	-19.311	24.171	-0.037272	5.075
Our Rank15 K15	-35.071	15.61	-35.621	18.762	-20.062	-1.7129	9.6919	8.5031	-14.704	28.726	3.2691	7.0315
Our Rank15 K20	-35.163	17.04	-36.355	20.867	-22.968	-6.0137	10.02	8.6059	-13.197	26.59	2.0193	8.3061
Our Rank15 K25	-35.084	21.627	-34.13	23.949	-15.77	-2.7586	15.358	13.808	-17.659	34.255	5.9703	10.596
Our Rank20 K5	-51.186	-21.906	-51.807	-19.799	-40.259	-34.537	-23.998	-25.301	-37.816	-11.934	-28.943	-26.165
Our Rank20 K10	-45.577	-8.3173	-48.513	-11.14	-35.072	-27.918	-14.708	-14.936	-35.246	-2.4537	-19.405	-15.65
Our Rank20 K15	-46.179	-7.273	-45.155	-6.4185	-30.085	-24.318	-13.268	-12.753	-31.29	0.88858	-20.254	-15.221
Our Rank20 K20	-46.401	-5.8307	-46.194	-3.2289	-38.486	-30.412	-17.639	-14.493	-28.108	-1.5868	-21.415	-13.979
Our Rank20 K25	-43.096	-6.4927	-44.097	-5.8828	-29.099	-18.047	-10.223	-11.513	-27.273	6.1926	-15.324	-12.668
Our Rank25 K5	-53.572	-28.541	-54.391	-29.088	-42.373	-37.072	-29.13	-30.962	-41.106	-18.326	-34.895	-33.093
Our Rank25 K10	-51.533	-23.83	-52.299	-22.202	-40.194	-34.629	-26.172	-26.872	-38.605	-11.089	-31.573	-28.101
Our Rank25 K15	-51.033	-22.831	-45.632	-5.3809	-37.588	-34.62	-25.161	-26.161	-43.859	-17.046	-32.198	-29.11
Our Rank25 K20	-47.854	-20.48	-50.16	-19.447	-35.24	-26.219	-17.275	-21.842	-38	-11.749	-28.288	-25.737
Our Rank25 K25	-50.153	-21.037	-49.389	-20.45	-36.261	-30.967	-22.691	-23.843	-35.676	-3.2621	-24.925	-24.667
Average	-30.201	33.853	-31.264	36.190	-14.222	1.475	18.051	20.452	-12.149	32.218	10.065	18.139

Table 3.10: Bjontegaard DSNR savings for the proposed compression scheme with respect to HEVC on *Bikes* data. (negative values represent gain)

	C2		H2		C4				H4			
	Subset 1	Subset 2	Subset 1	Subset 2	Subset 1	Subset 2	Subset 3	Subset 4	Subset 1	Subset 2	Subset 3	Subset 4
Our Rank5 K5	3.1858	-0.64819	3.3135	-0.75924	2.2456	1.3276	0.27811	0.055206	2.2956	0.49604	0.86727	0.23801
Our Rank5 K10	0.10433	-3.5577	0.36794	-3.647	-0.93802	-1.7129	-2.7215	-2.8549	-1.2597	-2.9325	-2.0563	-2.6661
Our Rank5 K15	-1.0361	-4.8883	-1.056	-4.8387	-2.0943	-2.905	-3.7461	-4.0313	-2.6755	-5.5732	-3.9089	-4.1377
Our Rank5 K20	-1.6952	-5.3044	-1.6733	-5.2954	-2.9825	-3.6848	-4.5336	-4.6039	-2.5055	-5.504	-4.1142	-4.3889
Our Rank5 K25	-1.9583	-5.9713	-2.6928	-6.0021	-2.9553	-3.8807	-4.8233	-4.9635	-3.3235	-5.4973	-4.4206	-4.9528
Our Rank10 K5	2.8709	-0.91965	2.995	-0.90207	1.5934	0.48147	-0.19297	-0.35531	1.4902	-0.84537	0.12971	-0.22785
Our Rank10 K10	1.461	-2.4919	1.471	-2.5255	0.28202	-0.86149	-1.751	-1.8927	-0.2398	-2.8676	-1.5797	-1.8527
Our Rank10 K15	0.8317	-2.9931	1.0256	-3.0485	-0.49927	-1.6585	-2.5021	-2.4865	-0.24677	-3.2583	-1.9027	-2.294
Our Rank10 K20	0.52914	-3.1415	0.70063	-3.2215	-0.46096	-1.6467	-2.46	-2.5458	-0.77263	-3.5207	-2.2205	-2.5215
Our Rank10 K25	0.29754	-3.4197	0.66539	-3.4783	-0.70869	-2.1683	-2.8188	-2.8281	-0.57277	-3.3225	-2.1988	-2.5902
Our Rank15 K5	4.2149	0.19088	4.3654	0.098622	3.1998	1.812	0.90918	0.74316	3.0562	-0.32068	1.1035	0.72279
Our Rank15 K10	3.0638	-0.87861	3.0925	-1.0107	1.9528	0.35851	-0.42696	-0.48056	1.6743	-1.4612	-0.16428	-0.4106
Our Rank15 K15	2.7772	-0.92855	2.9027	-1.0545	1.6544	-0.0079474	-0.57916	-0.54031	1.2122	-1.6668	-0.36925	-0.53141
Our Rank15 K20	2.7704	-1.0103	3.0332	-1.163	1.925	0.23977	-0.62275	-0.58026	1.0459	-1.5479	-0.26879	-0.58442
Our Rank15 K25	2.7713	-1.1818	2.7236	-1.2684	1.223	0.013689	-0.90877	-0.84246	1.4451	-1.9385	-0.51487	-0.7029
Our Rank20 K5	5.446	1.46	5.6711	1.3046	4.3901	2.7147	1.8077	1.8143	4.1359	0.93694	2.1244	1.8523
Our Rank20 K10	4.3794	0.45242	5.0244	0.64116	3.5387	2.0174	1.0202	0.96433	3.7859	0.24222	1.2674	0.98026
Our Rank20 K15	4.4862	0.37677	4.3975	0.33445	2.8689	1.6768	0.88699	0.78211	3.1838	-0.055937	1.3202	0.93879
Our Rank20 K20	4.5144	0.29115	4.577	0.14145	4.0589	2.2793	1.2571	0.93523	2.6787	0.15393	1.4243	0.83941
Our Rank20 K25	3.9782	0.34397	4.2229	0.28962	2.6881	1.1676	0.68384	0.71858	2.5867	-0.40283	0.93862	0.75909
Our Rank25 K5	6.0378	2.1074	6.3201	2.1637	4.7988	3.0349	2.3366	2.4195	4.7416	1.5342	2.7954	2.5789
Our Rank25 K10	5.5821	1.6651	5.861	1.5263	4.3985	2.7714	2.052	2.0036	4.29	0.90153	2.4348	2.0777
Our Rank25 K15	5.4698	1.5886	4.4761	0.2688	3.9696	2.7644	1.9389	1.9326	5.2986	1.4118	2.4938	2.1875
Our Rank25 K20	4.8316	1.392	5.3789	1.2939	3.601	1.8476	1.2219	1.5386	4.2668	0.94557	2.0922	1.8571
Our Rank25 K25	5.2938	1.4177	5.2198	1.3736	3.7429	2.3635	1.7138	1.7138	3.8127	0.25342	1.7712	1.7498
Average	2.808	-1.041	2.895	-1.151	1.659	0.333	-0.479	-0.535	1.576	-1.353	-0.118	-0.443

CHAPTER 4

CONCLUSION AND FUTURE WORK

4.1 Conclusion

In our study of Light field compression problem using FDL, we have studied and proposed an algorithm to deal with problem LF compression, lossy light field compression based on FDL using Hybrid Tucker TS via sketching. In our proposed method compression is based on FDL using Hybrid tucker decomposition we have tried to use the good energy packing property of SVD and de-correlating property of KLT, which could be a good tool to compress light field data. We discussed about the motivation behind the idea and then we discussed proposed scheme. Finally we have shown the comparative analysis of our proposed scheme with Dib et al. and HEVC, along with graphs and tables, we have used objective analysis Bjontegaard metric to show the significant gain in PSNR and bitrate.

4.2 Future Work

Based on the work presented in this thesis, the following research issues which can be explored by researchers in the field of light field compression in near future. The quantitative analysis done in our work shows the ability of our proposed scheme. More test can be done with large number of data sets to explore the generalization of our proposed method. Despite getting significant gain in bitrate and PSNR for some Rank and K (sketch dimension) values. There is a lot of improvement that could be done in order to get gain in bitrate. Our proposed scheme failed to achieve gain in bitrate for higher Rank. We need to explore more idea to save bitrate and de-correlate data among the images and within the image.

REFERENCES

- [1] M. Le Pendu, C. Guillemot, and A. Smolic, “A fourier disparity layer representation for light fields,” arXiv:1901.06919 (under review in IEEE Trans. Image Proc.), 2019.
- [2] R. Verhack, T. Sikora, L. Lange, R. Jongebloed, G. VanWallendael, and P. Lambert, “Steered mixture-of-experts for light field coding, depth estimation, and processing,” in IEEE Int. Conf. Multimed. Expo Workshops (ICMEW), 2017, pp. 1183–1188.
- [3] Dansereau, D.G.; Pizarro, O.; Williams, S.B. Decoding, calibration and rectification for lenselet-based plenoptic cameras. Proceedings of the IEEE conference on computer vision and pattern recognition, 2013, pp. 1027–1034.
- [4] Bjontegaard, G. Calculation of average PSNR differences between RD-curves. VCEG-M33 2001.
- [5] E. Dib, M. L. Pendu and C. Guillemot, "Light Field Compression Using Fourier Disparity Layers," 2019 IEEE International Conference on Image Processing (ICIP), 2019, pp. 3751-3755, doi: 10.1109/ICIP.2019.8803756.
- [6] A. Aggoun, “A 3D DCT compression algorithm for omnidirectional integral images,” in IEEE Inter. Conf. on Acoustics, Speech and Signal Processing, 2006.
- [7] M. Magnor, A. Endmann, and B. Girod, “Progressive compression and rendering of light fields,” in Vision, Modeling, and Visualization Proceedings, 2000
- [8] M. Magnor and B. Girod, “Data compression for light-field rendering,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 10, no. 3, pp. 338C–343, 2000.
- [9] Sullivan, G.J.; Ohm, J.R.; Han, W.J.; Wiegand, T. Overview of the high efficiency video coding (HEVC) standard. IEEE Transactions on circuits and systems for video technology 2012, 22, 1649–1668.

- [10] Sharma, M.; Ragavan, G. A Novel Randomize Hierarchical Extension of MV-HEVC for Improved Light Field Compression. 2019 International Conference on 3D Immersion (IC3D). IEEE, 2019, pp. 1–8.
- [11] X. Jiang, M. Le Pendu, R. Farrugia, C. Guillemot, "Light Field Compression with Homography-based Low Rank Approximation", IEEE Journal of Selected Topics in Signal Processing (J-STSP), vol. 11, No. 7, pp. 1132-1145, Oct. 2017
- [12] E. Dib, M. Le Pendu, X. Jiang, and C. Guillemot, "Super-ray based low rank approximation for light field compression," in Data Compression Conference (DCC), Mar. 2019.
- [13] C. Conti, P. Nunes, and L. D. Soares, "HEVC-based light field image coding with bi-predicted self-similarity compensation," in IEEE Int. Conf. Multimed. Expo Workshops (ICMEW), Jul. 2016.
- [14] I. Tabus, P. Helin, and P. Astola, "Lossy compression of lenslet images from plenoptic cameras combining sparse predictive coding and jpeg 2000," in IEEE Int. Conf. Image Process. (ICIP). IEEE, 2017, pp. 4567–4571.
- [15] C. Battaglino, G. Ballard, and T. Kolda. A practical randomized CP tensor decomposition. SIAM Journal on Matrix Analysis and Applications, 39(2):876–901, 2018.
- [16] author = Malik, Osman Asif and Becker, Stephen, booktitle = Advances in Neural Information Processing Systems, editor = S. Bengio and H. Wallach and H. Larochelle and K. Grauman and N. Cesa-Bianchi and R. Garnett, pages = , publisher = Curran Associates, Inc., title = Low-Rank Tucker Decomposition of Large Tensors Using TensorSketch, volume = 31, year = 2018
- [17] Rerabek, M.; Ebrahimi, T. New light field image dataset. 8th International Conference on Quality of Multimedia Experience (QoMEX), 2016, number CONF.
- [18] Waldemar and T. A. Ramstad, "Hybrid KLT-SVD image compression," 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing, 1997, pp. 2713-2716 vol.4, doi: 10.1109/ICASSP.1997.595349.