VISUAL DATA REPRESENTATION AND CODING BASED ON TENSOR DECOMPOSITION AND SUPER-RESOLUTION

By

Abo Talib Mahfoodh

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Electrical Engineering—Doctor of Philosophy

2016

ABSTRACT

VISUAL DATA REPRESENTATION AND CODING BASED ON TENSOR DECOMPOSITION AND SUPER-RESOLUTION

By

Abo Talib Mahfoodh

Tensor based methods have been used in a wide range of signal processing applications. A particular area of interest is tensor decomposition, which can be used to reduce the dimensionality of the massive multidimensional data. Hence, tensor decomposition can be considered as a high dimension extension of popular Singular Value Decomposition (SVD) methods used for matrix analysis. The lower dimension representation of tensors resulting from tensor decomposition can be used for classification, pattern recognition, and reconstruction. Our objective in the first part of this thesis, is to develop a tensor coding framework based on a tensor decomposition method for visual data efficient representation and compression.

As part of the proposed tensor coding framework, we developed a tensor decomposition algorithm that decomposed the input tensor into a set of rank-one tensors. The proposed decomposition is designed to be efficient specifically for visual data. The proposed tensor decomposition algorithm is applied in a block-wise approach. Two partitioning methods are proposed for tensor coding framework which are uniform and adaptive tree partitioning. The former subdivide a region into a set of equal size blocks while the later subdivide a region into a set of variable size blocks. The decision whether to subdivide the region or not is made based on the existing amount of the information and the overall available bitrate. A tree data structure stores the partitioning structure information which is required for the tensor reconstruction process.

Furthermore, an encoder/decoder framework is proposed for compressing and storing the decomposed data. The proposed framework provides a number of desirable properties especially at the decoder side which can be critical for some applications. Low complexity reconstruction, random access, and scalability are the main properties that we have targeted. We demonstrate the viability of the proposed tensor coding framework by employing it for the representation and coding of three types of data sets: hyperspectral/multispectral images, bio-metric face image ensembles, and low motion videos. These data sets can be arranged as either three or four dimensional tensors. For each application, we show that the compression efficiency along with the inherited properties of the proposed tensor coding framework, provide a competitive approach to the current standard methods.

In the second part of the thesis, we propose an example-based super-resolution algorithm for a new framework of scalable video streaming. The proposed method is applicable to scalable videos where the enhancement layer of some frames might be dropped due to changing network conditions. This leads to a streaming scenario that we call Inconsistent Scalable Video (ISV) streaming. At the decoder, the frames with the enhancement layer are used as a dictionary for super-resolving other video frames whose enhancement layers were dropped. The proposed super-resolution framework is integrated with Google VP9 video codec. Then it is applied to various High Definition (HD) videos to estimate the dropped enhancement layer. Our simulation results show an improvement visually and in terms of PSNR over traditional interpolation up-sampling filters.

To my family ...

ACKNOWLEDGMENTS

I would like to thank professor Hayder Radha who has been my adviser during the PhD program. It was a great learning opportunity for me to work with him. His supportive and constructive guidance was a major contributor to my success in completing this thesis. I also would like to thank my guidance committee members professor Selin Aviyente, professor Yiying Tong, and professor Lalita Udpa for their efforts, support, and for what I have learned from them. My special gratitude to professor Yiying Tong for his gracious support that left a great impact on me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Tensor Coding

Tensor decomposition, which is a generalization of matrix SVD decomposition, has been receiving growing attention recently [1–5]. Tensors are multi-dimensional set of data and generalizations of vectors and matrices, which are 1D and 2D tensors, respectively. There are two main tensor decomposition methods. The first one is known as Higher Order SVD (HOSVD) or Tucker decomposition [1, 6]. The second decomposition approach is known as Canonical-decomposition Parallel-factor (CP) [7, 8]. The CP decomposition factorizes a tensor onto a number of rank-one tensors [1]. A rank-one tensor can be thought of as a "basis-tensor" that can be represented efficiently and reconstructed perfectly using 1D vectors. In particular, an n dimensional rank-one tensor can be represented by n 1D vectors only. Like matrix SVD, tensor decomposition methods are used in a wide range of applications. Tensor decomposition has been used for data analysis, fusion, representation and classification [9].

In this thesis, we propose a framework based on tensor decomposition for representation and coding of n dimensional visual data. There are various types of data that can fit in this category of "n dimensional visual data". For example, a set of 2D video frames over time is inherently a 3D data set. Also, a set of images stored within a common image database (e.g., for storing images of human faces; or images of certain class of visual objects) can be categorized as a high dimension data set. In general, for a set of large number of images or

2D video frames stored in a common image database, we call these types of imaging/video data as a "visual data ensemble". Hence, in this thesis, an ensemble is a 3D or higher dimensional set of tensor data. We call the 2D matrices along a particular dimension, slice. For example a 3D visual data ensemble consists of 2D matrices along the third dimension, which are stacked on top of each other. In our experiments, we used 3D and 4D visual data ensembles to illustrate the concept and show the experimental results.

Our goal is to develop a compression approach for visual data ensembles while achieving desirable properties for applications like web browsing and image retrieval. A tensor based coding framework can achieve:

- **Random access:** It is the ability to reconstruct any slice within an ensemble without the need to reconstruct or access any other slice from the same ensemble. Random access is crucial for some applications, not only for storage efficiency, but also to reduce bandwidth across networks for scalable search and retrieval engines.

- **Coding efficiency:** achieving higher compression ratio by exploiting any potential correlation that may exist among the slices within the same data set.

- **Low complexity:** The ability to reconstruct the original tensor data with low complexity can provide fast access and enable low end devices to benefit from the proposed approach.

As part of the proposed tensor coding framework, we developed a Progressive Canonical/Parallel (PCP) tensor decomposition, which is based on the popular CP tensor factorization [1], for decomposing any arbitrary given visual data ensemble. Similar to CP, PCP factors an n-dimensional tensor into a set of R rank-one tensors, each of which is represented by n one dimensional vectors. We apply PCP in a block-wise basis, and each tensor-block

is decomposed into a different number of rank-one tensors. Depending on the nature of the data, a uniform partitioning or an adaptive tree partitioning can be employed to partition the tensor into a set of equal size blocks or variable size blocks respectively. The block-wise tensor decomposition strategy leads to the need for deriving the optimal distribution of rank-one tensors among the different visual data ensemble tensor blocks. Thus, we formulated the rank distribution as an optimization problem. A greedy algorithm is developed for identifying the optimal number of rank-one tensors that should be used for decomposing the ensemble blocks.

Tensor decomposition is well suited for approximating low rank data. If the signal contains a large amount of high frequency data, then the rank of the corresponding tensor is high. For the high rank tensors, at some point we would not be able to capture the high frequency information, no matter how many rank-one tensors are used. Meanwhile, there are cases and applications that would not require the presence of high frequency data, enabling the possibility of near lossless and lossless compression is desirable. The flexibility of the reconstruction quality that can range from low quality to high quality can serve different types of applications. A residual coding module is presented as part of the proposed tensor coding framework to enable near-lossless coding.

Data encoding based on tensor decomposition has different applications for 3D and higher dimensional visual data ensembles. For example, a low motion video holds a high correlation among the frames which can be exploited for low complexity coding [10]. Similarly, an image ensemble of face images, can be coded with tensor-factorization based method [11] to exploit the existing correlation among a large number of images. The potential applications of the proposed tensor coding framework are illustrated using three types of data sets:

1. **Hyperspectral/multispectral images data set:** These types of data set have a

high correlation in the spectrum spectral and time temporal dimension. The proposed tensor coding framework exploits this correlation to create an efficient representation and coding that holds the desirable random access property.

2. **Bio-metric face image data set:** A set of face images taken under different conditions, which can be used for recognition and analysis, form a 3D tensor. These images contain a high amount of similarities, espcially for face images of the same person.

3. **Low complexity video compression:** The proposed tensor coding framework can be used in video coding scenarios where low complexity decoding and random access are important.

1.2 Tensor coding motivations

Visual data is a core component of many applications and services. For example image databases of faces and fingerprints are used for security applications, and hyperspectral/multispectral images are used for scientific analysis. Such databases store a large number of images of the same type. In many such applications, the traditional compression standards are used to store these images without exploiting the correlation. The goal of the proposed tensor coding framework is to develop a compression method to provide:

- Random access to any slice without decoding other slices.

- Coding efficiency by exploiting any potential correlation that may exist among the slices within the same data set.

- Low complexity and fast decoding.

- Scalable reconstruction in which a lower quality can be reconstructed with part of the encoded data.

- Enable fast access and browsing.

1.3 Tensor coding related work

Tensor based algorithms have been used widely in image processing and computer vision applications [3, 12]. Our proposed framework belongs to the area of multidimensional signal decomposition. In [13] a video encoder based on 2DSVD decomposition was proposed. 2DSVD [14] decomposes a GOP onto two eigenvector matrices and group of coefficient matrices. Their framework was based on coding the factored matrices. Our method has two advantages over 2DSVD encoder. (a) It is faster than 2DSVD. (b) It can be extended to higher dimensions. Furthermore the proposed tensor coding framework provides higher PSNR.

A rank R decomposition method was proposed in [15] for video dimension reduction. Based on the presented experimental results, the method works well for texture videos, which are known to be low rank. The value of R in the proposed method is fixed. The same method was used in [16] for compact representation of video textures.

An efficient rank R decomposition was proposed in [17] for compact representation of image ensembles. The method was used for compact representation of face datasets and toy video sequence.

HOSVD analysis (Tucker decomposition) was proposed in [17] for modeling dynamic textures. Similarly, HOSVD decomposition was used in [18] for representing facial images in computer vision problems.

A D-1 factorization was used in [19] for video compression and classification. However, the method was used for video classification not coding. A rank-one decomposition was used for compact representation of multidimensional data in [20]. Their method is based on standard rank-one decomposition, which is not as efficient as our proposed PCP. They applied their approach for decomposing video textures.

A compression method based on tucker decomposition was proposed in [21] for hyperspectral images. The tucker decomposition was applied on the wavelet transform coefficients of the hyperspectral images to compact the energy of the sub images. This will result in loss of random access property.

Furthermore, [22, 23] proposed tensor based methods for hyperspectral image compression. Both works were based on proposing a decomposition algorithm without presenting a complete coding framework. It is important to measure the compression efficiency based on a storage requirement not just the number of coefficients. This is because of the fact that the decomposed values are float numbers as opposed to the visual data values which are usually 8 bits integers.

Lei Wang et al. presented a hyperspectral image compression system based on the lapped transform and Tucker decomposition [24]. The lapped transform decorrelate the hyperspectral image bands. Then they arranged the transformed coefficients of different frequencies into a 3D wavelet sub-band tensor. Finally Tucker decomposition was used to decompose the tensor into a core tensor and three factor matrices. The bit-plane coding algorithm was used to encode the core tensor.

Vasilescu, M. Alex O., and Demetri Terzopoulos proposed a multilinear modeling technique that employs an N-mode SVD [18]. They presented the multilinear analysis of facial images ensembles which contain different types, like different facial geometries, expressions,

head poses, and lighting conditions. They call the representation TensorFaces and argue that multilinear analysis can be an effective framework for computer vision problems.

Hazan Tamir et al. presented an algorithm for a non-negative 3D tensor decomposition which extracts a local parts feature decomposition from a set of object images [25]. They showed that this feature can be used for face detection using SVM and Adaboost classifiers. They argue that their tensor factorization has a unique factorization and it preserves the 2D representations of images. They argue that the proposed algorithm improve the sparsity level, ghost residue, and compression around comparing to NMF.

Wang, Hongcheng, et al. presented an out-of-core algorithm to approximate high dimensional tensors [26]. The algorithm preserve the original dimensionality of the data items and hence exploit existing spatial redundancy more effectively to reduce the computation complexity. They partition a tensor into a set of blocks and complete the tensor operations in a block-wise approach. Their experimental results showed the advantage of the proposed method for three graphics models which are 6D bidirectional texture functions, 7D dynamic BTFs and 4D volume simulation sequences. The proposed method can process out-of-core data and achieve higher compression ratios comparing to the previous methods.

Inoue, Kohei, and Kiichi Urahama proposed a dyadic singular value decomposition (DSVD) that reduces the dimensionality of a set of matrix data [27]. Their experimental results showed the method application in image compression and face recognition. The DSVD algorithm is derived from the higher Order SVD (HOSVD) of a three dimensional tensor. It provides a low rank approximation for data matrices. They showed that the DSVD can provides better results in terms of the computational complexity and accuracy in image compression comparing to the other dimensionality reduction methods. They argue that their results are better than the result derived from the eigenface method.

Hou Junhui, et al. proposed a compact and progressive representation of the motion capture data in video coding by employing a tensor decomposition method [28]. They arranged the motion capture sequence in a three dimensional tensor. They argue that this type of data has strong correlation within and across slices of the tensor. Then, they performed tensor decomposition iteratively to take advantage of the existing correlation. Their experimental results showed that their proposed method provides better results in terms of scalability and storage requirement comparing to the existing algorithms.

[29] Suter Susanne K., et al. proposed a multi-scale volume representation in a GPU-accelerated out-of-core multi-resolution rendering framework. The proposed method is based on the tensor approximation. They argue that the proposed hierarchical tensor decomposition can achieve large volume data pre-processing, GPU accelerated tensor reconstruction, and effective tensor quantization for data transfer bandwidth reduction. They showed that the proposed multi-scale representation can perform the extraction, analysis and display of structural features. Their experimental results showed the application of the proposed methos on a gigabyte-sized micro-tomographic volumes data set.

Wu Qing, et al. developed a hierarchical tensor transformation for compact data representation [30]. In order to show the existing multiscale structures, a multidimensional data set is transformed into a set of hierarchical signals.At each level, the signal is further divided into a set of smaller tensors. Furthermore a tensor approximation method is used to transform these smaller tensors. Their method has the advantage of progressive reconstruction. Their experimental results showed that the proposed method can provide higher compression ratios and quality when compared to wavelet transforms, wavelet packet transforms, and single-level tensor approximation.

[31] Sivalingam, Ravishankar, et al. proposed a sparse representation of positive defi-

nite matrices that preserves the inherent structure of the data unlike vectorization. They formulated the sparse decomposition of a positive definite matrix as a convex optimization problem. An efficient interior point algorithms can solve the formulated problem. Their experimental results showed the advantage of the new model for extending the sparsity-based algorithms for positive definite matrices.

1.4 Inconsistent scalable video streaming

Video traffic has been growing continuously especially during the last few years. According to CISCO's Visual Networking Index [32], by 2018 the sum of all forms of video traffic will be in a range of 80 to 90 percent. Another important factor in CISCO's index is the fact that by 2018 over half of all traffic will originate from non-PC devices. Furthermore, the mobile data will increase 11-fold and the traffic from wireless devices will exceed the traffic from wired devices.

The reported video traffic increase accounts for both consumer growth, the emerging of high quality video streaming, and the development of devices that can capture 360 degree videos as virtual reality contents. There are already many 4k, 8k videos, and virtual reality contents available on major Internet portals such as YouTube, Amazon, and Netflix.

These statistics support the fact that consumers are requesting video contents from various devices ranging from small cell-phones to large smart TVs. Furthermore the network connection quality of these devices may vary tremendously based on the region and consumer preferences. Consequently, a video streaming server has to ensure sending the best quality video based on the user device and network quality.

Scalable Video Coding (SVC) [33–35] is increasingly emerging as a viable solution to

address the aforementioned variability in network conditions and device capabilities. Some examples among the various types of SVC are temporal, spatial, and SNR methods.

In this thesis, we address the important case when some enhancement layers (e.g., within a GOP) are dropped while other enhancement layers can reach the receiver. This leads to an In-consistent Scalable Video (ISV) streaming. In such scenario, the decoder can exploit the higher quality/resolution SVC decoded pictures to assist in a super-resolution driven reconstruction of the lower quality/resolution pictures. The proposed framework was developed for spatial SVC. However, it can be extended and used within other forms of scalability, and in particular SNR SVC.

The proposed frame-work is applicable when network conditions change, and consequently, a streaming server can perform ISV streaming by choosing to drop some of the enhancement-layer frames. Under these circumstances, the frames with dropped layers need to be scaled up to the display size. A simple interpolation method can be used for ISV. However the final frame would be blurry and missing the high frequency data. The proposed framework, which was applied in block-wise approch, employes an example based super-resolution method to scale up those frames with a better quality. The block partition structure was drived from the quad tree block structure of the encoder. Our experimental results show that this partitioning strategy can improves the reconstruction quality when compared to using uniform size blocks.

The proposed framework was implemented within VP9 spatial SVC [36]. VP9 is an open source video codec which has been developed and supported by Google. The source code is available as part of Googles WebM project and can be obtained from [37].

1.5 ISV related work

Many prior efforts have been proposed for single image super-resolution with high quality reconstruction results [38]. The better quality comes at the expense of higher computation complexity. This complexity makes it difficult to employ these methods for videos.

In [39], the authors proposed a fast super-resolution method based on multi-frames that can be used for video. However, the complexity for even a low resolution video was high. With the emergence of high resolution videos (HD and above), using this type of super-resolution is not applicable. On the other hand, many of the classical image super-resolution methods do not perform well on the high-frequency component [40]. This is an important drawback especially for high resolution videos since this type of videos tend to have more high frequency components.

A video super-resolution framework has been proposed in [41–43] based on coding some of the frames at a high resolution (key-frames) to be used as the dictionary. The non-key-frames are super resolved with the high frequency data found in the dictionary. In our work, we are focusing on estimating the high frequency component as a mean of super resolving the video. We use a similar example based super-resolution as in [41] to scale up the frames whose enhancement layer is dropped.

Freeman William T., et al. proposed a fast algorithm for one-pass super-resolution that is based on a training-based super-resolution algorithm [44]. The algorithm perform a nearest-neighbor search in the training set to find a feature vector that is obtained from each patch of local image data. Their experimental results showed the application of the proposed method in natural image quality enhancement.

Xiong, Zhiwei, et al. proposed a robust single image super-resolution method to increase

the quality of low quality web images and videos [45]. Their method combines adaptive regularization and learning-based super-resolution. During the iterative regularization process, the image energy change characteristics is analyzed. This analysis provide the convergence property of the energy change ratio. It leads to the regularization parameter which balance quality enhancement and primitive components preservation. Also, the adaptive regularization improve the pair matching accuracy in learning-based super-resolution. Their experimental results showed that the proposed method can enhance the visual quality of the degraded web images and videos.

Shan, Qi, et al. proposed an upsampling method for image and video resolution enhancement that preserve the essential structural information [46]. Their proposed feedback-control framework recovers the high frequency details without additional local structure constraints. They argue that the proposed method is independent of the quality and number of the selected examples. Their experimental results showed that the proposed method can achieve high quality images enhancement without observable artifacts. Also they argue that the method can extends to video upsampling while maintaining the temporal coherence.

1.6 Summary of contributions

The contributions made in this thesis are summarized as follows:

1. Developing a Progressive Canonical-decomposition Parallel-factor (PCP) tensor-decomposition for representing and coding visual data ensembles efficiently.

2. Developing an adaptive tree partitioning algorithm for sub dividing an input tensor into a set of smaller size blocks. For three and four dimensional tensors, it translates to octree and hextree respectively. For n dimensional tensors it translates to 2^n-ary

tree.

3. Formulating the problem of finding the global optimal number of PCP eigenfibers for the blocks of a visual data tensor as an optimization problem.

4. Developing a greedy solution to solve the above optimization problem.

5. Developing a complete coding framework, based on the proposed PCP tensor decomposition.

6. Applying the proposed tensor coding framework on three types of data sets which are: hyperspectral/multispectral images, bio-metric images, and low motion videos.

7. Developing an example based super-resolution framework to approximate the missing high frequency data.

8. Employing the proposed super-resolution framework, which was developed within Google's VP9 scalable video coding software, in an inconsistent scalable video streaming scenario.

1.7 Thesis organization

The thesis is organized in two main parts. In the first part, we propose a tensor coding framework for representation and coding of visual data ensembles. In the second part of the thesis, an example based super-resolution algorithm is proposed for enhancing the quality of scalable video streaming when some of the enhancement layers are dropped.

The remaining of the thesis is organized as follows:

- In chapter 2, a summery of CP tensor decomposition is presented. Next we define the problem and propose a progressive tensor decomposition. Later in the chapter we define the problem of finding the optimal number of rank-one tensors for all the sub blocks as an optimization problem and propose a greedy solution for it. The theories are presented for three dimensional tensors first for illustrating purpose. The result are extended for the general n dimensional tensors afterward.

- In chapter 3, the different modules of the tensor coding framework are presented. A uniform and adaptive tree partitioning methods are discussed. Furthermore decomposed vectors arrangement and coding are explained in details along with residual coding module. We also present the main properties of the tensor coding framework.

- In chapter 4 three different applications of the proposed tensor coding framework are presented. For each application, we compare our experimental results with some of the standard coding methods used within the application.

- Chapter 5 consist of the second part of this thesis. The ISV streaming problem is defined under which some of the enhancement layers of a scalable video are dropped due to poor network quality. Later in the chapter, an example based super-resolution algorithm is proposed to reconstruct the missing high frequency data of the video. Finally, the experimental results are shown and compared with simple interpolation methods.

- In chapter 6 we discuss the conclusions of the two main proposed frameworks in this thesis.

Chapter 2

Tensor decomposition for visual data

In this chapter a brief introduction on CP tensor decomposition is presented before we discuss the details of the proposed tensor decomposition. At each section, we first develop and illustrate the theories and the results for the three dimensional tensors. Then, we extend the results for a general case of the n dimensional tensors.

2.1 CP decomposition

CP decomposes a 3D tensor $\chi \in \mathbb{R}^{v_1 \times v_2 \times v_3}$ onto a number of rank-one tensors, each of which can be written as an outer product of three vectors [1]. The original tensor is approximated by summation of these rank-one tensors as shown in eq. (2.1).

$$\hat{\chi} = \sum_{r=1}^{R} \lambda_r \left(a_r^{(1)} \circ a_r^{(2)} \circ a_r^{(3)} \right) \tag{2.1}$$

Where $\circ$ is an outer product, and λ_r is a normalization factor such that to maintain an ℓ_2 unit norm for the vectors $a_r^{(d)}$, $d \in \{1, 2, 3\}$ [1]. Hence, the tensor χ is approximated using a linear combination of rank-one tensors $\left(a_r^{(1)} \circ a_r^{(2)} \circ a_r^{(3)} \right)$; and the rank parameter R is the number of rank-one tensors used to approximate χ. Figure 2.1 shows the reconstruction of a three dimensional tensor from a set of rank-one tensors.

The vectors $a_r^{(d)}$ can be arranged as column vectors of a corresponding set of matrices

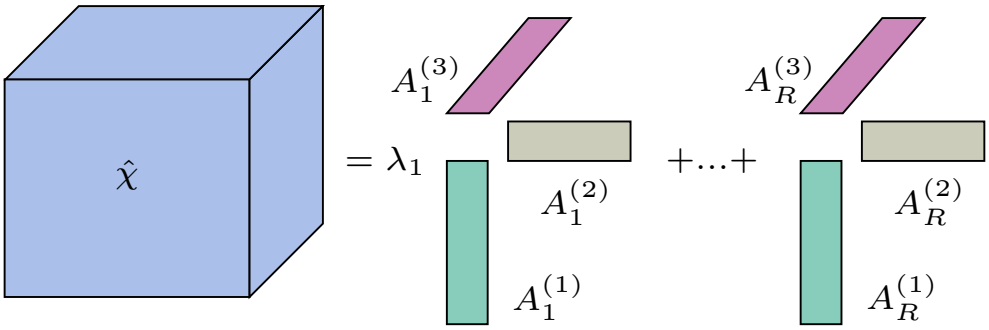


Figure 2.1: A three dimensional tensor reconstruction from a set of R rank-one tensors. Each 3D rank one tensor is an outer product of three vectors. The reconstructed tensor is a linear combination of these rank one tensors.

(i.e. $A^{(d)}$ where $d = 1, 2, 3$). For example, $A^{(1)} = [a_1^{(1)}\ a_2^{(1)} \ldots a_R^{(1)}]$ is a $v_1 \times R$ matrix. In general, $A^{(d)} \in \mathbb{R}^{v_d \times R}$. These matrices can be found using eq. (2.2).

$$A^{*(d)} = \underset{A^{(d)}}{argmin} \left\| X_{(d)} - A^{(d)} \left(A^{(d_1)} \odot A^{(d_2)} \right)^T \right\|_F \tag{2.2}$$

Where, $\odot$ is Khatri-Rao product, $d \in \{1, 2, 3\}$, $d_1 \in \{1, 2, 3\} - \{d\}$, and $d_2 \in \{1, 2, 3\} - \{d, d_1\}$. $X_{(d)}$ is a matrix that results from unfolding the tensor χ with respect to the d^{th} dimension. For example, $X_{(1)} \in \mathbb{R}^{v_1 \times (v_2 v_3)}$ is a matrix that results from unfolding the original tensor χ with respect to the first dimension (i.e. v_1). Similarly, $X_{(2)}$ and $X_{(3)}$ are the unfolded original tensors with respect to the second (v_2) and third (v_3) dimensions [1], respectively. For a given rank parameter R, the Alternative Least Square (ALS) [7] approach can be used to solve for the set of matrices in eq. (2.2). It solves for $A^{(1)}$ by fixing $A^{(2)}$ and $A^{(3)}$ and similarly for $A^{(2)}$ and $A^{(3)}$ as shown in (2.3).

$$A^{*(d)} = X_{(d)} \left(\mathcal{Z}^{(d)} \right)^T \left(\left(\mathcal{Z}^{(d)} \right)^T \left(\mathcal{Z}^{(d)} \right) \right)^{-1} \tag{2.3}$$

Where $\mathcal{Z}^{(d)} = A^{(d_1)} \odot A^{(d_2)}$ and, as before, $d \in \{1, 2, 3\}$, $d_1 \in \{1, 2, 3\} - \{d\}$, and $d_2 \in$

$\{1, 2, 3\} - \{d, d_1\}$.

An n-dimensional tensor $\chi \in \mathbb{R}^{v_1 \times \ldots \times v_n}$ is decomposed by CP onto a set of of rank-one tensors. An n dimensional rank-one tensor is equal to outer product of n vectors [1]. The original tensor is approximated by summation of these rank-one tensors as shown in eq. (2.4).

$$\hat{\chi} = \sum_{r=1}^{R} \lambda_r \left(a_r^{(1)} \circ \ldots \circ a_r^{(n)} \right) \tag{2.4}$$

The vectors $a_r^{(d)}$ can be arranged as column vectors of a corresponding set of matrices $A^{(d)} \in \mathbb{R}^{v_d \times R}$ where $d = 1, 2, \ldots, n$ and $A^{(d)} = [a_1^{(d)}\ a_2^{(d)} \ldots a_R^{(d)}]$. Finding these matrices can be formulated in an optimization problem as shown in eq. (2.5).

$$A^{*(d)} = \underset{A^{(d)}}{argmin} \left\| X_{(d)} - A^{(d)} \left(A^{(1)} \odot \ldots \odot A^{(d-1)} \odot A^{(d+1)} \odot \ldots \odot A^{(N)} \right)^T \right\|_F \tag{2.5}$$

Where, $\odot$ is Khatri-Rao product, $d = 1, 2, \ldots, n$. $X_{(1)} \in \mathbb{R}^{v_d \times (v_1 v_3 \ldots v_{d-1} v_{d+1} \ldots v_n)}$ is the unfolded tensor χ with respect to the d^{th} dimension. For a given rank parameter R, the ALS approach can solve for the set of matrices in eq. (2.5). It solves for $A^{(1)}$ by fixing $A^{(2)}$, $A^{(3)}$, ..., and $A^{(n)}$ and similarly for $A^{(2)}, \ldots, A^{(n)}$ and so on as shown in eq. (2.6).

$$A^{*(d)} = X_{(d)} \left(\mathcal{Z}^{(d)} \right)^T \left(\left(\mathcal{Z}^{(d)} \right)^T \left(\mathcal{Z}^{(d)} \right) \right)^{-1} \tag{2.6}$$

Where $\mathcal{Z}^{(d)} = A^{(d_1)} \odot \ldots \odot A^{(d-1)} \odot A^{(d+1)} \ldots \odot A^{(n)}, d = 1, 2, \ldots, n$.

2.2 CP decomposition for visual data

A straightforward approach for tensor-based representation of visual data is to directly employ the CP decomposition onto an original data set. However, such straightforward approach does not result in an efficient coding method. There are three main disadvantages in using CP decomposition for visual data coding which are:

1. Independent of the form of tensor decomposition used, the rank parameter R should not be fixed throughout the tensor decomposition of the entire visual data tensor. In particular, the value of R directly influences the rate and efficiency of the original tensor representation since it determines the number of rank-one tensors used for this representation. Meanwhile, different parts of the multimedia tensor have different levels of spatial and temporal details. On the other hand, it is known that finding the tenor rank is an NP-complete problem [47]. A primitive way of finding the rank is to start at one and gradually increase the rank, until the optimal rank is found. However, in the case of CP decomposition this will result in a high time complexity.

2. CP decomposition requires all the composing rank-one tensors in order to approximate the original visual data tensor χ. This is true even if the actual rank of the tensor is smaller than the number of the composing rank-one tensors. In other words, if part of the decomposed data are missing, the reconstructed result will be degraded significantly. This will result in a non-progressive reconstruction in which all the eigenfibers are required prior to the reconstruction.

3. CP decomposition result in a set of decomposed vectors with real number values. The reconstruction quality is highly sensitive to small approximation in the decomposed

values. As the result, there should be enough number of bytes allocated for storing the decomposed values. This will decrease the storage efficiency especially considering the fact that most of the visual data are 8-bit and requires only one byte to store each point. Furthermore, because of the precision sensibility, the CP decomposed values are not noise robust. The noisy decomposed values will result in poor quality reconstruction.

We propose a CP-based decomposition for visual data that can address the before mentioned shortcomings of the CP decomposition. The details of the proposed decomposition is presented in the next section.

2.3 Progressive CP decomposition

Similar to CP, the approximated tensor ($\hat{\chi}$) under PCP is a sum of rank-one tensors. However, the PCP decomposition results in different rank-one tensors and corresponding vectors from what is generated by CP. We also use a different normalization as explained further below. To emphasize the difference between the two schemes, we express the PCP decomposition using different notations for rank-one tensors and normalization parameters as expressed in eq. (2.7).

$$\hat{\chi} = \sum_{r=1}^{R} \beta_r (b_r^{(1)} \circ b_r^{(2)} \circ b_r^{(3)}) \tag{2.7}$$

Under PCP, $R \in \{1, 2, \ldots, R_{max}\}$, where R_{max} is the maximum possible number of rank one tensors that are available for approximating the original tensor. It limits the total bitrate by limiting the number of eigenfibers to be coded. Similar to CP, $b_r^{(d)}$ can be arranged as column vectors of a corresponding set of matrices $B^{(d)}$ where $d = 1, 2, 3$. Under PCP though, there are two key differences.

The first difference is that the eigenfibers $b_r^{(d)}$ are computed individually as shown in eq. (2.8).

$$b_r^{*(d)} = \underset{b_r^{(d)}}{argmin} \left\| \left(X_{(d)} - \sum_{k=0}^{r-1} {X'}_{(d),k} \right) - b_r^{(d)} \left(z_r^{(d)} \right)^T \right\|_F \tag{2.8}$$

Where $z_r^{(d)} = b_r^{(d_1)} \odot b_r^{(d_2)}$, $d \in \{1,2,3\}$, $d_1 \in \{1,2,3\} - \{d\}$, and $d_2 \in \{1,2,3\} - \{d, d_1\}$. ${X'}_{(d),k}$ is the k^{th} rank-one unfolded-tensor over dimension d, and $k \in \{0,1,2,\ldots,R\}$. ${X'}_{(d),0} = 0$, ${X'}_{(d),k} = \beta_k b_k^{(d)} \left(z_k^{(d)} \right)^T$. Note that the vector product $b_r^{(d)} \left(z_r^{(d)} \right)^T$ in eq. (2.8) results in a matrix of size $v_d \times v_{d_1} v_{d_2}$, which is the size of the unfolded-tensor matrix $X_{(d)}$.

For a given rank parameter R_{max}, we modify the ALS approach to solve the minimization problem in eq. (2.8). Similar to CP, we fix $b_r^{(2)}$ and $b_r^{(3)}$ and solve for $b_r^{(1)}$; and similarly for $b_r^{(2)}$ and $b_r^{(3)}$ as in eq. (2.9).

$$b_r^{*(d)} = \left(X_{(d)} - \sum_{k=0}^{r-1} {X'}_{(d),k} \right) \left(z_r^{(d)} \right)^T \left(\left(z_r^{(d)} \right)^T z_r^{(d)} \right)^{-1} \tag{2.9}$$

At each iteration r we calculate the error $\epsilon_r = MSE\,(\chi - \hat{\chi}_r)$, where $\hat{\chi}_r$ is obtained from eq. (2.7). The PCP decomposition is applied in block-wise approach. The errors of all 3D-blocks are used in a procedure of finding the optimal global solution as will be discussed later. The solution may add another rank-one tensor to approximate the residual $\chi - \hat{\chi}_r$ in the next iteration. This approximation results in a progressive decomposition of χ as shown in figure 2.2.

Figure 2.3 shows an example of uniform partitioning of the tensor and the PCP decomposition on each block. As mentioned before, the value of R for each block could be different

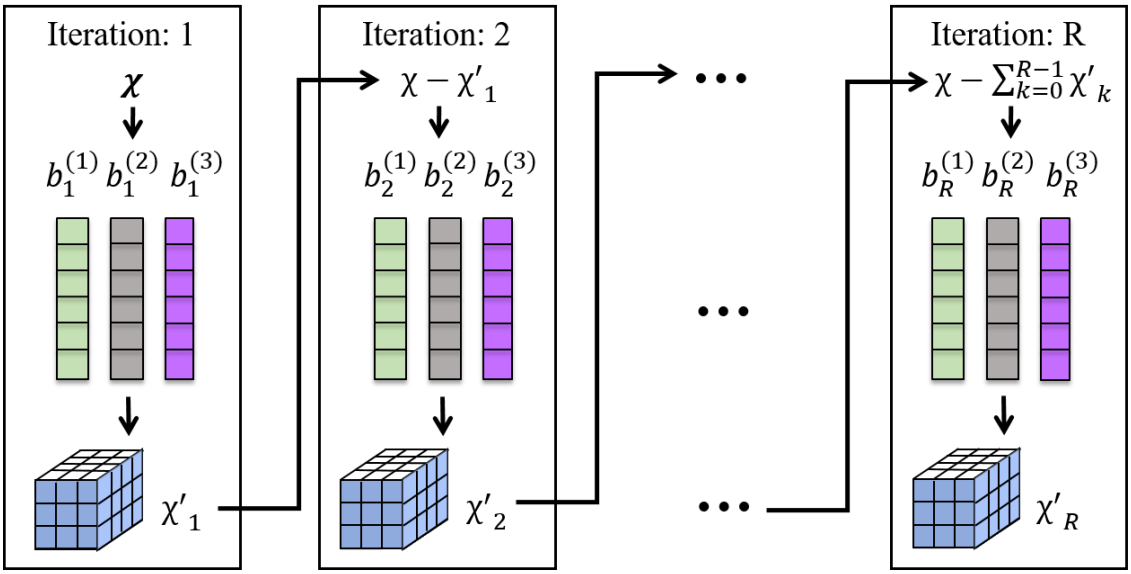


Figure 2.2: PCP decomposition progressive reconstruction of χ. At the first iteration the input tensor is approximated with only one rank one tensor. Then at each iteration a rank one tensor is added to approximate the residual.

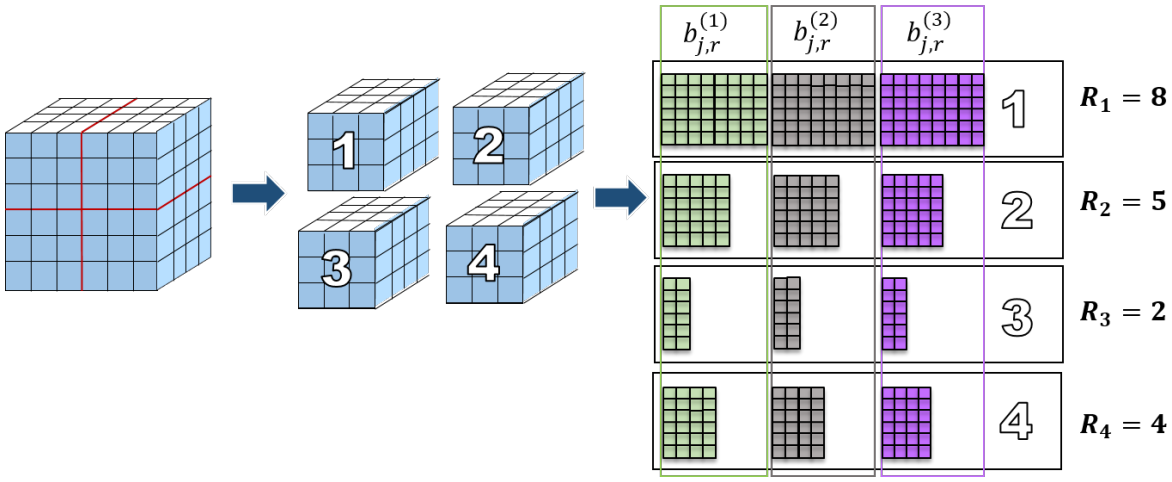


Figure 2.3: An example of uniform partitioning of an 3D tensor and the PCP decomposition on each block. PCP allocates different value of R_j for each block based on the amount of the information in that particular block.

based on its rank. The problem of finding the optimal number of rank-one tensors for each 3D-block is addressed in the next section.

The second difference between CP and the proposed PCP is the following. As mentioned above, under CP, the rank-one tensors are normalized by maintaining an ℓ_2 unit-norm vectors. This is captured through λ_r in eq. (2.1). Under PCP, we employ an ℓ_∞ norm instead. This leads to the following:

- The normalizing parameter β_r captures the maximum magnitudes of the entries of the corresponding vectors $b_r^{(d)}$, $d = 1, 2, 3$.
- The vectors $b_r^{(d)}$ have normalized values between -1 and $+1$. As we show later, the proposed PCP lends itself to more efficient coding when compared to traditional CP.

PCP approximates an n dimensional tensor χ as a linear combination of a set of rank one tensors as shown in eq. (2.10).

$$\hat{\chi} = \sum_{r=1}^{R} \beta_r (b_r^{(1)} \circ \ldots \circ b_r^{(n)}) \tag{2.10}$$

Where $R \in \{1, 2, \ldots, R_{max}\}$, where R_{max} is the maximum possible number of rank-one tensors that can be used. $b_r^{(d)}$ are column vectors of a corresponding set of matrices $B^{(d)}$ where $d = 1, 2, \ldots, n$. The PCP eigenfibers $b_r^{(d)}$ can be computed as follows:

$$b_r^{*(d)} = \underset{b_r^{(d)}}{argmin} \left\| \left(X_{(d)} - \sum_{k=0}^{r-1} {X'}_{(d),k} \right) - b_r^{(d)} \left(z_r^{(d)} \right)^T \right\|_F \tag{2.11}$$

Where $z_r^{(d)} = b_r^{(1)} \odot \ldots \odot b_r^{(d-1)} \odot b_r^{(d+1)} \odot \ldots \odot b_r^{(n)}$, $d \in \{1, 2, \ldots, n\}$. ${X'}_{(d),k}$ is the k^{th} rank-one unfolded-tensor over dimension d. ${X'}_{(d),0} = 0$, ${X'}_{(d),k} = \beta_k b_k^{(d)} \left(z_k^{(d)} \right)^T$. The

vector product $b_r^{(d)} \left(z_r^{(d)}\right)^T$ in eq. (2.11) results in a matrix of size $v_d \times v_1 v_2 \dots v_{d-1} v_{d+1} \dots v_N$, which is the size of the unfolded-tensor matrix $X_{(d)}$.

For a given rank parameter R_{max}, ALS approach can solve the minimization problem in eq. (2.11). We fix $b_r^{(2)}, b_r^{(3)}, \dots, b_r^{(n)}$ and solve for $b_r^{(1)}$; and similarly for the other eigenfibers as shown in eq. (2.12).

$$b_r^{*(d)} = \left(X_{(d)} - \sum_{k=0}^{r-1} {X'}_{(d),k} \right) \left(z_r^{(d)}\right)^T \left(\left(z_r^{(d)}\right)^T z_r^{(d)} \right)^{-1} \tag{2.12}$$

2.4 The Rank-Distortion optimization problem

In the proposed tensor coding framework, a 3D visual data tensor is partitioned into a set of 3D sub-tensor blocks. Block j has s_j elements where $s_j = v_{1j} v_{2j} v_{3j}$. PCP decomposes a given 3D block, which is indexed by j, onto R_j rank-one tensors, each of which has three eigenfibers. The total number of elements in the PCP rank-one decomposition for 3D block j is s'_j where $s'_j = R_j(v_{1j} + v_{2j} + v_{3j})$. Assuming that we use the same precision for the original visual data pixels and for the elements of the PCP decomposition (e.g., eight bits/pixel and eight bits/element in an eigenfiber), then using the eigenfibers instead of the original block will result in the compaction ratio shown in eq. (**??**).

$$\frac{s_j}{s'_j} = \frac{v_{1j} v_{2j} v_{3j}}{\left(R_j \left(v_{1j} + v_{2j} + v_{3j}\right)\right)} \tag{2.13}$$

We would like to increase the block size and minimize the value of R_j to have larger compaction ratio. However, a larger block potentially has higher rank and requires a larger value of R_j to be coded with acceptable quality. This will decrease the compaction ratio.

One option is to have the compaction ratio larger than some regularization parameter $\gamma > 1$. This leads to an upper bound for R_j that keeps the compaction ratio larger than γ as shown in eq. (2.14).

$$R_j < \frac{1}{\gamma}\left(\frac{v_1 v_2 v_3}{v_1 + v_2 + v_3}\right) \tag{2.14}$$

We employ the inequality in eq. (2.14) as a constraint for our optimization problem. Another constraint is related to the reconstruction error. We use the average error constraint shown in eq. (2.15) for reconstructing the original tensor.

$$\frac{1}{N}\sum_{j=1}^{N}\left\|\chi_j - \sum_{i=1}^{R_j} b_{i,j}^{(1)} \circ b_{i,j}^{(2)} \circ b_{i,j}^{(3)}\right\|_F \leq \epsilon_{max} \tag{2.15}$$

For a given data set with N 3D tensor blocks, the goal is to find the global optimum R, where R is a vector of dimension N. Each entry of $R = (R_1, R_2, \ldots, R_N)$ corresponds to the number of rank-one tensors which are used to reconstruct a 3D-block of the data set. We formulate the rank-distortion optimization problem to find the global optimum R as in eq. (2.16).

$$\begin{gathered} min\left(\sum_{j=1}^{N} R_j\right) \quad s.t. \\ \frac{1}{N}\sum_{j=1}^{N}\left\|\chi_j - \sum_{i=1}^{R_j} b_{i,j}^{(1)} \circ b_{i,j}^{(2)} \circ b_{i,j}^{(3)}\right\|_F \leq \epsilon_{max} \\ \sum_{j=1}^{N} R_j < \left(\frac{1}{\gamma}\sum_{j=1}^{N}\frac{v_{1j} v_{2j} v_{3j}}{v_{1j} + v_{2j} + v_{3j}}\right) \triangleq R_{max} \end{gathered} \tag{2.16}$$

Where ϵ_{max} is the average, overall acceptable error. The second inequality in eq. (2.16) captures an upper bound for the total number of eigenfibers that can be used. Note that If

all the blocks have the same size, R_{max} can be simplified to $((v_1 v_2 v_3)N) \,/\, (\gamma(v_1 + v_2 + v_3))$.

An n dimensional block j has s_j elements where $s_j = v_{1j} v_{2j} \ldots v_{nj}$. PCP decomposes this block onto R_j rank-one tensors, each of which has n eigenfibers. The total number of elements in the PCP rank-one decomposition for block (j) is s'_j where $s'_j = R_j(v_{1j} + v_{2j} + \ldots + v_{nj})$. Using the eigenfibers instead of the original block will result in the following compaction ratio $s_j/s'_j = v_{1j} v_{2j} \ldots v_{nj} / \left(R_j \left(v_{1j} + v_{2j} + \ldots + v_{nj}\right)\right)$. The upper bound for R_j that keeps the compaction ratio larger than γ is $R_j < \frac{1}{\gamma} \left(\left(v_{1j} v_{2j} \ldots v_{nj}\right) / \left(v_{1j} + v_{2j} + \ldots + v_{nj}\right)\right)$. For an n dimensional tensor, we use the average error constraint shown in eq. (2.17) for reconstructing the original tensor.

$$\frac{1}{N} \sum_{j=1}^{N} \left\| \chi_j - \sum_{i=1}^{R_j} b_{i,j}^{(1)} \circ b_{i,j}^{(2)} \circ \ldots \circ b_{i,j}^{(n)} \right\|_F \leq \epsilon_{max} \tag{2.17}$$

For a given n dimensional tensor with N n dimensional sub-tensor blocks, the goal is to find the global optimum vector R, where R is a vector of dimension N. Each entry of $R = (R_1, R_2, \ldots, R_N)$ corresponds to the number of rank-one tensors which are used to reconstruct an n dimensional block of the data set. The rank-distortion optimization problem to find the global optimum R is as shown in eq. (2.18).

$$\begin{gathered} min \left(\sum_{j=1}^{N} R_j \right) \quad s.t. \\ \frac{1}{N} \sum_{j=1}^{N} \left\| \chi_j - \sum_{i=1}^{R_j} b_{i,j}^{(1)} \circ b_{i,j}^{(2)} \circ \ldots \circ b_{i,j}^{(n)} \right\|_F \leq \epsilon_{max} \\ \sum_{j=1}^{N} R_j < \left(\frac{1}{\gamma} \sum_{j=1}^{N} \frac{v_{1j} v_{2j} \ldots v_{nj}}{v_{1j} + v_{2j} + \ldots + v_{nj}} \right) \triangleq R_{max} \end{gathered} \tag{2.18}$$

As mentioned earlier in the case of a 3D tensor, ϵ_{max} represents the average overall

acceptable error. The second inequality is the upper bound for the total number of eigenfibers that can be used. If the tensor is partitioned uniformly to a set of equal size blocks, R_{max} can be simplified to $((v_1 v_2 \dots v_n)N) \,/\, (\gamma(v_1 + v_2 + \dots + v_n))$.

A solution to this optimization problem can be found by searching for the optimum R_j which satisfies the constraints. The search is done by starting from $R_j = 1,\ j = 1, \dots, N$ and increasing R_j of block j that meets a certain criterion gradually until the constraints are satisfied. The details are presented in the next section.

2.5 Rank-Distortion optimization problem solution

A greedy algorithm is developed in this section to solve eq. (2.16). For an 3D input tensor that has been partitioned to N 3D tensor blocks, the algorithm starts initially by $R = \overrightarrow{\mathbf{1}}$. This initialization is along with the fact that each 3D tensor block should be represented at least with one rank-one tensor. Furthermore, E_j is defined as block j error decrement if R_j increased by one as shown in eq. (2.19).

$$E_j = \epsilon_{j,R_j} - \epsilon_{j,R_j+1} \tag{2.19}$$

Where ϵ_{j,R_j} is the reconstruction error when R_j rank one tensors are used in block j reconstruction. Initially $E_j = \epsilon_{j,1} - \epsilon_{j,2}$ for $j = 1, 2, \dots, N$. Iteratively we find block j^* that has the maximum E_{j^*} and increase its corresponding R_{j^*} by one. This greedy choice provides the largest possible error reduction at each iteration. After R_{j^*} is incremented for block j^*, the inequalities are checked. If the first inequality is satisfied or the second inequality is not satisfied, the algorithm will be terminated. The PCP tensor decomposition algorithm for 3D tensors is shown in algorithm 1.

Algorithm 1 3D tensor decomposition with Rank-Distortion Optimization

Input: A set of 3D tensor blocks (i.e. $\chi_j,\ j = 1, \ldots, N.$), and γ

Output: A set of eigenfibers to represent the input tensor.

$R = \overrightarrow{\mathbf{1}}$

Find $b_{r,j}^{(1)},\ b_{r,j}^{(2)}$, and $b_{r,j}^{(3)}$ for $r = 1, 2$ and $j = 1 \ldots N$ from eq. (2.9).

$\epsilon_{j,r} = \left\| \chi_j - \sum_{i=1}^{r} b_{i,j}^{(1)} \circ b_{i,j}^{(2)} \circ b_{i,j}^{(3)} \right\|_F$

for $r = 1, 2$ and $j = 1 \ldots N$ **do**

$E_j = \epsilon_{j,1} - \epsilon_{j,2}.\ \ for\ j = 1 \ldots N$

end for

while first inequality in eq. (2.16) is not satisfied and the second inequality in eq. (2.16) is satisfied **do**

$j^* = argmax_j\, E_j$

$r = R_{j^*} + 1$

Find $b_{r,j^*}^{(1)},\ b_{r,j^*}^{(2)}, b_{r,j^*}^{(3)}$ from eq. (2.9)

$\epsilon_{j^*,r} = \left\| \chi_{j^*} - \sum_{i=1}^{r} b_{i,j^*}^{(1)} \circ b_{i,j^*}^{(2)} \circ b_{i,j^*}^{(3)} \right\|_F$

$\epsilon_{j^*,r+1} = \left\| \chi_{j^*} - \sum_{i=1}^{r+1} b_{i,j^*}^{(1)} \circ b_{i,j^*}^{(2)} \circ b_{i,j^*}^{(3)} \right\|_F$

$E_{j^*} = \epsilon_{j^*,r+1} - \epsilon_{j^*,r}$

$R_{j^*} = r$

end while

For an n dimensional tensor, the optimization problem shown in eq. (2.18) can be solved with the greedy algorithm shown in algorithm 2. Similar to the algorithm for a 3D tensor, the algorithm starts initially by $R = \overrightarrow{\mathbf{1}}$. It iteratively finds block j^* that has the maximum E_{j^*} and increase its corresponding R_{j^*} by one. The time complexity of calculating the error decrements at each iteration can be reduced by storing them in a vector. The vector $\mathbf{E}$ stores the values E_j for $j = 1, 2, \ldots, N$. At each iteration, only the value of the entry point j^* needs to be updated.

Algorithm 2 n dimensional tensor decomposition with Rank-Distortion Optimization

Input: A set of n dimensional tensor blocks (i.e. $\chi_j,\ j = 1, \ldots, N.$), and γ

Output: A set of eigenfibers to represent the input tensor.

$R = \overrightarrow{\mathbf{1}}$

Find $b_{r,j}^{(1)},\ b_{r,j}^{(2)}, \ldots, b_{r,j}^{(n)}$ for $r = 1, 2$ and $j = 1 \ldots N$ from eq. (2.12).

$\epsilon_{j,r} = \left\| \chi_j - \sum_{i=1}^{r} b_{i,j}^{(1)} \circ b_{i,j}^{(2)} \circ \ldots \circ b_{i,j}^{(n)} \right\|_F$

for $r = 1, 2$ and $j = 1 \ldots N$ **do**

$E_j = \epsilon_{j,1} - \epsilon_{j,2}.\ \ for\ j = 1 \ldots N$

end for

while first inequality in eq. (2.18) is not satisfied and the second inequality in eq. (2.18) is satisfied **do**

$j^* = argmax_j\, E_j$

$r = R_{j^*} + 1$

Find $b_{r,j^*}^{(1)},\ b_{r,j^*}^{(2)}, \ldots, b_{r,j^*}^{(n)}$ from eq. (2.12)

$\epsilon_{j^*,r} = \left\| \chi_{j^*} - \sum_{i=1}^{r} b_{i,j^*}^{(1)} \circ b_{i,j^*}^{(2)} \circ \ldots \circ b_{i,j^*}^{(n)} \right\|_F$

$\epsilon_{j^*,r+1} = \left\| \chi_{j^*} - \sum_{i=1}^{r+1} b_{i,j^*}^{(1)} \circ b_{i,j^*}^{(2)} \circ \ldots \circ b_{i,j^*}^{(n)} \right\|_F$

$E_{j^*} = \epsilon_{j^*,r+1} - \epsilon_{j^*,r}$

$R_{j^*} = r$

end while

Chapter 3

Tensor coding framework

PCP is the core transform used in the tensor coding framework. Other tasks of the framework are handled in various modules. Among this tasks are tensor partitioning, eigenfibers arrangements and coding, header data management, etc. Figure 3.1 shows the diagram of the tensor coding framework with its various modules. The details of these modules are presented in this chapter.

3.1 Tensor partitioning

The first step of the tensor coding framework is the input tensor partitioning. It helps to balance the rate in the optimization problem in eq. (2.2) and eq. (2.5). It also balance the complexity based on the block size. We employed two types of partitioning:

- **Uniform partitioning**: the input tensor is divided into a set of equal size blocks. The blocks have the same dimension as the input tensor.
- **Adaptive tree partitioning**: initially the input tensor is partitioned into a set of equal size blocks, then for each block a recursive algorithm is employed to subdivide it if required. The decision of whether to subdivide a block or not is made based on a criteria. A tree structure represents the partitioning structure. For 3D and 4D input tensors, the partitioning can be represented by octree and hextree correspondingly. A higher dimension tensor partitioning can be represented by an 2^n-ary tree where n is

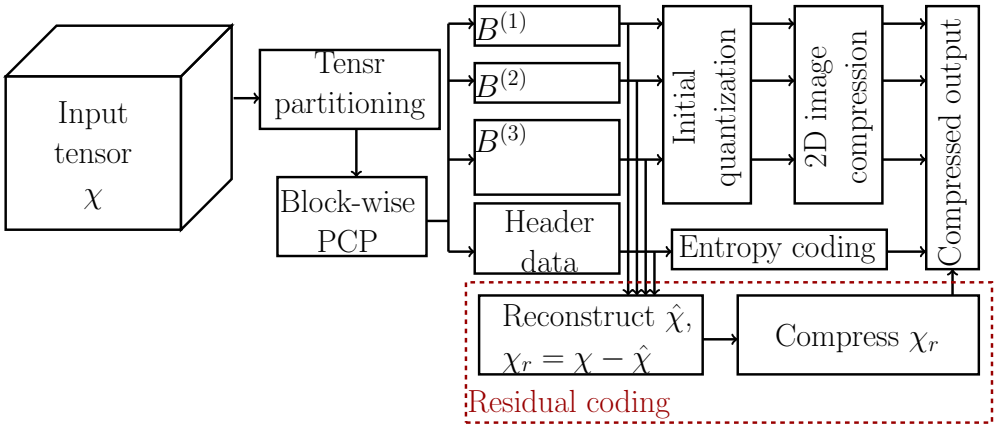


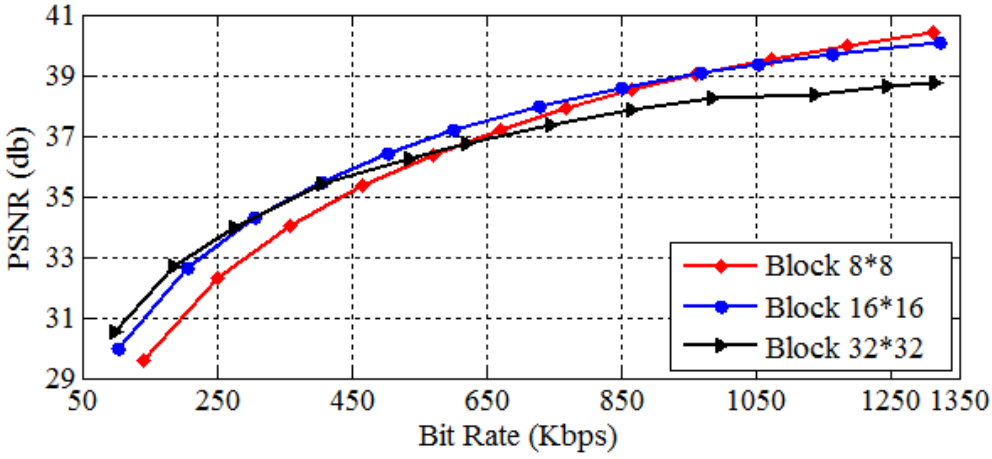
Figure 3.1: Tensor coding framework for visual data coding and representation. The framework consist of various modules that operate at different stages. Among these modules are tensor partitioning, decomposition, and eigenfibers coding. It also shows the residual coding process.

the tensor dimension. Similar to the uniform partitioning, each block has the same dimension as the input tensor. The details of the algorithm will be discussed later.

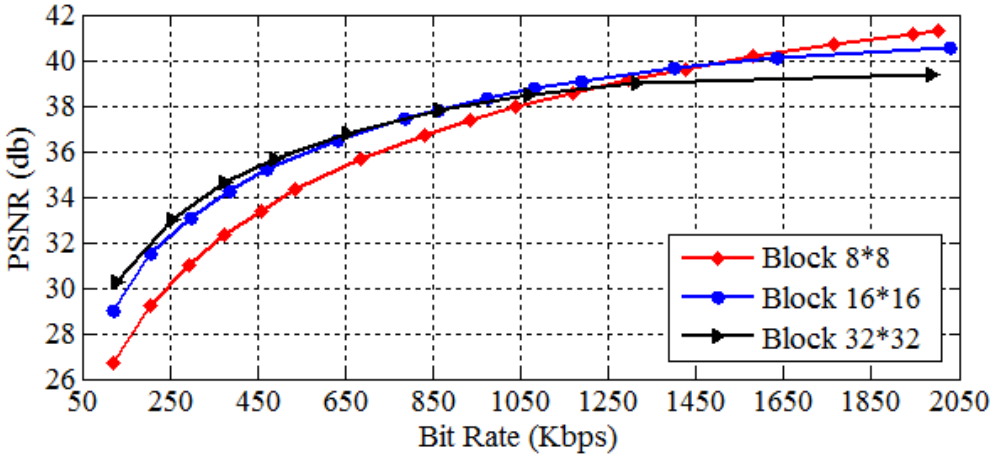
3.1.1 Uniform partitioning

Uniform partitioning divides the input tensor into a set of equal size blocks. Intuitively, we would like to increase the block size to have a larger compaction ratio. However, increasing the block size may increase the required number of rank-one tensors to code it.

Suppose a 3D region covered by a block of size $v_1 \times v_2 \times v_3$ requires R rank-one tensors to be coded. Also, if the same region covered by eight blocks of size $\frac{v_1}{2} \times \frac{v_2}{2} \times \frac{v_3}{2}$ requires R' rank-one tensors to be coded. Here R' is the total number of rank-one tensors for all the eight 3D blocks. The compaction ratio for both scenarios is as follows, respectively: $C = \frac{v_1 v_2 v_3}{R(v_1+v_2+v_3)}$ and $C' = \frac{1}{4}\frac{v_1 v_2 v_3}{R'(v_1+v_2+v_3)}$. The larger block results in better compression when $C > C'$ and consequently $R < 4R'$. This implies that if it is possible to code a region using the large block size while the error is not larger than when it is coded with the small block size and if $R < 4R'$, the large block size would result in higher compression than the



(a)

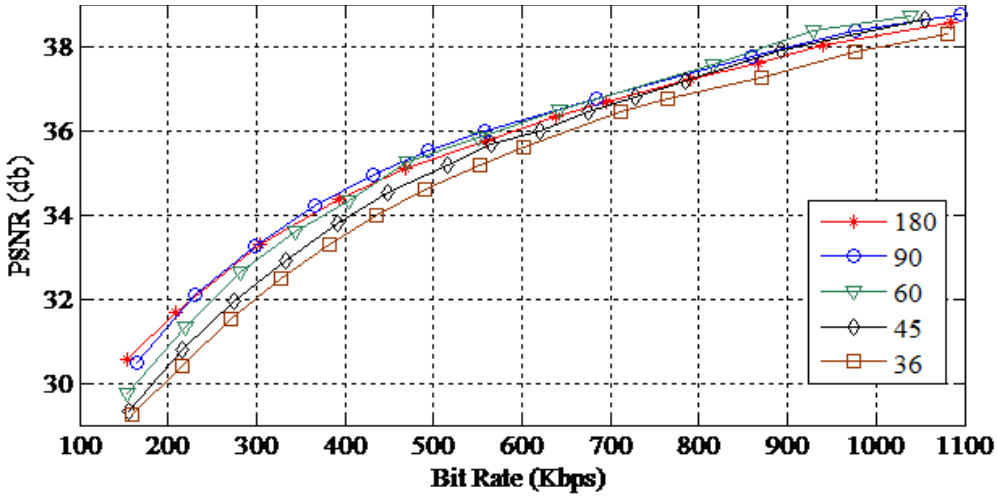


(b)

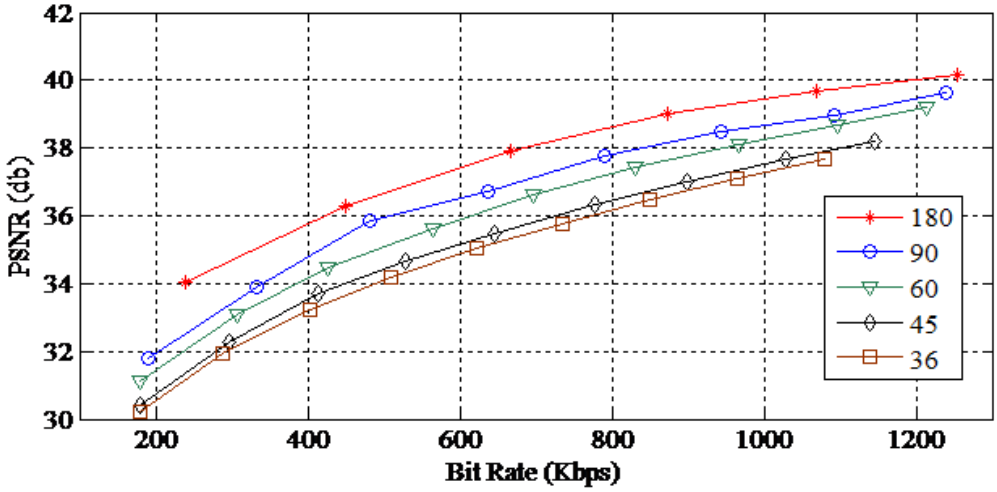
Figure 3.2: PSNR vs bitrate plots of (a) Silent and (b) Container CIF videos employing three different block sizes. Larger block sizes result in higher PSNR at lower bitrates however at some bitrate point they cross over.

small one. More generally, for a block size ratio d (i.e. $d = \left(v_1/v_1'\right) = \left(v_2/v_2'\right) = \left(v_3/v_3'\right)$), if $R < d^2 R'$ and the reconstruction error is relatively close; then, the larger block would be a better choice in terms of coding efficiency.

Figure 3.2 shows PSNR plots of Silent and Container CIF videos, encoded with three block sizes which are 8×8×180, 16×16×180, and 32×32×180. The plot shows that larger block sizes result in higher PSNR at low bitrates, however they cross over at some bitrate point.



(a)



(b)

Figure 3.3: PSNR plots of (a) Silent and (b) Container CIF videos employing blocks of different time dimension size. The plots show that if the video has high redundancy across time, then larger block sizes will result in higher PSNR.

In another experiment the size of the third dimension is varied while the spatial size is 16×16. The block size over the third dimension of size 180, 90, 60, 45, and 36 were evaluated. Figure 3.3 shows the PSNR results for Container and Silent video. The results show when the video has large number of blocks that do not change with time, expanding the block in time dimension will result in taking advantage of the redundancy and increases the PSNR. In the case of Container video that has linear and texture movement; at low bitrates, the larger

block sizes results in higher PSNR than smaller block sizes. However at some bitrate points, they crossover. Using variable block size for different regions of a video has the advantage of obtaining the highest possible PSNR.

Similar block size analysis can be done for an n dimensional tensor. Assuming that an n dimensional region covered by a block of size $v_1 \times v_2 \times \ldots \times v_n$ requires R rank-one tensors to be coded, if the same region covered by d^n blocks of size $\frac{v_1}{d} \times \frac{v_2}{d} \times \ldots \times \frac{v_n}{d}$ requires R' rank-one tensors to be coded where R' is the total number of rank-one tensors for all the d^n n dimensional blocks. The compaction ratio for both scenarios is as follows, respectively: $C = \frac{v_1 v_2 \ldots v_n}{R(v_1+v_2+\ldots+v_n)}$ and $C' = \frac{1}{d^{(n-1)}} \frac{v_1 v_2 \ldots v_n}{R'(v_1+v_2+\ldots+v_n)}$. The larger block results in better compression when $C > C'$ and consequently $R < d^{(n-1)} R'$.

3.1.2 Adaptive tree partitioning

Adaptive tree subdivide the input tensor into a set of variable size sub-blocks. In the case of a 3D tensor, it translate to octree, for 4D tensor it translate to hextree and in general for an n dimensional tensor it translate to 2^n-ary tree. For the illustration purpose the partitioning algorithm is developed for octree first and the result is extended afterward.

Octree is the 3D analogous of quadtree in which a 3D block is recursively subdivided into eight adjacent disjoint 3D sub-blocks [48]. Figure 3.4 illustrates an example 3D block subdivision and its corresponding octree representation.

The octree divides a $2^{v_1} \times 2^{v_2} \times 2^{v_3}$ block into a $v_1 - v_1^{(l_0)} + 1$ levels tree. Where $2^{v_1^{(l_0)}} \times 2^{v_2^{(l_0)}} \times 2^{v_3^{(l_0)}}$ is the smallest possible block. Depending on the application, it is desirable to have the flexibility of allocating different size along each dimensions. For example, for a hyperspectral images tensor, since the nature of the third dimension is different than the spatial dimensions, we would like to allocate different size along the third dimension.

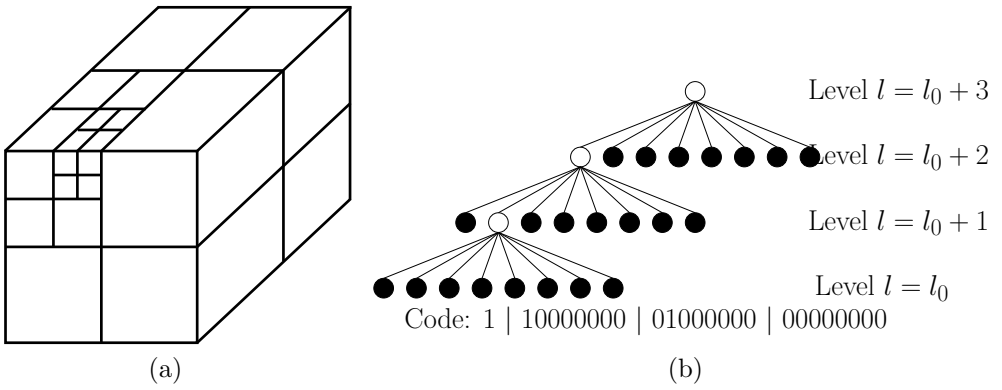


Figure 3.4: (a) An example of 3D block recursive subdivision, and (b) its corresponding octree representation. The leaf node indicate no subdivision while the branching node subdivided and it has eight children. A leaf is represented by 0 and a branching node is represented by 1. The octree representation code is also shown.

However the tree level is similar in all directions, $v_1 - v_1^{(l_0)} + 1 = v_2 - v_2^{(l_0)} + 1 = v_3 - v_3^{(l_0)} + 1$. The blocks at level l have the size $2^{v_1^{(l)}} \times 2^{v_2^{(l)}} \times 2^{v_3^{(l)}}$ where$0 \leqslant v_d^{(l_0)} \leqslant v_d^{(l)} \leqslant v_d, d = 1, 2, 3$. Each node can be a leaf or it can be divided to eight $2^{(v_1^{(l)}-1)} \times 2^{(v_2^{(l)}-1)} \times 2^{(v_3^{(l)}-1)}$ sub-blocks.

Similar to quadtree, the octree can be represented with a series of bits in which 0 indicates leaf and 1 indicates branching node. The top down approach is used for subdividing the blocks. It starts with a maximum size block and divide it if a criteria is met. The procedure is applied to the eight sub-blocks recursively until the sub-blocks do not require further division or they reach the maximum tree level l_0.

Since we are trying to approximate a tensor block by a linear combination of rank-one tensors, a natural choice for division criteria is the rank. However, as mentioned eralier, finding tensor rank is an NP-complete problem. As an alternative, weighted directional variance is used as the core of the decision criteria. For a given tensor block $\chi \in R^{(2^{v_1} \times 2^{v_2} \times 2^{v_3})}$, the average variance along each dimension is defined as in eq. (3.1).

$$s_d = \sum_{k=0}^{K} VAR(X_{(d)}^{jk}), \quad d = 1, 2, 3 \tag{3.1}$$

Where $X_{(d)}$ is a matrix that results from unfolding the tensor χ with respect to the d^{th} dimension and K is the number of the columns. The weighted directional variance is defined as in eq. (3.2).

$$\Delta = \sum_{d=1}^{3} w_d s_d \tag{3.2}$$

Where w_d is the dimension weight and $\sum_{d=1}^{3} w_d = 1$. w_d regulates the contribution of each direction in the decision. For each block we calculate the average variance at level l (i.e. Δ^l), then we subdivide it into eight sub-blocks and calculate the average variance for each of them (i.e. $\Delta_j^{(l-1)}, j = 1, \ldots, 8$). If sum of the average variance of the sub-blocks is smaller than the average variance of the block, then the block will be divided into eight sub-blocks. In general a relaxed criteria is used as in eq. (3.3).

$$\Delta^n - \sum_{j=1}^{8} \Delta_j^{n-1} < \frac{\tau}{\rho} \tag{3.3}$$

Where τ is a threshold and $\rho = 4 \times R_{max}/N$. The value of the ρ is derived from the block analysis in section 3.1.1. Note that the ratio R_{max}/N is the average available rank-one tensors per block. As mentioned earlier, when the available rank-one tensors per block is small, larger blocks will provide better approximation of the input data. The parameter ρ will enforce less aggressive block subdivision at lower rates and more aggressive block subdivision at higher rate. If the absolute value of the differential average is smaller than τ/ρ, then subdivision is not required. Otherwise, the block is divided into eight sub-blocks and the procedure is repeated recursively for each of them.

Figure 3.5 shows an example of the hextree and its corresponding code. Figure 3.6 shows an exmple of the general 2^n-ary tree and its corresponding code. As mentioned before,

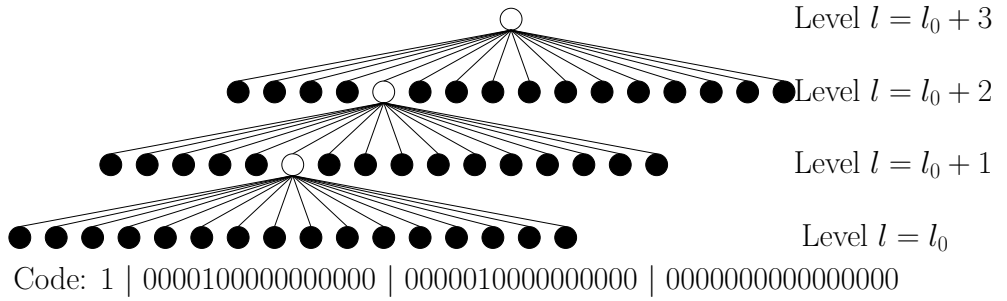


Figure 3.5: An example three levels hextree structure with its corresponding code. Each branching node has 16 children nodes which can be either a leaf or a branching node as well.

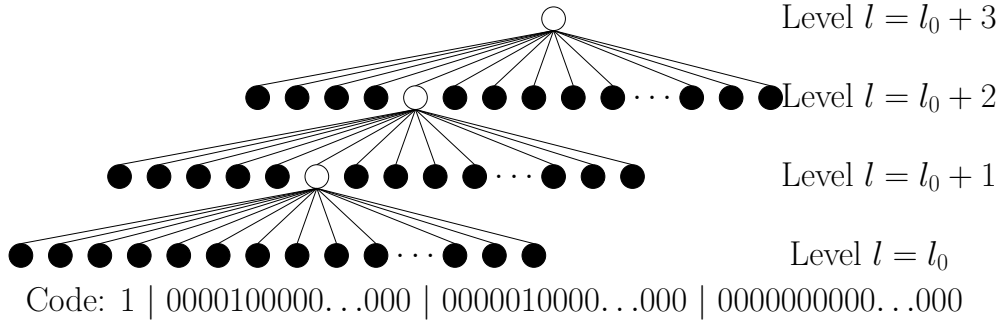


Figure 3.6: An example three levels 2^n-ary tree structure with its corresponding code. Each branching node has 2^n children nodes which can be either a leaf or a branching node as well

Depending on the application and its associated data, an n dimensional input tensor can have different nature along each direction. Therefore, the adaptive tree partitioning should be able to handle a block with different size along each direction. We use a 2^n-ary tree for partitioning an n dimensional tensor. It divides a $2^{v_1} \times 2^{v_2} \times \ldots \times 2^{v_n}$ block into a $v_1 - v_1^{(l_0)} + 1$ levels tree. Although the block size may not be equal along the dimensions, we assume that the tree levels are equal in all directions, $v_1 - v_1^{(l_0)} + 1 = v_2 - v_2^{(l_0)} + 1 = \ldots = v_n - v_n^{(l_0)} + 1$. Where $2^{v_1^{(l_0)}} 2^{v_2^{(l_0)}} \ldots 2^{v_n^{(l_0)}}$ is the smallest possible block. The blocks at level l have the size $2^{v_1^{(l)}} \times 2^{v_2^{(l)}} \times \ldots \times 2^{v_n^{(l)}}$. For dimension d, $0 \leqslant v_d^{(l_0)} \leqslant v_d^{(l)} \leqslant v_d, d = 1, 2, \ldots, n$. Each node can be a leaf or it can be divided to 2^n child sub-blocks with a size of $2^{(v_1^{(l)}-1)} \times 2^{(v_2^{(l)}-1)} \times \ldots \times 2^{(v_n^{(l)}-1)}$.

The dividing procedure is applied to the 2^n sub-blocks recursively until the sub-blocks

do not require further division or they reach the maximum tree level l_0. For a given tensor block $\chi \in R^{(2^{v1} \times 2^{v2} \times \ldots \times 2^{vn})}$, the average variance along each dimension is defined as in eq. (3.4).

$$s_d = \sum_{k=0}^{K} VAR(X_{(d)}^k), \quad d = 1, 2, \ldots, n \tag{3.4}$$

Where $X_{(d)}$ is a matrix that results from unfolding the tensor χ with respect to the d^th dimension and K is the number of the columns. The weighted directional variance is defined as in eq. (3.5).

$$\Delta = \sum_{d=1}^{n} w_d s_d \tag{3.5}$$

Where $\sum_{d=1}^{n} w_d = 1$. For each block we calculate the average variance at level l (i.e. Δ^l), then we subdivide it into 2^n sub-blocks and calculate the average variance for each of them (i.e. $\Delta_j^{(l-1)}, j = 1, 2, \ldots, 2^n$). If sum of the average variance of the sub-blocks is smaller than the average variance of the block, then the blocks will be divided into 2^n sub-blocks. In general a relaxed criteria is used as shown in eq. (3.5).

$$\Delta^l - \sum_{j=1}^{2^n} \Delta_j^{l-1} < \frac{\tau}{\rho} \tag{3.6}$$

Where $\rho = d^{(n-1)} \times R_{max}/N$ and τ is a threshold. If the absolute value of the differential average is smaller than τ/ρ, then subdivision is not required. Otherwise, the block is divided into 2^n sub-blocks and the procedure is repeated recursively for each of them.

Similar to octree, the 2^n-ary tree can be represented with a series of bits. However, as the dimension grows number of the bits grows exponentially. Arithmetic coding and zero

trail termination code can be used to reduce the size of the encoded tree.

3.2 Eigenfibers arrangement and coding

Once the eigenfibers are evaluated for all the blocks of a tensor, we can arrange them onto the columns of a 2D matrix B. Thus, we can apply a 2D compression scheme to this matrix B, treating it as an image. Subsequently, we can decode its columns, which represent the eigenfibers $b_r^{(d)}$, to reconstruct the original tensor. Two important questions need to be answered:

1. How should the eigenfibers $b_r^{(d)}$ be arranged within the matrix B?

2. How much correlation does exist among these eigenfibers?

Recall that under PCP each block has its own rank, and hence we denote R_j to represent the number of rank-one tensors used for approximating block j. Consequently, the total number of rank-one tensors used for approximating the whole tenor is: $\sum_{j=1}^{N} R_j$, where N is the total number of blocks. Each 3D rank-one tensor requires three eigenfibers: $(b_{r,j}^{(1)} \circ b_{r,j}^{(2)} \circ b_{r,j}^{(3)})$, for $r = 1, \dots R_j$ and $j = 1, \dots N$; then we have a total of $3\sum_{j=1}^{N} R_j$ eigenfibers to code.

There are many options for arranging these eigenfibers $b_{r,j}^{(d)}$ onto the matrix B that we plan to compress as a 2D image. Here, we employ the arrangement shown in the example of Figure 3.7. In this example, we generated the eigenfibers of 180 frames of the Container CIF video.

We divided the video into 16×16×180 3D blocks, which results into 396 tensor blocks. For clarity and ease-of-illustration purposes, we are only showing the eigenfibers for the

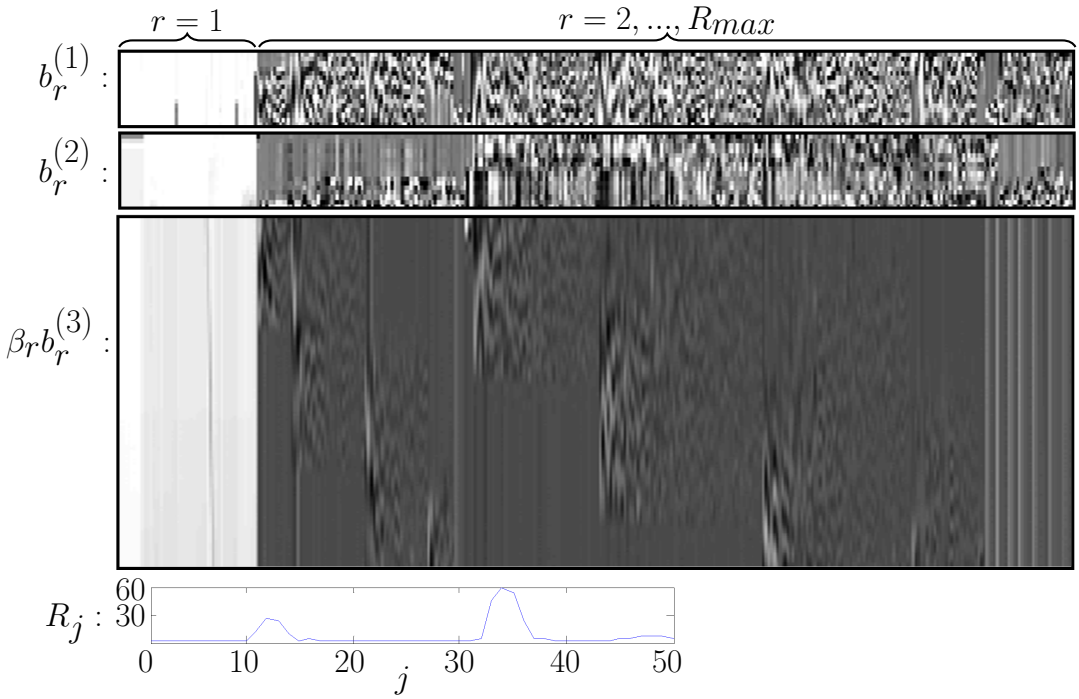


Figure 3.7: Eigenfibers of the PCP tensor decomposition and the rank values R for 50 blocks of the Container CIF video with 180 frames. The figure shows a possible eigenfibers arrangement for coding and storage.

first 50 blocks chosen in a raster-scan order. All eigenfiber values are mapped from their normalized $[-1, +1]$ range onto the traditional $[0, 255]$ pixel values. As shown in the figure, we employ the following arrangement:

- Vertical arrangement: The eigenfibers $b_{r,j}^{(1)}$ are put at the top of the 2D image; each fiber is of height 16. Next, the second eigenfibers $b_{r,j}^{(2)}$, also with height 16, are placed below $b_{r,j}^{(1)}$. These two groups of eigenfibers capture the 16×16 spatial information of the video-blocks. Meanwhile, the third eigenfibers $b_{r,j}^{(3)}$ with height 180 are placed below; and these later eigenfibers capture the temporal information of the 3D video-blocks.

- Horizontal arrangement: More importantly, we separate the eigenfibers associated with the first rank-one tensors (i.e., for $r = 1$) from the rest of all other eigenfibers with higher rank index (i.e. for $r > 1$). This separation is analogous to differentiating

between "DC" and "AC" coefficients in traditional image and video coding. The left bright area in the images of figure 3.7 corresponds to these eigenfibers. For higher rank indices, $r > 1$, we simply place the eigenfibers according to the blocks they belong to in a raster-scan order. We have also experimented with other horizontal arrangements for eigenfibers with $r > 1$. For example, one can group eigenfibers with $r = 2$, followed by ones with $r = 3$, and so on. We observed little improvement in coding efficiency while using such arrangement; meanwhile it increases the complexity due to the need of performing matrix permutations at both the encoding and decoding sides.

Note that, in addition to the eigenfibers, we also have the normalization parameter β_r. Under the proposed tensor coding framework, we absorb the parameters $\beta_{r,j}$ onto the third direction eigenfibers $b^{(3)}_{r,j}$, as shown in figure 3.7. There are two benefits for absorbing these parameters onto the eigenfibers. First, we eliminate the need for coding these parameters separately. Second, this multiplication process improves the correlation among the eigenfibers within the 2D image as we discuss below.

It is important to note that for a given 3D block, the three eigenfibers $(b^{(1)}_r, b^{(2)}_r, b^{(3)}_r)$ are expected to be uncorrelated. However, if we consider different 3D blocks of similar spatial and temporal characteristics, then we anticipate that the eigenfibers across such blocks to be correlated. Figure 3.8a shows the correlation among the columns of matrix A that correspond to traditional CP factored vectors $(a^{(1)}_{r,j}, a^{(2)}_{r,j}, a^{(3)}_{r,j})$. Figure 3.8b is the correlation among the columns of matrix B, which are the PCP eigenfibers $(b^{(1)}_{r,j}, b^{(2)}_{r,j}, b^{(3)}_{r,j})$. The bright upper-left region corresponds to the correlation among the first eigenfibers of each block. These particular (first) eigenfibers, which represent the *principle eigenfibers*, are highly correlated. However, they have low correlation with most of the other eigenfibers

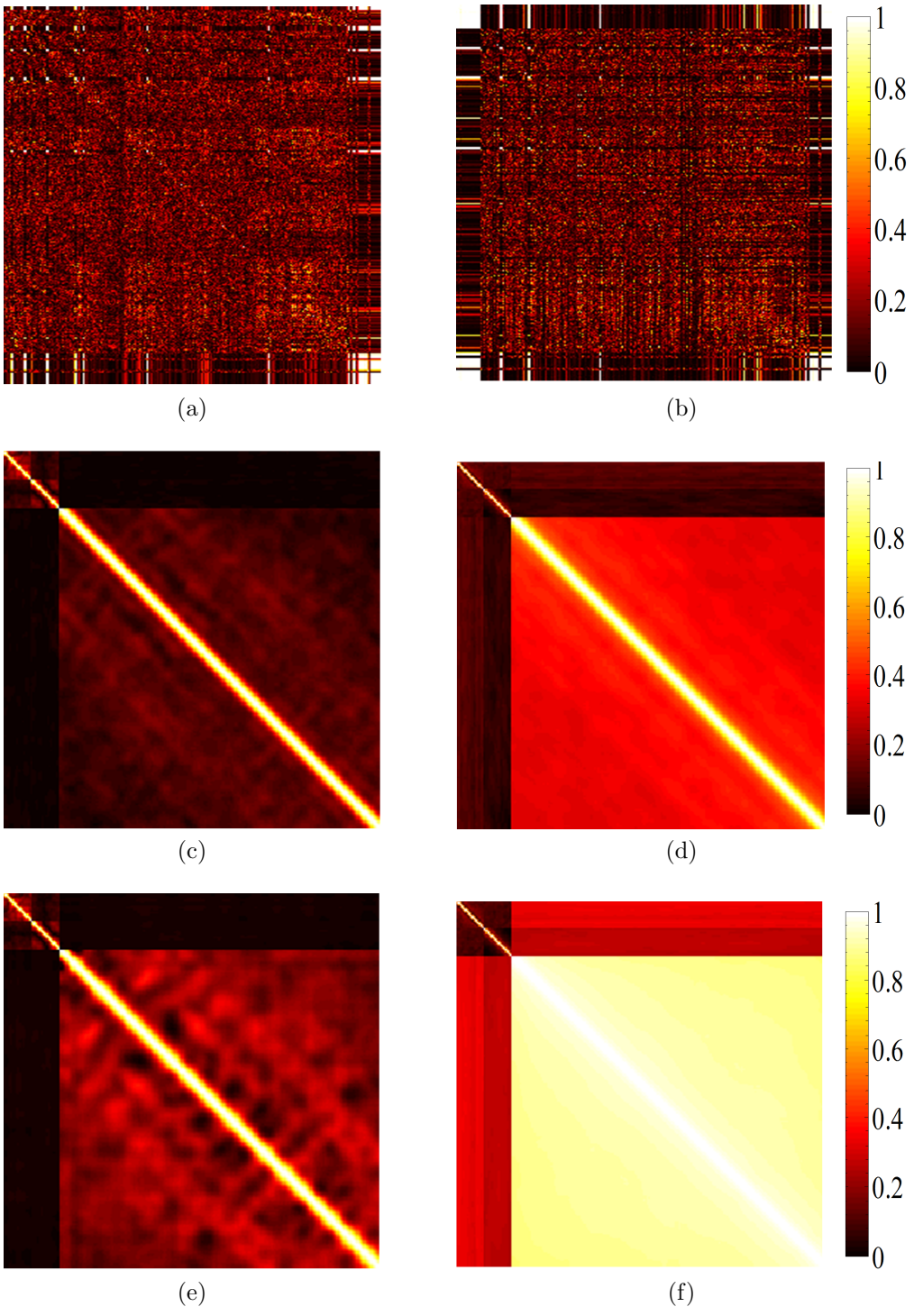


Figure 3.8: The correlation among (a) matrix A columns (i.e. CP vectors); (b) matrix B columns (i.e. PCP eigenfibers); The correlation among (c) matrix A rows; (d) matrix B rows; (e) after $\lambda_{(r,j)}$ absorption onto $a^{(3)}_{(r,j)}$ vectors; and (f) after $\beta_{(r,j)}$ absorption onto $b^{(3)}_{r}$. The video is Container CIF.

(darker upper panel). As expected, and similar to the original CP decomposition, PCP provides uncorrelated vectors within each block.

Meanwhile, the entries within the eigenfibers can be correlated. In other words, if we consider the first row of matrix B then it will be desired to have this row correlated with other rows within the same matrix. Such intra-eigenfiber correlation is captured by the correlation among the rows of the eigenfiber matrix shown in figures 3.8c-3.8f. The proposed PCP decomposition provides higher intra-eigenfibers correlation than what can be achieved under CP. Assuming R^* is known, we decomposed the video using both CP and PCP with the same R^* to obtain a comparable number of factored vectors and eigenfibers in this experiment. Figure 3.8c is the correlation among the rows of matrix A and figure 3.8d is the correlation among the rows of matrix B. From figures 3.8c and 3.8d, PCP provides eigenfibers with higher row correlation than the standard CP. The three square regions on the diagonal are the intra-eigenfiber correlation of $B^{(1)}$, $B^{(2)}$, and $B^{(3)}$ respectively. The other three regions above the diagonal is the intra-eigenfiber correlation of $B^{(1)}$ with $B^{(2)}$, $B^{(1)}$ with $B^{(3)}$, and $B^{(2)}$ with $B^{(3)}$. Figures 3.8e and 3.8f show the impact of absorbing the normalization parameters $\lambda_{r,j}$ (for CP) and $\beta_{r,j}$ (for PCP) within the corresponding eigenfibers.

The next step is to code the rank parameters R_j for all blocks. We simply arrange these values onto a vector and entropy code them in a lossless manner.

For a higher dimension input tensor we simply add the eigenfibers at the bottom of the 2D matrix. For example, decomposing an n dimensional input tensor will result in n eigenfibers. Based on the partitioning method we have two different settings:

1. Uniform partitioning: assuming each block is of size $v_1 v_2 \dots v_n$, the vector that stack all the eigenfibers of a rank-one tensor will be of size $v_1 + v_2 + \dots + v_n$. Since all the blocks are of the same size we can arrange this vectors as columns of the 2D matrix.

2. Adaptive tree partitioning: in this settings, each block can have a different size. However based on the tree structure we know the maximum block size and the tree level. Consequently we can obtain the minimum block size. For a minimum size block we simply follow the arrangement steps of uniform partitioning. For larger blocks, in order to maintain consistent column size, we simply store the corresponding eigenfibers as multiple columns. Each column has the same size as the vector of eigenfibers of the minimum size block. For example, assume the size of the minimum block is $v_1 \times v_2 \times \ldots \times v_n$ and a particular block j has a size of $2v_1 \times 2v_2 \times \ldots \times 2v_n$. The eigenfibers vector of the block j will be of size $2v_1 + 2v_2 + \ldots + 2v_n$, therefor we store this vector as two vectors of size $v_1 + v_2 + \ldots + v_n$ each to be consistent with the minimum size block.

3.3 Residual coding

The proposed framework is based on approximating a tensor with a set of rank-one tensors. If the original tensor is of low-rank we would be able to reconstruct it exactly. However, if the tensor is close to full rank, at some point no matter how many rank-one tensors we add, the approximation will not converge to the original one.

The exact or near lossless reconstruction is important for some applications. In order to present a complete coding framework that would be able to code the visual data tensors with high quality (near lossless), we employ a residual coding process as shown in figure 3.1. After approximating a visual data tensor χ with the proposed tensor coding framework, we obtain the final residual $\chi_r = \chi - \hat{\chi}$ where $\hat{\chi}$ is the approximation tensor.

Since PCP captured the existing correlation in the input tensor, we code each residual

slice separately. This would also maintain the random access property. Note that, these slices can be along any dimensional direction depending on the type of the random access required in a particular application. For example, image ensemble decoding requires random access along the third dimension. Consequently, the residual slices are coded separately along the third dimension. While the rate of coding the residual slices would not be as high as coding the original images, the scalability of the proposed method provides flexibility for different applications to reconstruct the data based on their requirement. In this thesis we used the Amplitude and Group Partitioning (AGP) image coding method [49] to code the residual.

3.4 Tensor coding properties

The proposed tensor coding framework has some desirable properties. These properties, individually or combined, result in advantages over standard compression methods in some applications. In this section Random access, progressive reconstruction, and time complexity will be discussed.

3.4.1 Random access

Tensor coding has the advantage of random access, which is not the case for the motion based coding methods and many 3D transform based coding approaches. The motion based coding methods codes a GOP using a single key frame and the rest are predicted frames. In order to decode any predicted frame, the decoder need to have all of the previous frames in the same GOP. The 3D transform based coding methods, take advantage of the existing correlation along all the dimensions. However they require all the transform domain coefficients to

reconstruct the original tensor.

In the case of tensor coding, to reconstruct a slice from a 3D tensor, the decoder requires the first two eigenfibers and a value of the third eigenfiber to decode a slice independently. In general, any slice can be decoded as shown in eq. (3.7).

$$Slice_{i,j} = \sum_{r=1}^{R_j} (B^{(1)}_{j,r} \circ B^{(2)}_{j,r} \circ B^{(3)}_{i,j,r}), j = 1, \ldots, N \tag{3.7}$$

Where j is the block index, $B^{(1)}_{j,r}$ and $B^{(2)}_{j,r}$ are vectors, $B^{(3)}_{j,r,i}$ is a single value from row i of $B^{(3)}$. The column indices are either obtained from the octree structure in the case of adaptive tree partitioning, or from the block number in the case of the uniform partitioning. The first two eigenfibers can be thought of as basis, while the values from the third eigenfiber are the coefficients.

For the more general case of an n dimensional tensor, a particular slice can be decoded as shown in eq. (3.8).

$$Slice_{i,i_3,\ldots,i_4} = \sum_{r=1}^{R_j} (B^{(1)}_{j,r} \circ B^{(2)}_{j,r} \circ B^{(3)}_{i_3,j,r} \circ \ldots \circ B^{(n)}_{i_n,j,r}), j = 1, \ldots, N \tag{3.8}$$

The random access property is not limited to a slice of the first and second dimension and based on the application a subset of eigenfibers can be selected to reconstruct a slice in a particular dimension. For example, in hyperspectral images, a vector of values across all bands for a particular pixel called pixel signature. It provide a valuable information about the type of the earth materials or the vegetation. When a hyperspectral images is coded with tensor coding framework, a pixel signature can be reconstructed from a set of eigenfinbers as shown in eq. (3.9).

$$Signature_{i_1,i_2,j} = \sum_{r=1}^{R_j} (B^{(1)}_{i_1,j,r} \circ B^{(2)}_{i_2,j,r} \circ B^{(3)}_{j,r}), j = 1, \ldots, N \tag{3.9}$$

Note that the reconstructed signature is a one dimensional vector and j is the block index, $B^{(1)}_{i_1,j,r}$ and $B^{(2)}_{i_2,j,r}$ are single values from matrices $B^{(1)}$ and $B^{(2)}$ correspondingly, and $B^{(3)}_{j,r}$ is a vector.

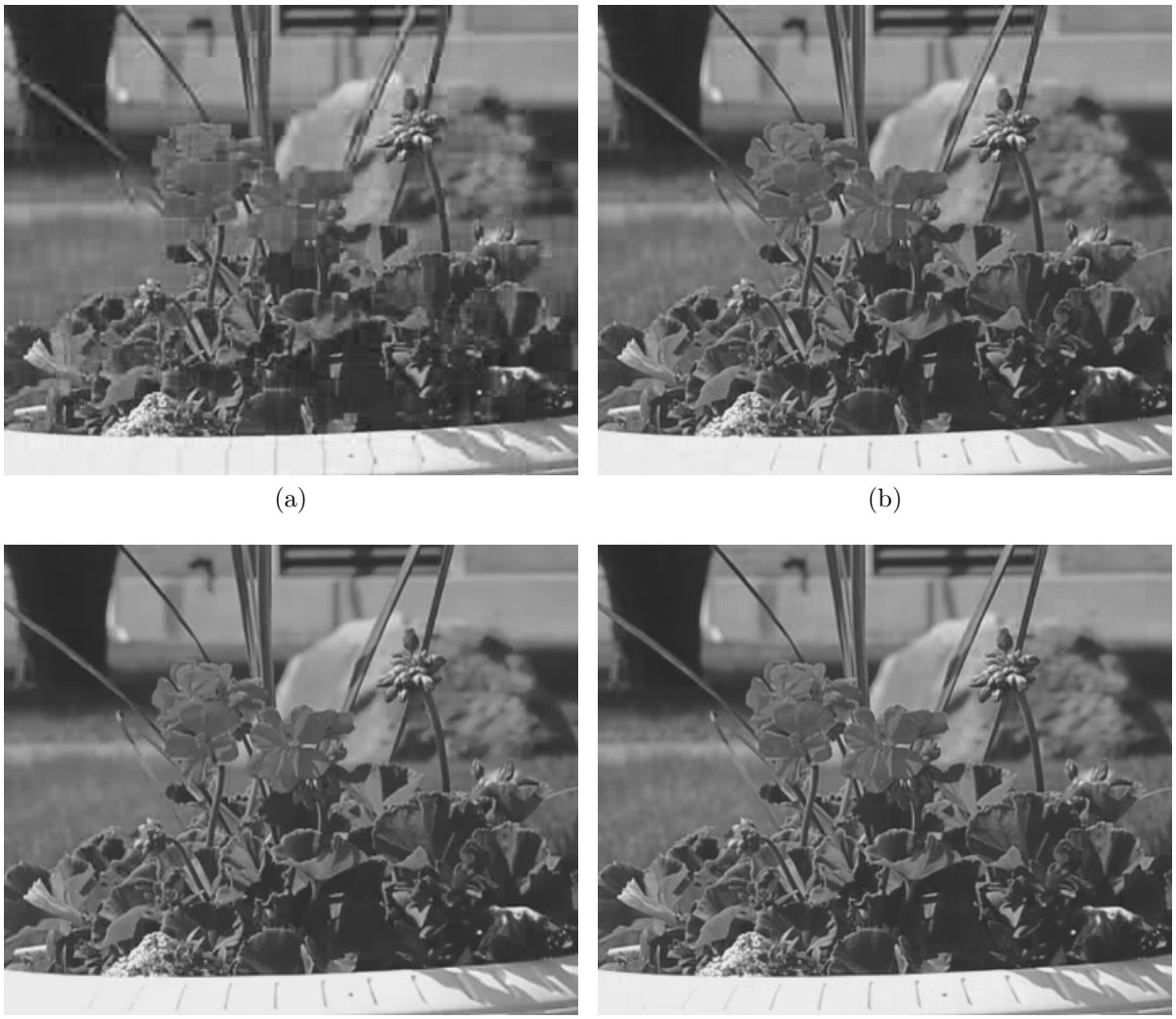
3.4.2 Progressive reconstruction and decoding

Unlike CP decomposition PCP factorizes the tensor in a progressive scheme. In other words, if one derives the CP decomposition using a given rank parameters R, then all of the R rank-one tensors must be used for reconstructing the approximated tensor $\hat{\chi}$; otherwise, the quality of the reconstructed tensor is significantly degraded. This is true even if the 3D block has a low rank; all of the factored vectors should be used in the reconstruction to give reasonable results. On the other hand, PCP simply improves the quality of the reconstructed tensor, when increasing the number of rank-one tensors used for reconstruction. Therefore, a subset of the eigenfibers can be used to reconstruct a lower quality result.

Figure 3.9 shows a frame from the "Red Flower" CIF video encoded by the proposed tensor coding framework with a total number of rank-one tensors equal to 6000 and then decoded using (a) 1000, (b) 2000, (c) 3000, and (d) 4000 rank-one tensors. Increasing number of rank-one tensors, result in gradual increment of the video details.

3.4.3 PCP time complexity

The most time-consuming operation of the PCP decomposition is the Khatri-Rao product in eq. (2.9). For example to find $b^{*(1)}_r$ we need to calculate $z^{(1)}_r$ and $X'_{(1),k}$. The first term



(a) (b)

(c) (d)

Figure 3.9: "Red Flower" CIF video encoded by tensor coding with a total of 4000 rank-one tensors. Uniform partitioning was used with block size of 1616180. The video decoded with (a) 1000 (93 Kbps 30.43 dB), (b) 2000 (205 Kbps 35.76 dB), (c) 3000 (320 Kbps 38.49 dB), (d) 4000 (433 Kbps 40 dB), rank-one tensors.

is of order $O(v_2 v_3)$ while the second one is of order $O(v_1 v_2 v_3)$. Recall that $v_1 \times v_2 \times v_3$ is the size of the 3D tensor block. The time complexity of all three decomposed eigenfibers and all other extra operations can be captured by a constant term (ρ). The total complexity of coding a visual data ensemble with tensor coding framework is $O\left(\rho v_1 v_2 v_3 \sum_{j=1}^{N} R_j\right)$ where N is the number of blocks, and R_j is the number of rank-one tensors used to reconstruct

block j. Note that to solve for $b_r^{*(1)}$ we are using ALS, which is an iterative approach. The complexity of all the iterations is captured by ρ.

PCP reconstructs a 3D tensor as in eq. ((2.7)). It consists of outer product of decomposed eigenfibers and then addition of rank-one tensors. Overall there are $v_1 v_2 v_3 \sum_{j=1}^{N} R_j$ multiplications and $v_1 v_2 v_3 \sum_{j=1}^{N} R_j$ additions; where N is the number of blocks. Hence, the complexity of the decoding algorithm is of order $O(2v_1 v_2 v_3 \sum_{j=1}^{N} R_j)$.

For an n dimensional tensor, the PCP reconstruction shown in eq. (2.10) has a time complexity of $O\left(\rho v_1 v_2 \dots v_n \sum_{j=1}^{N} R_j\right)$. Where N is the number of the blocks. Note that this time complexity accounts for the PCP decomposition only and despite the used tensor partitioning approach. The adaptive tree partitioning time complexity should be considered as part of the overall tensor coding complexity.

Chapter 4

Tensor coding applications

The proposed tensor coding framework is optimized for visual data, however it can be applied to any type of data. It is designed to be able to represent and code input tenor with any dimension efficiently specially 3D and above. In this thesis, the framework is applied to three applications which are

1. Hyperspectral and multispectral images. Multiple experiments have been done for both 3D and 4D input data tensors.
2. Biometric face image ensembles which are 3D tensors.
3. Low complexity video coding in which a video is treated as a 3D tensor.

4.1 Hyperspectral image coding

Unlike the conventional imaging systems that measure the energies in the visible light spectral bands, the hyperspectral imaging system measures the energies in a broad range of spectral bands. The number of spectral bands can range from 8 in Landsat data sets up to 200 in the multispectral data sets. While most of the bands do not provide visual information, they contain a vital scientific information about the earth and atmosphere. An efficient system for representation and coding of hyperspectral images is essential especially considering the fact that the amount of this type of images are growing rapidly.

While each band has its characteristics, there is a high amount of correlation among them that can be exploited for compression. Various 3D-wavelet based compression methods are proposed to exploit this correlation [50–53]. However, the random access property is not preserved in these methods. Fast random access property is playing a key role in providing efficient browsing experience. Suppose that we want to download a specific band of a hyperspectral images available on a server. Under the 3D-wavelet methods, the whole compressed data should be available to reconstruct the original data and then access that particular band. Depending on the size of the image, network quality, and the server bandwidth/traffic, obtaining the complete compressed data can be inconvenient.

The proposed tensor representation and coding can provide a random access while delivering a better compression than the single image compression methods [54]. Further more, the inherited scalable coding property of the proposed method can provide reconstruction with variable quality. The scalability property contribute in efficient functionality. For example we can use a lower quality reconstruction for classification. Also, the proposed method capable of fast decoding which is a key component of the efficient browsing. The inherited scalable coding property of the proposed method can provide reconstruction with variable quality. Furthermore, with the residual coding capability the high frequency data can be reserved and delivered whenever desired.

4.1.1 Experimental results

The proposed tensor coding framework was employed to code a set of 3D hyperspectral images from AVIRIS data set [55] and multispectral images from Landsat data set [56]. All the experiments were evaluated at a desktop computer with 12 GB of memory and an Intel Core i7 2600 CPU (8MB Cache, 3.4 GHz).

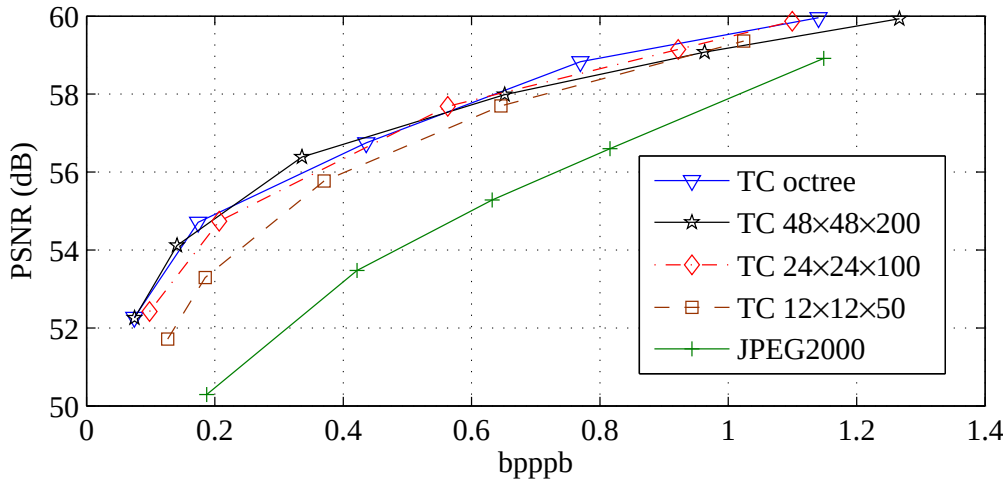


Figure 4.1: Indian_pine PSNR vs bpppb comparison among TC with different block sizes, TC with octree and JPEG2000.

In the first experiment we decomposed the Indian_pine hyperspectral image data with different number of eigenfibers to limit the rate and consequently the final storage size. The Indian_pine data set consists of 200 spectral bands and the spatial resolution is 144×144. Figure 4.1 shows the PSNR vs bit per pixel per band (bpppb) plot. Tensor coding results with different block sizes and octree block structure are compared to the JPEG2000. We used an octree with three levels and the largest block size of $48 \times 48 \times 200$. The weights in eq. (3.2) are equally distributed and the threshold in eq. (3.3) is set as 1% of the parent block variance.

Similar to the discussion in the section 3.1.1, Figure 4.1 shows that for smaller rate, the larger block sizes provide better compression at lower bitrates. However they cross over at some point and at higher bitrates smaller block sizes provide better compression. On the other hand, octree delivers a consistent better compression by choosing a variable block size based on the region and the available bitrate.

A critical illustration for how effective such decomposition can be is how does it impact the spectral profile of individual physical locations across bands. The spectral profile is a

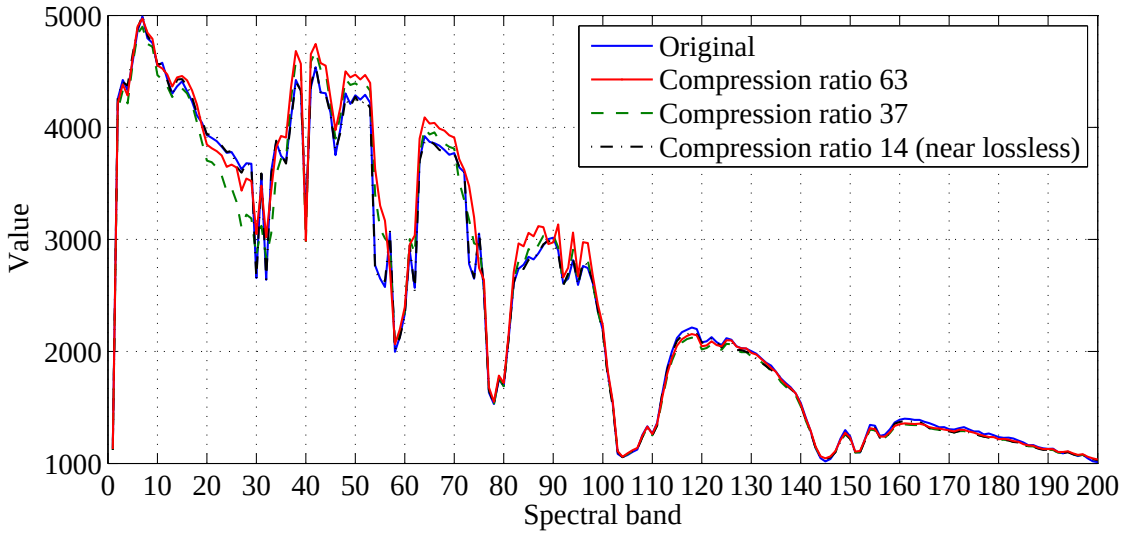


Figure 4.2: The spectral profile over 200 bands of a particular spatial location from the Indian_pine data shown in Figure 4.3. Three reconstructed spectral profiles based on the proposed framework at different bitrates are shown.

key piece of information that scientists rely on to analyze the data. The spectral profile of the Indian_pine is shown in Figure 4.2. Our simulation results show that reconstruction with two order of magnitudes of compression ratio provides spectral profiles that track the original one very closely therefore it can be used in applications like classification and pattern recognition. The scalability of the proposed tensor coding method provide a flexibility on the reconstructed signature quality. However there is a trade off between the the reconstructed signature precision and the compression ratio.

A similar experiments were done for different hyperspectral images from AVIRIS image data sets. Table 4.1 shows the bitrate comparison between tensor coding and JPEG2000 at a particular PSNR for these hyperspectral images. The results show the advantage of tensor coding in exploiting the correlation over a standard 2D compression method.

In different encoding scenario, we set the R_{max} value to 100 and 1000 which result in average of 1.39 and 6.95 eigenfibers per block respectively. Furthermore to achieve near lossless compression, when coded with $R_{max} = 1000$, we encoded each residual slice (difference

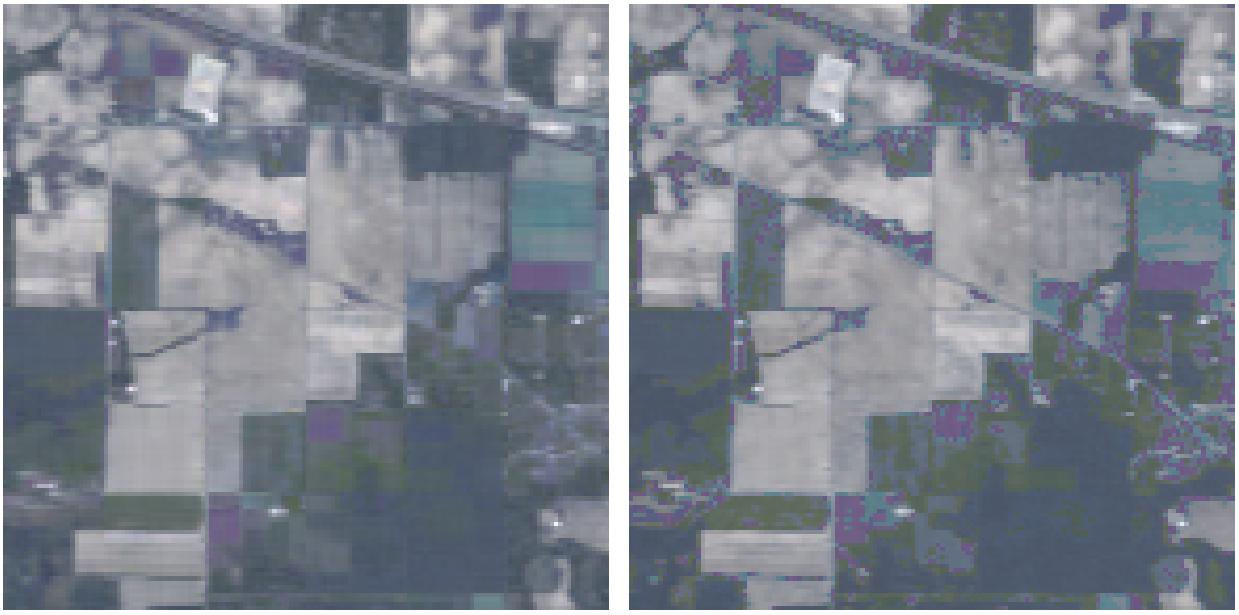
Table 4.1: bitrate comparison between tensor coding and JPEG2000 for various hyperspectral images from AVIRIS data set.

Image	**Tensor coding**		**JPEG2000**	
	bitrate(bpppb)	PSNR(dB)	bitrate(bpppb)	PSNR(dB)
Cuprite	0.20	42.82	0.41	42.83
Pavia University	0.20	51.90	0.53	51.94
Botswana	0.20	52.95	0.42	52.94
Salinas	0.20	66.17	0.90	66.17
Indian pines	0.20	55.29	0.66	55.32

between original and reconstructed slices) using AGP image coding [49]. Figure 4.3 shows the reconstruction results. Note that for illustration purpose only RGB bands are shown. In addition to the efficient and scalable representation ranging from more than two orders of magnitude to one order of magnitude in compression ratios (for near lossless), the time for reconstruction ranges from 0.6 to 1.35 per one 2D slice (using MatLab). This enables reconstruction of 100 low-resolution slices to more than 40 high-resolution slices per second. The decoding process time complexity can be further reduced through using C/C++ implementation with parallel computing capability. In this experiment, for $R_{max} = 100$, we obtained a compression ratio equal to 228 and the average decoding time of 0.6 msec/slice. For $R_{max} = 1000$, the compression ratio was 61.5 and the average decoding time was 1.35 msec/slice. Finally, the near lossless coding compression ratio was 10. Note that here the compression ratio is the ratio between the actual bits needed to store the compressed data and the original image bits.

In another experiment, we conducted 3D representation of high-resolution $2K \times 2K$ pixels

(a) (b)



(c) (d)

Figure 4.3: A hyperspectral 3D imaging data and its reconstructed representation using a progressive number of 3D eigenfiber sets. Compression ratios raging from one order of magnitude (near lossless) to more than two orders of magnitude. (b) Original Indian_pine, (b) 100 eigenfibers, compression ratio 213, (c) 1000 eigenfibers, compression ratio 37, (d) 1000 eigenfibers plus residual coding, compression ratio 10.

of Landsat data across 20 time instances. The Landsat images are multispectral that have fewer spectral bands comparing to hyperspectral. However, Landsat takes periodic images of the same region. While the spectral direction of the data set have some correlation, there is a large amount of correlation in the time direction. For this experiment we used the below coding scenario to evaluate and compare the higher dimension tensor coding.

1. The Landsat multispectral images have six spectrum. We aligned the 20 time instances of each spectrum in a 3D tensor. As a result, six 3D tensors were encoded with the proposed framework.

2. The Landsat multispectral images were aligned in a 4D tensor. The dimensions are two spatial, time, and spectrum. Then a hextree was used to represent the tensor partitioning and a 4D tensor coding framework was applied to code the data.

We compared the compression ratio of these two scenarios at different bitrates. Figure 4.4 shows the bitrate vs. PSNR plot for the Lake multispectral image. The lowest bitrate point achieved by the 4D tensor coding framework was 0.0072 bpppb. It translate to a compression ratio of 1111. The corresponding PSNR was 31.5. The encoding time at this particular bitrate point was 13.2 seconds per slice and the decoding time was 114 milliseconds per slice. In other words, the 4D tensor coding framework can decode about 9 slices per second at the bitrate of 0.0072 bpppb. Figure 4.5 shows the decoding time comparison along different bitrates between the 3D and 4D tensor coding for the Lake multispectral image. The decoding times of 4D and 3D tensor coding at lower bitrates were close. However the gap was gradually increased as the bitrate was increased.

Figure 4.6 shows the original image along with the 3D tensor coding reconstruction results at different compression ratios. The compression ratios were ranging from 1142 to 340 which

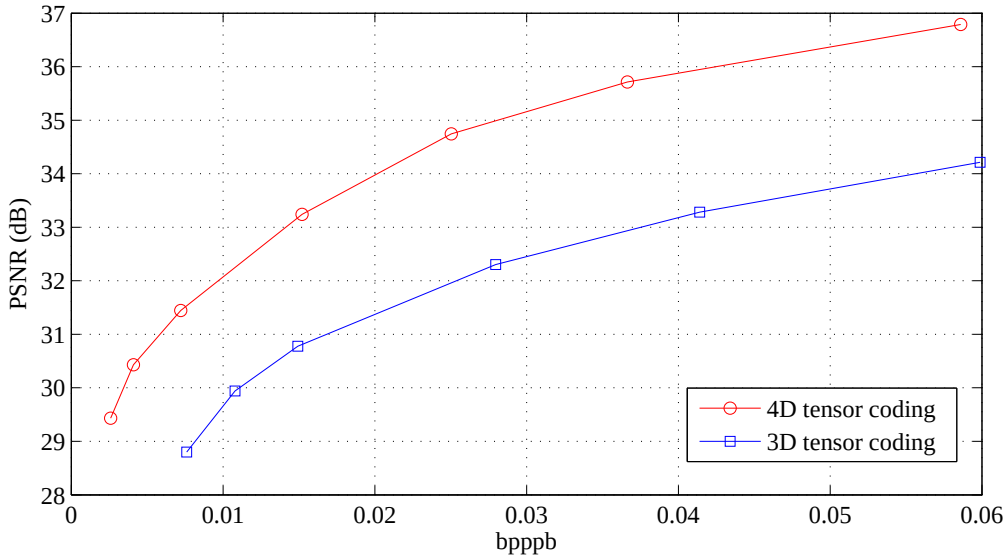


Figure 4.4: Lake multispectral image PSNR vs bpppb comparison. The compression result of 4D tensor coding with hextree partitioning is compared with 3D tensor coding with octree partitioning. 4D tensor coding achieves higher compression by exploiting the existing correlation along all the dimension of the lake multispectral image.

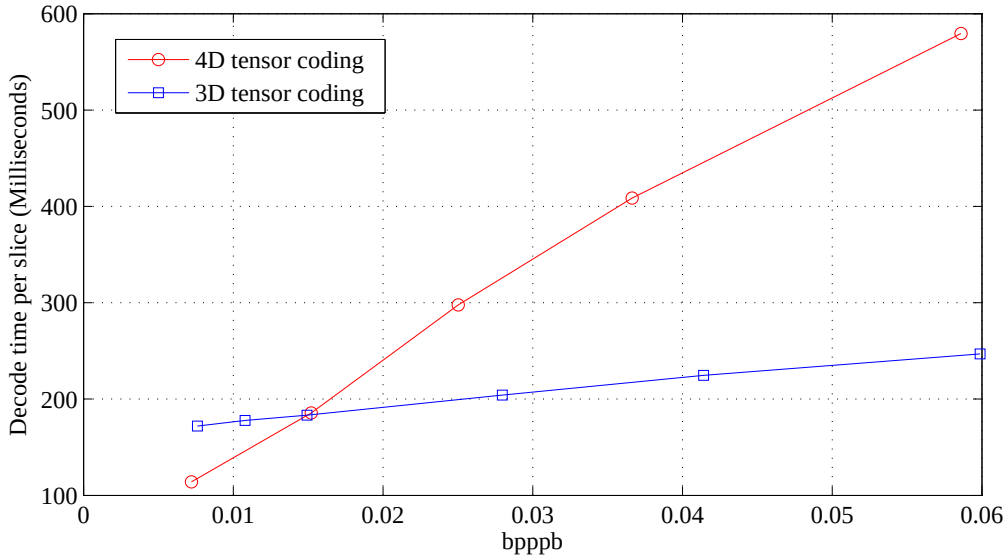


Figure 4.5: Lake multispectral image decoding time comparison. The decode time per slice is shown in milliseconds for 4D and 3D tensor coding. At lower bitrates the decoding time is close while in the higher bitrates the 4D tensor coding results in higher decoding time.

are quite significant. Note that at compression ratio 1142, each block is represented by only one rank one tensor. Since each block needs to be represented at least with one rank one tensor, this compression ratio represent a lower boundary on how much compression the 3D tensor coding can achieve.

Figure 4.7 shows the original image along with the 4D tensor coding reconstruction results at different compression ratios. The compression ratios were ranging from 6154 to 769. The 4D tensor coding can achieve better quality reconstruction at the same compression ratio compared to the 3D tensor coding. This is due to the fact that the 4D tensor coding takes advantage of the correlation along the fourth dimension. Also, note that 4D tensor coding can encode at lower bitrates compared to 3D tensor coding lower boundary.

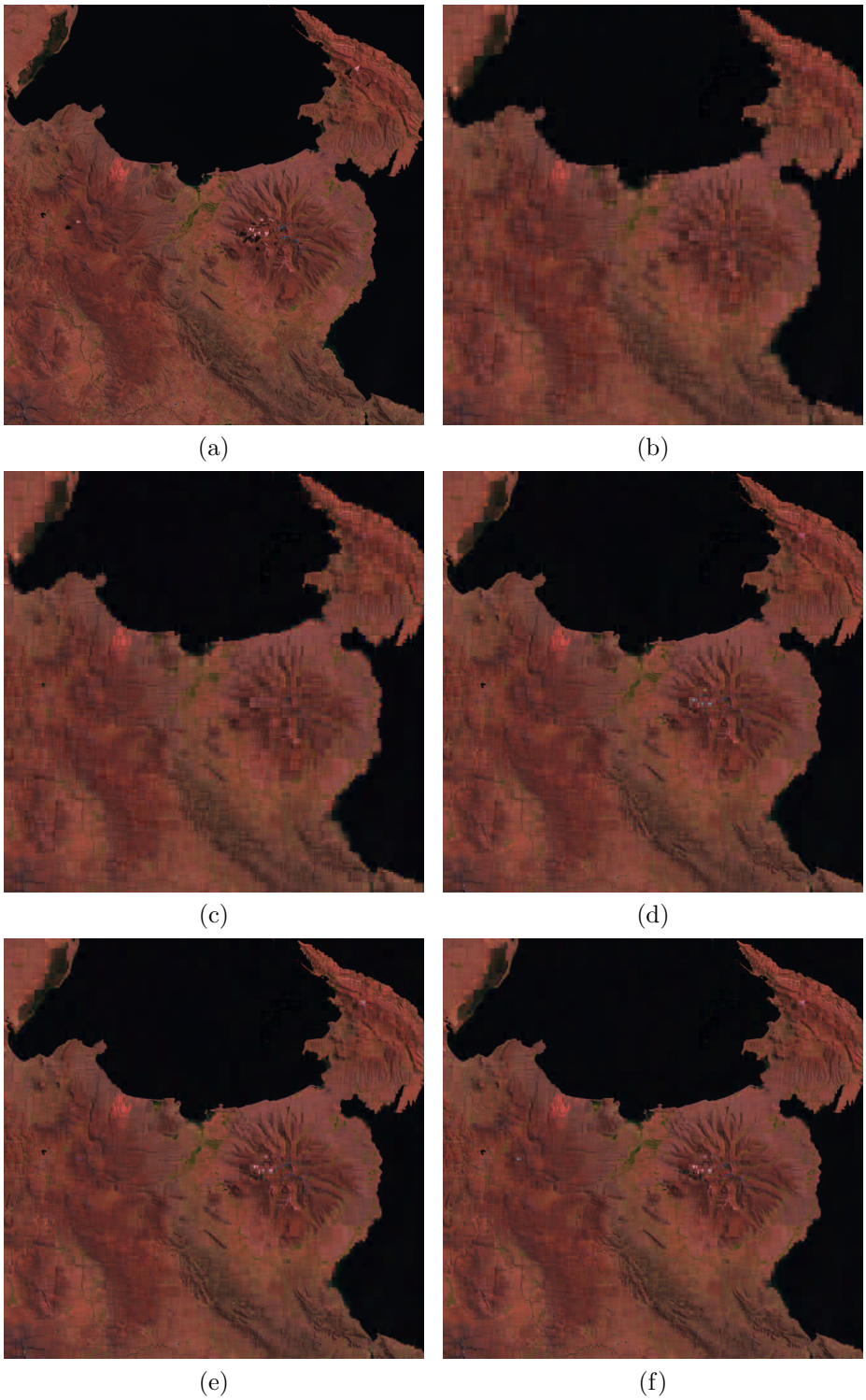


Figure 4.6: 3D tensor coding of high-resolution $2K \times 2K$ pixels across 20 time instances of Landsat data. (a) original image, (b) compression ratio = 1142 and PSNR = 29, (c) compression ratio = 727 and PSNR = 30.83, (d) compression ratio = 534 and PSNR = 31.73, (e) compression ratio = 421 and PSNR = 32.37, (f) compression ratio = 340 and PSNR = 32.88.

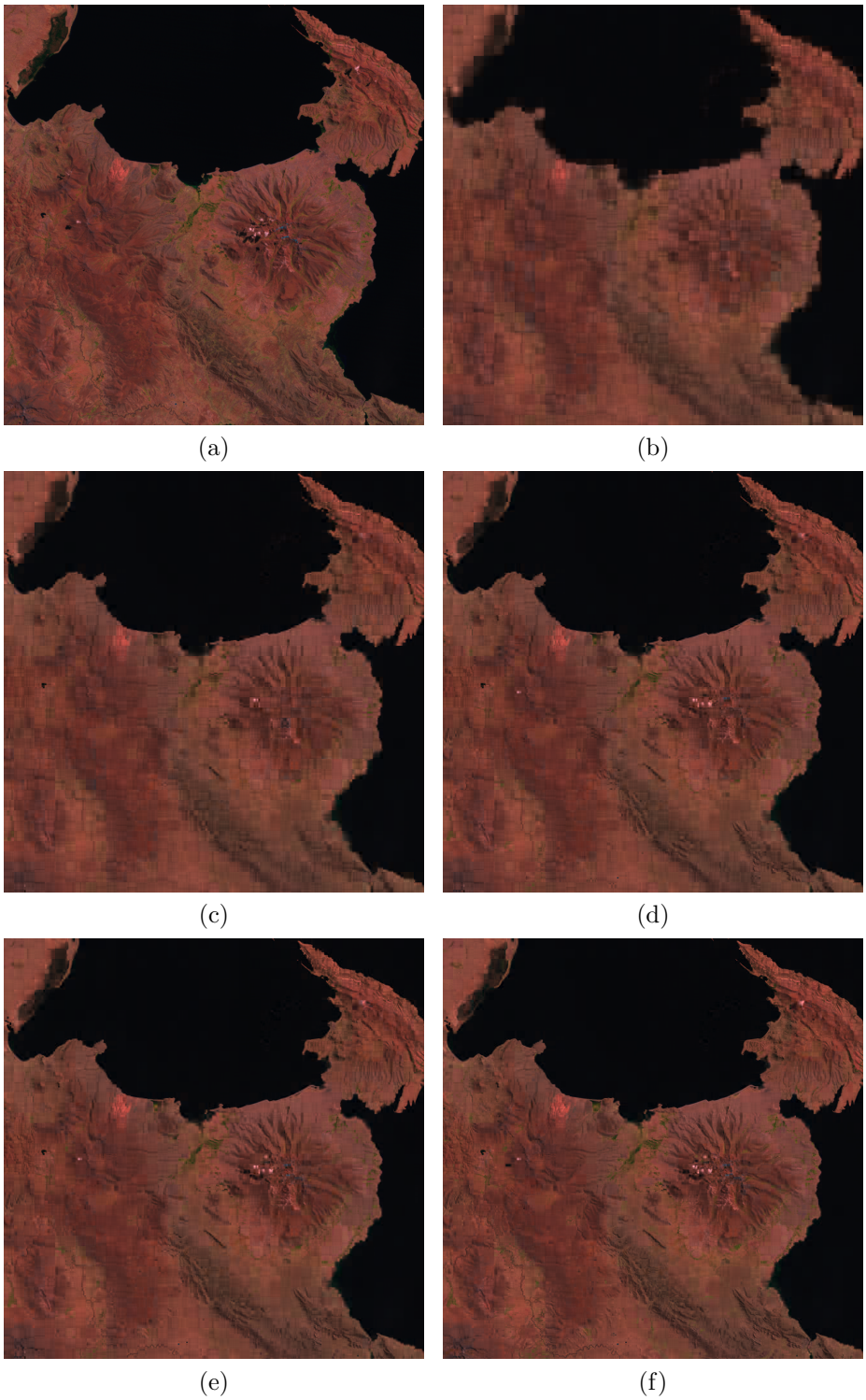


Figure 4.7: 4D tensor coding of high resolution $2K \times 2K$ pixels across 20 time instances of Landsat data. (a) original image, (b) compression ratio = 6154 and PSNR = 27.29, (c) compression ratio = 4210 and PSNR = 29.43, (d) compression ratio = 3077 and PSNR = 30.43, (e) compression ratio = 1429 and PSNR = 31.02, (f) compression ratio = 769 and PSNR = 32.22.

4.2 Image ensembles coding

Image databases represent a core component of many well-established and emerging applications and services including ecommerce and security. For example, image databases of faces, fingerprints, and eye retinas are used extensively for biometric and other security-related applications. Such databases store a vast number of images of the same type, and yet, traditional compression standards are used to compress and store these images without exploiting the correlation that potentially exists among the images within the same database.

For example, the ISO/IEC 19794 standard on biometric data interchange format defined JPEG and JPEG2000 as admissible lossy compression methods. A key driver for encoding each image in isolation of other images within the same database is the ability to access and decode any image without the need to access/decode other images. Such requirement eliminates popular video coding standards as viable candidates for coding still-image databases. Employing the proposed tensor coding framework can achieve both: (a) random access to any image within a collection of images coded jointly and (b) coding efficiency by exploiting any potential correlation that may exist among the images within the same database.

4.2.1 Experimental results

The proposed method was applied to the Yale Face Database B [57]. The database has images of 38 persons. Each of them has 64 images of size 192×168. These images vary in expression and illumination condition. After stacking the images on top of each other, we have a 3D tensor of size $192 \times 168 \times 2432$. The resulting tensor is decomposed using PCP and the eigenfibers are arranged in 2D matrices. Then the result is compressed by JPEG2000. Within the context of our proposed image-ensemble tensor based compression, we compare the

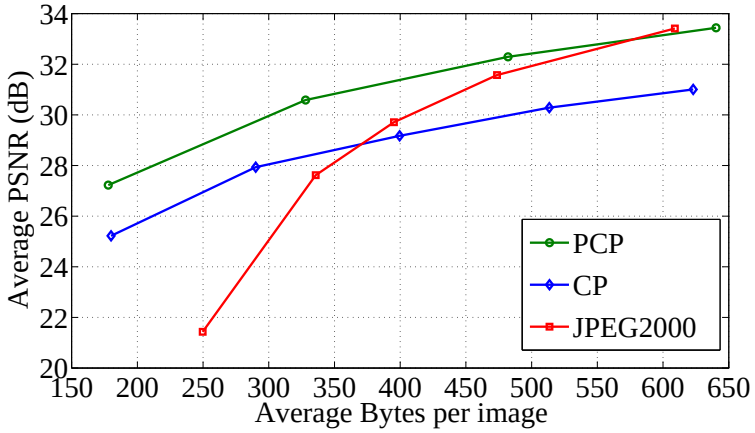


Figure 4.8: Average PSNR plot of 38 persons face images from Yale Face Database B versus the average storage size required per image.

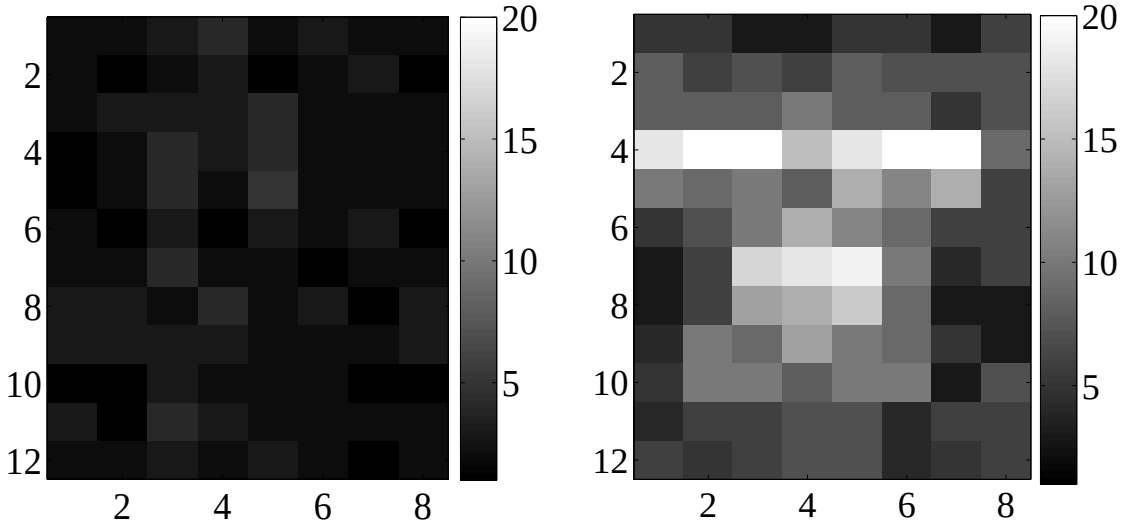
following tensor decomposition approaches and existing still-image compression standards used in image databases:

1. Block-wise PCP-decomposition. The block size is $16 \times 21 \times 64$ and the value of γ is changed to control the final storage size.

2. Block-wise CP-decomposition. The block size is the same as in method (1) and the value of γ is changed to compare the results for different compaction ratios. Here, we used the same structure as in 3.1 except the decomposition method is replaced with CP and JPEG2000 lossless mode is used since small changes in the CP decomposed vectors can lead to large error in reconstruction.

3. Storing each image separately using JPEG2000 standard. The MATLAB implementation is used.

Figure 4.8 shows the reconstruction PSNR averaged over 38 persons versus the required space (in Kbytes) for all the 64 images of a person averaged over 38 persons. Over a wide-range of bitrates, PCP outperforms other methods. Figure 4.9 shows the PCP R values for the blocks of an image encoded at two different bitrates. Notice that when we increase



(a) (b)

Figure 4.9: PCP R values for the blocks of an image encoded at a) 288 Bytes b) 979 Bytes. As the allocated number of rank one tensors is increased, the tensor coding framework allocate a higher number of rank one tensors to the blocks with more information. For example the blocks that contains the eyes and the mouth

the R_{max} value, our algorithm allocate larger R values for the blocks that contain more information. In the case of the Yale Face Database, those regions are around the eyes, nose and the mouth. Figure 4.10 shows one of the reconstructed images using above methods and standard JPEG along with MATLAB implementation of Motion JPEG2000. Figure 4.11 shows the reconstruction results at higher bitrates. At higher bitrates, except for JPEG, all of the methods have close PSNR and the visual quality is similar.

Based on the progressive nature of PCP, its time complexity is linear as a function of the number of rank-one tensors. CP factorization (i.e., the encoding side) has a quadratic complexity as a function of R. Either case (PCP or CP), the decoding complexity is on the same order as a traditional JPEG2000 decoding. Figure 4.12 shows the time complexity, where the decomposition methods are evaluated at a desktop computer with 12 GB of memory and an Intel Core i7 2600 CPU (8MB Cache, 3.4 GHz).

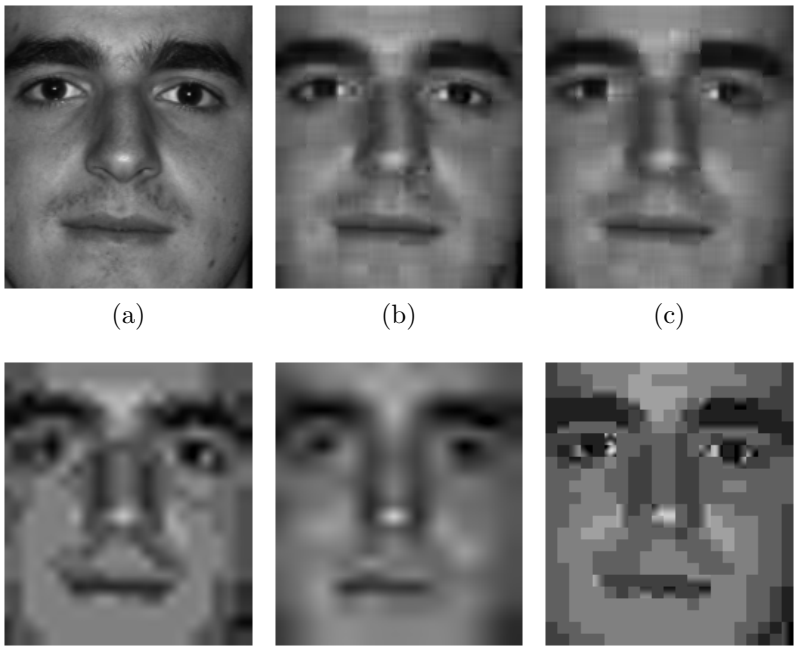


Figure 4.10: (a) Original image; (b) tensor coding with PCP (288 Bytes, 30.1 dB) ; (c) tensor coding with CP (286 Bytes, 28.98 dB); (d) JPEG2000 (291 Bytes, 25.68 dB); (e) motion JPEG2000(292 Bytes, 24.63 dB); (f) JPEG (790 Bytes, 25.19 dB).

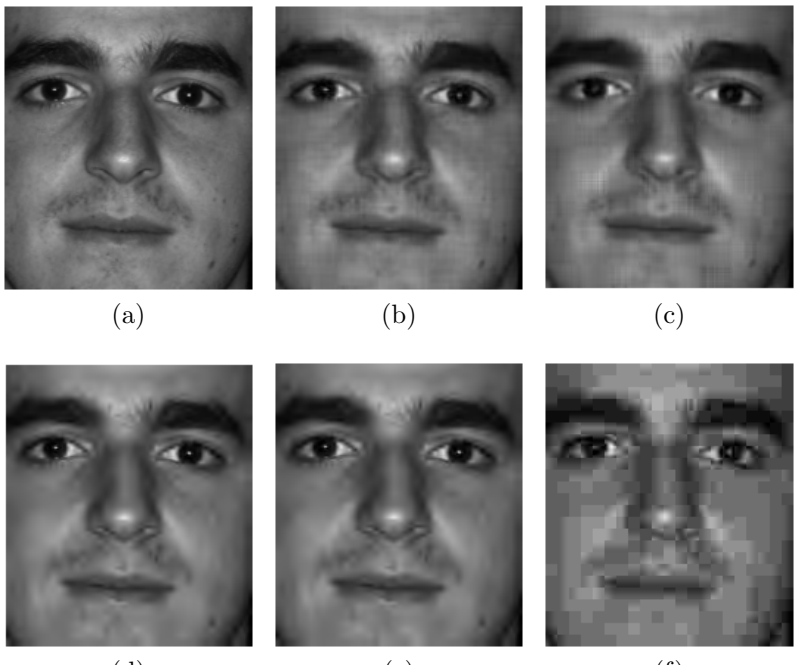


Figure 4.11: (a) Original image; (b) tensor coding with PCP (979 Bytes, PSNR: 34.5); (c) tensor coding with CP (986 Bytes, PSNR: 32.6); (d) JPEG2000 (975 Bytes, PSNR: 35.27); (e) motion JPEG2000 (990 Bytes, PSNR: 35.1); (f) JPEG (999 Bytes, PSNR:29.1).

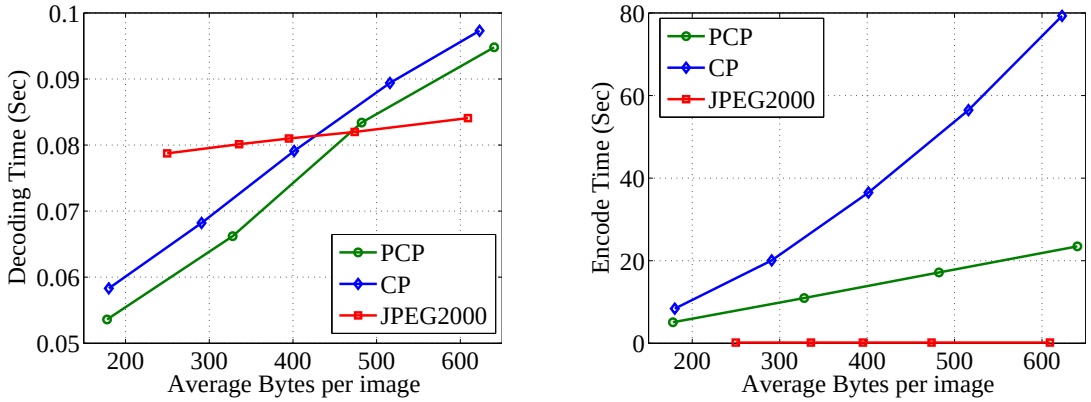


Figure 4.12: Average a) decoding b) encoding time of 38 persons face images from Yale Face Database B versus the average storage size required per image.

These simulations confirm the conjecture that one can achieve highly-efficient progressive coding of image ensembles while maintaining low-complexity random access to any desired image when employing tensor-based decomposition.

4.3 Low complexity video coding

The proposed framework does not rely on any form of Motion Estimation (ME) or Motion Compensation (MC). It can be targeted for applications and devices that tolerate delay but require low-complexity at both the encoder and decoder. We show that for low rank videos, TC outperforms the video coding schemes that do not employ ME. Among such video coding schemes, H.264/AVC-Intra [58, 59] are based on coding each frame without exploiting temporal redundancy. Distributed Coding for Video Services (DISCOVER) codec [60] is based on Wyner-Ziv coding with side information [61, 62]. H.264/AVC-no motion [58] codes a Group Of Pictures (GOP) such that the first frame is coded as a reference (i.e. I frame) and the other frames in the same GOP are coded as predicted pictures. This latter coding scheme exploits the temporal redundancy without employing any ME. Another set of low

complexity video coding schemes are based on dimensionality reduction by means of multidimensional decomposition. Two-dimensional Singular Value Decomposition (2DSVD) [14,63] video coding is one of the recent developed frameworks in this family [13].

4.3.1 Experimental results

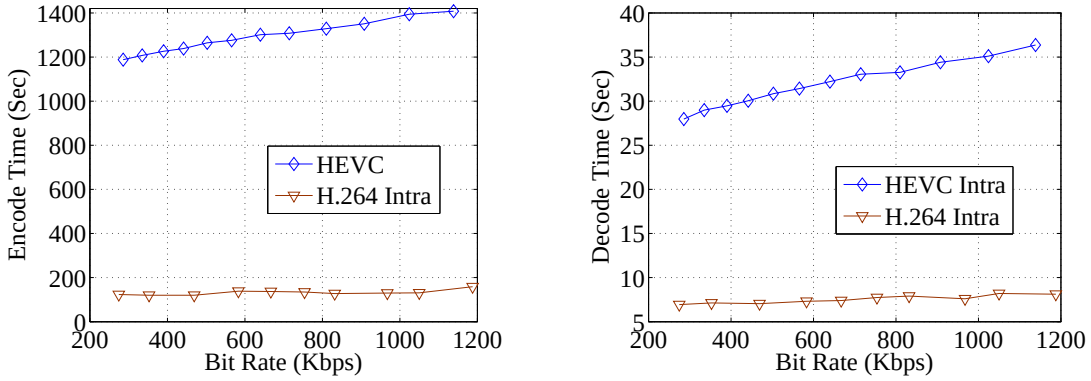
In our simulations, we simply employed JPEG2000 to compress the eigenfibers' matrix. Huffman and run-length coding was applied to compress the rank-values R_j. The proposed tensor coding was evaluated by comparing it with four encoders. They have in common the property of low complexity encoding. We did not show comparison results with the High Efficiency Video Coding (HEVC) standard because of its encoding complexity. Even Intra coding of HEVC has higher complexity in comparison with H.264. Figure 4.13 shows (a) encoding, (b) decoding time comparison between HEVC Intra and H.264 Intra for Container video. While HEVC Intra is significantly more complex than H.264 Intra, it achieves 1.24 dB average PSNR increase. In this experiment HHI HEVC software revision 3604 [64] was used at a desktop computer with 12 GB of memory and an Intel Core i7 2600 CPU (8MB Cache, 3.4 GHz).

The following five video codecs were used in our simulations:

1. H.264/AVC-no motion with high profile, GOP of size 24 and number of reference frames equals to one. The JM18.4 version implemented in C/C++ available at [65] was used.

2. H.264/AVC-Intra with high profile (JM18.4 version).

3. Scalable video coding with 3D set partitioning in hierarchical trees (3D SPIHT) [66–68]. The C/C++ implementation is available at [69]. In this implementation the GOP size



(a) (b)

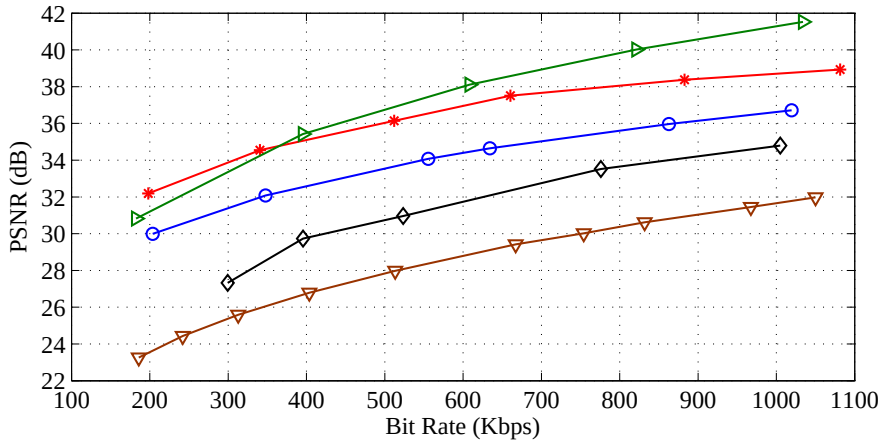
Figure 4.13: HEVC Intra and H.264 Intra time complexity comparison for container video. (a) Encoding time, (b) Decoding time.

is 16. The value of bit per pixel (bpp) was changed accordingly to achieve different bitrates.

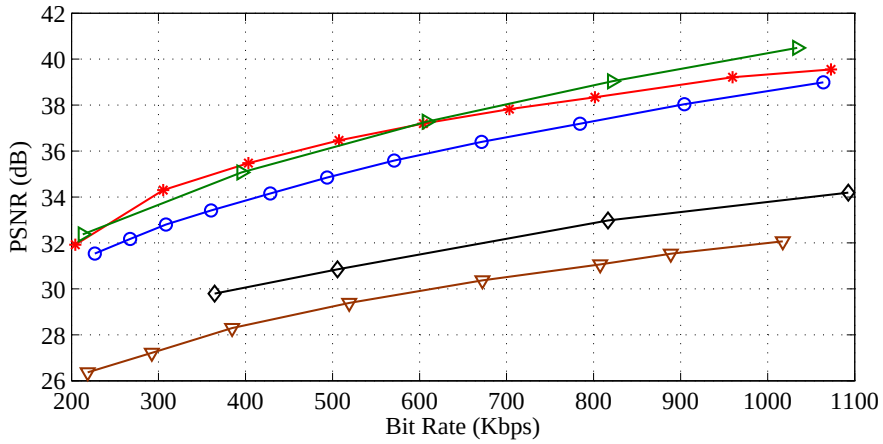
4. DISCOVER DVC intra codec [60] with GOP size of two. The C/C++ implementation available at [70] was used.

5. The proposed tensor coding framework implemented in MATLAB.

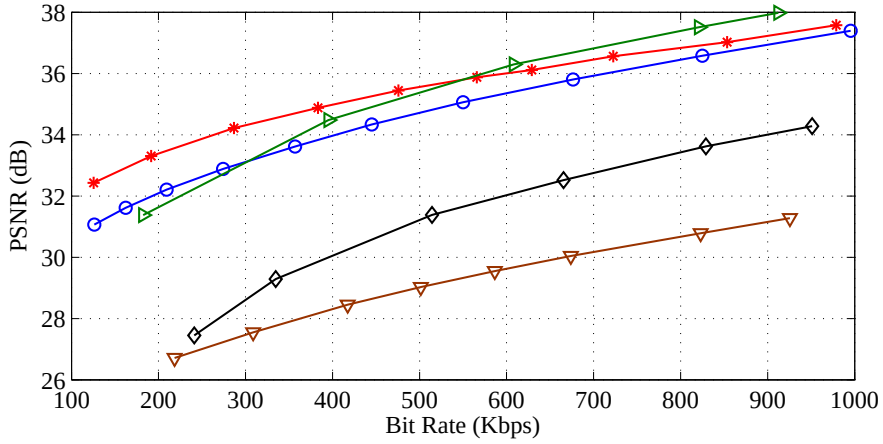
First, we present the PSNR results and corresponding plots for 180 frames of four CIF videos available at [71–73]. Note that the DISCOVER codec requires odd number of frames to code them. We used 179 frames of the test video sequences when coding with DISCOVER codec. Also, the parameters were assigned in a way to keep the PSNR values of both key and Wyner/Ziv frames close together. For tensor coding, we changed the block size to obtain the highest possible PSNR at different bitrates.



(a)



(b)



(c)

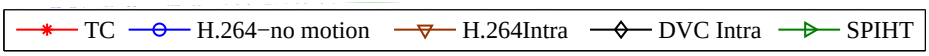


Figure 4.14: PSNR vs. bitrate plots of (a) Container,(b) Silent, (c) Bridge Close video.

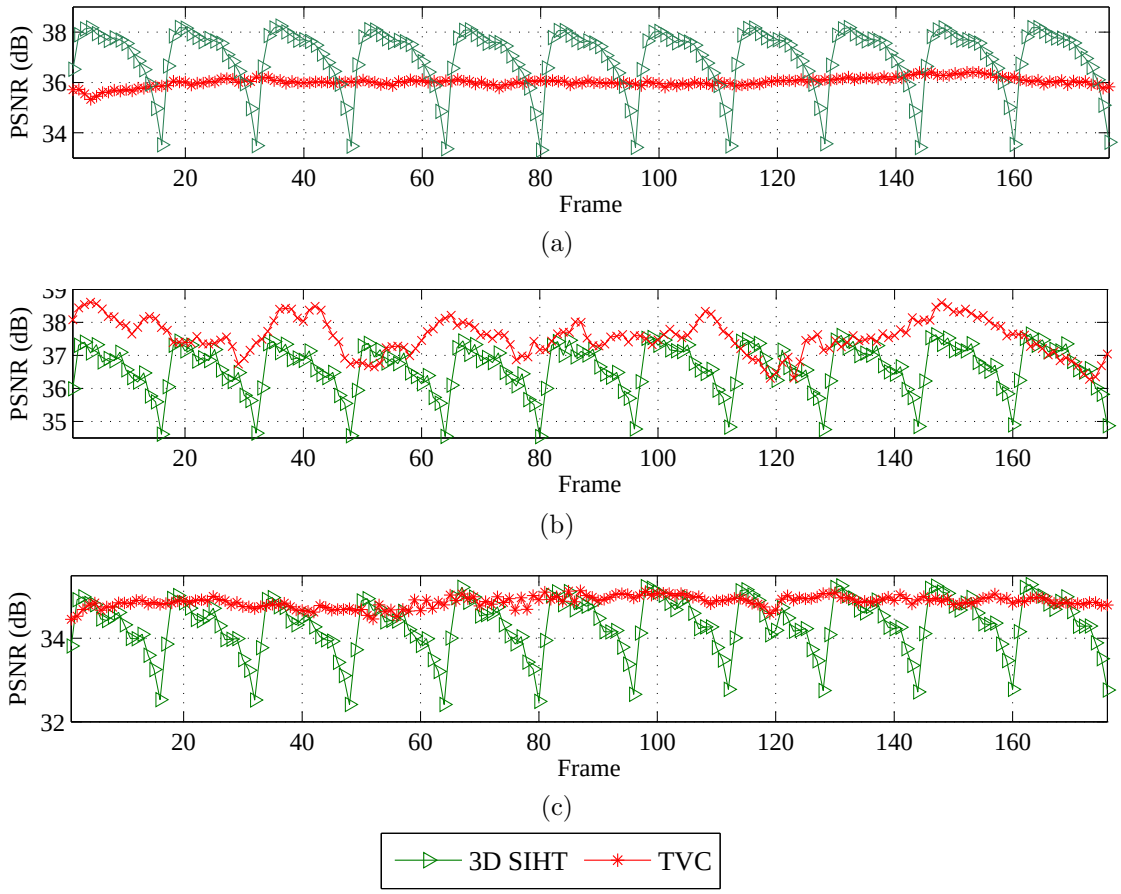
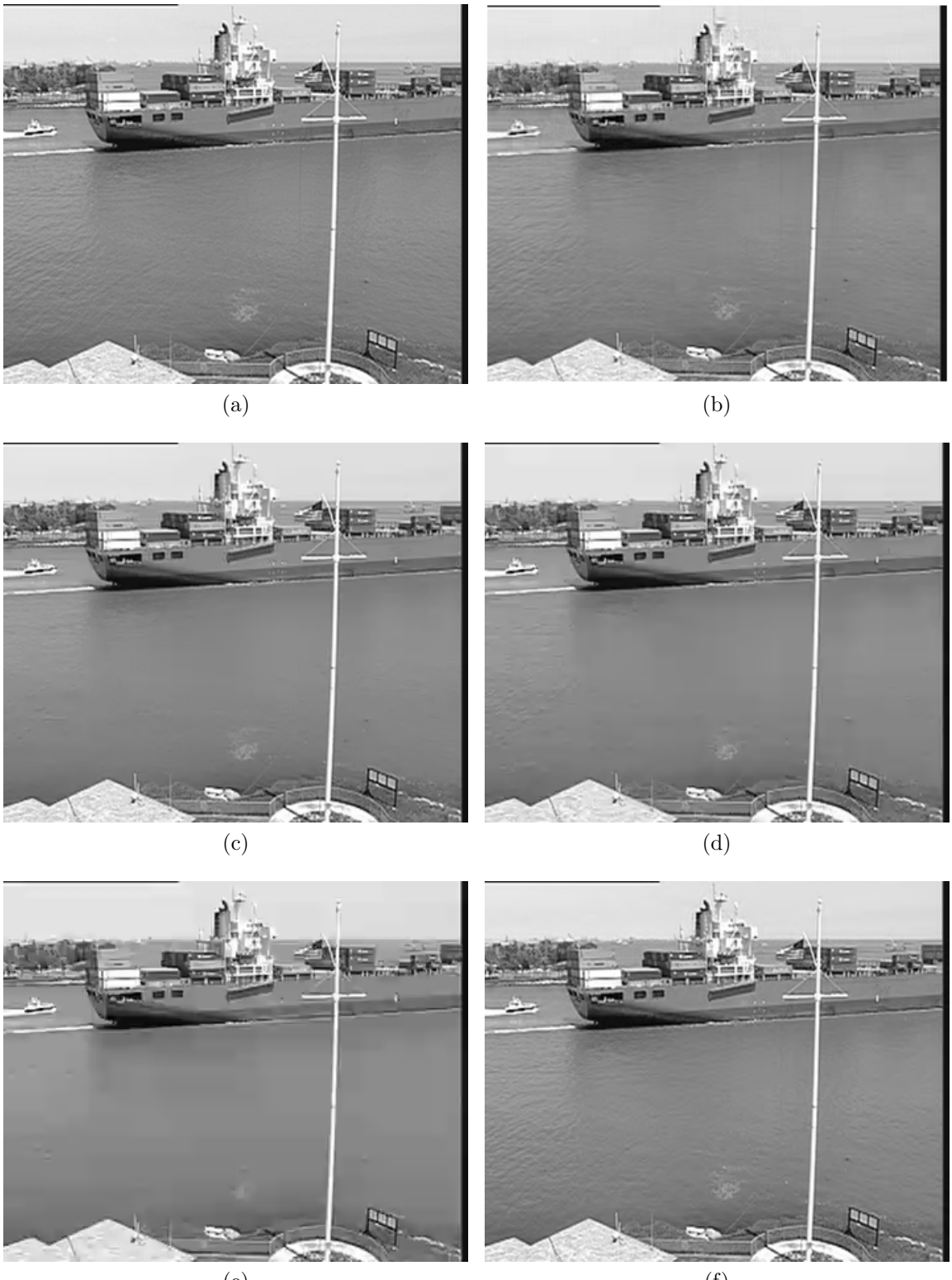


Figure 4.15: PSNR vs. frames plots of (a) Container (500 Kbps), (b) Silent (699 Kbps), (c) Bridge Close (380 Kbps) videos encoded with TC and SPIHT. Tensor coding maintains a more uniform PSNR across the frames while the SPIHT quality tends to degrade as it get closer to the end of the GOP.

The PSNR plots are shown in Figure 4.14. Our proposed method outperformed all other encoders except for higher bitrates of the Container video where 3D SPIHT provided higher *average* PSNR. Even though 3D SPIHT *on average* performed better than our method, it suffers from wide PSNR variation within each GOP, and with very visible degradation that is quite significant toward the end of the GOP. In particular, 3D SPIHT results in high PSNR differences among the frames at the beginning of the GOP in comparison with frames at the end of the GOP, especially for videos that have frames that tend to change within a GOP. Figure 4.15 shows the PSNR versus frame sequence of Container and Bridge Close



(a) (b) (c) (d) (e) (f)

Figure 4.16: Frame 14 of the Container CIF video. (a) original video, (b) tensor coding (731.07 Kbps; 37.39 dB), (c) H.264/AVC-no-motion (732.09 Kbps; 35.26 dB), (d)DISCOVER DVC (735.53 Kbps; 34.06 dB), (e) H.264/AVC-Intra (753.96 Kbps; 30.02 dB), (f) SPIHT (732.94 Kbps; 38.22 dB).

(a) (b)

(c) (d)

(e) (f)

Figure 4.17: The first frame of the Bridge Close CIF video. (a) original frame, (b) tensor coding (202.74 Kbps; 33.48 dB), (c) H.264/AVC-no-motion (209.29 Kbps; 32.21 dB), (d) DISCOVERDVC (274.77 Kbps; 28.32 dB), (e) H.264/AVC-Intra (218.41 Kbps; 26.71 dB), (f) SPIHT (203.77 Kbps; 31.81 dB).

videos encoded using tensor coding and 3D SPIHT. At low bitrates, the PSNR variation of 3D SPIHT was noticeable. Hence, such wide variation in visual quality is arguably unacceptable for many applications.

Figure 4.16 shows frame number 14 of container video. The original frame was compared with the decoded frame of five different codecs in this case. The video was coded at approximately 730 Kbps. For this particular frame, tensor coding had lower PSNR at this bitrate than the 3D SPIHT. Meanwhile, tensor coding decoded the frame with more details (especially the water region) in comparison with the other decoders.

Figure 4.17 shows the first frame of "Bridge Close" video. The original frame was compared with the decoded frame of the five encoders as in the previous experiment. The video was coded at approximately 210 Kbps. In this example, tensor coding outperformed all other five codecs both in terms of PSNR and visual quality. The details of the bridge are more visible in the frame decoded with tensor coding in comparison to the other methods. Due to the aforementioned issues with 3D SPIHT, and in particular the significant variation in visual quality within each GOP, we focus the remainder of this section on comparing tensor coding with the other three leading coding systems.

In another experiment, we encoded ten CIF videos at two levels of PSNR to compare the bitrate and the time complexity. The results for the relatively low PSNR are shown in Table 4.2. Table 4.3 shows the results corresponding to encoding the same videos but at relatively higher PSNR values. The encoders were evaluated at a desktop computer with 12 GB of memory and an Intel Core i7 2600 CPU (8MB Cache, 3.4 GHz). Tensor coding provides the best bitrate results and very competitive encoder/decoder times at low bitrates. As expected, DISCOVER DVC provided the best encoding times but at a significant penalty at the decoder side. For videos with low motion, tensor coding encoding time was even

smaller than DISCOVER DVC. However at higher bitrates the encoding time was higher than other methods. The tensor coding decoding time was consistently the lowest at both the low and high PSNR levels. Overall, tensor coding provided a good balance of coding efficiency and low-complexity despite its MATLAB implementation.

Table 4.2: The encoding/decoding time, PSNR and bitrate of ten CIF videos with 180 frames using four coding methods. Relatively-low and (almost) the same PSNR values were targeted for this experiment to compare the bitrate and time complexity.

Video	**Method**	**PSNR(dB)**	**bitrate (Kbps)**	**Encode time(s)**	**Decode time(s)**
Container	H.264/AVC-no motion	30.35	276	77	2.9
	H.264/AVC-Intra	30.02	754	96	7.7
	DISCOVER	30.29	424	**58**	803
	TC	30.18	**114**	81	**1.4**
Silent	H.264/AVC-no motion	30.96	192	88	4.2
	H.264/AVC-Intra	30.37	672	111	9.6
	DISCOVER	30.45	456	**54**	1367
	TC	30.97	**116**	68	**1.3**
Mother and Daughter	H.264/AVC-no motion	32.55	120	72	3.62
	H.264/AVC-Intra	32.38	284	89	7.7
	DISCOVER	32.94	227	**54**	1059
	TC	32.64	**98**	65	**1.2**
Bridge Close	H.264/AVC-no motion	32.12	209	75	2.5
	H.264/AVC-Intra	32.6	1264	104	8.3
	DISCOVER	32.64	788	**64**	1173
	TC	32.29	**100**	**64**	**1.3**
Bridge far	H.264/AVC-no motion	37.02	51	76	2.7
	H.264/AVC-Intra	37.03	638	97	7.9
	DISCOVER	36.84	493	58	1517

Table 4.2: (cont'd)

	TC	37.03	**33**	**25**	**0.7**
Waterfall	H.264/AVC-no motion	26.53	219	81	3.6
	H.264/AVC-Intra	26.43	433	93	8.7
	DISCOVER	26.37	276	**53**	971
	TC	26.68	**148**	80	**1.7**
Vassar View 0	H.264/AVC-no motion	32.56	109	89	3.23
	H.264/AVC-Intra	32.01	588	114	9.1
	DISCOVER	32.29	382	68	995
	TC	32.43	**83**	**45**	**1.1**
News	H.264/AVC-no motion	32.44	231	76	3.4
	H.264/AVC-Intra	32	785	97	7.9
	DISCOVER	32.04	486	**59**	1044
	TC	32.49	**230**	113	**2.5**
Red Flower	H.264/AVC-no motion	32.52	124	84	2.4
	H.264/AVC-Intra	32.06	1645	109	9
	DISCOVER	32.18	776	73	791
	TC	32.48	**100**	**38**	**1.2**
Hall	H.264/AVC-no motion	32.68	117	86	3.1
	H.264/AVC-Intra	32.11	672	95	7.8
	DISCOVER	32.38	434	58	922
	TC	32.67	**98**	**36**	**1.2**

Table 4.3: The encoding/decoding time, PSNR and bitrate of ten CIF videos with 180 frames using four coding methods. Relatively-high and (almost) the same PSNR values were targeted for this experiment to compare the bitrate and time complexity.

Video	**Method**	**PSNR(dB)**	**bitrate (Kbps)**	**Encode time(s)**	**Decode time(s)**
Container	H.264/AVC-no motion	39.88	1902	125	6.5
	H.264/AVC-Intra	38.91	3184	125	9.6
	DISCOVER	39.47	1800	**100**	1020
	TC	39.36	**1214**	448	**3.1**
Silent	H.264/AVC-no motion	39.74	1216	118	5.5
	H.264/AVC-Intra	39.16	3749	151	11.5
	DISCOVER	38.74	2074	**101**	1843
	TC	39.73	**1112**	320	**2.4**
Mother and Daughter	H.264/AVC-no motion	39.28	776	88	5.3
	H.264/AVC-Intra	39.70	1203	116	9.6
	DISCOVER	39.62	740	**72**	1143
	TC	39.97	**736**	154	**1.9**
Bridge Close	H.264/AVC-no motion	39.94	2038	125	5.9
	H.264/AVC-Intra	39.28	3964	137	10.4
	DISCOVER	39.28	2665	**119**	2069
	TC	39.96	**1975**	329	**3.73**
Bridge far	H.264/AVC-no motion	39.08	446	95	4.6
	H.264/AVC-Intra	39.23	1370	122	10.1
	DISCOVER	39.31	1045	**70**	2092

Table 4.3: (cont'd)

	TC	39.20	**332**	208	**3.6**
Waterfall	H.264/AVC-no motion	32.44	1984	112	8.6
	H.264/AVC-Intra	32.66	2389	129	10.9
	DISCOVER	32.38	**1093**	**72**	1219
	TC	32.50	1291	373	**3.4**
Vassar View 0	H.264/AVC-no motion	39.32	695	116	4.3
	H.264/AVC-Intra	39.17	2733	141	10.2
	DISCOVER	39.01	1799	**92**	1719
	TC	39.22	**636**	183	**1.9**
News	H.264/AVC-no motion	39.52	879	97	4.6
	H.264/AVC-Intra	39.38	2085	112	9.4
	DISCOVER	39.14	1308	**87**	1749
	TC	39.91	**850**	188	**2.2**
Red Flower	H.264/AVC-no motion	39.18	598	110	3.3
	H.264/AVC-Intra	39.06	4159	137	10.9
	DISCOVER	39.42	2169	99	1465
	TC	39.86	**494**	**87**	**1.6**
Hall	H.264/AVC-no motion	39.89	736	104	5.1
	H.264/AVC-Intra	39.43	1990	116	9.4
	DISCOVER	39.26	1467	**75**	2703
	TC	39.84	**634**	122	**2.2**

Chapter 5

Super-resolution for reconstructing high frequency data

We first present a brief introduction to spatial ISV in the context of the proposed framework. Subsequently, the overall architecture of the proposed super-resolution based spatial ISV system is explained in details.

5.1 Spatial ISV with Hybrid Super and Base Frames

Figure 5.1 shows the proposed hybrid two-layer spatial ISV with super and base frames. The base layer ($B(F_i)$) is the sequence of the down-sampled video where the F_i is the frame i. The down-sample ratio can be any reasonable value. The enhancement layer ($E(F_i)$) is the difference between the up-sampled base layer and the original video. In other words, the enhancement layer carries the high frequency components. Dropping the enhancement layer can reduce the required bandwidth to stream the video. However the mere up-sampled base layer would result in blurry frame and noticeable quality drop.

In the proposed framework, a few frames are encoded and/or transmitted/received with the enhancement layer. We refer to such frames as super-frames. The remaining frames are encoded and/or transmitted/received using the base layer only. We call these frames base-frames. Hence, the decoder receives a hybrid of super-frames and base-frames. This

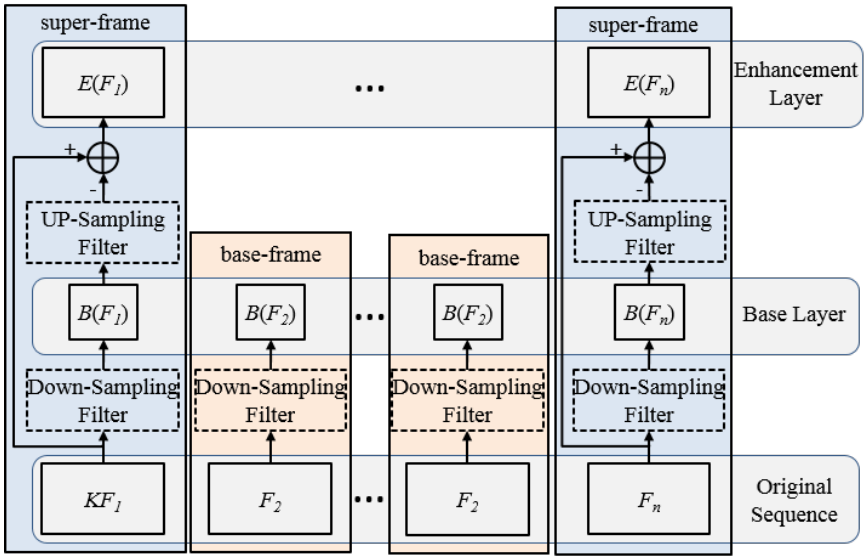


Figure 5.1: Encoder diagram of the proposed spatial ISV with hybrid two-layer spatial video coding that consist of the super-frames and the base-frames. The process of generating the base and the enhancement layer is shown.

encoding framework would enable the use of high frequency data in super-frames to enhance the base-frames. In other words, the super-frames can be used to build a dictionary for the super-resolution method. Encoding a few frames with enhancement layer enables saving bandwidth and/or reacting to changing network conditions.

Under the proposed framework, a Group Of Pictures (GOP) is defined as a set of one super-frame and all remainder frames are base-frames. In general, the super-frame could be placed within anywhere with a GOP. However, for simplicity of implementation and simulation, we place the super-frame as the first frame within a GOP. It is important to note that a larger GOP size would result in a smaller required bandwidth. At the receiver side, the decoded video consists of frames with two different spatial sizes. The super-frames are added to the display sequence directly since they have the same resolution as of the final display sequence. They are also used to extract the high frequency data. The base-frames are up-sampled and then super resolved using the proposed framework. Finally, they are

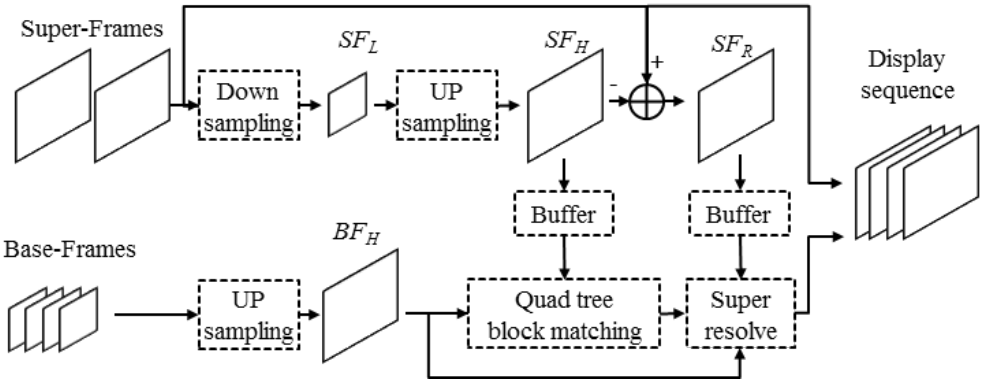


Figure 5.2: The proposed decoder structure with super-resolution framework for a two-layer ISV encoded video. The final reconstructed display video consists of super-frames and base-frames. The super-frames are added directly while the base-frames are super resolved to improve their quality.

added to the display sequence. Figure 5.2 shows the proposed decoder structure and the process of reconstructing the final display video.

5.2 Super-resolution framework for ISV

The proposed framework is similar to an unsharp masking where the blurred copy of the frame is obtained by a down-sampling filter followed by an up-sampling filter. A bilinear or bicubic filter can be used. The unsharp mask SF_R is obtained using the difference between the decoded super-frame and an up-sampled version of its low-resolution picture as in eq. (5.1).

$$SF_R = SF - SF_H \tag{5.1}$$

Here, SF_R is the residual (unsharp mask), SF is the decoded super-frame, and $SF_H = UP(DOWN(SF))$. Both SF_R and SF_H are stored in buffers to be used for super resolving base-frames. Once a new super-frame is decoded, the process of extracting high frequency data will be repeated and the buffers will be updated with the new data.

Each base-frame was up-sampled and then super resolved. The super-resolution was applied in a block-wise approach. We used the blocks that are defined by the quad tree structure of the encoder. The blocks are of different sizes ranging from 4×4 up to 64×64. Since the quad tree structure is on the base-frame with smaller size, up-sampling results in larger block sizes. For example if the base-frame is scaled up by 2 at each direction, then the block sizes would be from 8×8 up to 128×128. Based on experiment, we find out that in this case, dividing the block size by two at each direction and maintaining the 16×8 and 8×16 as the minimum block size gives the best result.

For each block of BF_H, a block-wise search was applied to find a match in SF_H. Here, BF_H is the up-sampled base-frame. The window of the search can be restricted to decrease the complexity. In our experiments, the window was 64 pixels from each direction of the current block. Sum of the absolute value was used as the error measure. Eq. (5.2) shows the block-matching minimization.

$$k^* = \underset{k}{argmin}\, SUM\left(\left|SF_H^k - BF_H^j\right|\right) \tag{5.2}$$

Where j is the index of the block to be super resolved, k is the index of a matching block in SF_H, $SUM()$ is the sum of all the elements in a block. After finding k, we super resolve block j by adding the corresponding high frequency data from SF_R:

$$\hat{F}^j = BF^j + SF_R^k \tag{5.3}$$

The overall proposed algorithm is as follow:

Algorithm 3 super-resolution for ISV

Input: Hybrid spatial SVC encoded video.

Output: Super resolved decoded video

if super-frame **then**

Decode the frame and send to the output stream

Compute and store SF_H in a buffer.

Compute SF_R as in Eq. (5.1) and store the result in a buffer.

else

Decode and up-sample the frame

for block k in the quad tree structure **do**

Search for a match in SF_H as in Eq. (5.2)

Find the corresponding high frequency data from SF_R.

Super resolve the current block as in Eq. (5.3).

end for

Send the super resolved base-frame to the output stream.

end if

5.3 Experimental results

The proposed framework was implemented within spatial SVC of VP9 version 1.3.0. A video was encoded in two spatial layers. The base layer was half of the display video at each direction. The enhancement layer was the same size as the display video. The enhancement layer of the base-frames was dropped based on the GOP size. GOP sizes 2, 5, 10, 15, 20, 25, and 30 were used for comparison.

The result of using quad tree blocks was compared with various fixed-size blocks. The

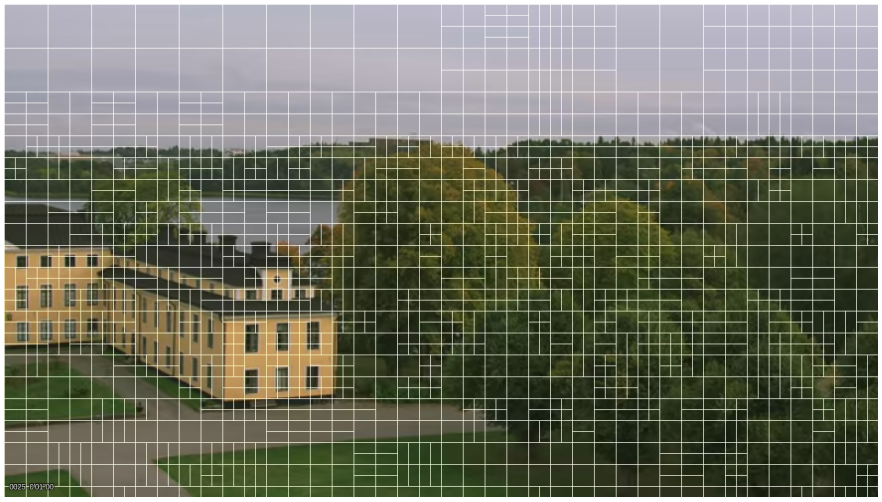


Figure 5.3: The quad tree block structure which was used for the proposed super-resolution framework. The frame is the 25th frame of the in_to_tree.

block sizes were 8×8, 16×16, 32×32, and 64×64. The quad tree block was divided into four sub-blocks to compensate for up-sampling. Figure 5.3 shows the structure of the quad tree on the 25th frame of the in_to_tree video. In this experiment 150 frames of in_to_tree 720p video were encoded at 4 Mbps. The search box for block matching was 64 pixels from each direction. For down and up sampling we used bilinear interpolation filters. Figure 5.4 shows the PSNR result versus GOP size. Note that for the smaller GOP, larger block size has higher PSNR result while for larger GOP the larger blocks result in lower PSNR. This is mainly because of the fact that at a longer distance from the super-frame, finding a good match for larger block sizes becomes harder. Using the quad tree keeps a good balance between the smaller and larger GOP.

In another experiment, 150 frames of in_to_tree, shields, and old_town videos were encoded with VP9 SVC at various bitrates. All the videos were 720p. They were encoded with two layers. The base layer was half of the enhancement layer at each direction. The enhancement layer of the base-frames was dropped. Then at the decoder, the following methods were used to up-sample the base-frames:

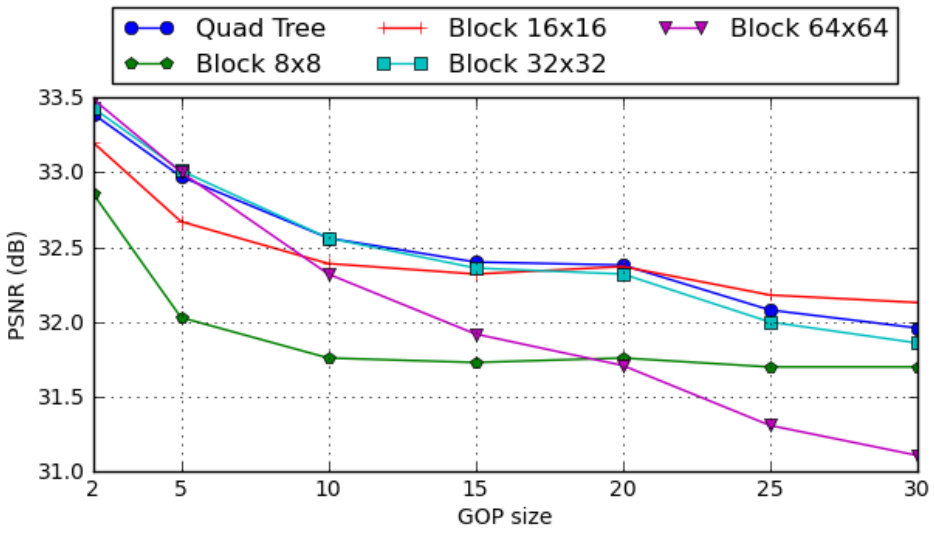
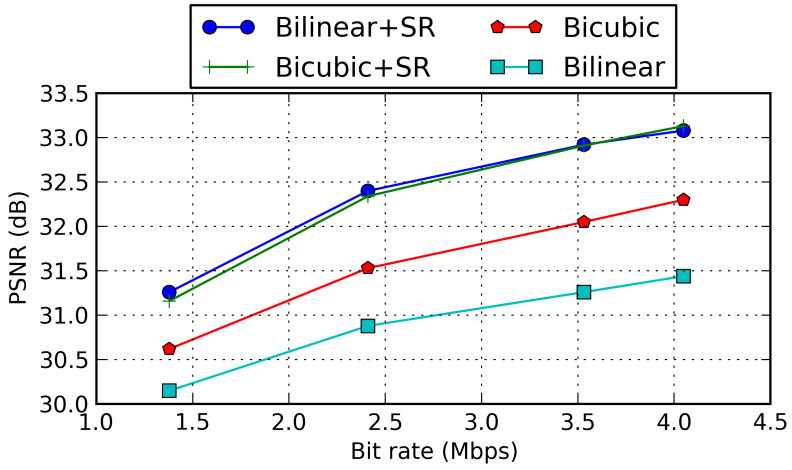


Figure 5.4: PSNR comparison of the proposed super-resolution framework when applied to quad tree and various fixed size blocks.

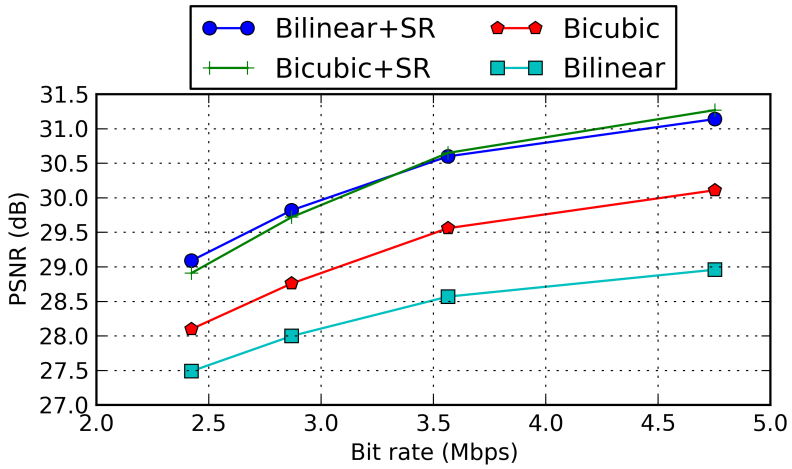
- Bilinear interpolation
- Bicubic interpolation
- Bilinear interpolation and the proposed super-resolution
- Bicubic interpolation and the proposed super-resolution

Figure 5.5 shows the PSNR vs. bitrate plots that compares the above up-sampling methods. In this experiment the GOP size was 15. Note that using bilinear or bicubic up-sampling with the proposed super-resolution framework, changes the PSNR result slightly. This is because of the fact that using bicubic up-sampling would result in more high frequency components in SF_H and consequently less high frequency components in SF_R. Also, BF_H in eq. (5.3) would have more high frequency components due to bicubic up-sampling. However, the SF_R high frequency components decrease almost cancel out the difference.

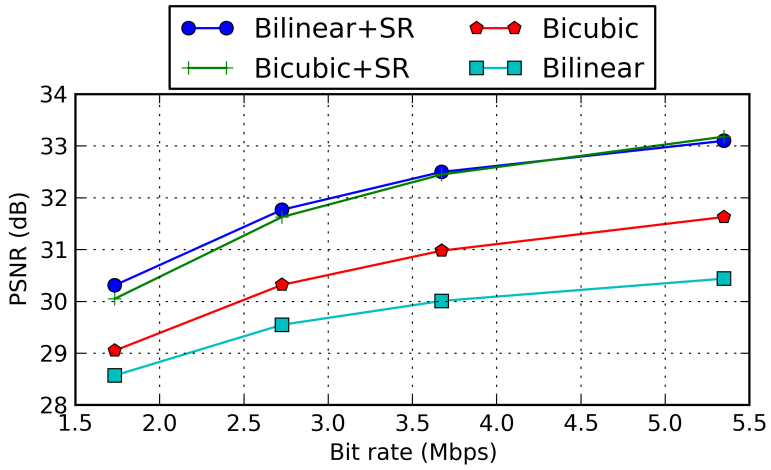
Table 5.1 shows the PSNR comparison between bicubic up-sampling and the proposed super-resolution framework for various videos with GOP of size 5.



(a)

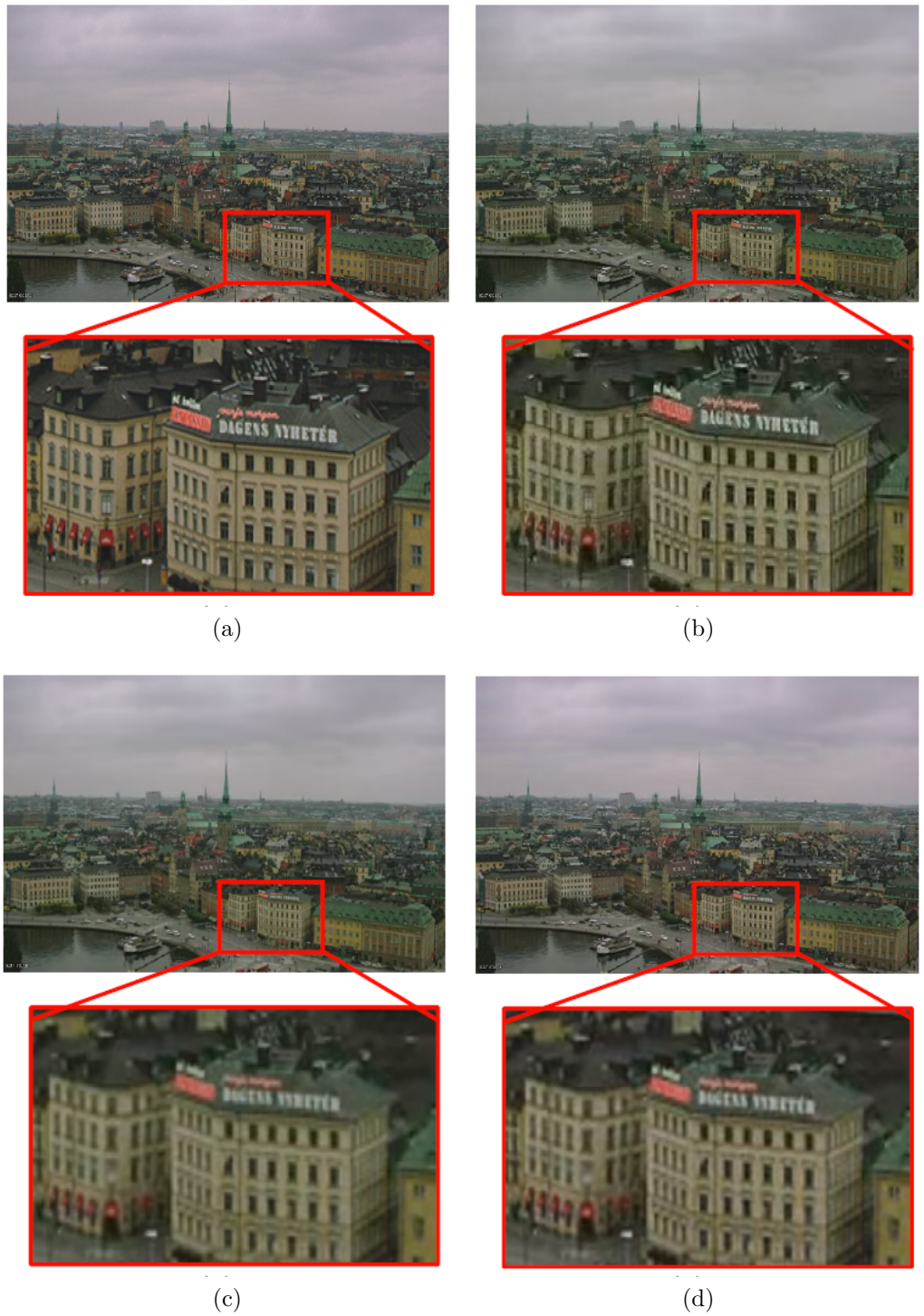


(b)



(c)

Figure 5.5: PSNR results of the super-resolution with bilinear up-sampling, super-resolution with Bicubic up-sampling, bilinear up-sampling, and Bicubic up-sampling of (a) Intotree, (b) shields, and (c) old_town.



(a) (b)

(c) (d)

Figure 5.6: Frame 27 of the (a) original, (b) bilinear up-sampling and the proposed super-resolution framework, (c) bilinear up-sampling, (d) bicubic up-sampling, of the old_town video. For (b), (c), and (d), the video is encoded at 2.7 Mbps.

Table 5.1: PSNR comparison between bicubic up-sampling and the proposed super-resolution framework.

Videos	**SR**	**Bicubic**	**Gain**
old_town_720p	34.33	32.71	1.62
shields_720p	32.82	31.31	1.51
Stockholm_720p	32.71	31.27	1.44
in_to_tree_720p	34.28	33.24	1.04
station_1080p	37.50	36.75	0.75
parkrun_720p	25.59	25.24	0.35
Pedestrian_area_1080p	36.51	36.16	0.35
crowd_run_1080p	26.9	26.6	0.3
tractor_1080p	33.21	32.96	0.25

Figure 5.6 shows frame 27 of the old_town video. The result of bilinear up-sampling, bicubic up-sampling, bilinear up-sampling and the proposed framework were compared with the original frame. The video was encoded at 2.7 Mbps. In the highlighted part, it is obvious that the proposed framework reconstructed the frame with more high frequency components.

Chapter 6

Conclusion

We presented tensor coding as a low complexity framework for coding high dimensional visual data. Tensor coding is based on a proposed Progressive Canonical-decomposition Parallel-factor (PCP) decomposition that reduces an n dimensional tensor onto a 2D data set. The proposed PCP was applied in block-wise approach.

Two tensor partitioning methods which are uniform and adaptive tree were employed to divide the input tensor into a set of sub-tensor blocks. The uniform partitioning divide a tensor into a set of equal size blocks. The adaptive tree partitioning employs a top down approach to divide the tensor. At each level it decide whether a block need to be sub-divided into a set of smaller blocks or not. The decision is made based on a developed weighted directional variance and the allocated number of rank-one tensors. A 2^n-ary tree structure was used to store the partitioning information.

The partitioned sub-tensors have different amount of information and rank. Therefore they may require different number of rank-one tensors. The proposed framework utilizes a greedy algorithm to search for the global optimal number of rank-one tensors that represent all of the blocks of a tensor. After applying the PCP to all the blocks, the decomposed eigenfibers are arranged and then further compressed by a 2D compression method. The required side information for decoding were stored in a header file and then entropy coded.

The proposed tensor coding framework was supplemented with a residual coding module to enable near lossless compression. This addition can benefit the applications that require a

fine level of details. The residual coding module encoded the tensor's slices separately since the PCP has already captured the present correlation among the slices.

An important aspect of the proposed tensor coding framework is its desirable properties. We showed that the framework can achieve random access, progressive reconstruction or scalability, and low complexity reconstruction. These properties play an important role in applications like online browsing, streaming and scientific analysis.

We applied the proposed tensor coding framework to represent and code three type of data sets which were hyperspectral/multispectral images, bio-metric face images, and low motion videos. For each application we showed that the proposed tensor coding framework can achieve higher quality reconstruction, specially at the low bitrates, in comparison with the standard compression methods.

Our experimental results for the 4D multispectral images showed a possibility of three order of magnitude compression. This significant compression ratio is possible by taking advantage of the existing correlation along all the dimensions. The lower dimensions compression methods can not achieve this level of compression. Although the quality of the reconstructed image at this compression level was degraded, it can be acceptable for some applications like course level browsing and classification.

On the second part of this thesis, we proposed a super-resolution method for spatial inconsistent scalable video streaming. In this streaming scenario some of the frames can lose their enhancement layer due to network connection quality. This leads to an undesirable experience where some of the frames are at high quality while the others are not. In the worst case the streaming can keep switching between the two qualities in an effort to mitigate the network congestion.

The proposed method was used to reconstruct the frames with dropped enhancement

layer. This was done by using the high frequency data of the frames with enhancement layer (super-frames) as dictionary. The frames without enhancement layer (base-frame) were first up-sampled and then super resolved. The super resolution method was based on searching the dictionary for a match and then adding the corresponding high frequency data. The quad tree structure of the current frame was used for block-matching search with super-frame.

The proposed super resolution framework was integrated with Google VP9 video codec. Then we applied the framework to various High Definition (HD) videos to estimate the dropped enhancement layer. Our simulation results show an improvement visually and in terms of PSNR over traditional interpolation up-sampling filters.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.

[2] R. M. Bowen and C.-C. Wang, *Introduction to vectors and tensors*. Courier Corporation, 2008, vol. 2.

[3] A. Cichocki, D. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, and H. A. Phan, "Tensor decompositions for signal processing applications: From two-way to multiway component analysis," *Signal Processing Magazine, IEEE*, vol. 32, no. 2, pp. 145–163, 2015.

[4] O. Scherzer, *Handbook of Mathematical Methods in Imaging: Vol. 1*. Springer Science & Business Media, 2011.

[5] J. Liu, P. Musialski, P. Wonka, and J. Ye, "Tensor completion for estimating missing values in visual data," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 1, pp. 208–220, 2013.

[6] L. R. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, no. 3, pp. 279–311, 1966.

[7] J. D. Carroll and J.-J. Chang, "Analysis of individual differences in multidimensional scaling via an n-way generalization of eckart-young decomposition," *Psychometrika*, vol. 35, no. 3, pp. 283–319, 1970.

[8] R. A. Harshman, "Foundations of the parafac procedure: Models and conditions for an" explanatory" multi-modal factor analysis," 1970.

[9] A. Cichocki, D. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, and H. A. Phan, "Tensor decompositions for signal processing applications: From two-way to multiway component analysis," *Signal Processing Magazine, IEEE*, vol. 32, no. 2, pp. 145–163, 2015.

[10] A. T. Mahfoodh and H. Radha, "Tensor video coding," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 1724–1728.

[11] ——, "Compression of image ensembles using tensor decomposition," in *Picture Coding Symposium (PCS), 2013*. IEEE, 2013, pp. 21–24.

[12] S. Aja-Fernández, R. de Luis Garcia, D. Tao, and X. Li, *Tensors in image processing and computer vision*. Springer Science & Business Media, 2009.

[13] Z. Gu, W. Lin, B.-S. Lee, and C. Lau, "Low-complexity video coding based on two-dimensional singular value decomposition," *Image Processing, IEEE Transactions on*, vol. 21, no. 2, pp. 674–687, 2012.

[14] C. H. Ding and J. Ye, "2-dimensional singular value decomposition for 2d maps and images." in *SDM*. SIAM, 2005, pp. 32–43.

[15] B. Zhou, F. Zhang, and L. Peng, "Video dimension reduction and coding using multiple tensor rank-r decomposition," in *Image and Signal Processing (CISP), 2011 4th International Congress on*, vol. 1. IEEE, 2011, pp. 330–334.

[16] ——, "Compact representation for dynamic texture video coding using tensor method," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 23, no. 2, pp. 280–288, 2013.

[17] H. Wang and N. Ahuja, "Rank-r approximation of tensors using image-as-matrix representation," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2. IEEE, 2005, pp. 346–353.

[18] M. A. O. Vasilescu and D. Terzopoulos, "Multilinear analysis of image ensembles: Tensorfaces," in *Computer VisionECCV 2002*. Springer, 2002, pp. 447–460.

[19] C. Ding, H. Huang, and D. Luo, "Tensor reduction error analysisapplications to video compression and classification," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.

[20] H. Wang and N. Ahuja, "Compact representation of multidimensional data using tensor rank-one decomposition," *vectors*, vol. 1, p. 5, 2004.

[21] A. Karami, M. Yazdi, and G. Mercier, "Compression of hyperspectral images using discerete wavelet transform and tucker decomposition," *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of*, vol. 5, no. 2, pp. 444–450, 2012.

[22] L. Zhang, L. Zhang, D. Tao, X. Huang, and B. Du, "Compression of hyperspectral remote sensing images by tensor approach," *Neurocomputing*, vol. 147, pp. 358–363, 2015.

[23] H. Chen, W. Lei, S. Zhou, and Y. Zhang, "An optimal-truncation-based tucker decomposition method for hyperspectral image compression," in *Geoscience and Remote Sensing Symposium (IGARSS), 2012 IEEE International*. IEEE, 2012, pp. 4090–4093.

[24] L. Wang, J. Bai, J. Wu, and G. Jeon, "Hyperspectral image compression based on lapped transform and tucker decomposition," *Signal Processing: Image Communication*, vol. 36, pp. 63–69, 2015.

[25] T. Hazan, S. Polak, and A. Shashua, "Sparse image coding using a 3d non-negative tensor factorization," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 1. IEEE, 2005, pp. 50–57.

[26] H. Wang, Q. Wu, L. Shi, Y. Yu, and N. Ahuja, "Out-of-core tensor approximation of multi-dimensional matrices of visual data," in *ACM Transactions on Graphics (TOG)*, vol. 24, no. 3. ACM, 2005, pp. 527–535.

[27] K. Inoue and K. Urahama, "Dsvd: A tensor-based image compression and recognition method," in *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*. IEEE, 2005, pp. 6308–6311.

[28] J. Hou, L.-P. Chau, N. Magnenat-Thalmann, and Y. He, "Scalable and compact representation for motion capture data using tensor decomposition," *Signal Processing Letters, IEEE*, vol. 21, no. 3, pp. 255–259, 2014.

[29] S. K. Suter, J. A. I. Guitián, F. Marton, M. Agus, A. Elsener, C. P. Zollikofer, M. Gopi, E. Gobbetti, and R. Pajarola, "Interactive multiscale tensor reconstruction for multiresolution volume visualization," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 17, no. 12, pp. 2135–2143, 2011.

[30] Q. Wu, T. Xia, C. Chen, H.-Y. S. Lin, H. Wang, and Y. Yu, "Hierarchical tensor approximation of multi-dimensional visual data," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 14, no. 1, pp. 186–199, 2008.

[31] R. Sivalingam, D. Boley, V. Morellas, and N. Papanikolopoulos, "Tensor sparse coding for region covariances," in *Computer Vision–ECCV 2010*. Springer, 2010, pp. 722–735.

[32] V. Cisco, "Forecast and methodology, 2013–2018.(2014)."

[33] H. M. Radha, M. Van der Schaar, and Y. Chen, "The mpeg-4 fine-grained scalable video coding method for multimedia streaming over ip," *Multimedia, IEEE Transactions on*, vol. 3, no. 1, pp. 53–68, 2001.

[34] J.-R. Ohm and M. van der Schaar, "Scalable video coding," in *Tutorial material, Int. Conf. Image Processing ICIP*, vol. 2001, 2001.

[35] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the h. 264/avc standard," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 17, no. 9, pp. 1103–1120, 2007.

[36] D. Mukherjee, J. Bankoski, A. Grange, J. Han, J. Koleszar, P. Wilkins, Y. Xu, and R. Bultje, "The latest open-source video codec vp9-an overview and preliminary results," in *Picture Coding Symposium (PCS), 2013.* IEEE, 2013, pp. 390–393.

[37] "webm/libvpx," https://chromium.googlesource.com/webm/libvpx/, accessed: 2015-01-04.

[38] S. C. Park, M. K. Park, and M. G. Kang, "Super-resolution image reconstruction: a technical overview," *Signal Processing Magazine, IEEE*, vol. 20, no. 3, pp. 21–36, 2003.

[39] S. Farsiu, M. D. Robinson, M. Elad, and P. Milanfar, "Fast and robust multiframe super resolution," *Image processing, IEEE Transactions on*, vol. 13, no. 10, pp. 1327–1344, 2004.

[40] S. Baker and T. Kanade, "Limits on super-resolution and how to break them," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 9, pp. 1167–1183, 2002.

[41] E. M. Hung, R. L. De Queiroz, F. Brandi, K. F. De Oliveira, and D. Mukherjee, "Video super-resolution using codebooks derived from key-frames," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, no. 9, pp. 1321–1331, 2012.

[42] K. F. De Oliveira, F. Brandi, E. M. Hung, R. L. de Queiroz, and D. Mukherjee, "Bipredictive video super-resolution using key-frames," in *Proc. SPIE Symposium on Electronic Imaging, Visual Information Processing and Communication*, 2010.

[43] F. Brandi, R. de Queiroz, and D. Mukherjee, "Super-resolution of video using key frames and motion estimation," in *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on.* IEEE, 2008, pp. 321–324.

[44] W. T. Freeman, T. R. Jones, and E. C. Pasztor, "Example-based super-resolution," *Computer Graphics and Applications, IEEE*, vol. 22, no. 2, pp. 56–65, 2002.

[45] Z. Xiong, X. Sun, and F. Wu, "Robust web image/video super-resolution," *Image Processing, IEEE Transactions on*, vol. 19, no. 8, pp. 2017–2028, 2010.

[46] Q. Shan, Z. Li, J. Jia, and C.-K. Tang, "Fast image/video upsampling," in *ACM Transactions on Graphics (TOG)*, vol. 27, no. 5. ACM, 2008, p. 153.

[47] J. Håstad, "Tensor rank is np-complete," *Journal of Algorithms*, vol. 11, no. 4, pp. 644–654, 1990.

[48] M. J. Atallah and M. Blanton, *Algorithms and Theory of Computation Handbook, Volume 2: Special Topics and Techniques*. CRC press, 2009.

[49] A. Said and W. A. Pearlman, "Low-complexity waveform coding via alphabet and sample-set partitioning," in *Electronic Imaging'97*. International Society for Optics and Photonics, 1997, pp. 25–37.

[50] X. Tang, W. A. Pearlman, and J. W. Modestino, "Hyperspectral image compression using three-dimensional wavelet coding," in *Electronic Imaging 2003*. International Society for Optics and Photonics, 2003, pp. 1037–1047.

[51] E. Christophe, C. Mailhes, and P. Duhamel, "Hyperspectral image compression: adapting spiht and ezw to anisotropic 3-d wavelet coding," *Image Processing, IEEE Transactions on*, vol. 17, no. 12, pp. 2334–2346, 2008.

[52] X. Tang, S. Cho, and W. A. Pearlman, "3d set partitioning coding methods in hyperspectral image compression," in *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, vol. 2. IEEE, 2003, pp. II–239.

[53] X. Tang and W. A. Pearlman, "Three-dimensional wavelet-based compression of hyperspectral images," in *Hyperspectral Data Compression*. Springer, 2006, pp. 273–308.

[54] Q. Du and J. E. Fowler, "Hyperspectral image compression using jpeg2000 and principal component analysis," *Geoscience and Remote Sensing Letters, IEEE*, vol. 4, no. 2, pp. 201–205, 2007.

[55] Aviris - airborne visible / infrared imaging spectrometer - data. [Online]. Available: http://aviris.jpl.nasa.gov/data/index.html

[56] Landsat global archive consolidation. [Online]. Available: http://landsat.usgs.gov//Landsat_Global_Archive_Consolidation.php

[57] A. S. Georghiades, P. N. Belhumeur, and D. J. Kriegman, "From few to many: Illumination cone models for face recognition under variable lighting and pose," *Pattern*

Analysis and Machine Intelligence, IEEE Transactions on, vol. 23, no. 6, pp. 643–660, 2001.

[58] D. Marpe, T. Wiegand, and G. J. Sullivan, "The h. 264/mpeg4 advanced video coding standard and its applications," *Communications Magazine, IEEE*, vol. 44, no. 8, pp. 134–143, 2006.

[59] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the h. 264/avc video coding standard," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, no. 7, pp. 560–576, 2003.

[60] X. Artigas, J. Ascenso, M. Dalai, S. Klomp, D. Kubasov, and M. Ouaret, "The discover codec: architecture, techniques and evaluation," in *Picture Coding Symposium (PCS" 07)*, no. MMSPL-CONF-2009-014, 2007.

[61] L. Liu, Z. Li, and E. J. Delp, "Efficient and low-complexity surveillance video compression using backward-channel aware wyner-ziv video coding," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 19, no. 4, pp. 453–465, 2009.

[62] M. Tagliasacchi, A. Trapanese, S. Tubaro, J. Ascenso, C. Brites, and F. Pereira, "Intra mode decision based on spatio-temporal cues in pixel domain wyner-ziv video coding," in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, vol. 2. IEEE, 2006, pp. II–II.

[63] J. Ye, "Generalized low rank approximations of matrices," *Machine Learning*, vol. 61, no. 1-3, pp. 167–191, 2005.

[64] "HEVC," http://hevc.hhi.fraunhofer.de, accessed: 2013-08-09.

[65] "H.264/AVC JM Reference Software," http://iphome.hhi.de/suehring/tml/, accessed: 2013-01-18.

[66] A. Said, W. Pearlman *et al.*, "An image multiresolution representation for lossless and lossy compression," *Image Processing, IEEE Transactions on*, vol. 5, no. 9, pp. 1303–1310, 1996.

[67] B.-J. Kim, W. Pearlman *et al.*, "An embedded wavelet video coder using three-dimensional set partitioning in hierarchical trees (spiht)," in *Data Compression Conference, 1997. DCC'97. Proceedings.* IEEE, 1997, pp. 251–260.

[68] Y. Chen and W. A. Pearlman, "Three-dimensional subband coding of video using the zero-tree method," in *Visual Communications and Image Processing'96*. International Society for Optics and Photonics, 1996, pp. 1302–1312.

[69] "Image compression programs," http://www.cipr.rpi.edu/research/SPIHT/spiht3.html, accessed: 2013-09-02.

[70] "Discover," http://www.discoverdvc.org/, accessed: 2012-11-20.

[71] "Xiph.org video test media," http://media.xiph.org/video/derf/, accessed: 2012-11-20.

[72] "Yuv video sequences," http://www.discoverdvc.org/, accessed: 2012-11-20.

[73] "Video quality experts group (vqeg) - its," ftp://vqeg.its.bldrdoc.gov/MM/cif/, accessed: 2012-11-20.