University of Southern California

Ph.D. Dissertation

COMPRESSION OF SIGNAL ON GRAPHS WITH THE APPLICATION TO IMAGE AND VIDEO CODING

Author:
Yung-Hsuan Chao

Supervisor:
Dr. Antonio Ortega

A Dissertation Presented to the
FACULTY OF THE USC GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the Requirements for the Degree

Doctor of Philosophy
(ELECTRICAL ENGINEERING)

December 2017

Abstract

In this Ph.D. dissertation, we discuss several graph-based algorithms for transform coding in image and video compression applications. Graphs are generic data structures that are useful in representing signals in various applications. Different from the classic transforms such as Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT), graphs can represent signals on irregular and high dimensional domains, *e.g.* social networks, sensor networks. For regular signals such as images and videos, graphs can adapt to local characteristics such as edges and therefore provide more flexibility than conventional transforms. A frequency interpretation for signal on graphs can be derived using the Graph Fourier Transform (GFT). By properly adjusting the graph structure, *e.g.* connectivity and weights, based on signal characteristics, the GFT can provide compact representations even for signals with discontinuities. However, the GFT has high implementation complexity, making it less applicable in signals of large size, *e.g.* video sequences. In our work, we develop a transform coding scheme based on a low complexity lifting transform on graph. More specifically, we focus on two important problems in the design of a lifting transform, namely, the design of bipartition and the bipartite graph approximation. The two parts are optimized in terms of energy compaction for Gaussian Markov Random Field (GMRF), which has been widely utilized in modeling the statistics of image data.

As application, we consider two types of multimedia signals, including both regular and irregularly distributed signals. Among the first type of signal, we consider the compression of intra-predicted video residuals, which is regular with pixels residing on the 2D grid. However, these signals contain significant edge structures, which cannot be efficiently represented with existing transform coding standards. With the proposed graph lifting transform based on local edges, we demonstrate significant gains as compared to the state of the art DCT based coding, with comparable performance to that achieved by the high complexity GFT. We also discuss different types of edge models for video residuals and propose a new model for ramp edges, which shows promising results in GFT, as compared to the conventional step edge model. As a second type of signal, we propose a coding scheme for non-demosaicked light field images. Similar to the traditional digital camera, a light field camera captures color information using a photo sensor embedded with a color filter array (CFA). On the captured image, each pixel contains one single color component (out of R,G, and B) which are distributed based on Bayer pattern. However, through the conversion to an array of sub-aperture images, which is a representation commonly used for light field processing and display, the distribution of Bayer pattern no longer holds and pixels of each color component are distributed irregularly in space. In order to compress such data, a conventional scheme

using DCT requires demosaicking during conversion, which highly increases the amount of data for coding. With a graph based approach, the original signal can be efficiently encoded without any pre-processing step, avoiding the redundancies introduced by demosaicking. We also discuss an intra-prediction algorithm and optimal graph construction for irregularly spaced pixels. The results using the proposed scheme with graph based lifting transform show huge gains in compression as compared to DCT based coding in high bit rates, which are critical for archival scenario and instant camera storage.

To my family . . .

Acknowledgements

This dissertation would not have been possible without the help and support of my teachers, collaborators, family, and friends. First and foremost, I would like to thank my advisor Professor Antonio Ortega, for his supervision, patience, and guidance. His training on critical thinking, formulating and solving problems has helped me evolve from a naive student to a mature researcher. I wouldn't say the process was easy. Actually, there were some really tough moments when I felt lost, and almost gave up on my Ph.D. studies, but his considerable encouragement has kept me going. I could not have asked for a better advisor and mentor.

It was my pleasure to be given the opportunity to work with Professor Gene Cheung in National Institute of Informatics (NII), for the topic of light field image compression during the last year of my Ph.D.. The discussion with him have been insightful. The work would not be completed without his generous feedback and suggestions.

I would also like to thank Professor C.-C. Jay Kuo, and Professor Ramesh Govindan for serving as my dissertation committee members, as well as Professor Justin P. Haldar, Professor Shrikanth S. Narayanan for being the committee members and Professor David Taubman from University of New South Wales (UNSW), for being the guest member in my qualifying exam. Their comments has been a great help in improving this dissertation.

I would like to express my gratitude to all my teachers in USC. I have greatly benefited from their lectures. I must also acknowledge all the input from my collaborators and group members, Dr. Hilmi E. Egilmez, Dr. Sunil K. Narang, Dr. Akshay Gadde, Aamir Anis, Jiun-Yu (Joanne) Kao, Eduardo Pavez, Dr. Wei Hu, and Jin Zeng. I would like to give special thanks Dr. Yuwen He, the mentor of my internship in InterDigital. His training really stimulate my interest in video coding, and I would never forget his encouragement in my pursuit of Ph.D.

Finally, I would like to take this opportunity to express the profound gratitude to my family for their love and support. I would especially like to thank my brother, Wei-Lun (Harry) Chao. Thank you for the advice in research. Thank you for always being there for me. Thank you for being the role model for me in every way.

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

DCT	**D**iscrete **C**osine **T**ransform
DWT	**D**iscrete **W**avelet **T**ransform
GFT	**G**raph **F**ourier **T**ransform
HEVC	**H**igh **E**fficiency **V**ideo **C**oding
JPEG	**J**oint **P**hotographic **E**xperts **G**roup
CDF	**C**ohen-**D**aubechies-**F**eauveau
MST	**M**aximum **S**panning **T**ree
NM	**N**oise **M**odel
MA	**M**oving **A**verage
EA-GBT	**E**dge **A**daptive **G**raph **B**ased **T**ransform
GMRF	**G**aussian **M**arkov **R**andom **F**ield
AR	**A**uto**R**egressive
LF	**L**ight **F**ield
CFA	**C**olor **F**ilter **A**rray
ML	**M**aximum **L**ikelihood

List of Symbols

General Rule	
$i, j, N, \cdots$	Scalar (lower and uppercase normal)
$\mathcal{S}, \mathcal{N}, \mathcal{P}, \cdots$	Set (uppercase $\mathcal{C}$alligraphic)
$\mathbf{f}, \mathbf{c}, \mathbf{s}, \cdots$	Vector (lowercase **b**old)
$f_i, c_i, s_i, \cdots$	Vector element at index i
$\mathbf{f}_{\mathcal{S}}, \mathbf{c}_{\mathcal{S}}, \mathbf{s}_{\mathcal{S}}, \cdots$	Sub-vector extracted from positions indexed by set Sc
$\mathbf{A}, \mathbf{T}, \mathbf{P}, \cdots$	Matrix (uppercase **B**old)
$A_{i,j}, T_{i,j}, P_{i,j}, \cdots$	Matrix element at index (i, j)
$\mathbf{A}_{\mathcal{S},\mathcal{N}}, \mathbf{T}_{\mathcal{S},\mathcal{N}}, \mathbf{P}_{\mathcal{S},\mathcal{N}}, \cdots$	Sub-matrix from row/column positions indexed by set $\mathcal{S}/\mathcal{N}$

Symbol List	
G | $\mathcal{V}$ | $\mathcal{E}$	Graph | Node set | Link set
$\mathbf{A}$	Adjacency matrix
$\mathcal{N}(i)$	Set of neighbouring nodes connecting to node v_i
$\deg(i)$ | $\mathbf{D}$	degree on node v_i | Degree matrix
$\mathbf{L}$	Combinatorial Laplacian
$\mathcal{L}_{\text{sym}}$	Symmetric normalized Laplacian
$\mathbf{L}_{\text{G}}$	Generalized Laplacian
$\mathcal{P}$ | $\mathcal{U}$	Prediction set | Update set in lifting
$\mathbf{P}$ | $\mathbf{U}$	Prediction | Update filter in lifting
$\text{Tr}(\cdot)$ | $\det(\cdot)$	Trace | Determinant
$\text{diag}(\mathbf{v})$	Diagonal matrix with the elements of $\mathbf{v}$ on the main diagonal
$\sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	Multivariate Gaussian with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$
$\mathbf{Q}$	Precision (inverse covariance) matrix of Gaussian
$O(\cdot)$	big O notation in complexity analysis

Chapter 1

Introduction

A graph is a data structure that consists of a set of nodes (or vertices) connected by links (or edges). Graphs provide natural representations for data in many modern fields such a sensor network and social network, where data samples are distributed irregularly in high dimensional space, and also for traditional regular signals such as audio and images. In an image, each data point or data patch, *e.g.*, pixel and pixel patch, can be denoted as a node, and a graph signal is a function associated to each node, *e.g.*, pixel intensity. A link can be weighted with a value defined according to the similarity between the two nodes connected. The definition of similarity depends on the application. For example, in social networks, two nodes (users) can be considered similar, and thus have large link weight between them, if the users reside in the same city or have mutual friends. In images, two nodes (pixels) can be considered similar if the geometric distance between them is small and/or if the pixels have similar intensity values. In the last decade, there has been a growing interest in generalizing the commonly used theories and algorithms that were first developed for traditional signal processing to signals on graphs. Some of these techniques include de-noising, filtering, clustering and compression. In this thesis, we will discuss the problem of compression for data defined on graphs. Specifically, we will be focusing on applications for image and video compression.

Transform coding techniques including those based on the Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT) are widely used nowadays for multimedia compression. These methods exploit the redundancies within smooth signals, projecting signals onto a frequency domain in which they have compact representation. However, these transforms are limited to signals on regular 1D and 2D grids. Moreover, the transforms are designed based on a stationary assumption of signal statistics. In order to represent data with locally variable characteristics and on defined irregular domains, researchers have proposed the Graph Fourier Transform (GFT). Similar to DCT, frequency interpretation for signals on graphs can be derived with GFT. By properly designing link weights based on

signal characteristics, structures such as image edges, which are represented by relatively high frequencies in conventional transforms, can be coded with fairly low bit rate.

However, the GFT is a global transform and requires high complexity in implementation. Hence it is difficult to apply it for large data or in applications such as video coding, where complexity is of significant concern. In this dissertation, we develop a low complexity graph based transform using the lifting scheme. The lifting scheme is a technique to obtain multi-resolution signal representation and is composed of three building blocks, namely, bipartition, prediction, and update. In bipartition, nodes of the input graph signal are split into an update set and a prediction set. Then, signal in the prediction set is predicted from the update set, resulting in prediction residuals stored as high frequency coefficients. Next, the signal in update set is updated using the filtered prediction residuals, giving rise to the smooth approximation stored as low frequency coefficients. In predicting (updating) the prediction (update) set, only the signal in the opposite set will be utilized, *i.e.* only the weights (similarities) on links connecting nodes in different sets are exploited. Therefore, the prediction and update processes for a graph signal can be seen as processes applied on an approximated bipartite graph with only links connecting a node in prediction (update) set to a node in update (prediction) set. Transforms implemented with the lifting scheme are guaranteed to be invertible regardless of the the design of the three building blocks, and therefore provide lots of flexibility in design. In order to address the complexity concern in the GFT, in this thesis, we design the prediction and update filters to be localized, and focus on two problems:

1. How to design proper bipartition in lifting;
2. How to construct a bipartite approximation for a graph given the vertex bipartition

in order to improve coding efficiency, given the use of localized prediction and update filters.

As an application, we will discuss the application of the proposed graph based lifting transform in both regular and irregular multimedia signals. As an example of regular signals, we consider the compression of intra-predicted video residuals, where graphs are designed based on edge structure within data. We also describe a hybrid coding scheme that provides optimized transform selection according to coding efficiency. As for irregular signals, we discuss the compression of non-demosaicked light field (LF) images. We develop a coding scheme for raw LF data without going through pre-processing steps such as demosaicking, which significantly increase redundancy within data. Without demosaicking, the LF data, after being converted to an array of sub-aperture images, contains pixels with sparse distribution. In our scheme, the pixels will be considered vertices of a graph, and encoded using a graph-based lifting transform. We also describe the intra-prediction algorithm and optimal graph

construction applied for pixels with irregular distribution. In the last part of the dissertation, we consider the problem of edge modeling for different types of predicted video residuals, namely the intra and inter-predicted residuals, and the corresponding optimization of graph structure in GFT to achieve improved coding efficiency. In the next section, we will review some related work in image and video compression, followed by a detailed description of the research problems in this dissertation and our contributions.

1.1 Related Work

Transforms such as Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT) have been widely used in conventional multimedia signal compression. For example, in image and video coding standards including JPEG, MPEG, H.264, and the most recent HEVC, DCT is used to obtain sparse representation of signals in a frequency domain. However, for signals containing high frequency structures, *e.g.*, edges in images, these standard transforms are likely to produce large coefficients in high frequency basis in the transformed domain, which require high bitrate to represent them. Besides, the standard transforms can only deal with conventional signals that are defined in regular domain such as the 1 dimensional space and 2D grids. In order to deal with signals having high frequency edges, researchers have proposed designing *directional transforms*, which can incorporate the edge directions in the basis functions. Some examples include the directional DCT [83], bandelet [42], and curvelets [38]. Filtering in these works is performed along the edge directions, where neighbouring pixels tend to have high correlation, thus avoiding filtering across edges. However, many of these directional transforms require a high complexity pre-classification, which divides signals into multiple regions of uniform geometric flow. Also, most of these transforms are restricted to a given number of edge directions, which limits the adaptation to signals with more complicated characteristics, *e.g.*, corners.

Edge-Adaptive Graph based Transform

In [28], an edge adaptive transform based on graphs is proposed for depth image coding. The use of graphs allows the incorporation of signal characteristics into the link weight, which removes limitations on the specific edge directions that can be represented. Furthermore, graphs provide natural representations for signals in irregular and high dimentional spaces, and therefore can be applied in applications such as social and sensor networks. For signal defined on graphs, a frequency response can be derived using the eigenvectors and eigenvalues of the graph Laplacian, which models the variation on graphs taking into account their connection. The transform is called the Graph Fourier Transform (GFT). A signal is considered smooth

on a graph if the variation is small between two nodes connected with large weight link. Such signals have energy mostly compacted on the basis associated to small eigenvalues. Therefore, one can obtain sparse representations for signals with edges if the associated graph is designed in a way that the links connecting nodes separated by image edges have small weights. The GFTs of graphs designed based on edge structures are also called Edge-Adaptive Graph based Transforms (EA-GBTs), and have been successfully applied in applications such as depth map compression [28, 76], image compression [27], and video compression [32]. In [84], a theoretical analysis of the optimality of GFT in compression is provided in a probabilistic point of view. The authors show that given a graph, there is a unique underlying Gaussian Markov Random Field (GMRF) that satisfies a conditional independent assumption defined by graph connection. The GFT associated to the graph is shown to be equivalent to the KLT of such GMRF model, and therefore is optimal in terms of decorrelation.

1.2 Motivation

1.2.1 Low Complexity Graph Transform for Image/Video coding

Despite the advantages described in the previous section, a major challenge in using the GFT is its high complexity in computation. There are two sources of complexity. First, the transform basis vectors are generated using eigen-decomposition, with complexity up to $O(N^3)$. Second, the application of transform to a signal has complexity $O(N^2)$, because in general the corresponding matrix multiplication does not have a fast algorithm. Although the first source of complexity is more significant, this computation can be performed offline if graph connectivity does not change. Therefore, in a real application, the complexity can be reduced using some pre-computing techniques. Examples include the work in [76] and [32]. In [76] one pre-selects the k most utilized graph structures in advance, pre-computing the eigen-decomposition off-line. In [32], on the other hand, graph templates are generated based on statistical observation. Nevertheless, the second source of complexity, *i.e.* computing the transform coefficients for each block, is unavoidable, and therefore limits the usage of GFT based algorithms for large N or in applications with major concern in complexity.

To address the complexity issue in GFT, we aim to design a low complexity transform in signal projection, and at the same time still keep the advantages of GFT in signal adaptation. We design such transform using the lifting approach. The lifting transform for graph signals has been applied successfully in signal compression for different applications, including data gathering in Wireless Sensor Network (WSN) [67], and video coding [53, 54]. In these works,

the transforms are designed to be localized in the sense that the filtering of one node only takes information from the nodes that are directly connected to it. As a result, for locally connected graphs, the computation of transformed coefficients requires only $O(N \log N)$ in complexity, while for GFT, the computation costs up to $O(N^2)$ in complexity. Moreover, the prediction and update function are functions of the link weights, modeling the pair-wise similarities between nodes. Therefore, the transform avoids filtering across dissimilar nodes, *e.g.*, pixels across image edges, and thus few high frequency coefficients will be produced. Although the lifting transform has provided good performance in compression for many applications, only a limited amount of work has addressed its optimality in compression. In this dissertation, we will discuss the optimization of lifting transform. Specifically, we focus on optimizing the bipartition step in the lifting transform design in order to ensure energy compaction in the transformed frequency domain.

1.2.2 Application for Light Field Image Compression

Light Field (LF) imaging separately captures light rays arriving from different directions at each pixel in an image. With the additional information of ray directions within the light field image, some useful applications including multi-view rendering [7, 75], depth estimation [6, 29] and re-focusing [5, 11] become possible. In a lenselet-based plenoptic light field camera [63], which is the most commonly used light field camera nowadays, an array of microlenses is placed in front of the image sensor, in order to separately capture different directional rays arriving at an image pixel. Due to the structure of microlenses, each point on the focal plane will be mapped onto a patch of pixels, in which each pixel corresponds to a specific ray direction, instead of one pixel position as for traditional digital camera. The captured raw image is commonly called a *lenselet image*, and will typically be converted into a series of 2D *sub-aperture images* before compression and display. Each sub-aperture image collects pixels of the same ray direction.

Similar to the traditional digital camera, a light field camera uses Bayer patterned color filters on its sensor to capture color information, and on the resulting lenselet image, each pixel position will contain only one color component out of R, G, and B, as shown in Fig. 1.1. In order to generate full color RGB light field image, a lenselet image typically goes through *demosaicking* before being converted to sub-aperture images. Each sub-aperture image can be seen as one traditional 2D picture of a specific ray angle. Therefore, in the existing compression schemes [3, 15, 21, 30, 48], the existing video compression standards have been adopted to encode the series of sub-aperture images. For example, in [3], sub-aperture images are arranged, in *Raster* and *Spiral* orders, into pseudo-sequences that can

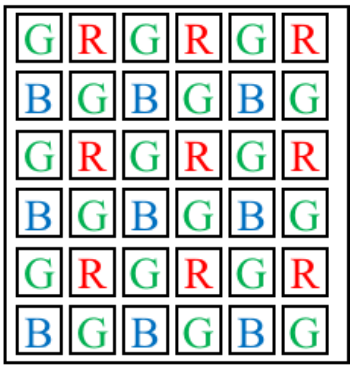


FIGURE 1.1: RGB color components arranged by Bayer pattern *GRBG*

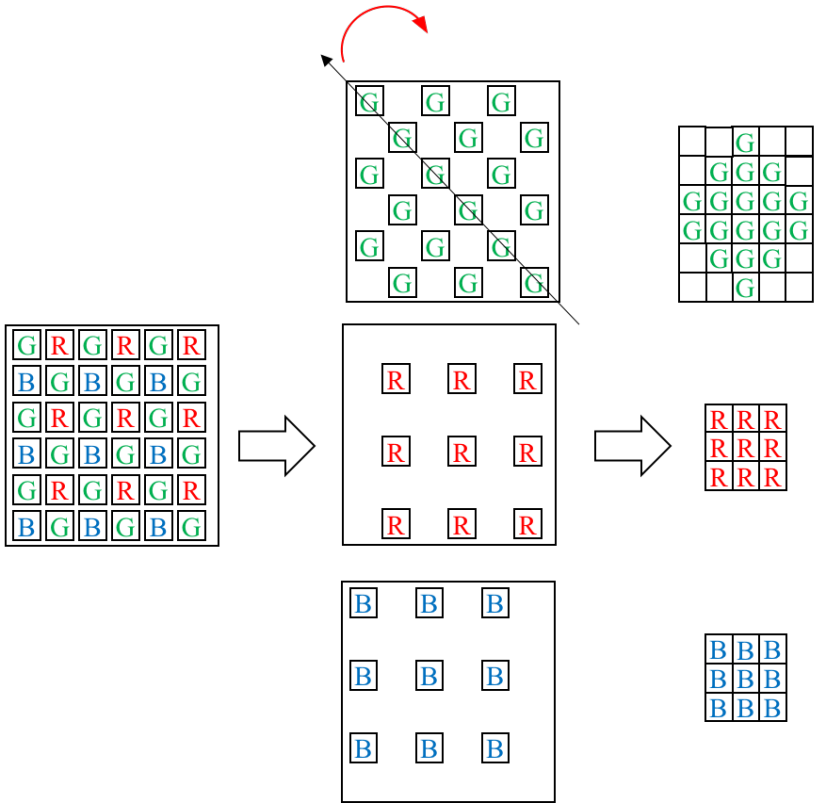


FIGURE 1.2: Transformation of Bayer patterned RGB color components into formats suitable for image/video codecs

be input into video codecs. In [15], a new mode, called *Self Similarity* (SS) has been proposed and included as an intra-prediction mode in HEVC standard, in order to exploit the similarity between pixels from adjacent ray directions. However, the number of pixels is highly increased by demosaicking, in which missing color components at each pixel are interpolated by the neighbouring pixels. Although in theory the same compression rate can be achieved before and after interpolation, existing image and video coding techniques including JPEG and HEVC usually do not exploit this knowledge in the encoding algorithm. Therefore, in this thesis, we propose a novel coding scheme that directly encodes the raw lenselet image before demosaicking, avoiding the redundancies introduced by the demosaicking process. A similar idea has been applied for conventional image [43, 44] and video compression [13]. In these works, the algorithms directly encode the non-demosaicked (Bayer patterned) raw image/video, and apply the demosaicking on the decoder side for the decoded image and video with Bayer pattern. For encoding, the RGB components arranged based on Bayer pattern will first be transformed into formats (rectangular) suitable for image and video codecs, as shown in Fig. 1.2, through simple down-sampling and rotation, before being compressed with the corresponding standards such as JPEG and HEVC. However, unlike image and video, pixels in light field data, after the conversion to sub-aperture images, will no longer be distributed based on Bayer pattern. Instead, the distribution will be highly irregular within each sub-aperture image, making it difficult to be encoded using conventional coding scheme. In our work, we develop a novel coding scheme for non-demosaicked light field image, where sparsely and irregularly distributed pixels within each sub-aperture image are connected as a graph and coded with a graph based transform. We also extend intra-prediction to pixels that are distributed sparsely to exploit the correlation among close-by pixels within each sub-aperture image.

1.2.3 Edge Models in the Graph based Transform

The Edge Adaptive Graph based Transform (EA-GBT) has shown promising results in compressing images with edges. The edge model in most graph designs is based on the assumption that all edges can be represented with ideal step functions [28, 76]. However, edges with sharp step transitions rarely exist in natural images. Instead, most of the edges in images are ramps, as pointed out in [61]. In this thesis, we present an alternative graph construction based on ramp edge model. As an application, we consider the compression of videos with different types of prediction. In addition, the signaling of graph geometries for ramp edges is also discussed.

1.3 Contributions

1.3.1 Graph based Lifting Transform with Optimized Bipartition

In our work, we introduce an optimization problem for bipartition in lifting scheme, based on a generative GMRF model. The optimal bipartition maximizes the energy compaction in the update set. We also provide a greedy solution for finding the right bipartition for different size of prediction and update sets. The resulting bipartition has a nice physical interpretation in that the regions with more variation in graph structure, which are more difficult to predict, will have more nodes selected as predictors and included into the update set. An extension of the bipartition algorithm to multi-level decompositions is also presented. In addition, we propose a novel bipartite graph formulation using Kron reduction based reconnection for each node in the prediction set. The method gives promising results reducing the prediction residue, which requires high cost for encoding. A probability interpretation is also provided for the proposed reconnection technique. The experiments on intra-predicted video coding show that the proposed lifting scheme outperforms the standard DCT based encoding, and provides comparable performance to the high complexity GFT.

1.3.2 Graph based Compression for Pre-demosaic Light Field Image

We develop a coding system for non-demosaicked light field data with sparsely and irregularly distributed pixels. Without demosaicking, we map raw sensed color data captured by plenoptic camera directly to sub-aperture image 2D grids, within which the color pixels are sparsely distributed. A novel intra-prediction scheme is performed on the irregularly distributed pixels, exploiting the correlation of each block with its decoded neighbouring reference blocks. A gradient estimation is calculated within each block based on the structure tensor, which is utilized later for edge direction estimation and directional prediction using adaptive kernel. For transform coding, sparsely distributed pixels within each block in a sub-aperture image are connected as a graph and encoded with the localized graph based lifting transform proposed. The optimal graph constructions are derived based on Maximum Likelihood (ML) criteria applied to a GMRF model.

1.3.3 Ramp Model in the Graph based Transform

We propose an 1D autoregressive model (AR) based on ramp edges, and estimate the optimal parameters in the model using the training sets of residual video sequences. The new model is utilized in the graph construction and has shown outperforming results in the intra-predicted

video sequences than the one based on step edges. To signal the overhead information, a new edge coding technique, called *Arithmetic Ramp Edge Coding* (AREC) is presented, which is an extension of the AEC approach [18] utilized for step edge coding.

1.4 Outline of the thesis

In this chapter, we described the motivation and our contributions in this dissertation. In Chapter 2, we will review the concepts of the Graph Fourier Transform, the lifting scheme, and its extension to signal on graphs. In Chapter 3, the problem of finding the optimal bipartition in lifting scheme in terms of energy compaction is defined, and a solution based on greedy approximation is proposed. Besides, we also describe the proposed method for bipartite graph formulation based on Kron reduction. The optimality of prediction filter based on the bipartite formulation is discussed. In Chapter 4, the proposed lifting transform is applied to the problem of video compression and we also discuss the graph construction and coding scheme designed. In Chapter 5, we address the problem of light field image coding and the improvement of coding efficiency using the graph based transform. A novel intra-prediction and graph construction algorithms are described in detail. In Chapter 6, we describe the design of Edge Adaptive Graph based Transform (EA-GBT) based on step and ramp edge models, and study the edge characteristics in different types of residual video sequences. We conclude the dissertation in Chapter 7 with possible directions for future work.

Chapter 2

Graph based Transforms

In this chapter, we provide an introduction to the Graph Fourier Transform and the lifting scheme, which are necessary backgrounds for this dissertation. We will start from some notation definitions for graphs and the lifting scheme in Section 2.1. In Section 2.2, we will introduce the concept of the Graph Fourier Transform (GFT), which defines the frequency interpretation for signals on graphs. This will give us an idea of how to construct a suitable graph in order to get compact representation of the associate signal in the frequency domain. We will also discuss the optimality of GFT computation in compression in terms of a probabilistic model. The same model will be adopted later in our design of optimal lifting transform. In Section 2.3, we describe the concept of lifting transform, including its invertibility, and the commonly used approaches for prediction and update filters. In Section 2.4, we will discuss the extension of lifting scheme on graphs and related research problem.

2.1 Notations

A graph $G = (\mathcal{V}, \mathcal{E})$ is a collection of nodes indexed by $v_i \in \mathcal{V} = \{v_1, v_2, v_3, \cdots, v_N\}$, and links $e_{i,j}$, connecting nodes v_i and v_j. The number of nodes N defines the size of the graph. For a weighted graph, there is a non-negative weight $w_{i,j}$ associated to each link $e_{i,j}$, modeling the similarity between the connected node pair. The connection of graph G can be represented by an $N \times N$ *adjacency matrix* $\mathbf{A}$, which has zero diagonal elements and the element $A_{i,j} = w_{i,j}$ for off-diagonal terms. For undirected graphs, the adjacency matrix is symmetric, and the degree $\deg(i)$ of node v_i is defined as $\deg(i) = \sum_j w_{i,j}$. A *degree matrix* $\mathbf{D}$ is a $N \times N$ diagonal matrix with $D_{i,i} = \deg(i)$. A *combinatorial Laplacian matrix* is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$, which is the most commonly used Laplacian matrix in literature. In Chapter 6, we also consider the generalized graph Laplacian $\mathbf{L}_G$ where there are non-negative *self loop* weights associated to each node. A graph signal $\mathbf{f} \in \mathbb{R}^N$ can be represented as a vector, where each element $\mathbf{f}_i$ is the signal value associated to node v_i.

We focus only on linear transforms $\mathbf{T} : \mathbb{R}^N \rightarrow \mathbb{R}^M$, where the operation on node v_i is defined as a linear combination of the signal value on node v_i and its nearby nodes $v_j \in \mathcal{N}_i$. *i.e.*

$$y_i =< \mathbf{T}_{i,:}, \mathbf{f} >= T_{i,i} f_i + \sum_{j \in \mathcal{N}_i} T_{i,j} f_j, \tag{2.1}$$

where $\mathbf{T}_{i,:}$ represents the i_{th} row of the transform matrix $\mathbf{T}$, and y_i is the output of the operation at node v_i. In the lifting transform, the nodes in $\mathcal{V}$ are divided into two disjoint sets $\mathcal{P}$ and $\mathcal{U}$, where $\mathcal{P}$ stands for the *prediction set*, and $\mathcal{U}$ is the *update set*. The corresponding signals are denoted as $\mathbf{f}_{\mathcal{P}} \in \mathbb{R}^n$ and $\mathbf{f}_{\mathcal{U}} \in \mathbb{R}^m$, where n and m are the number of nodes in $\mathcal{P}$ and $\mathcal{U}$ respectively. The prediction filter is defined as a linear transform $\mathbf{P} : \mathbb{R}^m \rightarrow \mathbb{R}^n$ that predicts signal $\mathbf{f}_{\mathcal{P}}$ from $\mathbf{f}_{\mathcal{U}}$. The update filter is defined as a linear transform $\mathbf{U} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ that updates $\mathbf{f}_{\mathcal{U}}$ using the prediction residual coefficients stored in $\mathcal{P}$. The bipartition, prediction, and update processes can be repeated for the smooth coefficients stored in the update set for multi-level lifting transform. We use the superscript to represent signals and operations in each level, *e.g.* $\mathbf{f}^\ell$, G^ℓ and $\mathbf{P}^\ell$ indicate the signal, the associated graph, and the prediction transform in level ℓ, respectively

2.2 Graph Fourier Transform (GFT)

2.2.1 Signal Variation on Graphs

In order to obtain the frequency interpretation for signals represented by graphs, similar to the DCT transform for one dimensional array, it is necessary to extend the concept of variation from conventional signals to graph signals, considering both the connection and pair-wise similarity between nodes. The most commonly used variation operator for graphs is the combinatorial Laplacian matrix $\mathbf{L}$. The variation of signal $\mathbf{f}$ on the graph given the associated Laplacian matrix $\mathbf{L}$ is written as

$$\begin{aligned} \text{var}(\mathbf{L}, \mathbf{f}) &= \mathbf{f}^T \mathbf{L} \mathbf{f} \\ &= \frac{1}{2} \sum_{i,j} w_{i,j} (f_i - f_j)^2. \end{aligned} \tag{2.2}$$

The graph connectivity is taken into account since the difference between two nodes that are strongly connected, *i.e.* with large link weight, will be emphasized in computing the variation. An $N \times N$ Laplacian matrix is diagonalizable with non-negative eigenvalues $\lambda_1 = 0, \lambda_2, \cdots, \lambda_N$, where $\lambda_i \leq \lambda_j$ for $i \leq j$. The corresponding eigenvectors are denoted

as $\mathbf{U} = \{\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_N\}$. Note that for two eigenvectors $\mathbf{u}_i$ and $\mathbf{u}_j$ with $i \leq j$, the variation of the vectors on the graph follows the order of eigenvalues, *i.e.*, $\text{var}(\mathbf{L}, u_i) \leq \text{var}(\mathbf{L}, u_j)$. Therefore, these eigenvectors provide a Fourier-like basis with frequency quantified by the corresponding eigenvalues, thus $\mathbf{U}$ is called the Graph Fourier Transform (GFT). The GFT coefficients $\tilde{\mathbf{f}}$ of signal $\mathbf{f}$ are obtained as $\tilde{\mathbf{f}} = \mathbf{U}^{-1}\mathbf{f}$. Note that for undirected graph, the combinatorial Laplacian matrix is symmetric and thus $\mathbf{U}^{-1} = \mathbf{U}^T$. There are other types of Laplacian matrices that have been used in literature, including the random walk Laplacian $\mathcal{L}_{\text{rw}} = \mathbf{D}^{-1}\mathbf{L}$ and the symmetric normalized Laplacian matrix $\mathcal{L}_{\text{sym}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$, which is commonly used in graph cut related works [70]. Both matrices have non-negative eigenvalues $\lambda_i \in [0, 2]$. Note that the combinatorial Laplacian $\mathbf{L}$ and the normalized Laplacian $\mathcal{L}_{\text{sym}}$ are both symmetric, with orthogonal GFT basis, while the GFT for the random walk Laplacian $\mathcal{L}_{\text{rw}}$ is non-orthogonal.

2.2.2 Optimality of GFT in Signal Compression

In order to achieve good performance in compression, it is important to have a transform that can represent signals sparsely. In traditional signal processing applications, the DCT is useful in compressing smooth signals such that signal variations are small between neighbouring data points, since on average most of the signal energy is concentrated on the low frequency bases. For GFT, the signal variation is defined in (2.2) as a function of signal and graph structure. In order to have compact representation with most energy in the low frequency basis for a given signal $\mathbf{f}$, we aim to choose a graph that leads to small variation on the resulting Laplacian matrix $\mathbf{L}^*$, *i.e.* the value of $\text{var}(\mathbf{L}^*, \mathbf{f})$ is small. This can be easily achieved by assigning large weights to links connecting nodes with similar signal values, and small weights to links connecting nodes with large differences. The flexibility in graph construction provides many advantages to GFT for compression in applications such as image and video compression [27, 28, 32, 76, 77]. Natural images usually contain edges, where multiple neighboring pairs of pixels have large intensity difference. Edges are not efficiently represented by conventional transforms such as the DCT. We can express an image as a graph where one pixel is denoted as a node, and an sparse representation in GFT can be achieved by assigning small weights to links across edges.

In [84, 85], optimality of GFT in signal compression is analyzed. For a random signal with known correlation, described by its covariance matrix, the *Karhunen–Loève Transform* (KLT) is optimal in terms of de-correlation. The KLT is an optimal orthogonal transform in energy compaction in terms of mean squared error, *i.e.* the mean squared error of the reconstructed signal using k transformed coefficients is minimized among all orthogonal

transforms. However, correlation models need to be estimated from training data for real signals. In order to obtain a fixed transform without high complexity in computation, the DCT is often used. The DCT is equivalent to KLT for signals that can be modeled as a stationary Markov sequence with correlation equal to 1 [1, 14]. The optimality of the GFT can be shown in a similar way. Given a graph G, the associated GFT is shown to be equivalent to the KLT for an underlying Gaussian Markov Random Field (GMRF) defined based on the graph connectivity, *i.e.* the inverse covariance matrix (precision matrix) of this GMRF is equivalent to the graph Laplacian $\mathbf{L}$. In fact, GFT can outperform KLT in practice, since fewer parameters are required to estimate, leading to more robust signal modeling [34].

2.3 Lifting Transforms

In this section, we will introduce the lifting scheme, including the invertibility property, the design of localized transforms and the extension to multi-level decomposition.

2.3.1 Preliminaries

A lifting scheme consists of three stages:

1. Bipartition: Data points are divided into two disjoint sets, called *prediction set* ($\mathcal{P}$) and *update set* ($\mathcal{U}$)

2. Prediction: This step is used to remove redundancy in the signal. The signal $\mathbf{f}_{\mathcal{P}}$ in $\mathcal{P}$ is predicted using signal $\mathbf{f}_{\mathcal{U}}$ in $\mathcal{U}$ with prediction transform denoted as $\mathbf{P}$. Then, the prediction error, *i.e.* $\mathbf{d} = \mathbf{f}_{\mathcal{P}} - \mathbf{P}\mathbf{f}_{\mathcal{U}}$, is stored in set $\mathcal{P}$. If the signals in the two sets are highly correlated with each other, the prediction error is expected to be small.

3. Update: The signal $\mathbf{f}_{\mathcal{U}}$ in $\mathcal{U}$ is updated using the prediction residue $\mathbf{d}$ and the update transform $\mathbf{U}$. The process generates a smooth approximation $\mathbf{s}$ to the original signal $\mathbf{f}$, and the result is stored in $\mathcal{U}$ as transformed coefficients.

A one level lifting scheme is summarized in Fig. 2.1. The transformed coefficients $\mathbf{c} = [\mathbf{s}^T, \mathbf{d}^T]^T$, contain the prediction error $\mathbf{d} = \mathbf{f}_{\mathcal{P}} - \mathbf{P}\mathbf{f}_{\mathcal{U}}$ and the smooth coefficients $\mathbf{s} = \mathbf{f}_{\mathcal{U}} + \mathbf{U}\mathbf{d}$. In matrix form, the whole process can be written as

$$\mathbf{c} = \begin{bmatrix} \mathbf{s} \\ \mathbf{d} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{U} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{P} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{f}_{\mathcal{U}} \\ \mathbf{f}_{\mathcal{P}} \end{bmatrix}. \tag{2.3}$$

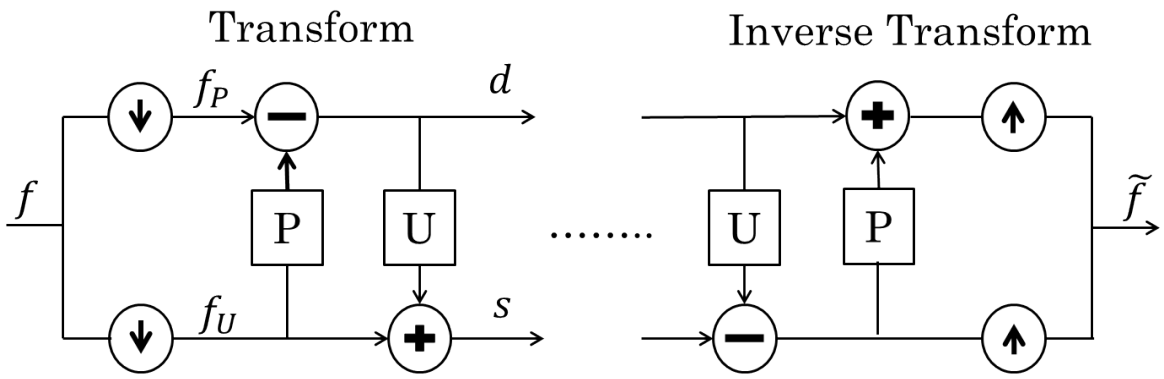


FIGURE 2.1: 1 level lifting scheme: forward and inverse transforms

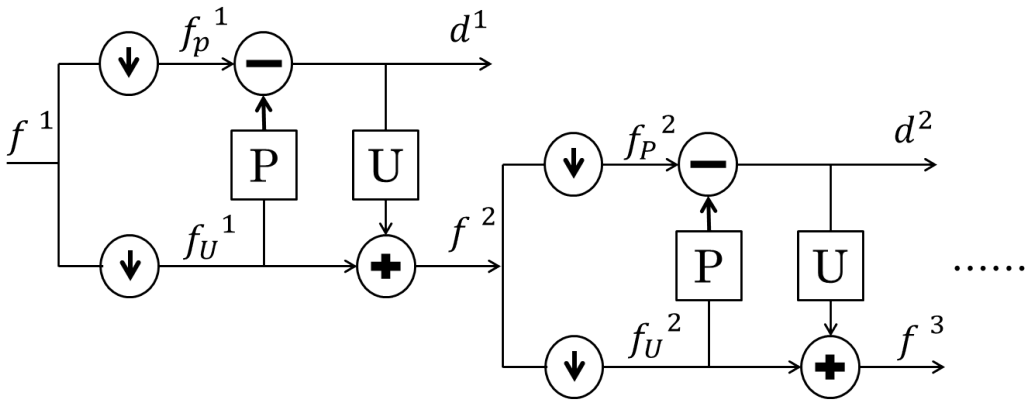


FIGURE 2.2: The lifting scheme with multi-level decomposition

The inverse transforms for both prediction and update processes can be immediately derived by inverting the operations (addition replaced by subtraction and subtraction by addition) and orders in the forward process, as shown in Fig. 2.1. In matrix form this is written as

$$\begin{aligned} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{P} & \mathbf{I} \end{bmatrix}^{-1} &= \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{P} & \mathbf{I} \end{bmatrix}, \\ \begin{bmatrix} \mathbf{I} & \mathbf{U} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}^{-1} &= \begin{bmatrix} \mathbf{I} & -\mathbf{U} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}. \end{aligned} \tag{2.4}$$

Note that the invertibility is guaranteed regardless of the selection of the prediction transform $\mathbf{P}$ and update transform $\mathbf{U}$, thus providing a lot of flexibility in transform design.

If the predictor is properly designed such that the prediction error $\mathbf{d}$ has low energy on average, fewer bits will be needed for representing $\mathcal{P}$, thus reducing the overall cost in

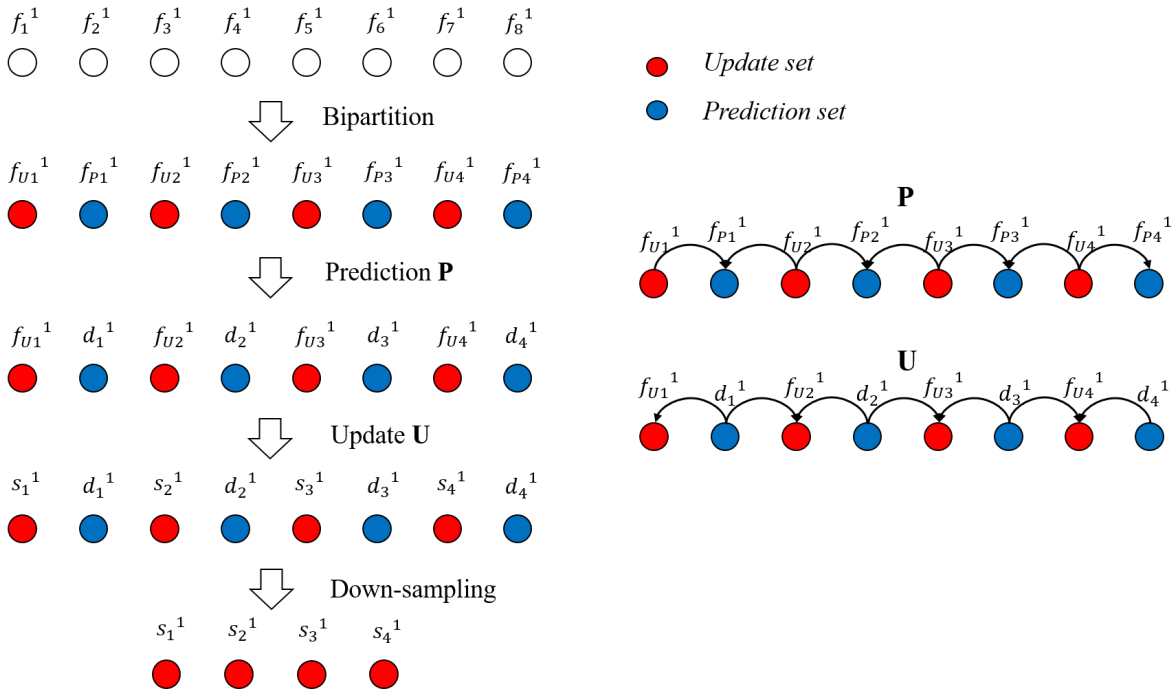


FIGURE 2.3: Example of localized lifting transform on 1 dimensional signal

compression. A multi-resolution representation for signals can be derived by repeatedly applying the lifting scheme on the smooth coefficients $\mathbf{c}$, as shown in Fig. 2.2. The smooth coefficients $\mathbf{c}^{\ell}$ serve as the input signal $\mathbf{f}^{\ell+1}$ for the $(\ell+1)^{\text{th}}$ level lifting. Moreover, a more compact representation can be obtained from the multi-level decomposition if the predictor $\mathbf{P}^{\ell}$ in each level ℓ is selected properly. The invertibility property will still hold for multi-level decomposition.

2.3.2 Localized Transform Design

The lifting scheme can be used to implement any invertible transform. One of the commonly used transform is the 5/3 biorthogonal filterbank of Cohen-Daubechies-Feauveau (CDF) [2], which has been adopted in JPEG2000 standard for lossless compression [9, 73]. Given a one dimensional signal at first level $\mathbf{f}^1 = [f_1^1, f_2^1, \cdots f_N^1]^T$, the lifting scheme for CDF5/3 can be represented as shown in Fig. 2.3 for $N = 8$. For the bipartition, the data points at odd locations are assigned to *update set* ($\mathcal{U}^1$), and the even points are put into the *prediction set* ($\mathcal{P}^1$). Each data point in $\mathcal{P}^1$ is predicted using its adjacent points, resulting in a prediction error $\mathbf{d}^1 = [d_1^1, d_2^1, d_3^1, d_4^1]^T$. The prediction error will then be used to filter the adjacent data points in $\mathcal{U}^1$, generating the smooth approximation $\mathbf{s}^1 = [s_1^1, s_2^1, s_3^1, s_4^1]^T$. For multi-level

decomposition, the lifting scheme for the 2^{nd} level will be applied on the smooth coefficients $\mathbf{s}^1$. In detail, the prediction for signal value $f_{\mathcal{P}i}$ is computed with the two adjacent points $f_{\mathcal{U}i}$ and $f_{\mathcal{U}(i+1)}$ as $\hat{f_{\mathcal{P}i}} = \frac{1}{2}(f_{\mathcal{U}i} + f_{\mathcal{U}(i+1)})$. For the update stage, $f_{\mathcal{U}i}$ is updated with the prediction error from the two neighbours. The smooth coefficients for interior points are computed as $c_i = f_{\mathcal{U}i} + \frac{1}{4}(d_i + d_{(i+1)})$. Compared to other types of wavelet filterbanks such as Haar transform and CDF9/7, the CDF5/3 filterbanks have several advantages:

1. Locality: The transform is highly localized and therefore has low complexity in computation

2. Rational-valued: The transform coefficients are rational-valued, and therefore it is easier to create lossless transform in real implementation. Also, for coefficients equal to $\frac{1}{2^n}$, the transform can be implemented with simple shifts.

3. Symmetry: The filter response is symmetric, which enables a simple generalization onto undirected graphs, where each node might have different number of connected neighbours.

Due to these nice properties, for the prediction and update transforms design throughout this dissertation we will apply the generalization of CDF5/3 filterbanks to graphs, which will be described in detail in the next section. The main contribution for our work will be focusing on the bipartition algorithm and the graph construction, given the use of CDF5/3 like transforms for prediction and update.

2.4 Lifting Transform on Graphs [56]

The generalization of lifting scheme to signals on graphs is in general not trivial. Before we go into the detail, here we first describe the problems that will be encountered during the generalization, and the criterion for the design of graph-based lifting.

1. Graph construction: In Section 2.2, we have discussed the criterion for designing a good graph on which to define the GFT, where low weights are assigned to links connecting nodes with large signal difference. The criterion also hold for graphs in the lifting transform. Since we consider only linear transforms in the design of prediction and update filters, in order to obtain good prediction, the filtering for each node in $\mathcal{P}$ should select higher weights for the neighbours in $\mathcal{U}$ that are similar. Besides, since the main goal in our work is to design low complexity and localized transforms, the graph

connectivity is required to be localized. The details of graph construction in application of video compression will be addressed in Chapter 4.

2. Design of prediction and update transforms (**P** and **U**): As mentioned above, a useful feature of CDF5/3 is its locality, *i.e.*, the transform of data at each position $x \in \mathcal{P}$ requires only the signal at x and values at the adjacent data points $x-1$ and $x+1$ in $\mathcal{U}$. This property is preserved in the generalization to signal on graphs: the transform for node $v_i \in \mathcal{P}$ takes information only from f_i and f_j, $v_j \in \mathcal{N}(i) \cap \mathcal{U}$, where $\mathcal{N}(i)$ consists of nodes that are directly connected to v_i. The same property holds for $v_i \in \mathcal{U}$ such that only neighbors from $\mathcal{P}$ are considered during the transform. Note that for general graphs, $\mathcal{N}(i)$ for different v_i may contain different number of nodes. Also, the weight on the link connecting v_i and $v_j \in \mathcal{N}(i)$ varies based on pair-wise similarities. In order to have better prediction, the predictor for node v_i should obtain more information from those connected neighbors that are likely to be most similar to v_i. Besides exploiting the correlation between data samples, it is also important to have transforms that are orthogonal or nearly orthogonal, in order to reduce the distortion in reconstruction caused by quantization. In [68], the authors propose a method to design update filters that promote orthogonality. Later in this section, we will describe the generalization of CDF5/3 predictor and the orthogonalized update filter in more detail.

3. Bipartition: In the lifting scheme, the transform (Prediction or Update) for node v_i in $\mathcal{P}$ (respectively $\mathcal{U}$) is a function of only the signal values at v_i and nodes in $\mathcal{U}$ (respectively $\mathcal{P}$) in order to ensure the invertibility. In using the CDF5/3 filterbank, the transform at each node acquires neighbouring information only from the connected nodes in the opposite bipartite set. For graphs that are not bipartite, this means that the links that connect nodes in the same set will not be utilized during the filtering. In other words, applying generalized CDF5/3 in graphs is equivalent to applying the transform on an approximated bipartite graph, containing only links connecting nodes in $\mathcal{U}$ to nodes in $\mathcal{P}$. The bipartition algorithm directly affects the bipartite approximation and the performance of prediction and update filtering. In order to have better prediction, each node v_i in $\mathcal{P}$ should have enough high correlated neighbouring nodes in $\mathcal{U}$ after the bipartite approximation. In Chapter 3, we will review related work for bipartite approximation in signal compression, and describe our approach in designing an optimized bipartition in terms of energy compaction in the transformed domain.

4. Graph representation for signals in level $\ell > 1$: After the update stage, the signal will be downsampled, keeping only the smooth signal in $\mathcal{U}$ for processing in the next level.

Those nodes in $\mathcal{U}$ might not be directly connected in the original graph. Therefore, a graph needs to be constructed for the downsampled data in order to capture pair-wise correlation. In Section 2.4.3, we will discuss some commonly used graph reduction after downsampling.

2.4.1 Prediction Filter Design

Given the bipartite sets $\mathcal{P}$ and $\mathcal{U}$ and the approximated bipartite graph G_{bpt} (with adjacency matrix $\mathbf{A}_{\text{bpt}}$), which contains only links connecting nodes in $\mathcal{P}$ to nodes in $\mathcal{U}$, the predictor used in our work for node $v_i \in \mathcal{P}$ is defined as

$$\hat{f}_i = \frac{1}{\deg(i)} \sum_{v_j \in \mathcal{N}_{\text{bpt}}(i)} w_{i,j} f_j, \tag{2.5}$$

where $w_{i,j} = \mathbf{A}_{\text{bpt}}(i,j)$ and $\mathcal{N}_{\text{bpt}}(i)$ is the set of neighboring node of v_i in G_{bpt}. The transformed coefficient at v_i is computed as $d_i = f_i - \hat{f}_i$. Note that for a one dimensional signal, which can be represented with a line graph with all link weights equal to 1, the predictor is simplified into the predictor used in the traditional CDF5/3 filterbank described in 2.3. Therefore, we will call this transform the *generalized CDF5/3 predictor*, which has been applied for graph based lifting by several authors [53, 54, 56, 67]. Later in Section 5.5.3, we will also discuss the generalized CDF5/3 filterbank developed for generalized graphs with non-zero self loop weights on nodes.

2.4.2 Update Filter Design

For the design of update filter, we can also generalize the update filters used in CDF5/3 filterbanks. The transformed coefficient at $v_r \in \mathcal{U}$, written as s_r, stores the smooth signal computed as

$$s_r = f_r + \frac{1}{2\deg(r)} \sum_{v_j \in \mathcal{N}(r)} w_{r,j} d_r, \tag{2.6}$$

which has been applied in literature [54, 56, 67]. However, in [68], the authors show that for the CDF5/3 design applied to graphs, the transform's orthogonality will be reduced if each node has more than 2 connected neighbours, Therefore, in the paper, the authors proposed an orthogonalized update transform. Assume the signal in ℓ^{th} lifting level is $\mathbf{f} = [\mathbf{f}_{\mathcal{U}}^T, \mathbf{f}_{\mathcal{P}}^T]^T$, then

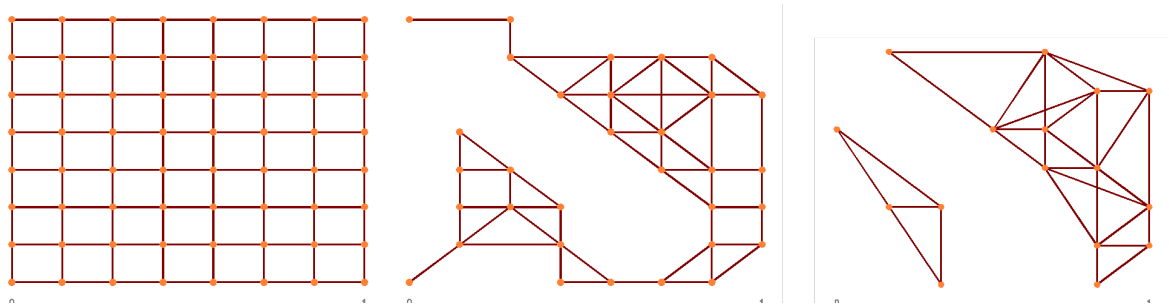


FIGURE 2.4: Example of graph downsampling by connecting 2-hops neighbors in the previous level

the corresponding transform matrix $\mathbf{T}$ in the lifting scheme can be written as

$$\mathbf{T} = \begin{bmatrix} -\mathbf{t}_1^T- \\ -\mathbf{t}_2^T- \\ \vdots \\ -\mathbf{t}_N^T- \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{U} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{P} & \mathbf{I} \end{bmatrix}, \tag{2.7}$$

where the first m rows correspond to the filter response of nodes in $\mathcal{U}$, and the last n rows are the filter responses for nodes in $\mathcal{P}$. The filter response for node v_j in $\mathcal{P}$ can be written as

$$\mathbf{t}_j^T = \mathbf{e}_j^T \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{P} & \mathbf{I} \end{bmatrix}, \tag{2.8}$$

where $\mathbf{e}_j$ is a column vector with element $e_{jj} = 1$ and $e_{jk} = 0$ for $k \neq j$. The orthogonalized update filter $\mathbf{U}$ is computed for each node $v_i \in \mathcal{U}$ such that the filter response $\mathbf{t}_i$ is orthogonal to the filter response $\mathbf{t}_j$ of its neighbouring nodes $v_j \in \mathcal{P}$. The computation can be done by solving a linear equation. In orthogonalizing $\mathbf{t}_i$, only the responses of its neighbouring nodes are considered. For localized $\mathbf{P}$, the nodes in $\mathcal{P}$ that are not neighbors of v_i have filter responses that have no common support or small common support with $\mathbf{t}_i$, and thus have little effect on its orthogonality. In the lifting scheme used in our work, we will apply the generalized CDF5/3 predictor with orthogonalized update filter, since the orthogonalized update filter has been shown to have better performance empirically compared to the one without orthogonalization.

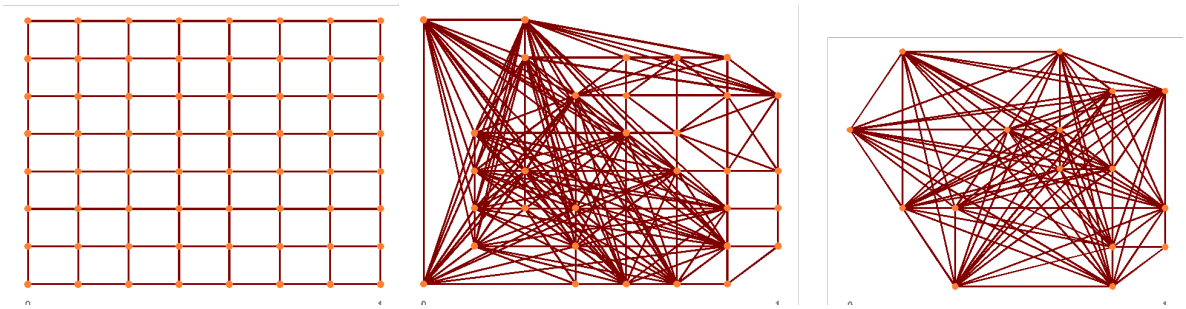


FIGURE 2.5: Example of graph downsampling using Kron reduction

2.4.3 Graph Reduction

To obtain a multi-resolution decomposition for signals, the lifting scheme is applied iteratively onto the downsampled, smooth signal from the update set in the previous lifting level. The transform further exploits the similarity within the downsampled signal. For the 1 dimensional signal discussed in Section 2.3.2, where the bipartition divides data into even and odd samples, the CDF5/3 filterbank exploits the correlation between pairs of data points that are 1 point away in the first level. For the transform in the 2^{nd} level, the de-correlation considers the similarity between data samples that are 2 points away in the first level. Similarly for the ℓ^{th} level, the transform exploits the correlation between samples that are 2 points away in the $(\ell-1)^{\text{th}}$ level. The concept can be extended to signals on graphs [54, 57, 58], where for the graph construction of input signal in level ℓ, the nodes that are 2 hops away in the previous level are connected. However, the approach cannot maintain graph connectivity as the decomposition goes to higher levels. In Fig. 2.4, we show an example of graph construction using such approach, where the update set in each level is chosen by random sampling. In this example, multiple disconnected components are produced when the decomposition reaches level 3. As a result, the transform will not be able to exploit the correlation between disconnected components, losing the opportunity of further energy compaction.

In this work, we apply another graph reduction method, initially proposed in [20], and designed from a probabilistic viewpoint. We first define a generative Gaussian Markov Random Field (GMRF) model with inverse covariance (or precision) matrix $\mathbf{Q} = \mathbf{L} + \delta\mathbf{I}$, where $\mathbf{L}$ is the graph Laplacian and δ is a small constant used to ensure matrix invertibility. Given a random signal $\mathbf{f} = [\mathbf{f}_{\mathcal{U}}^T, \mathbf{f}_{\mathcal{P}}^T]^T$ produced by such GMRF model, where $\mathbf{f}_{\mathcal{U}}$ and $\mathbf{f}_{\mathcal{P}}$ are the signals in the update and prediction sets respectively, the covariance matrix $\mathbf{\Sigma}$ and the inverse

covariance matrix $\mathbf{Q}$ can be written in block form as follows:

$$\mathbf{\Sigma} = \begin{bmatrix} \mathbf{\Sigma}_{\mathcal{U},\mathcal{U}} & \mathbf{\Sigma}_{\mathcal{U},\mathcal{P}} \\ \mathbf{\Sigma}_{\mathcal{P},\mathcal{U}} & \mathbf{\Sigma}_{\mathcal{P},\mathcal{P}} \end{bmatrix}, \quad \mathbf{Q} = \begin{bmatrix} \mathbf{Q}_{\mathcal{U},\mathcal{U}} & \mathbf{Q}_{\mathcal{U},\mathcal{P}} \\ \mathbf{Q}_{\mathcal{P},\mathcal{U}} & \mathbf{Q}_{\mathcal{P},\mathcal{P}} \end{bmatrix}. \tag{2.9}$$

After removing signal in $\mathcal{P}$, the downsampled signal $\mathbf{f}_{\mathcal{U}}$ is also a GMRF, with covariance matrix $\mathbf{\Sigma}_{\mathcal{U},\mathcal{U}}$, and the inverse covariance matrix, denoted as $\mathbf{Q}_{\mathrm{d}}$, can be written as

$$\mathbf{Q}_{\mathrm{d}} = \mathbf{\Sigma}_{\mathcal{U}}^{-1} = \mathbf{Q}_{\mathcal{U},\mathcal{U}} - \mathbf{Q}_{\mathcal{U},\mathcal{P}} \mathbf{Q}_{\mathcal{P},\mathcal{P}}^{\dagger} \mathbf{Q}_{\mathcal{P},\mathcal{U}}. \tag{2.10}$$

We can derive the corresponding graph Laplacian of $\mathbf{f}_{\mathcal{U}}$ as $\mathbf{L}_{\mathrm{d}} \approx \mathbf{Q}_{\mathrm{d}}$. The graph connection in the downsampled signal therefore is based on the partial correlation specified in $\mathbf{Q}_{\mathrm{d}}$. The corresponding adjacency matrix $\mathbf{A}_{\mathrm{d}}$ is defined as

$$\begin{aligned} \mathbf{A}_{\mathrm{d}i,i} &= 0 \\ \mathbf{A}_{\mathrm{d}i,j} &= -\mathbf{Q}_{\mathrm{d}i,j}, \text{ for } i \neq j. \end{aligned} \tag{2.11}$$

This method is also known as the *Kron reduction* in the literature, and was first proposed for application in electrical networks [22]. The graph derived from *Kron reduction* has several desirable properties in particular, and most relevant for our application:

1. If the original graph is connected in the first level, the reduced graph will still be connected.

2. Two nodes v_i and v_j are connected in level ℓ if there is a path through the removed nodes in $\mathcal{P}^{\ell-1}$.

3. For two nodes that are connected though a path of large weighted links in level $\ell - 1$, the link connecting them in level ℓ will also have large weight.

One drawback in the *Kron reduction* is that the downsampled graph becomes dense as the decomposition goes to higher levels. An example is shown in Fig. 2.5, where the downsampled sets are the same as in Fig. 2.4. This loss of sparsity can increase the computation complexity in the transform. Also, the transform will fail to capture the local connectivity information. Therefore, we apply a simple sparsification after downsampling in each level by keeping the largest k links for each node. In Fig. 2.6, we show an example of sparsification in the reduced graph, where at least 4 links are kept for each node.

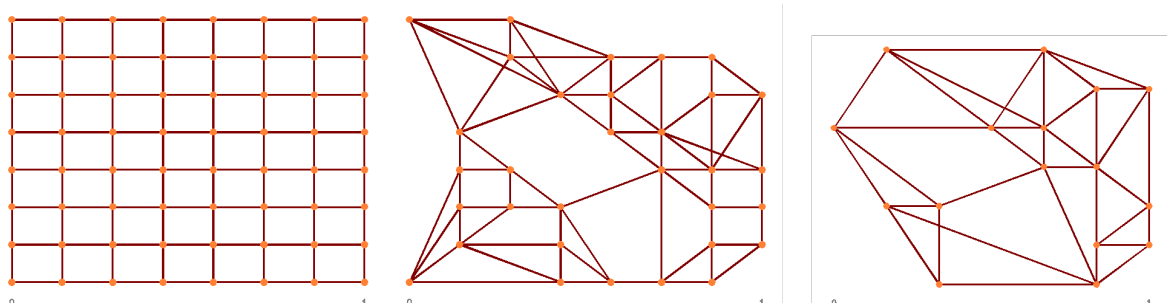


FIGURE 2.6: Example of graph downsampling using Kron reduction and sparsification

2.4.4 Complexity of Lifting Scheme

As mentioned in Section 1.2.1, the GFT requires multiplying the input signal (written as a vector) with a dense matrix, which leads to $O(N^2)$ complexity. For lifting, on the other hand, the transform is highly localized. If half of the nodes are selected into the update set in each level, at most $\log N$ transform levels will be required. For the computation of each coefficient in level ℓ, the number of operations is proportional to the degree $\deg(i)$ of node v_i, which is usually a constant for graphs of interest in image and video processing and due to the sparsification after graph reduction in high lifting levels. As a result, only $O(N \log N)$ complexity is needed for lifting application. Besides, the GFT coefficients are real valued, while in lifting based on CDF 5/3 filterbank, the coefficients are rational, which leads to additional reduction in implementation cost. In our experiments in Chapter 4, we will show that with the lifting transform, the quality of reconstruction after compression is comparable to GFT, so that we pay no penalty for these significant reductions in complexity.

2.5 Summary

In this chapter, we have introduced the concept of graphs, and a notion of frequency for graph signals using the Graph Fourier Transform (GFT). We have discussed the optimality of GFT in signal energy compaction. In Sections 2.3 and 2.4, we gave an introduction for the lifting scheme and its generalization to signals on graphs. The design of prediction and update transforms and graph downsampling used in our work are also described. In the next chapter, we will focus on the optimization of bipartition in the lifting transform in terms of energy compaction.

Chapter 3

Bipartition in Lifting Transforms

In Section 2.4, we have described the problems in applying the lifting scheme to signals on graphs and the algorithms for prediction, update and graph downsampling utilized in our experiments. In this chapter, we will be focusing on the optimization of bipartition in order to maximize energy compaction.

3.1 Related Work

3.1.1 Max Cut based Bipartition

The first bipartition technique proposed for graph based lifting was [56], where a denoising application was considered. In this work, the authors use a bipartition algorithm called *conflict minimization*, which aims to minimize the total weight of links connecting nodes in the same set ($\mathcal{P}$ or $\mathcal{U}$), *i.e.*, those links that will not be used in the lifting transform. In [53, 54, 57], a similar idea is applied for the lifting scheme in the application of low pass approximation and video compression, where the bipartition is done by maximizing the total weights on links connecting nodes in $\mathcal{P}$ to nodes in $\mathcal{U}$. In other words, the algorithm tries to utilize as many pair-wise similarities as possible after discarding *conflicting* links to achieve better decorrelation. This is is equivalent to solving a maximum cut problem (*Max Cut*). The *Max Cut* problem is known to be NP-complete, and therefore some greedy approximations have been proposed in the literature. In [59], the authors propose a solution for the *Max Cut* problem based on *maximum spanning tree (MST)*, whose optimal solution can be formed using methods such as Prim's algorithm. The optimality of the *Max Cut* approach is also analyzed in [59] for interpolation, where the authors show that by maximizing the cut value between $\mathcal{P}$ and $\mathcal{U}$, a lower bound of the ℓ_1 norm error for cross-linear interpolation will be minimized assuming the samples are *i.i.d*. However, the signals typically exhibit local correlation, and therefore the *i.i.d* assumption does not hold in general. Taking images as an example, there exists high

correlation between pixels that are adjacent to each other. Moreover, no justification was provided for optimality in terms compression efficiency in [59].

3.1.2 Prediction Error Minimization

In [23, 52], Martínez-Enríquez et al. propose an optimal bipartition for lifting transform in video compression. The objective function is based on minimizing the energy of prediction error stored in $\mathcal{P}$, thus promoting the energy compaction into the low frequency set $\mathcal{U}$. Defining $\hat{\mathbf{f}}_{\mathcal{P}}$ to be the prediction of $\mathbf{f}+\mathcal{P}$, the problem can be expressed as

$$\begin{aligned}(\mathcal{P},\mathcal{U})^* &= \arg\min_{(\mathcal{P},\mathcal{U})} \mathbb{E}[(\mathbf{f}_{\mathcal{P}} - \hat{\mathbf{f}_{\mathcal{P}}})^T(\mathbf{f}_{\mathcal{P}} - \hat{\mathbf{f}_{\mathcal{P}}})] \\ &= \arg\min_{(\mathcal{P},\mathcal{U})} \mathbb{E}[(\mathbf{f}_{\mathcal{P}} - \mathbf{P}\mathbf{f}_{\mathcal{U}})^T(\mathbf{f}_{\mathcal{P}} - \mathbf{P}\mathbf{f}_{\mathcal{U}})] \\ &= \arg\min_{(\mathcal{P},\mathcal{U})} \sum_{v_i\in\mathcal{P}} \mathbb{E}[(f_i - \hat{f}_i)^2].\end{aligned} \tag{3.1}$$

In this work, the authors consider two signal models, namely the noise model (*NM*), and moving average (*MA*) model. The prediction $\hat{f}_{i\in\mathcal{P}}$ is computed as the normalized average of its neighbouring nodes in $\mathcal{U}$. A greedy solution for (3.1) is proposed. In general, the transform with *MA* model performs better than the one with *NM* based bipartition in terms of minimizing the expected prediction errors in $\mathcal{P}$, showing the significance for considering local correlation in the designing of signal model. However, the moving average (*MA*) model assumes marginal independence for samples that are not immediate neighbors, which is in general not true for image and video signal where correlation usually also exits between pixels that are few pixels away. In the next section, we will discuss in detail the difference of the two models in [23, 52] from our proposed signal model.

3.2 Optimized Bipartition for GMRF modeled signal

In this thesis, we propose a bipartition scheme that is optimal in terms of energy compaction under the assumption that an N dimensional signal $\mathbf{f}$ can be modeled as GMRF: $\mathbf{f} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. A random vector $\mathbf{f} = [f_1, f_2, \cdots, f_N]^T$ can be modeled by a GMRF if its probability density function can be written as

$$p(\mathbf{f}) = (2\pi)^{-\frac{N}{2}} \det(\mathbf{Q})^{\frac{1}{2}} \exp(-\frac{1}{2}(\mathbf{f}-\boldsymbol{\mu})^T\mathbf{Q}(\mathbf{f}-\boldsymbol{\mu})), \tag{3.2}$$

where $\boldsymbol{\mu}$ is its mean vector and $\mathbf{Q}$ is the inverse covariance matrix $\mathbf{Q} = \boldsymbol{\Sigma}^{-1}$, also called precision matrix. The precision matrix $\mathbf{Q}$ defines the conditional pair-wise correlations between signal values, *i.e.*

$$p(f_i, f_j|\mathbf{f}/\{f_i, f_j\}) = -\frac{Q_{i,j}}{\sqrt{Q_{i,i}Q_{j,j}}}. \tag{3.3}$$

3.2.1 Relationship to Noise Model (NM) and Moving Average (MA) models

It can be shown that the *NM* and *MA* signal models considered in [23, 52] both correspond to specific GMRF models. Under the *NM* case, a signal is defined as

$$\mathbf{f} = \mathbf{c} + \boldsymbol{\eta}, \tag{3.4}$$

where $\mathbf{c}$ consists of constant values $[c_1, c_2, \cdots, c_N]^T$ and vector $\boldsymbol{\eta}$ contains *i.i.d* zero mean Gaussian noises $[\eta_1, \eta_2, \cdots, \eta_N]^T$ with variance $\sigma^2_{\eta,i}$ for i^{th} element. The model can be expressed as a GMRF with mean $\boldsymbol{\mu}_{NM} = \mathbf{c}$ and covariance

$$\begin{aligned}\boldsymbol{\Sigma}_{NM} &= \mathbb{E}[(\mathbf{f} - \mathbf{c})(\mathbf{f} - \mathbf{c})^T] &(3.5)\\ &= \mathbb{E}[\boldsymbol{\eta}\boldsymbol{\eta}^T] &(3.6)\\ &= \mathrm{diag}([\sigma^2_{\eta,1}, \sigma^2_{\eta,2}, \cdots, \sigma^2_{\eta,N}]). &(3.7)\end{aligned}$$

The covariance matrix contains non-zero values only for diagonal elements, *i.e.*, the model assumes there is no pair-wise correlation between pixels. The assumption is usually not true for image and video signals, where neighboring pixels usually possess high similarity. For *MA* model, on the other hand, the signal value for sample (node) i is defined as

$$f_i = \frac{1}{|\mathcal{N}_{[i]}|}\sum_{j\in\mathcal{N}_{[i]}} \epsilon_j + \alpha\eta_i, \tag{3.8}$$

where $\mathcal{N}_{[i]}$ is the closed set of spatial neighbors of node i ($\mathcal{N}_{[i]} = \mathcal{N}(i) \cup i$) and α is an adjustable parameter. ϵ_j and η_i are both $i.i.d$ zero-mean Gaussian noise with variance $\sigma^2_{\epsilon,i}$ and $\sigma^2_{\eta,i}$ respectively. The equation can be expressed in matrix form using the adjacency matrix $\mathbf{A}$ and degree matrix $\mathbf{D}$ as

$$\mathbf{f} = (\mathbf{D} + \mathbf{I})^{-1}(\mathbf{A} + \mathbf{I})\boldsymbol{\epsilon} + \alpha\boldsymbol{\eta}, \tag{3.9}$$

where vectors $\boldsymbol{\epsilon}$ and $\boldsymbol{\eta}$ contain the variance of $i.i.d$ Gaussian noises: $\boldsymbol{\epsilon} = [\epsilon_1, \epsilon_2, \cdots, \epsilon_N]^T$ and $\boldsymbol{\eta} = [\eta_1, \eta_2, \cdots, \eta_N]^T$. In [23, 52], the adjacency matrix for video signals contains only links between each pixel and its four immediate neighbors (pixels on the top, right, left, and bottom), *i.e.*, pixels within 1 pixel width. Defining $\tilde{\mathbf{A}} = (\mathbf{D} + \mathbf{I})^{-1}(\mathbf{A} + \mathbf{I})$, it can be shown that the model is also a specific GMRF, with zero mean and covariance matrix

$$\begin{aligned}
\boldsymbol{\Sigma}_{MA} &= \mathbb{E}[\mathbf{f}\mathbf{f}^T] && (3.10)\\
&= \mathbb{E}[(\tilde{\mathbf{A}}\boldsymbol{\epsilon} + \alpha\boldsymbol{\eta})(\tilde{\mathbf{A}}\boldsymbol{\epsilon} + \alpha\boldsymbol{\eta})^T] && (3.11)\\
&= \tilde{\mathbf{A}}\mathbb{E}[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T]\tilde{\mathbf{A}}^T + \alpha^2\mathbb{E}[\boldsymbol{\eta}\boldsymbol{\eta}^T] && (3.12)\\
&= \tilde{\mathbf{A}}\,\mathrm{diag}([\sigma^2_{\epsilon,1}, \sigma^2_{\epsilon,2}, \cdots, \sigma^2_{\epsilon,N}]])\tilde{\mathbf{A}}^T + \alpha^2\,\mathrm{diag}(\sigma^2_{\eta,1}, \sigma^2_{\eta,2}, \cdots, \sigma^2_{\eta,N}]). && (3.13)
\end{aligned}$$

In (3.13), the second term is a diagonal matrix consisting of noise variances, while the first term, which performs left and right multiplication of a diagonal matrix with adjacency matrix $\tilde{\mathbf{A}}$, contains only non-zero elements between each node and its neighbors within 2-hops (2 pixel width). In other words, the model assumes marginal independence between pixel that are more than two pixels away from each other. The assumption, although taking into account some local similarities between adjacent pixels, is also not realistic in general for image and videos, where similarities usually exist between pixels that are few pixels (more than 2 pixel width) away from each other.

3.2.2 Gaussian Markov Random Field (GMRF) Model

In our work, we define a generative GMRF model where the partial correlation, defined in the inverse covariance matrix (precision matrix) $\mathbf{Q}$, is based on the graph connectivity, *i.e.* $Q_{i,j} = 0$ if there is no connection on the graph between node v_i and v_j. In the applied GMRF, $\mathbf{Q}$ is defined as

$$\mathbf{Q} = (\mathbf{L} + \delta\mathbf{I}), \tag{3.14}$$

where δ is a small constant ≈ 0 used to ensure the invertibility of $\mathbf{Q}$. Based on the chain rule in probability theory, the covariance matrix for the proposed GMRF has non-zero elements for nodes connected with a path within the graph, which provides a more realistic model than *NM* and *MA* models considered in the related work. The described GMRF model has been adopted with great success in many applications in multimedia signal processing including [10, 84, 85].

3.2.3 Optimized Bipartition for GMRF

To find the optimal bipartition in terms of energy compaction, we apply the same objective function used in (3.1), which minimizes the magnitude of prediction residuals in $\mathcal{P}$. The bipartition $(\mathcal{P}, \mathcal{U})$ and prediction transform $\mathbf{P}$ are jointly optimized. For the GMRF model, given a chosen bipartition, the optimal predictor $\hat{\mathbf{f}}^*_{\mathcal{P}}$ for $\mathbf{f}_{\mathcal{P}}$ is the conditional expectation of $\mathbf{f}_{\mathcal{P}}$ given $\mathbf{f}_{\mathcal{U}}$, which is expressed as

$$\begin{aligned}\hat{\mathbf{f}}^*_{\mathcal{P}} &= \mathbb{E}[\mathbf{f}_{\mathcal{P}}|\mathbf{f}_{\mathcal{U}}] \\ &= \boldsymbol{\Sigma}_{\mathcal{P},\mathcal{U}}\boldsymbol{\Sigma}^{-1}_{\mathcal{U},\mathcal{U}}\mathbf{f}_{\mathcal{U}} \\ &= -\mathbf{Q}^{-1}_{\mathcal{P},\mathcal{P}}\mathbf{Q}_{\mathcal{P},\mathcal{U}}\mathbf{f}_{\mathcal{U}},\end{aligned} \tag{3.15}$$

i.e., the maximum a posteriori (MAP) estimation. The prediction error in (3.1) with the optimal prediction

$$\mathbf{P}^* = \boldsymbol{\Sigma}_{\mathcal{P},\mathcal{U}}\boldsymbol{\Sigma}^{-1}_{\mathcal{U},\mathcal{U}} = -\mathbf{Q}^{-1}_{\mathcal{P},\mathcal{P}}\mathbf{Q}_{\mathcal{P},\mathcal{U}} \tag{3.16}$$

can therefore be rewritten as

$$\begin{aligned}(\mathcal{P}, \mathcal{U}) &= \arg\min_{(\mathcal{P},\mathcal{U})} \Theta(\mathcal{P}, \mathcal{U}) \\ &= \arg\min_{(\mathcal{P},\mathcal{U})} \mathrm{Tr}(\mathbb{E}[(\mathbf{f}_{\mathcal{P}} - \mathbf{P}^*\mathbf{f}_{\mathcal{U}})(\mathbf{f}_{\mathcal{P}} - \mathbf{P}^*\mathbf{f}_{\mathcal{U}})^T]) \\ &= \arg\min_{(\mathcal{P},\mathcal{U})} \mathrm{Tr}(\boldsymbol{\Sigma}_{\mathcal{P},\mathcal{P}} - \boldsymbol{\Sigma}_{\mathcal{P},\mathcal{U}}\boldsymbol{\Sigma}^{-1}_{\mathcal{U},\mathcal{U}}\boldsymbol{\Sigma}_{\mathcal{U},\mathcal{P}}) \\ &= \arg\min_{(\mathcal{P},\mathcal{U})} \mathrm{Tr}(\mathbf{Q}^{-1}_{\mathcal{P},\mathcal{P}}),\end{aligned} \tag{3.17}$$

where the step going from 3^{rd} to the 4^{th} line above is based on the Schur complement [86]. This is an NP-hard problem, and therefore an approximation is required. In our work, we apply a greedy approximation for solving (3.17), which is summarized in Algorithm 1. In the algorithm, given a graph $G = (\mathcal{V}, \mathcal{E})$, the *update set* is initialized as an empty set $\mathcal{U}^0 = \emptyset$, and the *prediction set* $\mathcal{P}^0$, on the other hand, is initialized as the whole node set $\mathcal{V}$. The superscript denotes the iteration. In iteration t, the algorithm selects an optimal node v from $\mathcal{P}^{t-1}$ that minimizes the prediction error $\boldsymbol{\Theta}(\mathcal{P}^{t-1}/\{v\}, \mathcal{U}^{t-1} \cup \{v\})$ of the remaining prediction nodes given the signal in update set with node v included. The optimal node will then be added to the update set: $\mathcal{U}^t = \mathcal{U}^{t-1} \cup \{v\}$.The process continues until the target size $|\mathcal{U}|$ is reached.

Similar optimization approaches for partitioning have been used in applications such as dynamic networks [51] and active learning [39]. However, to solve the problem in (3.17)

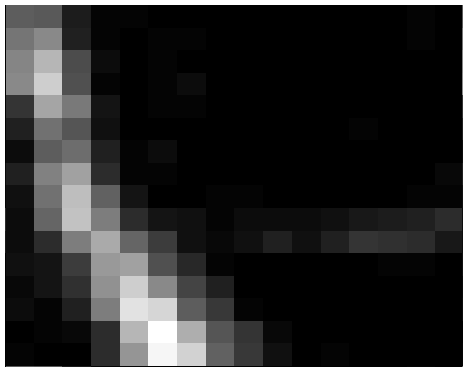
Algorithm 1 Greedy solution for bipartition

Input Graph $G = (\mathcal{V}, \mathcal{E})$ and target $|\mathcal{U}| = m$
Output Bipartition $(\mathcal{U}, \mathcal{P})$
1: Initialize $\mathcal{U}^0 = \emptyset$; $\mathcal{P}^0 = \mathcal{V}$
2: **for** $t = 1 : 1 : m$ **do**
3: Select $v* = \arg\min_v \Theta(\mathcal{P}^{t-1}/\{v\}, \mathcal{U}^{t-1} \cup \{v\})$
4: $\mathcal{P}^t \leftarrow \mathcal{P}^{t-1}/\{v*\}$
5: $\mathcal{U}^t \leftarrow \mathcal{U}^{t-1} \cup \{v*\}$
6: **end for**

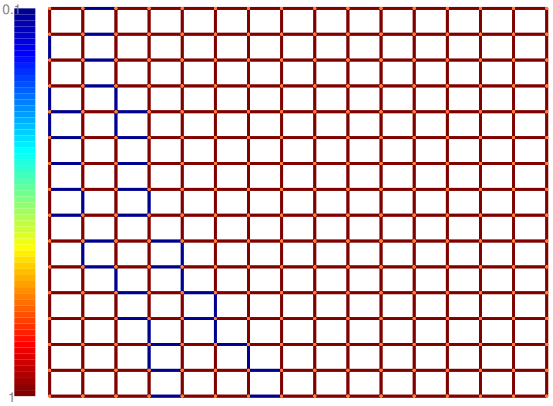
requires computing the inverse $\mathbf{Q}^{-1}_{\mathcal{P},\mathcal{P}}$ in every iteration t for each potential node v in $\mathcal{P}^{t-1}$, which would be too complex in practice. Therefore, in [39], the authors propose an efficient sequential optimization scheme. In this algorithm, eigen-decomposition of $\mathbf{Q}$ is performed at the beginning. For the remaining iterations, only matrix-vector multiplication for each candidate node in $\mathcal{P}^{t-1}$ is required. As a result, the computation complexity for bipartition can be reduced to $O(N^3)$.

3.2.4 Analysis of Proposed Bipartition

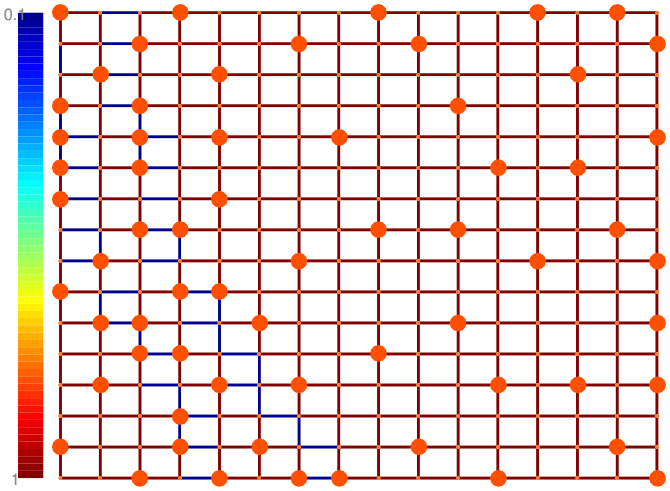
We observe that the selected nodes for $\mathcal{U}$ using the proposed algorithm are usually not distributed evenly. In fact, the distribution varies according to the local correlation defined in the GMRF model, as shown in the toy example in Fig. 3.1. In the example, pixel intensities have higher variation on the left side compared to variation on the right side of the block. We use a 4-connected grid graph to represent the block (Fig. 3.1(b)). The link weights are decided based on edges detected: the weight on links across edges are assigned a small weight $w < 1$, while other links have link weight 1. Then, we apply the proposed algorithm discussed in the previous section and select $\frac{1}{4}$ of the nodes to be included in $\mathcal{U}$ (Fig. 3.1(c)). As a result, the density of nodes selected for $\mathcal{U}$ is higher in the high variance regions (left) as compared to the smooth area. Intuitively, the pixels in low variance region have similar intensities even though the pixels may be several hops away from each other, and therefore selecting a large number of samples in these regions can be redundant. For pixels in high variance regions, on the other hand, a large number of update pixels is required in order to ensure good quality in prediction.



(a) Block with edge structure

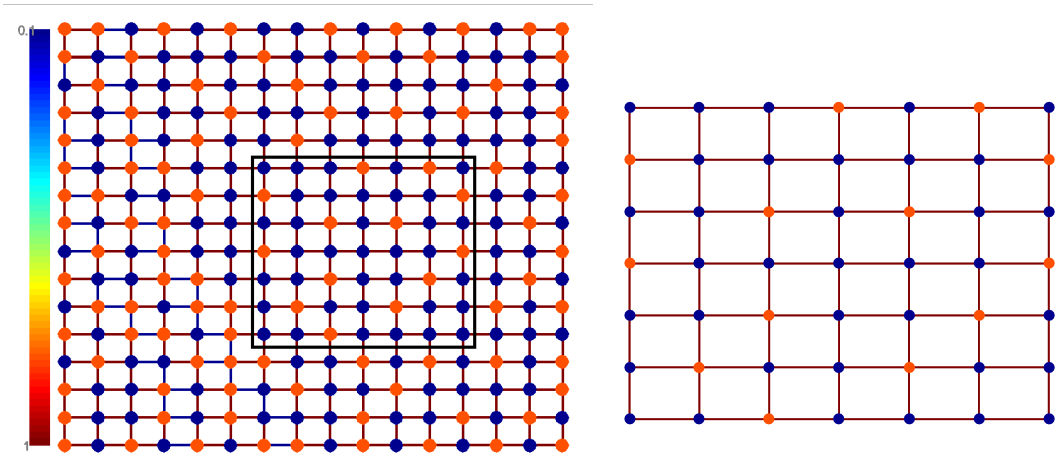


(b) 4 connected grid graph constructed based on edge structures in (a)



(c)

FIGURE 3.1: Example of $\mathcal{U}$ node selection by greedy algorithm of MAP error minimization



(a) (b)

FIGURE 3.2: Example of block with $\mathcal{P}$ nodes (blue nodes) with low connectivity to the update set (red nodes)

3.3 Bipartite Graph Formulation

In this section, we discuss the method for bipartite graph construction given the bipartition. In the previous work using lifting [53, 54, 56, 57], bipartite graphs were obtained by removing those links connecting two nodes in either $\mathcal{P}$ or $\mathcal{U}$, *i.e., conflicting links*. However, as mentioned in Section 3.2.4, using the proposed bipartition based on minimizing the MAP prediction error, the distribution density of nodes in the $\mathcal{U}$ varies depending on local correlation. As a result, nodes in $\mathcal{P}$ in the low variance areas, where distribution of $\mathcal{U}$ nodes have low density, tend to have low connectivity to $\mathcal{U}$. An example is shown in Fig. 3.2(a), where we consider the same graph used in Fig. 3.1(b), on which half of nodes are selected as $\mathcal{U}$ nodes. If we take a close look at the low variance area (marked with red box), it can be observed that there exist nodes in $\mathcal{P}$ that do not have any direct connection to $\mathcal{U}$, while most of the $\mathcal{P}$ nodes have only one neighbouring node in $\mathcal{U}$. That is, the nodes in $\mathcal{P}$ will have limited amount of information for prediction. As a result, high prediction error may occur in the smooth regions when using a localized prediction transform, *e.g.*, generalized CDF 5/3.

3.3.1 Kron Reduction based Reconnection

In order to ensure that every node in $\mathcal{P}$ has sufficient number of $\mathcal{U}$ neighbors for prediction, we propose a Kron reduction based reconnection approach in generating the bipartite graph G_{bpt} for transform. One important property of Kron reduction, as described in (2.10), is that it maintains graph connectivity, *i.e.* after removing nodes in $\mathcal{S}_c \subset \mathcal{V}$ ($\mathcal{V} = \mathcal{S} \cup \mathcal{S}_c$), two nodes

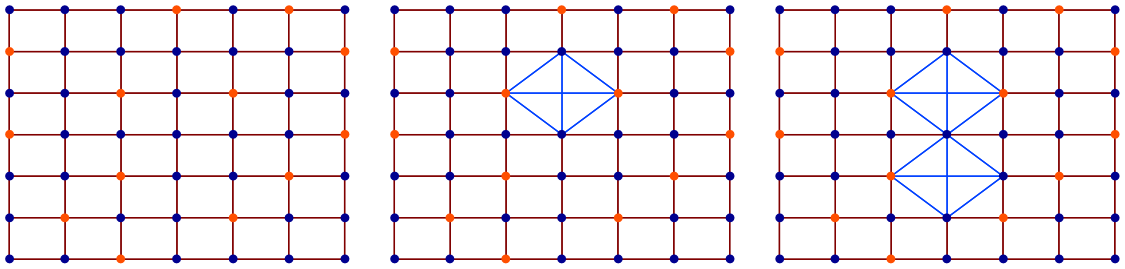


FIGURE 3.3: Example for iterative Kron reduction by removing 1 node at a time

Algorithm 2 Kron reduction based reconnection for $\mathcal{P}$

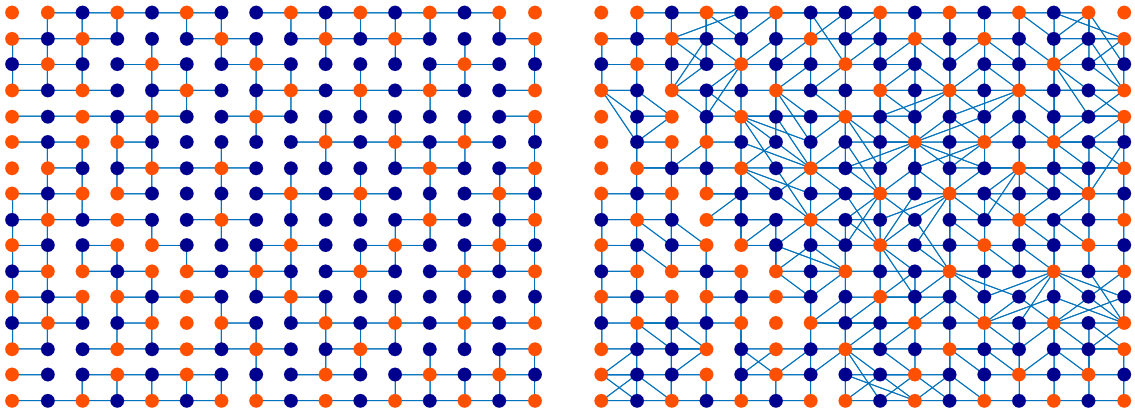
Input Adjacency matrix $\mathbf{A}$ of graph G and bipartition $(\mathcal{U}, \mathcal{P})$
Output Adjacency matrix $\mathbf{A}_{\text{bpt}}$ of the bipartite graph G_{bpt}

1: Initialize $\mathbf{A}_{\text{bpt}}$ as an empty $N \times N$ matrix
2: Compute degree matrix $\mathbf{D}$, where $D_{i,i} = \sum_j A_{i,j}$
3: Graph Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{A}$
4: **for** $v_i \in \mathcal{P}$ **do**
5: Define $\mathcal{U}^+ = \mathcal{U} \cup \{v_i\}$, and $\mathcal{P}^- = \mathcal{P}/\{v_i\}$
6: Compute the Kron reduction $\mathcal{L}_{\mathcal{U}^+} = \mathbf{L}_{\mathcal{U}^+,\mathcal{U}^+} - \mathbf{L}_{\mathcal{U}^+,\mathcal{P}^-}\mathbf{L}^{\dagger}_{\mathcal{P}^-,\mathcal{P}^-}\mathbf{L}_{\mathcal{P}^-,\mathcal{U}^+}$
7: Assign link weight connecting v_i and $v_j \in \mathcal{U}$ to be $\mathbf{A}_{\text{bpt}}(i,j) = -\mathcal{L}_{\mathcal{U}^+}(k,r)$, where k and r are the indices of v_i and v_j in the reduced Laplacian $\mathcal{L}$
8: Keep only the k links between v_i and $\mathcal{U}$ with the largest weights
9: **end for**

in $\mathcal{S}$ that were connected by a path through the removed nodes $\mathcal{S}_c$ will remain connected in the reduced graph. An example of Kron reduction after removing two nodes sequentially is shown in Fig. 3.3. Thanks to this property, for a node $v \in \mathcal{P}$ with low connectivity to the *update set* in G, we can generate links connecting v to $\mathcal{U}$ that are more than 1 hop away by removing other nodes in $\mathcal{P}$ using Kron reduction. The newly connected $\mathcal{U}$ nodes are those which have connection to v in the original graph by a path through the removed $\mathcal{P}$ nodes. In our construction for the bipartite graph, we apply the same process for every node v in $\mathcal{P}$. Then, a sparsification process, which for a given $\mathcal{U}$ keeps only k links with largest weights connecting to v, is applied for the resulting graph in order to reduce the complexity for the following transform stages. The detail of the algorithm is summarized in Algorithm 2.

3.3.2 Iterative Kron Reduction

Another useful property of the Kron reduction is that it can be computed iteratively, *i.e.* the Kron reduction applied after removing nodes in set $\mathcal{S}_c = \{v_1, v_2, \cdots, v_m\}$ leads to the same



(a) Bipartite graph built by removing links connecting nodes in the same set ($\mathcal{U}$ or $\mathcal{P}$

(b) Bipartite graph built using kron reduction and sparsification for nodes in $\mathcal{P}$

FIGURE 3.4: Example of bipartite graph construction using two different schemes

graph as removing node $v_i \in \mathcal{S}_c$ one at a time over m iterations and adapting the graph at each iteration. This avoids the computation of matrix inversion (the matrix inversion will become a simple divison by a constant). Note that the order of nodes removal does not affect the result. If the reduction is done in a certain order: $\{v_1, v_2, \cdots, v_m\}$, denoting $\mathcal{S}^t$ the node set kept in the t^{th} iteration, then the Kron reduction in iteration t is computed as

$$\mathcal{L}^t = \mathcal{L}^{t-1}_{-v_t,-v_t} - \frac{\mathcal{L}^{t-1}_{-v_t,v_t}\mathcal{L}^{t-1}_{v_t,-v_t}}{\mathcal{L}^{t-1}_{v_t,v_t}}, \tag{3.18}$$

where the index $-v_t$ corresponding to the nodes in $S^{t-1}/\{v_t\}$. In each iteration, the complexity depends on the number of nonzero elements in $\mathcal{L}^{t-1}_{-v_t,v_t}$, *i.e.* the number of links connecting to the eliminated node. Define c to be the maximum number of links connecting to the eliminated node through the iteration, the cost for eliminating one node takes $O(c^2)$ operations by performing the outer product $\mathcal{L}^{t-1}_{-v_t,v_t}\mathcal{L}^{t-1}_{v_t,-v_t}$ and the total complexity for Kron reduction will be $O(c^2N)$. In [22], the authors shows that the reduced graph from a graph with sparse connection will be sparse though Kron iteration in general, thus c is usually much smaller than N. For bipartite graph construction using the proposed reconnection, the process needs to be done for all the nodes in $v \in \mathcal{P}$. Note that during the Kron reduction at each node v, only the connection from v to $\mathcal{U}$ is considered, *i.e.*, only one row from (3.18) is calculated, so the cost for iterative Kron reduction at each node is reduced to $O(cN)$. Therefore, the overall complexity of re-connection for all the nodes in $\mathcal{P}$ is $O(cN^2)$ in one lifting level.

3.3.3 Probabilistic Interpretation

Using the Kron reduction based reconnection for every node in $\mathcal{P}$, we can assure that every node has enough neighbors in $\mathcal{U}$ for the prediction with the localized CDF5/3 transform to be used. Moreover, we can show that the generalized CDF5/3 predictor $\mathbf{P}_{\text{bpt}}$ applied on the bipartite graph G^*_{bpt} without sparsification, which is written as

$$\mathbf{P}_{\text{bpt}} = \mathbf{D}^{*\,-1}_{\text{bpt}\mathcal{P},\mathcal{P}} \mathbf{A}^*_{\text{bpt}\mathcal{P},\mathcal{U}}, \tag{3.19}$$

where $\mathbf{D}^*_{\text{bpt}}$ and $\mathbf{A}^*_{\text{bpt}}$ denote the degree and adjacency matrices of G^*_{bpt}, is equivalent to the MAP estimator in (3.15) for the defined GMRF model. A proof is provided in Appendix A. Therefore, the prediction transform we proposed (without bipartite graph sparsification) is optimal in terms of prediction error minimization. In addition, with the help of iterative algorithm, the Kron reduction based reconnection has lower complexity compared to computing directly the MAP estimator, which requires matrix inversion.

3.4 Experiments

In our experiments, we apply the lifting transform using the proposed bipartition and reconnection algorithms to test images and predicted residues from video data. The graph construction is based on the edge locations, which is the same approach used in our application for video compression in the next chapter. For the links across edges, a weight $w < 1$ is assigned, while the rest of the links are assigned weight 1. For baseline comparison, we consider the related approaches described in Section 3.1, namely the (1) Maximum spanning tree (MST), (2) Noise Model (*NM*) and (3) Moving Average (*MA*) model based methods. For the first approach, since there is no assumption for the types of prediction and update transforms in the bipartition design, we apply the same reconnection and transforms used in our proposed scheme. For the bipartition based on the *NM* and *MA*, on the other hand, we apply the same predictor as the assumption in the optimization described in [23, 52]: the predicted signal $\hat{f}_i$ on node $v_i \in \mathcal{P}$ is computed as

$$\hat{f}_i = \frac{1}{m_i} \sum_{j \in \mathcal{N}(i) \cap \mathcal{U}} f_j, \tag{3.20}$$

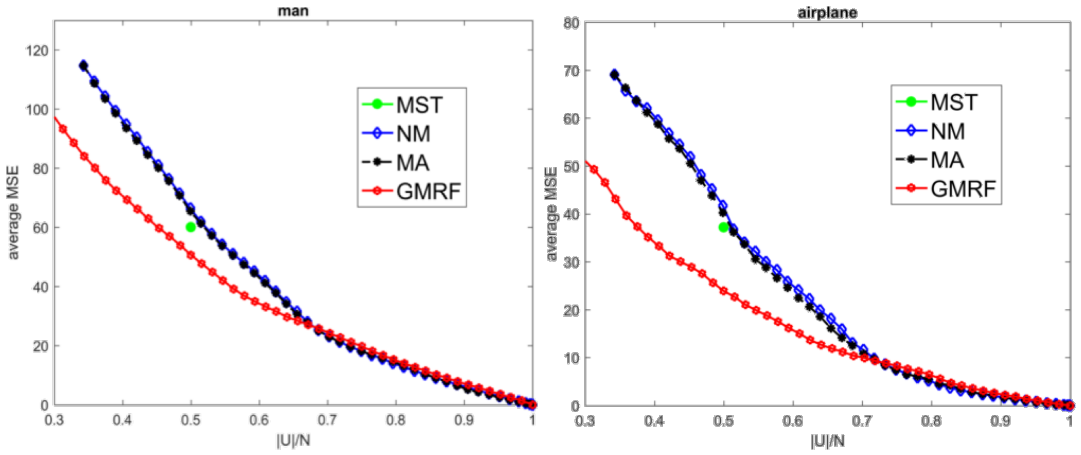
where $m_i = |\mathcal{N}(i) \cap \mathcal{U}|$. Same as the implementation in [23, 52], the weak links across image edges are removed before the bipartition and prediction. The update filter is orthogonalized using the method in [68] in all of these cases.

The comparison is based on the energy compaction in the *update set* $\mathcal{U}$ in the transformed domain, *i.e.* we compare the mean squared error of reconstruction after truncating the coefficients in $\mathcal{P}$ in the transformed domain. The test sets contains 4 images:*man, airplane, lena, peppers* and 2 intra-predicted residual sequences: *Cactus* and *Kimono*, consisting of 5 frames in each sequence. The images or video frames are firstly divided into multiple non-overlapping 8×8 blocks, where the lifting schemes are applied separately. In our results, we only consider blocks where some edges are present, and the mean squared errors are derived as the average for all the edge blocks. For MST, the number of nodes in set $\mathcal{U}$ is determined given the starting node in the implementation, while in *NM*, *MA*, and the proposed method, the size of $\mathcal{U}$ is decided by the users. In the experiments, we consider different bipartition rates $|\mathcal{U}|/N$ for the three methods. Note that for *NM* and *MA* model-based bipartition, $\mathcal{U}$ is initialized as the minimum Set Cover, and therefore the minimum bipartition rate is higher.

The results in Fig. 3.5 show that the proposed method using GMRF model consistently outperforms the baseline approaches especially for lower bipartition rate. This is because the proposed GMRF model captures more correlation between pixels that are further away from each other, which is usually high for pixels in smooth areas. Therefore, in the bipartition, the algorithm will select more nodes in high variation region for $\mathcal{U}$, as described in Section 3.2.4, resulting in better efficiency in prediction compared to other baseline approaches. Note that the experiment only considers a 1 level lifting transform. The extension to multi-level can be done by applying the same process of bipartition and transform iteratively from low level to high level on the downsampled graphs as done in [53, 54, 57, 68]. However, the optimization for bipartition considering multi-level is still an open question. In our application for video compression discussed in next chapter, we apply a different bipartition approach for multi-level lifting, which empirically gives better performance than the conventional method in [53, 54, 57, 68].

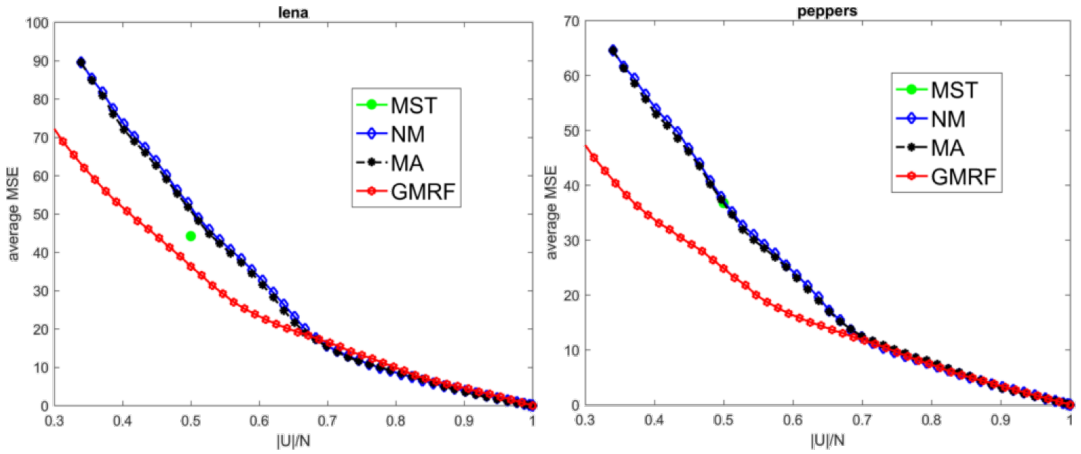
3.5 Summary

In this chapter, we defined the problem of optimal bipartition in the lifting transform in terms of energy compaction for the generative GMRF model. The model provides more accurate modeling for real signals. A greedy approximation was applied for selecting nodes to be included in the update set in the experiment. In addition, we proposed a reconnection approach for bipartite graph construction in order to solve the problem of connectivity loss caused by the uneven distribution of nodes in update set using the proposed bipartition. Our experimental results show that the proposed method outperforms the baseline bipartition approaches in terms of energy compaction.



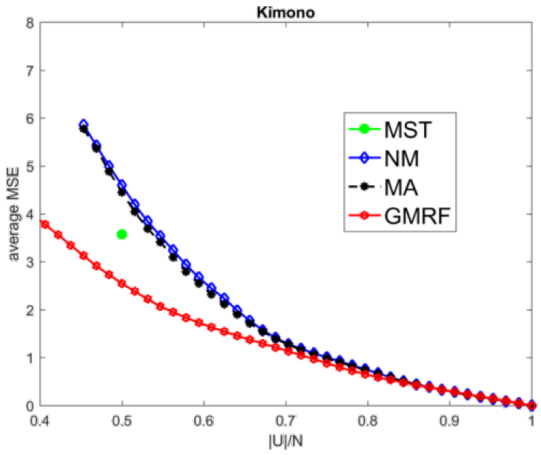
(a) Man

(b) Airplane

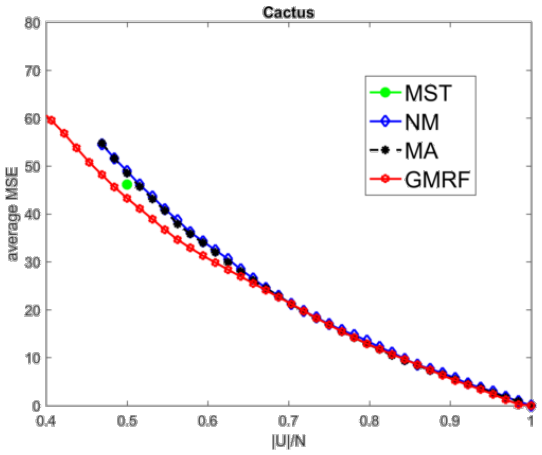


(c) Lena

(d) Peppers



(e) Kimono: intra-predicted residue



(f) Cactus: intra-predicted residue

FIGURE 3.5: Reconstruction error (MSE) after truncation coefficients in $\mathcal{P}$ for different bipartition rate

Chapter 4

Video Coding Application

In Chapter 3, we have discussed optimization of graph based lifting in terms of energy compaction in a probabilistic perspective. Specifically, we proposed an optimized bipartition based on a generative GMRF model derived from the graph connectivity, in order to maximize the energy compacted in the *update set* ($\mathcal{U}$) in the transformed domain. Besides, since the problem is NP-hard, we provided a greedy solution, which has shown promising results in energy compaction for different $|\mathcal{U}|$ as compared to the baselines methods. In this chapter, we apply the proposed lifting scheme to the application of intra-predicted video compression. In the design of image and video codecs, there are many components that need to be designed in addition to the transform scheme discussed in the previous chapter. These include 1) graph construction, 2) overhead signaling, 3) transformed coefficient scanning, and 4) rate distortion optimization.

The advantage of graph representation lies in its adaptation to different signal characteristics. A better graph representation in terms of the connection and weight assignment that captures the pair-wise similarity more accurately can enable better prediction and energy compaction. However, a more complex representation also increases the overhead required to describe the graph structure, *i.e.*, the information needed so that the decoder can construct the inverse transform. In this chapter, we will discuss a graph construction approache suitable for predicted video residues and its corresponding overhead signaling. Moreover, we will discuss the ordering of transformed coefficients and the associated entropy coding. Most of the work in this chapter was published in [80].

4.1 Graph Construction

We apply block based encoding, *i.e.* a video frame is divided into non-overlapping $m \times m$ blocks, on which transforms are applied. For graph construction in each block, we take a 4-connected grid graph, shown in Fig. 4.1, which has been widely adopted for image and video

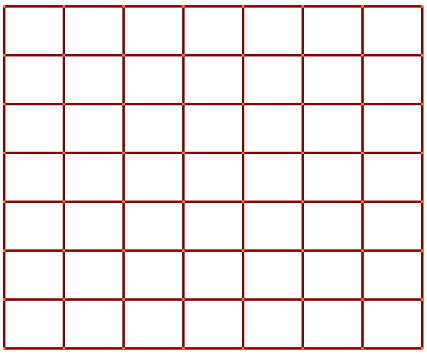


FIGURE 4.1: Example of 4-connected grid graph

frame representation, as a starting point. The weight assignment is based on the edge structure. For two nodes with an edge detected in between, the pair will be considered as having *weak correlation* and assigned a nonzero weight $w < 1$ on the link connecting them. While other node pairs will be considered as having *strong correlation*, with link weight equivalent to 1. This is similar to the method applied in [28] for depth map. The only difference is that instead of fully disconnecting links across edges, we assign a nonzero weak weight. This is because the intensity difference of pixels across edges in the intra-predicted video and other natural images is not as sharp as that encountered with depth maps. Also, edge structures are more complicated in intra-predicted residual sequences than in depth maps, and therefore by fully disconnecting the links across edges, lots of disconnected graph components may be produced.

The choice of w is based on the minimization of prediction error using the generalized CDF 5/3 transform for every node $v_i \in \mathcal{V}$ using the information from its one hop neighbors $v_j \in N(i)$. The CDF 5/3 predictor of node v_i is written as

$$\theta(i) = \frac{1}{\deg_i}\Big(\sum_{j \in \mathcal{N}(i)^{\mathrm{w}}} w f_j + \sum_{j \in \mathcal{N}(i)^{\mathrm{s}}} f_j\Big), \tag{4.1}$$

where $N(i)^{\mathrm{w}}$ and $N(i)^{\mathrm{s}}$ indicates the set of adjacent nodes of v_i with *weak correlation* and *strong correlation*, and $\deg(i)$ is the degree of v_i computed as

$$\deg(i) = w \cdot |\mathcal{N}(i)^{\mathrm{w}}| + |\mathcal{N}(i)^{\mathrm{s}}|. \tag{4.2}$$

The optimal weight w^* is found by solving the following minimization of the total prediction error:

$$w^* = \arg\min_{w} \sum_{i} (f_i - \theta(i))^2. \tag{4.3}$$

The optimization is performed on the intra-predicted frames from a set of training video sequences. Note that the problem in (4.3) is not convex since the normalization factor of (4.2) contains the unknown variable w. In our experiment, we apply gradient descent with multiple initial points generated randomly, and select the optimal w^* with minimum prediction error. For the construction of downsampled graphs in lifting level $\ell > 1$, we apply Kron reduction with sparsification, described in Section 2.4.3.

4.2 Bipartition Scheme

Algorithm 3 Boundary/Edge extension for sampling on graphs

Input Graph $G = (\mathcal{V}, \mathcal{E})$, and target $\mathcal{U}$ size m
Output $\mathcal{U}$ and $\mathcal{P}$ after sampling

1: Extend the Adjacency matrix $\mathbf{A}$ to include the extended nodes (called $\mathbf{A}_{ext}$) around boundaries and edges.
2: Compute the Laplacian matrix as $\mathbf{D}_{ext} - \mathbf{A}_{ext} + \delta\mathbf{I}$.
3: Initialize $\mathcal{U} = \emptyset$, $\mathcal{P} = \mathcal{V}$
4: **for** $t = 1 : 1 : m$ **do**
5: Choose the sample y s.t. y, along with its extended nodes $\{y', y'' \cdots\}$ and $\mathcal{U}^t$ in the previous, minimize the MAP error in set $\mathcal{P}$.
6: $\mathcal{U} = \mathcal{U} \cup \{y\}$, and $\mathcal{P} = \mathcal{P}/\{y\}$.
7: **end for**

For bipartition, we apply the optimized scheme based on MAP error minimization proposed in Section 3.2.3. Note that in GMRF, the diagonal element $Q_{i,i}$ in the precision matrix $\mathbf{Q} = \mathbf{L} + \delta\mathbf{I}$ can be interpreted as the inverse of the prediction error for node v_i given $\mathcal{V}/\{v_i\}$. Hence, the nodes around block boundaries and edges, which have lower degree, are considered to have large prediction error, and therefore are given higher priority in selection for update set. However, the pixels near boundaries tend to be further away from other pixels, thus having high density in sampling around boundaries reduces efficiency in prediction. To address this issue, we make the number of links for each node equal by using a symmetric boundary extension as shown in Fig. 4.2. As a result, the graph used for bipartition is augmented. The approach is consistent with the filterbank used later, which also uses a boundary extension with degree normalization. If a node v (*e.g.* node 11 in the example) is selected as

Algorithm 4 Bipartition in multi-level lifting transform

Input graph $G = (\mathcal{V}, \mathcal{E})$, maximum level q, and target size $\{|\mathcal{U}^m|\}_{m=1:q}$
Output $\{\mathcal{U}^m\}_{m=1:q}$

```
Initialize U = ∅ and P = V
for m = q : −1 : 1 do
    Initialize U^m = ∅
    for s = 1 : 1 : |U^m| do
        Select v_i* = arg min_{v_i} Θ(P/{v_i}, U ∪ {v_i})
        P = P/{v_i*}
        U = U ∪ {v_i*}
    end for
    U^m = U
end for
```

a sample to be included in $\mathcal{U}$, its mirrored nodes (denoted $11'$) are also selected. Note that the weight between extended node v' and a boundary node x (e.g., node 15) is equal to the weight between v and x. The same idea is also applied for nodes around edges. This method of bipartition including the boundary and edge extension is summarized in Algorithm 3. Note that in this work, we propose a novel bipartition strategy for multi-level decomposition. In [23, 52], the optimization for $\mathcal{U}^\ell$ in level ℓ considers only the minimization of errors stored in $\mathcal{P}^\ell$. It ignores the fact that since $\mathcal{U}^{\ell-1} = \mathcal{U}^\ell \cup \mathcal{P}^\ell$, the selection of $\mathcal{U}^\ell$ will also affect the prediction for $\mathcal{P}$ in the lower levels, *i.e.* $\{\mathcal{P}^{\ell-1}, \mathcal{P}^{\ell-2}, \cdots, \mathcal{P}^2, \mathcal{P}^1\}$. Therefore, in our bipartition scheme for multi-level decomposition, the update set $\mathcal{U}^\ell$ in level ℓ is optimized such that the prediction error in $\mathcal{S} = \{\mathcal{P}^\ell, \mathcal{P}^{\ell-1}, \cdots, \mathcal{P}^2, \mathcal{P}^1\}$ is minimized. The objective function is written as

$$\begin{aligned} \mathcal{U}^{\ell *} &= \arg\min_{\mathcal{U}^\ell} \mathbf{\Theta}(\mathcal{S}, \mathcal{U}^\ell) \\ &= \arg\min_{\mathcal{U}^\ell} \mathbb{E}(\|\mathbf{f}_\mathcal{S} - \mathbf{P}\mathbf{f}_{\mathcal{U}^\ell}\|)^2, \end{aligned} \tag{4.4}$$

where $\mathbf{f}_\mathcal{S}$ and $\mathbf{f}_{\mathcal{U}^\ell}$ correspond to the signals in $\mathcal{S}$ and $\mathcal{U}^\ell$ and $\mathbf{P}$ is the MAP estimator of $\mathbf{f}_\mathcal{S}$ given $\mathbf{f}_{\mathcal{U}^\ell}$. The greedy algorithm is summarized in Algorithm 4.

4.3 Transform Design

The bipartition approximation G_{bpt} is constructed by reconnecting $\mathcal{P}$ nodes using the Kron reduction, as discussed in Section 3.3. As mentioned, the method solves the problem of connectivity loss in $\mathcal{P}$ due to the uneven distribution of samples in $\mathcal{U}$. The prediction and

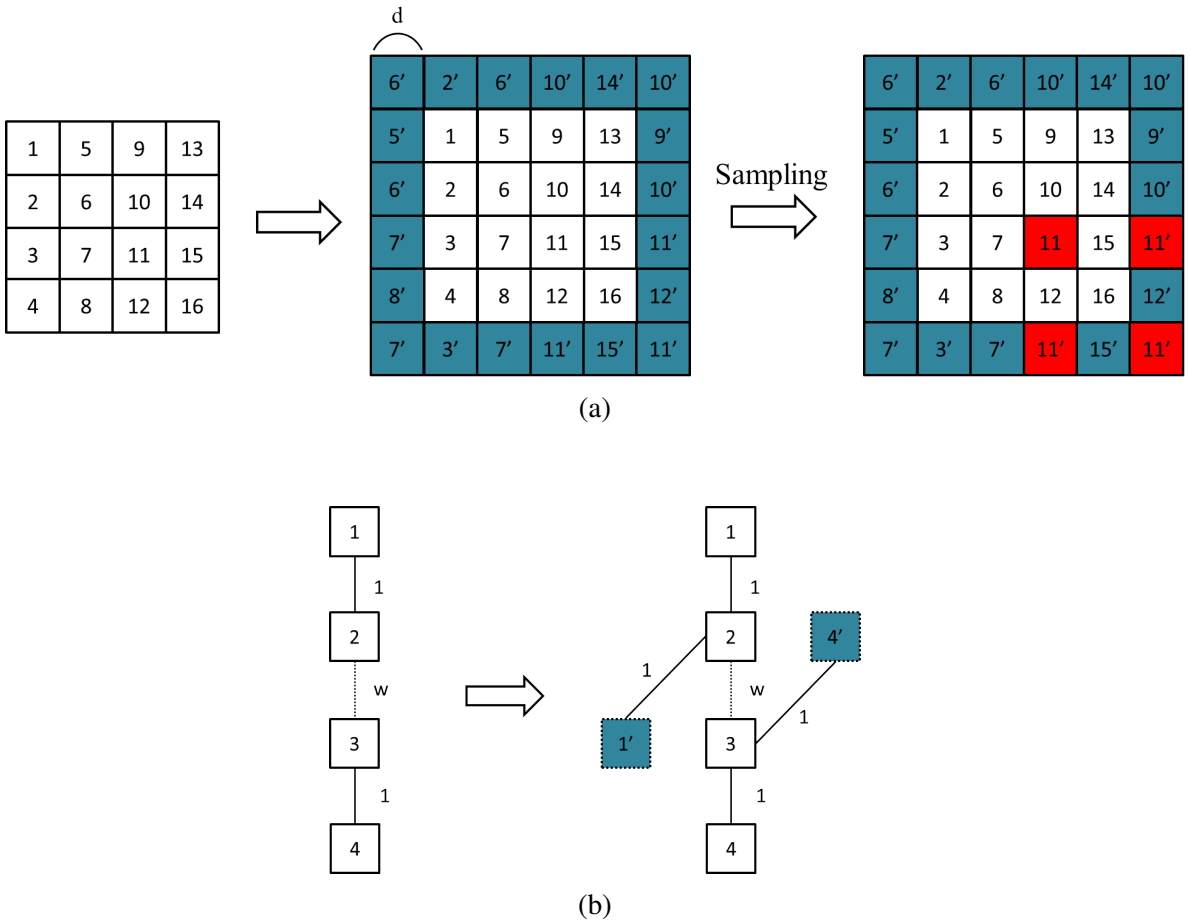


FIGURE 4.2: Boundary extension for pixels around (a) block boundaries and (b) edges

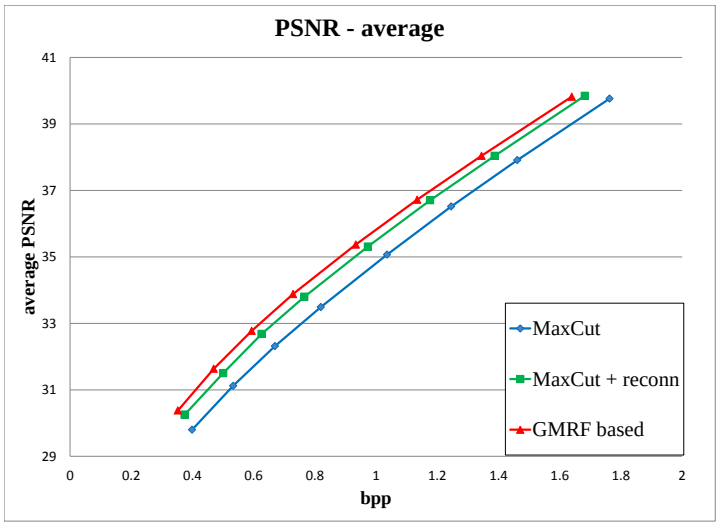


FIGURE 4.3: The comparison between proposed lifting scheme, MaxCut based lifting, and the MaxCut based lifting with proposed re-connection technique.

update transforms are based on the generalized CDF5/3 filterbanks and orthogonalization. We compare the coding gain of the proposed bipartition and re-connection approach (*reconn-GMRF*) with the MaxCut based lifting in [54] (*MaxCut*). The test sequences consist of 7 frames from 7 video sequences. The result is shown in Fig. 4.3. we also include results of MaxCut based bipartition with re-connection using Kron reduction (*reconn-MaxCut*). Note that even with a simple bipartition scheme such as MaxCut based method, using re-connection it leads to performance comparable to *reconn-GMRF*, making further simplification of bipartition a direction for future work.

4.4 Overhead Signaling and Entropy Coding

In a practical image/video codec, the transformed coefficients are usually scanned and entropy coded in a specific order, such as the zigzag scanning in the conventional DCT, where the coefficients are scanned from low frequency components to high frequency components. If the signal energy is well compacted into the low frequency subband, the high frequency subband will contains lots of zeros after quantization, leading to highly efficient encoding since the number of nonzero components for scanning is small. In the proposed lifting scheme, we optimize the energy compaction in the *update set*, and repeat the same process from the 1^{st} level to the highest level of decomposition. As a result, the lifting coefficients with large magnitude will tend to be concentrated in the high levels, while the small coefficients will most likely be contained in low levels. Therefore, we adopt the same coefficient reordering method proposed in [53]. Defining $\mathbf{d}^{\ell}$ and $\mathbf{s}^{\ell}$ as the detail and smooth coefficients stored in $\mathcal{P}$ and $\mathcal{U}$ in the ℓ^{th} level, the coefficients will be ordered as $[\mathbf{s}^{q}, \mathbf{d}^{q}, \mathbf{d}^{q-1}, \cdots, \mathbf{d}^{2}, \mathbf{d}^{1}]$ before scanning. The coefficients within each level will be ordered based on their *reliability*, defined as the average of weights on links connected to the node. In general, a node with low *reliability* has higher prediction error, and therefore should be scanned earlier than a node with high *reliability*. For entropy coding, we apply an amplitude group partition technique called AGP [66]. AGP can learn and adapt to different coefficient distributions, thus providing fair comparison between different transforms. Before quantization, the coefficients of the CDF53 lifting transform are first normalized based on [74] so as to compensate for the lack of orthogonality. For the signaling of graph geometries, we use the Arithmetic Edge Coding (AEC) proposed in [18].

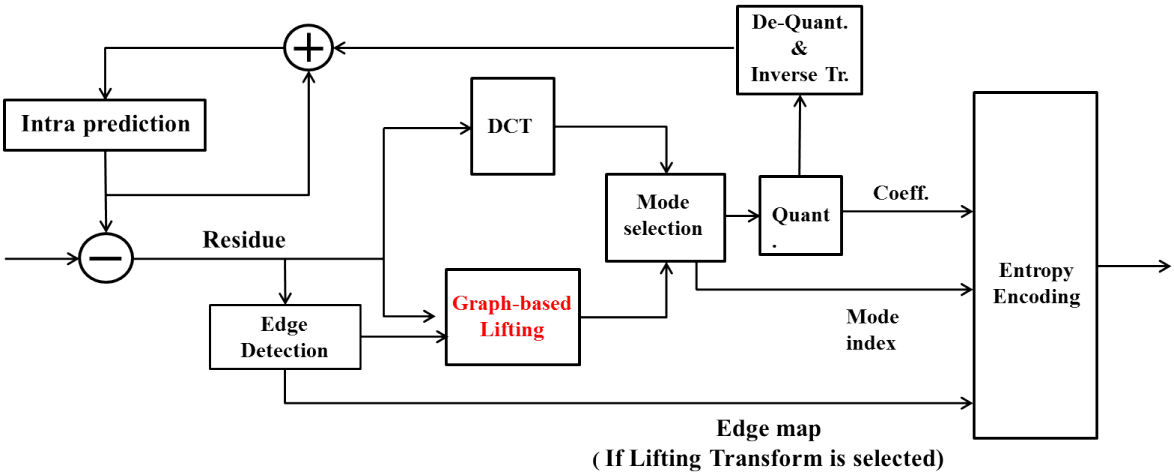


FIGURE 4.4: Encoder for intra-predicted videos with mode selection between DCT and the graph based lifting transform.

4.5 Experimental Results

In our experiments, we generate the intra-predicted residual frames for test sequences *Foreman*, *Mobile*, *Silent*, and *Deadline* using HEVC(HM-14) with transform unit size fixed as 8×8. The encoder system is shown in Fig. 4.4. In order to deal with the trade-off between transform quality and signaling overhead, for the transform coding, the encoder will select between the proposed graph based lifting scheme, which requires sending edge locations, and the conventional DCT. The selection is based on Rate Distortion Optimization (RDO). We define SSE to be the sum of squared error of the reconstructed signal, and R as the bitrate. For graph based lifting, both the bits for coefficient encoding and the edge information overhead are considered for the bitrate. The RD cost is computed as $\text{RD}_{\text{lifting}} = \text{SSE}_{\text{lifting}} + \lambda(\text{R}^{\text{coeff}}_{\text{lifting}} + \text{R}^{\text{edge}}_{\text{lifting}})$, where R^{coeff} and R^{edge} correspond to the bitrate needed for encoding transformed coefficients and the graph geometry based on edges. DCT is chosen if there is no edge component in the block or if its RD cost, computed as $\text{RD}_{\text{DCT}} = \text{SSE}_{\text{DCT}} + \lambda\text{R}^{\text{coeff}}_{\text{DCT}}$, is smaller than $\text{RD}_{\text{lifting}}$. The parameter λ is chosen as $0.85 \cdot 2^{(QP-12)/5}$, and the *QP* values used in this experiment are from 24 to 36 with step size 2. The flags for transform selection are signaled using arithmetic coding. In order to further reduce the overhead cost, in each 8×8 block only one contour is allowed for AEC.

We compare the the proposed lifting scheme (*reconn-GMRF*) with 4 baseline approaches, including the DCT, the GFT, *MaxCut*, and *reconn-MaxCut*. The DCT coefficients are zig-zag scanned, and the GFT coefficients are scanned from the smallest eigenvalue to the largest eigenvalues. The average PSNR gain and bitrate reduction are presented in Table 4.1. For

Methods	GFT		**reconn-GMRF**		reconn-MaxCut		MaxCut	
	Δ PSNR (dB)	**Δ rate (%)**	**Δ PSNR (dB)**	**Δ rate (%)**	**Δ PSNR (dB)**	**Δ rate (%)**	**Δ PSNR (dB)**	**Δ rate (%)**
Foreman	0.34	-7.28	0.29	-6.42	0.26	-5.77	0.17	-3.63
Mobile	0.17	-1.46	0.10	-0.97	0.10	-0.96	0.08	-0.51
Silent	0.22	-4.28	0.20	-3.88	0.18	-3.58	0.09	-1.66
Deadline	0.37	-4.97	0.31	-3.98	0.30	-3.90	0.24	-3.19

TABLE 4.1: PSNR-bitrate comparison with Bjontegaard metric. The negative value for Δrate indicates the average bitrate reduction against DCT, and the positive Δ PSNR shows the average PSNR gain.

videos with simple edge structures such as *Foreman* and *Deadline*, graph based lifting has around $0.3dB$ gain in PSNR. While in the videos with complicated edge structure such as *Mobile*, the gain is limited since the edge map dominates the cost. We also show that *reconn-MaxCut* provides a good approximation to high complexity *reconn-GMRF* and GFT.

4.6 Summary

In this chapter, we described the application of our proposed transform coding using graph based lifting transform in video coding. Novel approaches in graph weights assignment and multi-level bipartition in lifting are proposed. we also discussed detail in designing encoding system, *e.g.*, transform selection and signaling overhead. The results outperforms the conventional DCT based encoding, and their performance approximates that of the high complexity GFT based encoding.

Chapter 5

Application: Pre-demosaic Light Field Image Compression

5.1 Introduction

In this chapter, we introduce a new application of graph based transforms to light field (LF) image coding[1]. LF imaging separately captures light rays arriving from different directions at each pixel in the image sensor. With acquired LF data, multi-view rendering and re-focusing become possible post-capture. However, due to the additional information of light ray directions, acquired LF data are large in volume compared to conventional color images of the same resolution, and hence efficient compression of LF data is important for storage and transmission.

In the last decade, many hardware designs have been developed for LF acquisition, including multiple camera arrays, aperture cameras, and lenselet-based plenoptic cameras. Among them, the lenselet-based plenoptic camera is the most popular, and has been made commercially available by companies such as Lytro [50] and Raytrix [62]. In a plenoptic camera, a

[1]A part of the content in this chapter was published in [8]

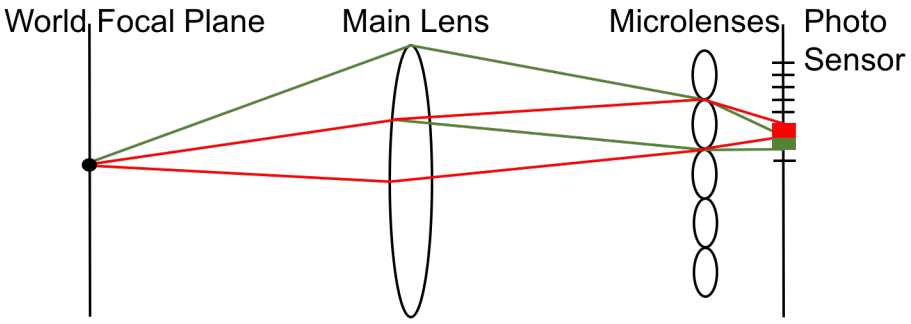


FIGURE 5.1: Conceptual system of lenselet-based plenoptic camera

microlens array is placed ahead of an otherwise conventional photo sensor embedded with Bayer color filter, as shown in Fig. 5.1. The resulting raw image is called *lenselet image*. Since light fields are commonly represented and processed as 4D functions [45, 63, 72], the acquired lenselet image typically undergoes demosaicking (pixelwise RGB interpolation) and conversion to multiple sub-aperture images on a 2D array. Each sub-aperture image can be seen as a typical 2D photo, gathering pixels from a specific light direction.

There exist two types of redundancies within the 4D LF representation: i) spatial redundancy among neighboring pixels in a sub-aperture image, *i.e. intra-view correlation*, and ii) angular redundancy among sub-aperture images of nearby directions, *i.e. inter-view correlation*. The existing works in literature for lenselet-based LF compression can be roughly grouped into two categories that make use of classic image and video coding techniques. The first one encodes the entire 2D array of sub-aperture images as an image using an image coding standard, *e.g.* JPEG or Main Still Picture Profile in HEVC [15, 16, 65]. The intra and inter-view correlations are exploited using the regular intra-prediction modes (Angular, Planar and DC modes) and a newly introduced *Self Similarity* (SS) mode, which is similar to the Intra Block Copy (IBC) in HEVC screen content extension. The other type is the pseudo sequence based approach [3, 21, 25, 30, 48, 69], where each sub-aperture image is treated as a video frame in a pseudo video sequence. The intra and inter-view correlations are exploited using the intra and inter prediction in the video coding standard, *e.g.* H.264 and HEVC.

Exploiting inter-view correlation in compression leads to high computation complexity (due to motion / disparity prediction) and creates dependencies among coded sub-aperture images, which is undesirable for random access. In particular, in an archiving scenario, a user may desire to quickly browse through viewpoint images, each of which can be synthesized in acceptably high quality using only a small subset of sub-aperture images. Thus, speedy extraction of this image subset from the LF data compressed in high quality is important. Furthermore, we note that most standard digital cameras use a low complexity codec (JPEG) operating by default at very high PSNR. Similarly, in this paper we will consider an intra-view only approach (which leads to faster encoding and better random access), operating at high rates/PSNR.

Another problem in the aforementioned existing works is that the compression schemes are applied on the full color sub-aperture images, where large redundancies are introduced by demosaicking. Moreover, to incorporate standard codecs, an RGB sub-aperture image must be converted to 4:2:0 YUV format, which induces distortions due to integer rounding and color sub-sampling.

In this paper, we propose a new coding scheme, where *compression is applied on the original lenselet images captured by the photo sensor*, without the aforementioned pre-processing that increases data volume or distorts captured pixel values. Our work is inspired by schemes proposed in [12, 13, 43, 44] for regular images, which also postpone the demosaicking step to the decoder. Specifically, we first map the raw captured pixels directly onto sparse locations in a series of sub-aperture images. Unlike the input images for compression in [12, 13, 43, 44], where R, G, and B pixels are regularly distributed based on the Bayer pattern, the color components after the mapping to sub-aperture images are irregularly placed, making it difficult to be encoded using conventional schemes, *e.g.,* JPEG or All-Intra mode in HEVC.

In our proposed scheme, novel intra-prediction and transform coding are designed for non-demosaicked LF data, consisting of sparsely distributed pixels. Specifically, for intra-prediction, we estimate the local characteristics of each block based on the structure tensor according to the available pixels. The information will then be used to adjust the shape of kernel functions used for prediction. For transform coding, the irregularly distributed pixels in a sub-aperture image will be connected as a graph, with the pixel values interpreted as a *graph-signal*. The graph weights, which reflect similarities between connected sample pairs, are optimized based on Gaussian Markov Random Field (GMRF) modeling of signal. The problem of learning the optimal graph structure, *i.e.*, graph connection and weights, based on statistical modeling for image and video signal has been well studied in the last decade [24, 34, 36, 60]. However, the methods can only be applied on data with regular pixel placement. In this work, we consider the graph learning problem based on sparsely distributed pixels, namely blocks with missing observations on some pixel positions. Due to the large amount of data size, the graph-signals are encoded using the low complexity graph-based lifting transform described in the previous few chapters. In our experiments, we apply the proposed LF coding scheme on a dataset captured with Lytro Illum. Compared to the state of the art HEVC-based coding, the results show noticeable gains at high PSNRs, which is useful for LF rendering in an archival scenario.

The rest of the chapter is organized as follows. In Section 5.2, we present the notation used throughout this chapter and a review of conventional approaches in lenselet-based LF image compression. Our proposed coding scheme is described in Section 5.3. In sections 5.4 and 5.5, we describe the proposed intra-prediction and transform coding based on sparsely distributed pixels. Experiments and conclusions are presented in sections 5.6 and 5.7 respectively.

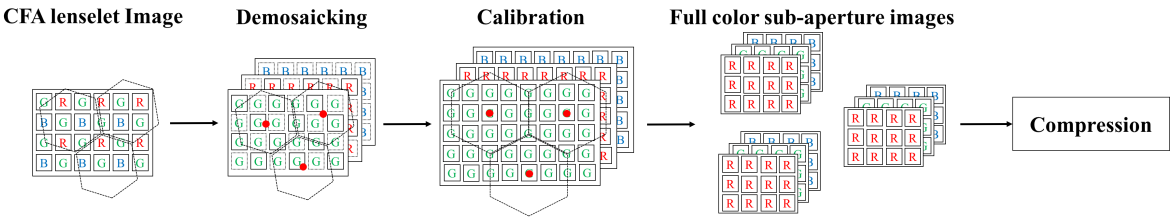


FIGURE 5.2: Conventional encoder for light field image. The demosaicking and calibration processes are applied before compression.

5.2 Notations and Background

5.2.1 Notations

In this chapter, the elements of a 2-dimensional vector (*e.g.,* $\mathbf{v} = [v_1, v_2]^T$) in the Euclidean space will represent the directions along row-axis and column-axis. The notation is commonly used for image coordinate systems. Compared to the definition in Section 2.1, we consider here a more general graph structure $G_{\mathrm{G}} = (\mathcal{V}_{\mathrm{G}}, \mathcal{E}_{\mathrm{G}})$, where for every node v in $\mathcal{V}_{\mathrm{G}}$, there is an associated non-negative *self loop* weight h_i. The corresponding graph Laplacian matrix, called *generalized graph Laplacian*, is defined as $\mathbf{L}_{\mathrm{G}} = \mathbf{D} - \mathbf{A} + \mathbf{H}$, where $\mathbf{H}$ is a diagonal self loop matrix with element $H_{i,i} = h_i$. $\mathbf{D}$ and $\mathbf{A}$ are the degree matrix and adjacency matrix defined in Section 2.1. Note that the commonly used combinatorial Laplacian $\mathbf{L}$ is a special case of $\mathbf{L}_{\mathrm{G}}$ with zero weight for all self loops.

5.2.2 Background: Compression after De-mosaicking

Fig. 5.2 depicts the conventional coding scheme for light field image compression, where the captured Bayer patterned lenselet image is first converted into an array of full color sub-aperture images, followed by compression using standard image/video codecs. In the figure, the conversion process is based on the method proposed by Dansereau et al. [17, 47]. Through the Bayer filter embedded on the photo sensor, each pixel on the captured lenselet image contains only one color component out of R, G, and B. In order to generate full color images, the missing color components at each pixel have to be interpolated using the nearby pixels where the target colors are available. The process is called *demosaicking*. In this work, we apply the demosaicking approach proposed by Malvar et al. [31] to the lenselet image. The number of pixels to be encoded will be increased threefold through the process regardless of the demosaicking algorithms used.

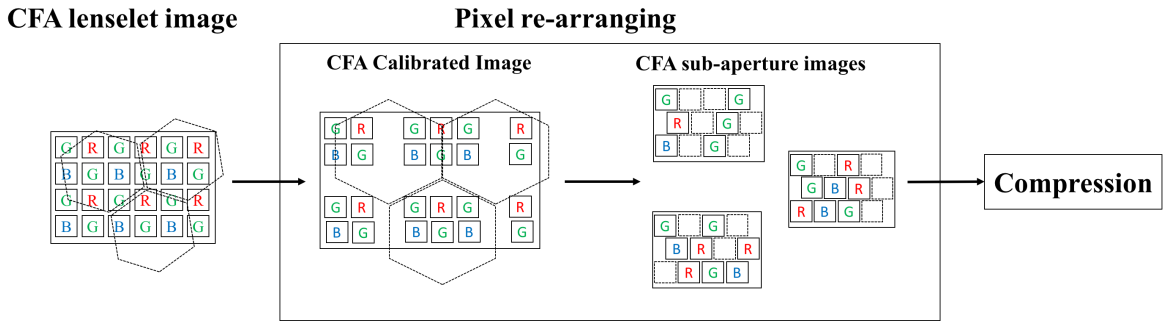


FIGURE 5.3: Proposed LF encoder, where compression is applied on the raw lenselet data without demosaicking

Projected from the microlens array in the plenoptic camera, a lenselet image consists of multiple hexagonally arranged pixel patches, which are called *macro-pixels* (represented with a dash line in Fig.5.2); each macro-pixel collects light for one image pixel arriving from different directions. However, due to manufacturing defects, the arrangement of macro-pixels is usually not aligned with the image coordinates, making it difficult to infer a pixel's corresponding position in the scene and the arriving light angle. In [17], the center of each macro-pixel is located through the help of white images, which are lenselet images taken through a white diffuser, or lenselet images of a white scene. The center locations are estimated by finding the local maximum of brightness on the white images. Then, the color lenselet image needs to be calibrated via rotation, translation and scaling, so that each macro-pixel center (denoted with red point in the figure) falls onto an integer pixel location and the arrangement of macro-pixels is aligned to the regular hexagonal grid. Through the calibration, the amount of data will also be increased due to the interpolation involved in scaling.

Each pixel in the calibrated image is indexed by its spatial and angular coordinates. The spatial coordinate is given by the position of the associated macro-pixel and the angular coordinate is the relative location within each macro-pixel. We then collect pixels of the same angular coordinate into one sub-aperture image, where the pixels are arranged according to their spatial coordinates. Each sub-aperture image can be viewed as a typical 2D picture, where large spatial correlation exists between neighbouring pixels.

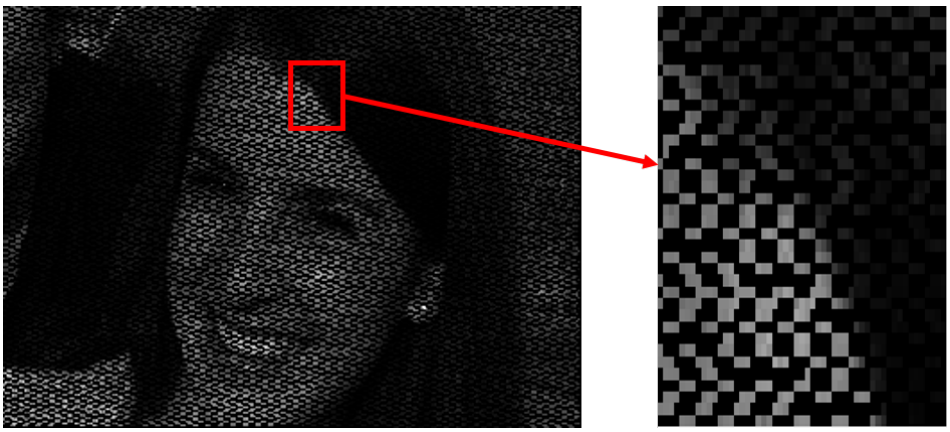


FIGURE 5.4: Sparsely distributed G components on one sub-aperture image (Figure *Friends1* from EPFL light field dataset)

5.3 Proposed scheme: Compression before De-mosaicking

In the pre-processing stage of the conventional coding scheme, the volume of LF data is increased greatly during demosaicking and the scaling operation needed for calibration. In order to avoid these redundancies, we propose a new coding scheme for LF in which compression is performed on the data collected in the original lenselet image instead of the pre-processed pixels in the full color sub-aperture images. The flow chart is shown in Fig. 5.3. Without demosaicking, we map raw pixels onto the calibrated lenselet image according to the transformation matrix applied in [17]. Pixels that fall onto non-integer locations after transformation will be rounded to the nearest integer positions. Then, based on the relative locations within the macro-pixels on the calibrated image, pixels are arranged onto multiple sub-aperture images, where redundancies between spatial neighbours can be exploited. Due to the placement of macro-pixels, the pixels in the sub-aperture images will be placed hexagonally, which is different from the conventional pipeline, where sub-aperture images are re-sampled into rectangular placement. Note that the pixels around boundaries of each macro-pixel, which have large noise due to underexposure, are discarded in the pipeline described in [17]. However, in the rearrangement process in our proposed scheme, the boundary pixels will be kept. The mapping will not change the number of pixels nor the intensity values of R, G, and B components from the raw data.

Since no interpolation is applied, some pixel locations are empty in the sub-aperture images, as shown in Fig. 5.4. Depending on the camera manufacturing, *i.e.,* the type of macro-pixel misalignment and calibration algorithm adopted, the spatial and angular coordinates for each pixel on the captured lenselet image may be different. Therefore, the pattern of pixel

distribution in sub-aperture images is not fixed and is also highly irregular. This is in contrast with the input signal considered in the pre-demosaic image coding schemes proposed in [12, 13, 43, 44], where R, G, and B pixels are distributed regularly based on Bayer pattern. Due to such irregularity of spatial distribution for LF data, existing coding techniques, *e.g.,* discrete cosine transform (DCT) and discrete wavelet transform (DWT), cannot be easily applied. This motivates the use of graphs, which can represent both regular and irregular data points as long as the pair-wise relations can be defined properly. In the next section, we will first describe the proposed intra-prediction that can predict irregularly spaced pixel based on pixels in the decoded neighbouring blocks. Then, in Section 5.5, the graph construction for graph-based coding of sparsely distributed pixels will be discussed.

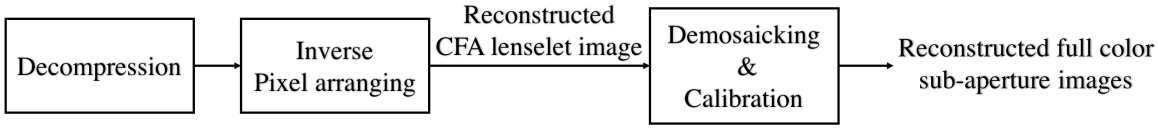


FIGURE 5.5: Decoder in the proposed LF coding scheme, where demosaicing and calibration are applied to generate full color sub-aperture image array

At the decoder side, pixels in sub-aperture images are de-compressed and inverse-mapped back to their original positions in the 2D lenselet image, as shown in Fig. 5.5. The image will then be demosaicked and calibrated [17, 47] in order to generate full color 4D LF for further processing, *e.g.,* multi-view rendering and re-focusing. Note that our scheme does not rely on a particular selection of demosaicking and calibration algorithms. In fact, multiple works have been proposed over the last few years in LF image calibration and demosaicking. For example, in [79] and [82], different methods are applied to locate macrolens centers by examing the dark pixels and line features from the white image. Therefore, the spatial and angular coordinates estimated are different from the ones in [17], which leads to different types of sparse distribution within sub-aperture images. In [55, 79], a different strategy of demosaicking is considered which performs color interpolation on each of the sub-aperture image instead of the whole lenselet image. Our coding scheme can be easily adapted by using different transform matrices for pixel rearrangement in the encoder side according to the calibration algorithm applied. The demosaicking and calibration strategies at the decoder side can also be adjusted accordingly.

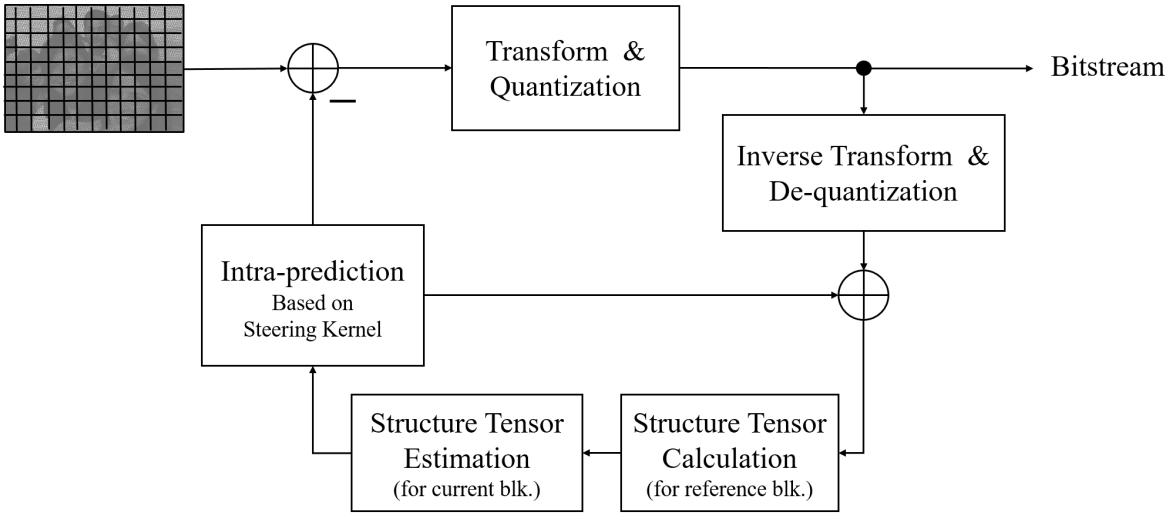


FIGURE 5.6: Proposed Intra-prediction system for sparsely distributed pixels

5.4 Proposed Intra-prediction Scheme

In order to facilitate random access for rendering, each sub-aperture image is encoded independently. A sub-aperture image will be divided into non-overlapping $m \times m$ blocks, as transform units. A coding scheme with variable block sizes, *e.g.*, the quad-tree partition in HEVC, is left as a possible future work.

5.4.1 Gradient Estimation

In the latest video standards, *e.g.*, H.264 and HEVC, an intra-prediction mode is selected through exhaustively searching all the possible directions to find the one with the minimum rate distortion (RD) cost. The process increases the implementation complexity significantly as the number of directions allowed increases. In order to reduce the complexity, many methods have been proposed in literature [4, 26, 37, 40], which exploit the local characteristics, *e.g.* edge direction, to reduce the number of candidate modes searched. In these works, the information of local gradients is used to determine the dominant edge direction within each block. The intra-modes closely aligned with the estimated edge direction are of higher priority in mode selection. However, all these methods are restricted to regular videos with a complete set of pixels within each block. In this section, we propose a new intra-prediction algorithm based on the structure tensor for intra-predicting the sparsely distributed pixels before demosaicking.

In Fig. 5.6, we show the flow chart of our proposed intra-prediction system. Pixels in the decoded neighbouring blocks are used as references for each block in the sub-aperture

images. Based on the structure tensor, which provides information about local gradient in the reference blocks, we estimate the structure tensor of the current block to be encoded. Therefore, overhead, such as indices for intra-prediction mode, does not need to be signaled. With structure tensor, edge direction and strength within a block can be estimated. This information allows us to define the rotations and stretches of the adaptive kernels used for intra-prediction. Besides the estimation of structure tensor, where we utilize the correlation between different color channels, all other steps are applied on R, G, and B components separately.

The structure tensor of a block B is calculated as

$$\mathbf{H}_{\mathrm{B}} = \Sigma_{i\in\mathrm{B}} \nabla\mathbf{f}(i) \nabla\mathbf{f}(i)^T = \Sigma_{i\in\mathrm{B}} \begin{bmatrix} d_r(i)^2 & d_r(i)d_c(i) \\ d_c(i)d_r(i) & d_c(i)^2 \end{bmatrix}, \tag{5.1}$$

where $d_r(i)$ and $d_c(i)$ denote the gradient of pixel intensities along row and column axis on pixel i.

In our work, the gradient $\nabla\mathbf{f}(i)$ for each pixel i is estimated with linear regression. Using Taylor's theorem, the pixel value at a given point $\mathbf{x}$ can be expressed as

$$f(\mathbf{x}) = f(\mathbf{a}) + \nabla\mathbf{f}(\mathbf{a}) \cdot (\mathbf{x} - \mathbf{a}) + O(\|\mathbf{x} - \mathbf{a}\|^2), \tag{5.2}$$

where $\nabla\mathbf{f}(\mathbf{a})$ is the gradient vector at point $\mathbf{a}$. For $\mathbf{a}$ sufficiently close to $\mathbf{x}$, the pixel value $f(\mathbf{x})$ can be well approximated with the first two terms

$$f(\mathbf{x}) \approx f(\mathbf{a}) + \nabla\mathbf{f}(\mathbf{a}) \cdot (\mathbf{x} - \mathbf{a}). \tag{5.3}$$

Based on the linear approximation, we estimate the gradient $\nabla\mathbf{f}(\mathbf{a})$ at point $\mathbf{a}$ by fitting a hyperplane that best satisfies (5.3) for a number of pixels $\{\mathbf{x}_1, \mathbf{x}_2, \cdots \mathbf{x}_k\}$ close to $\mathbf{a}$. The fitting can be represented as an overdetermined system:

$$\mathbf{F} = \mathbf{X} \cdot \nabla\mathbf{f}(\mathbf{a}) \tag{5.4}$$

$$= \begin{bmatrix} f(\mathbf{x}_1) - f(\mathbf{a}) \\ f(\mathbf{x}_2) - f(\mathbf{a}) \\ \vdots \\ f(\mathbf{x}_k) - f(\mathbf{a}) \end{bmatrix} = \begin{bmatrix} (\mathbf{x}_1 - \mathbf{a})^T \\ (\mathbf{x}_2 - \mathbf{a})^T \\ \vdots \\ (\mathbf{x}_k - \mathbf{a})^T \end{bmatrix} \cdot \nabla\mathbf{f}(\mathbf{a}). \tag{5.5}$$

The optimal gradient $\nabla \mathbf{f}(\mathbf{a})^*$ can be derived by solving the least square problem

$$\nabla \mathbf{f}(\mathbf{a})^* = \underset{\nabla \mathbf{f}(\mathbf{a})}{\arg\min} \|\mathbf{F} - \mathbf{X} \cdot \nabla \mathbf{f}(\mathbf{a})\|_2^2, \tag{5.6}$$

which has a closed form solution

$$\nabla \mathbf{f}(\mathbf{a})^* = (\mathbf{X}^T\mathbf{X})^\dagger \mathbf{X}^T\mathbf{F}. \tag{5.7}$$

For each pixel, we use 4 nearby pixels ($k = 4$) for hyperplane fitting in (5.5). The neighbouring pixels are selected as the 2 closest neighbours in terms of Euclidean distance in horizontal and vertical orientations, respectively. This choice is made in order to avoid the ill-conditioning that may result if the pixels picked were all aligned on the same line.

After computing the structure tensor in (5.1) using the estimated gradients, we apply eigen-decomposition on the 2×2 matrix $\mathbf{H}_\mathrm{B}$:

$$\mathbf{H}_\mathrm{B} = \begin{bmatrix} \mathbf{e}_1 & \mathbf{e}_2 \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} \mathbf{e}_1{}^T \\ \mathbf{e}_2{}^T \end{bmatrix} \tag{5.8}$$

$$= \lambda_1 \mathbf{e}_1 \mathbf{e}_1{}^T + \lambda_2 \mathbf{e}_2 \mathbf{e}_2{}^T. \tag{5.9}$$

The resulting eigenvectors $\mathbf{e}^1$ and $\mathbf{e}^2$ along with their corresponding eigenvalues λ_1 and λ_2, where $\lambda_1 < \lambda_2$, summarize the gradient distribution within block B. The eigenvector $\mathbf{e}_2$ represents the direction maximally aligned with the gradient, while the orthogonal direction $\mathbf{e}_1$ roughly represents the edge direction.

5.4.2 Structure Tensor Estimation

Leveraging the local redundancy among neighbouring pixels in the sub-aperture image, we can estimate the local gradient, and therefore the structure tensor, of each input block I using the information from its decoded neighbouring blocks $\{\mathrm{B}_1, \mathrm{B}_2, \cdots\}$. In our algorithm, the estimate of the structure tensor $\mathbf{H}_\mathrm{I}$ of block I is calculated as the weighted average of the structure tensors from its neighbouring blocks:

$$\mathbf{H}_\mathrm{I} = \frac{1}{c} \Sigma_i w_i \cdot (\frac{1}{n_i} \mathbf{H}_{\mathrm{B}_i}), \tag{5.10}$$

where n_i denotes the number of available pixels in reference block B_i, w_i is the weight associated to block B_i, and c is the normalization constant $c = \Sigma_i w_i$.

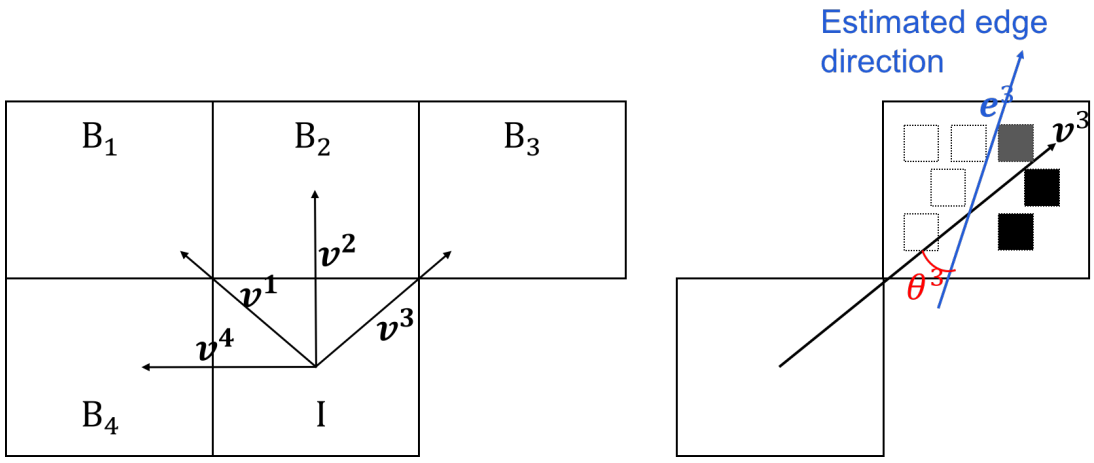


FIGURE 5.7: 4 decoded reference blocks and the vectors indication their relative locations to the input block I and edge direction estimated

The weight value w_i is a function of the edge orientation in B_i and the reference block's relative location from the input block I. This design is based on the observation that edges in natural images are continuous contours, and can be approximated locally with straight lines. In other words, the orientation of edges is mostly consistent locally. Therefore, if the edge direction calculated on one reference block B_i is consistent with its relative location from I, *e.g.*, if the reference block in the top-left corner has edge orientation from top-left to bottom-right, the input block to-be coded is more likely to have the same edge direction. In estimating the gradient in the input block, blocks with consistent edge orientation will be assigned larger weights. In our algorithm, we consider up to 4 decoded neighbouring blocks, *i.e.*, blocks on the top-left corner, top, top-right corner, and left, as the references for structure tensor estimation and the following intra-prediction. The unit vectors $\mathbf{v}^i$ from I to each of the B_i reference blocks are $[\frac{-1}{\sqrt{2}}, \frac{-1}{\sqrt{2}}]^T$, $[-1, 0]^T$, $[\frac{-1}{\sqrt{2}}, \frac{1}{\sqrt{2}}]^T$, $[0, -1]^T$, respectively. Denote $\mathbf{e}^i$ to be the edge direction calculated with the structure tensor in block B_i, and θ_i to be the angle between $\mathbf{v}^i$ and $\mathbf{e}^i$, calculated as

$$\theta_i = \arccos\left(\frac{\mathbf{v}^i \cdot \mathbf{e}^i}{\|\mathbf{v}^i\|_2}\right) \in [0, \pi], \tag{5.11}$$

depicted in Fig. 5.7. The weight w_i in (5.10) is defined as

$$w_i = \begin{cases} \exp(-\frac{\theta_i}{\delta}) & \text{if } \theta < \frac{\pi}{2} \\ \exp(-\frac{\pi - \theta_i}{\delta}) & \text{if } \theta > \frac{\pi}{2} \end{cases}, \tag{5.12}$$

where δ is an adjustable parameter. In (5.12), larger weights are assigned to the blocks

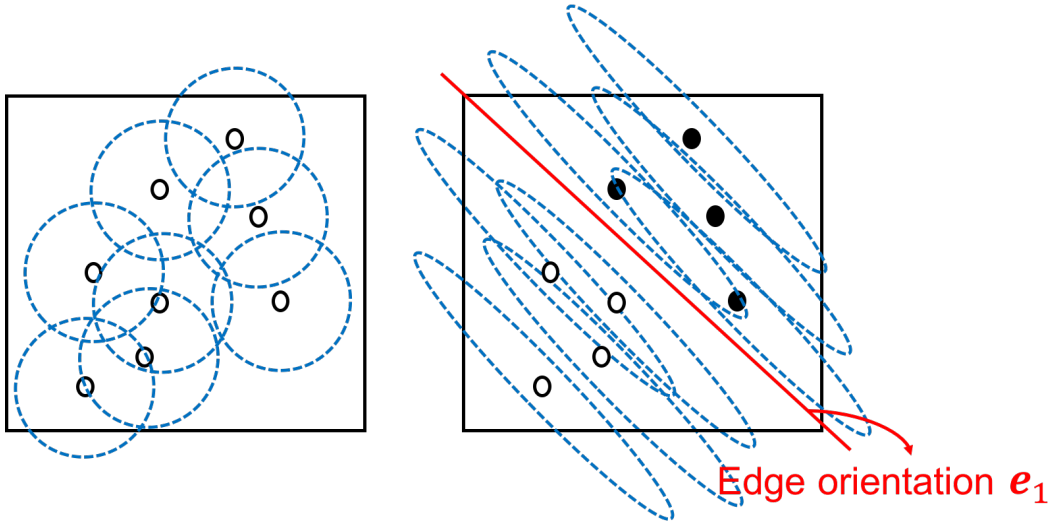


FIGURE 5.8: Illustration of kernel size and shape in the smooth block (left) and the block with strong edge (right)

with smaller θ_i, *i.e.*, the edge directions $\mathbf{e}^i$ and $\mathbf{v}^i$ are nearly consistent. The structure tensor estimation described in (5.10) is done only on the G channel. For R and B channels, the structure tensors are calculated based the reconstructed G channel pixels in the same block, since there exists high correlation between gradients in R, G, and B channels.

5.4.3 Data-adaptive Kernel Regression

For intra-prediction, we apply the same adaptive kernel regression on each pixel in a given block, similar to what was done in [35]. In the case of zero-order estimation, the prediction of the pixel intensity at $\mathbf{x}$ can be calculated by taking the weighted average of its neighbouring pixels $\mathbf{x}_i$, written as

$$\tilde{f}(\mathbf{x}) = \frac{1}{z_{\mathbf{x}}} \Sigma_{\mathbf{x}_i \in \mathcal{R}_{\mathbf{x}}} (K(\mathbf{x}_i - \mathbf{x}) \cdot f(\mathbf{x}_i)), \tag{5.13}$$

where $\mathcal{R}_{\mathbf{x}}$ denotes the set of neighbouring pixels of $\mathbf{x}$ in the decoded reference blocks, and $z_{\mathbf{x}}$ is the normalization constant $z_{\mathbf{x}} = \Sigma_{\mathbf{x}_i \in \mathcal{R}_{\mathbf{x}}} K(\mathbf{x}_i - \mathbf{x})$. A common choice of the kernel function $K(\cdot)$ is the Gaussian kernel

$$K(\mathbf{x}_i - \mathbf{x}) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(\mathbf{x}_i - \mathbf{x})^T (\mathbf{x}_i - \mathbf{x})}{2\sigma^2}\right), \tag{5.14}$$

which gives higher weights to nearby pixels than to pixels that are far away. However, such kernel selection assumes isotropy in image characteristics, ignoring local features such as edges. For data-adaptive kernel regression used in this work, the Gaussian kernel is adapted

based on the edge orientation and strength derived from structure tensor as

$$K(\mathbf{x}_i - \mathbf{x}) = \frac{\sqrt{\det(\mathbf{C})}}{2\pi\sigma^2} \exp(-\frac{(\mathbf{x}_i - \mathbf{x})^T \mathbf{C} (\mathbf{x}_i - \mathbf{x})}{2\sigma^2}), \tag{5.15}$$

where $\mathbf{C}$ is the local gradient covariance based on the eigenvectors and eigenvalues of structure tensor defined in (5.9), and can be decomposed as

$$\mathbf{C} = \phi \cdot \begin{bmatrix} \mathbf{e}_1 & \mathbf{e}_2 \end{bmatrix} \Lambda \begin{bmatrix} \mathbf{e}_1{}^T \\ \mathbf{e}_2{}^T \end{bmatrix} \tag{5.16}$$

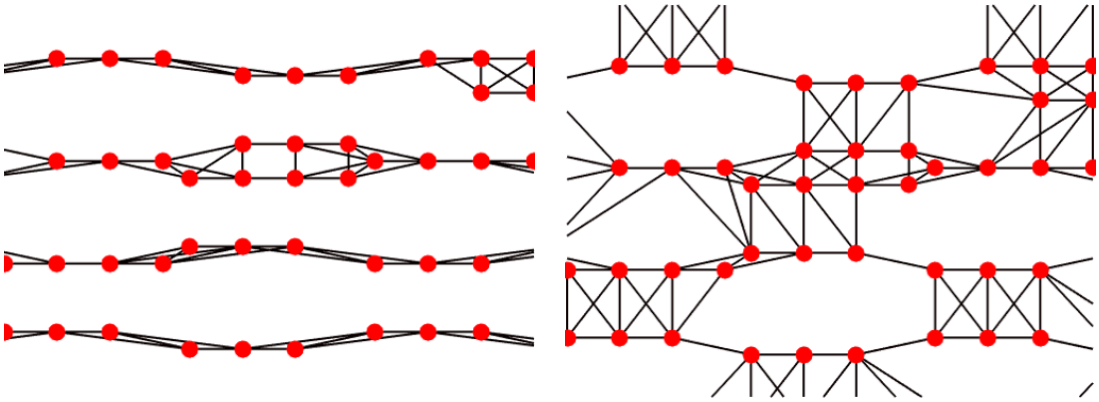
$$\Lambda = \begin{bmatrix} \frac{1}{\epsilon} & 0 \\ 0 & \epsilon \end{bmatrix} \quad , \epsilon = \frac{\lambda_2 + p_2}{\lambda_1 + p_2} \tag{5.17}$$

$$\phi = \frac{1}{n} \sqrt{\lambda_1 \lambda_2 + p_1}, \tag{5.18}$$

where $\begin{bmatrix} \mathbf{e}_1 & \mathbf{e}_2 \end{bmatrix}$ rotates the coordinates of Gaussian kernel along the edge direction and dominant gradient. ϵ is the ratio of two eigenvalues, representing the relative strength of gradient in $\mathbf{e}_2$ from the perpendicular direction $\mathbf{e}_1$. The kernel will be elongated for blocks of strong edges, where $\lambda_2 \gg \lambda_1$, and will be near-circular for smooth blocks ($\lambda_2 \approx \lambda_1 \approx 0$), as illustrated in Fig. 5.8. ϕ determines the scaling of the kernel size, where n is the number of available pixels in the block. p_1 and p_2 are two positive scalars used to ensure numerical stability.

5.5 Proposed Transform Coding

As described in Section 5.3, the pixels rearranged onto the sub-aperture images before demosaicking are distributed sparsely, which motivates the usage of graph-based transform in exploring data redundancies. In this section, we discuss the graph construction, on which the graph based transform will be applied. Each node on the graph represents one pixel and each link connects distinct pixels within the same color component. Note that the graphs for different blocks are constructed independently. For graph construction, we will discuss two algorithms: 1) a simple heuristic based on geometric distance between pixels and 2) graph learning from training data based on GMRF modeling.



(a) (b)

FIGURE 5.9: A part of graph constructed for irregularly placed R components. In (a), the one using 4 nearest neighbor method is shown. In (b), each pixel is connected to 2 neighbors in horizontal and vertical orientations respectively

5.5.1 Graph Construction base on Geometric Distance

In each sub-aperture image, similar to natural images, large local redundancies exist among pixels that are close in distance. Hence, the most straightforward approach in exploiting the pair-wise correlation is to connect each pixel with its k nearest neighbors in terms of Euclidean distance. For complexity reduction in the graph-based lifting transform, where the computation for each node depends on its connected neighbours, we consider mainly sparse graphs, *i.e.,* small k. However, the graph connection based on k-nearest neighbour with small k can be highly sensitive to the pixel arrangement. For example, in the cropped sub-aperture image shown in Fig. 5.9, R components are mostly aligned horizontally. The resulting graph, based on k-nearest neighbor ($k = 4$), thus consists of mostly horizontal links as shown in Fig. 5.9(a), and is unable to capture local similarity in regions with vertical features, *e.g.,* vertical edges.

In order to exploit similarity along different orientations, yet still keep connection sparse, we instead connect each pixel to an equal number of neighbours in horizontal and vertical directions, as shown in Fig. 5.9(b). The weight $w_{i,j}$ on the link connecting node v_i and v_j is defined as

$$w_{i,j} = \exp\left(-\frac{\text{dist}(i,j)^2}{\sigma^2}\right), \tag{5.19}$$

with the assumption that pixels that are closer in distance are more likely to be similar in pixel intensities. The function $\text{dist}(i,j)$ measures the Euclidean distance between node v_i and v_j. This approach has been adopted in our previous publication in [8].

5.5.2 Graph Learning based on Statistics Modeling

In addition to the simple heuristic using geometric distance, we also consider optimized graph construction based on the statistics of residual blocks in LF images. Since the information about the graph, *i.e.*, connection, link weights, and self loops, can be uniquely and fully represented using a graph Laplacian matrix $\mathbf{L}_\mathrm{G}$, the problem of finding the optimal graph can be seen as being equivalent to the optimization of $\mathbf{L}_\mathrm{G}$.

In the literature, many works have studied the problem of finding the optimal graph Laplacian matrix based on observations in different applications. In [24, 33, 34, 60, 78], a statistical assumption has been made on data, where each observation $\mathbf{f} \in \mathbb{R}^N$ is modeled as a realization of a GMRF, *i.e.* $\mathbf{f} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma} = \mathbf{Q}^{-1})$ and the problem for learning the graph Laplacian matrix $\mathbf{L}_\mathrm{G}$ for this type of data modeling boils down to the maximum likelihood estimation of the precision matrix $\mathbf{Q}$, which defines the partial correlation between pair-wise variables in $\mathbf{f}$. In [33, 34], Egilmez et al. proposed an algorithm specifically for finding a precision matrix $\mathbf{Q}$ with graph Laplacian structure, *i.e.*

$$\begin{cases} Q_{i,j} < 0 & \text{if } A_{i,j} > 0 \\ Q_{i,j} = 0 & \text{if } A_{i,j} = 0 \end{cases}, \tag{5.20}$$

with additional constraints on the graph connectivity. For a graph of N nodes and M links, given p *i.i.d* observations $\{\mathbf{f}_1, \mathbf{f}_2, \cdots \mathbf{f}_p\}$ of zero mean GMRF, the precision matrix can be found by solving the maximum likelihood (ML) problem:

$$\begin{aligned} &\arg\max_{\mathbf{Q}} \prod_{i=1:p} p(\mathbf{f}_i|\mathbf{Q}, \boldsymbol{\mu}_i = \mathbf{0}) \\ &= \arg\min_{\mathbf{w},\mathbf{h}} \quad \log\det(\mathbf{Q}) - \mathrm{Tr}(\mathbf{QS}) \\ &\text{subject to } \mathbf{Q} = \mathbf{B}\,\mathrm{diag}(\mathbf{w})\mathbf{B}^T + \mathrm{diag}(\mathbf{h}), \end{aligned} \tag{5.21}$$

where $\mathbf{S}$ is the $N \times N$ sample covariance matrix, and $\mathbf{B}$ is the $N \times M$ incidence matrix, specifying the connectivity between node pairs. The $M \times 1$ vector $\mathbf{w}$ contains the weights associated to each link, and the $N \times 1$ vector $\mathbf{h}$ contains the self loops associated to each node.

The aforementioned Laplacian learning algorithms are all based on the assumption that each N dimensional observation $\mathbf{f}_i$ contains a complete set of variables, *i.e.* a pixel value is available at each index in $\mathbf{f}^i$. In a block of a non-demosaicked sub-aperture image, however, pixels are distributed sparsely and each index in $\mathbf{f}_i$ contains at most one color component out of R, G, and B. Therefore some modifications are required for using the existing learning

algorithm. Without loss of generality, we can write the observed block as a column vector with its elements ordered as

$$\mathbf{f}_i = \begin{bmatrix} \mathbf{f}_{iO} \\ \mathbf{f}_{i\mathcal{M}} \end{bmatrix}, \tag{5.22}$$

where $\mathbf{f}_{iO}$ is a r dimensional vector containing the observed pixel intensities, and $\mathbf{f}_{i\mathcal{M}}$ of dimensional $N - r$ is the missing pixels. r and the indices specified by O and $\mathcal{M}$ are both variables that are block dependent. In order to optimize the graph for $\mathbf{f}_{iO}$, we assume $\mathbf{f}_{iO}$ to be a sub-sampled version of $\mathbf{f}_i$, which is modeled as a GMRF. In the statistics literature, many methods are proposed for estimating the inverse covariance matrix in a GMRF model based on observations with missing variables [41, 49, 71]. In the work by Kolar and Xing [41], a simple *plug-in* algorithm is proposed. The method consists of two steps: First, the sample covariance matrix is estimated using the incomplete observations. Define $\mathbf{r}_i$ to be the indicator vector of $\mathbf{f}_i$, where element

$$\begin{cases} r_{ia} = 1 & \text{if } f_{ia} \text{ is available} \\ r_{ia} = 0 & \text{otherwise} \end{cases}, \tag{5.23}$$

the estimated sample covariance matrix $\tilde{\mathbf{S}}$ is calculated as

$$\tilde{S}_{a,b} = \frac{\sum_{i=1:p} r_{ia} \cdot r_{ib} \cdot (f_{ia} - \mu_{ia})(f_{ib} - \mu_{ib})}{\sum_{i=1:p} r_{ia} \cdot r_{ib}}. \tag{5.24}$$

Readers are referred to [41] for more details on the theoretical justification of the estimation. In the second step, $\tilde{\mathbf{S}}$ will be *plugged* into the objective function of the maximum likelihood estimation of the precision matrix $\mathbf{Q}$. In this work, we will optimize $\mathbf{Q}$ based on (5.21) with $\mathbf{S}$ replaced by $\tilde{\mathbf{S}}$.

We can write the covariance and precision matrices of the estimated GMRF model in block form as

$$\begin{aligned} \mathbf{\Sigma} = \mathbf{Q}^{-1} &= \begin{bmatrix} \mathbf{\Sigma}_{O,O} & \mathbf{\Sigma}_{O,\mathcal{M}} \\ \mathbf{\Sigma}_{\mathcal{M},O} & \mathbf{\Sigma}_{\mathcal{M},\mathcal{M}} \end{bmatrix}, \\ \mathbf{Q} &= \begin{bmatrix} \mathbf{Q}_{O,O} & \mathbf{Q}_{O,\mathcal{M}} \\ \mathbf{Q}_{\mathcal{M},O} & \mathbf{Q}_{\mathcal{M},\mathcal{M}} \end{bmatrix}. \end{aligned} \tag{5.25}$$

Once the precision matrix $\mathbf{Q}$, and therefore the graph Laplacian matrix, is derived for the whole block, which includes the positions of both available and missing data, the graph Laplacian $\mathcal{L}$ for $\mathbf{f}_{iO}$ can be calculated by taking the Schur complement of the sub-matrix $\mathbf{\Sigma}_{O,O}$

of $\mathbf{\Sigma}$:

$$\mathcal{L} = \Sigma_{\mathcal{O},\mathcal{O}}^{-1} = \mathbf{Q}_{\mathcal{O},\mathcal{O}} - \mathbf{Q}_{\mathcal{O},\mathcal{M}}\mathbf{Q}_{\mathcal{M},\mathcal{M}}^{-1}\mathbf{Q}_{\mathcal{M},\mathcal{O}}. \tag{5.26}$$

In our experiment, light field images are divided into two groups: training set and test set. Blocks of intra-predicted residuals from the training data are classified, based on the structure tensor, into 8 directional modes:

$$\mathcal{M} = \{\mathcal{M}_\theta | \quad \theta = \frac{-3\pi}{8}, \frac{-\pi}{4}, \frac{-\pi}{8}, 0, \frac{\pi}{8}, \frac{\pi}{4}, \frac{3\pi}{8}, \frac{\pi}{2}\} \tag{5.27}$$

and one DC mode, $\mathcal{M}_{\text{DC}}$, if there is no dominant edge direction. Given the edge angle γ of a training block derived from its structure tensor, *i.e.*, the angle between the smallest eigenvector and the horizontal axis, and the eigenvalues λ_1 and λ_2 ($\lambda_1 \leq \lambda_2$), the associated class is determined based on

$$\begin{cases} \mathcal{M}_{\text{DC}} & \text{if } \frac{\lambda_2}{\lambda_1} < T \\ \mathcal{M}_\theta & \text{if } \gamma \in [\theta - \frac{\pi}{16}, \theta + \frac{\pi}{16}) \end{cases}. \tag{5.28}$$

Training blocks within a class are assumed to be samples from the same GMRF model. Therefore, in total 9 graph Laplacian matrices $\{\mathbf{L}_{\text{G}i} | i = 1, 2, \cdots 9\}$ are derived using the *plug-in* algorithm for intra-predicted residuals. We consider 8 connected graph for pixels on the hexagonal grid as the connectivity constraint in (5.21). For residual blocks in test set, blocks will be classified into 9 modes based on the edge angle derived from the estimated structure tensor, whose calculation is described in Section 5.4. The graph construction of pixels in each block is obtained with Schur complement from the Laplacian matrix $\mathbf{L}_{\text{G}i}$ of the corresponding class.

5.5.3 Graph-based Lifting Transform

Due to the large data size of light field images, we apply the localized graph based lifting transform with the Max Cut bipartition and re-connection (*reconn-MaxCut*) described in Chapter 4, which was shown to have comparable performance but lower complexity than *reconn-GMRF*. As described in Chapter 3, for the context of compression, the main objective in designing the lifting scheme is to compress most of the data energy in the low frequency band, *i.e.*, in the update set $\mathcal{U}$, or correspondingly reduce the energy of prediction residuals $\mathbf{d} = \mathbf{f}_{\mathcal{P}} - \mathbf{P}\mathbf{f}_{\mathcal{U}}$. For a signal $\mathbf{f}$ modeled as GMRF $\mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$, the minimum mean square error

estimator (also MAP estimator) of signal value f_i on node $v_i \in \mathcal{P}$ is expressed as

$$\begin{aligned}\hat{f}_i &= \mu_i - \sum_j \frac{Q_{i,j}}{Q_{i,i}}(f_j - \mu_j) \\ &= -\sum_j \frac{Q_{i,j}}{Q_{i,i}} f_j + (\mu_i + \sum_j \frac{Q_{i,j}}{Q_{i,i}}\mu_j).\end{aligned} \tag{5.29}$$

The matrix form of (5.29) for the case of zero mean is the same as in (3.15). If the precision matrix estimated satisfies the graph Laplacian structure, as described in (5.20), and if $\boldsymbol{\mu}_i \approx \boldsymbol{\mu}_j = \mu$, which is usually true for neighboring nodes of high correlation, the above equation can be simplified as

$$\begin{aligned}\hat{f}_i &= -\sum_j \frac{Q_{i,j}}{Q_{i,i}} f_j + \mu(1 + \sum_j \frac{Q_{i,j}}{Q_{i,i}}) \\ &= \frac{1}{D_{i,i} + H_{i,i}} \sum_j A_{i,j} f_j + \frac{H_{i,i}}{D_{i,i} + H_{i,i}}\mu.\end{aligned} \tag{5.30}$$

The estimator is simply a weighted average of the signal values f_j on the neighbouring nodes v_j and the associated mean μ of v_i. The self loop weight $H_{i,i}$ can be interpreted as a measurement of similarity to the mean μ. In our experiment, we assume $\mu = 0$ since the sub-aperture images after intra-prediction usually have average value close to 0. Note that if the graph is bipartite, *i.e.* links only exist between nodes from opposite sets, the prediction operation in (5.30) is equivalent to the low complexity CDF53 filterbank

$$\hat{f}_i = \frac{1}{D_{i,i}} \sum_{v_j \in \mathcal{U}} A_{i,j} f_j \tag{5.31}$$

described in Section 2.4, when self loop weight $H_{i,i} = 0$. In this work, we consider a more general CDF5/3 filterbank with non-zero self loops:

$$\hat{f}_i = \frac{1}{D_{i,i} + H_{i,i}} \sum_{v_j \in \mathcal{U}} A_{i,j} f_j + \frac{H_{i,i}}{D_{i,i} + H_{i,i}}\mu. \tag{5.32}$$

For bipartite graphs, the proposed filterbanks are optimal in terms of mean square error for the GMRF model, as shown in (5.30).

For graphs that are not bipartite, we use the re-connection algorithm with sparsification described in Section 3.3, which re-connects each $v_i \in \mathcal{P}$ to be predicted to nodes in $\mathcal{U}$, and apply CDF5/3 predictor in (5.32) on the newly formed bipartite graph. As proven in Appendix A, without the sparsification, the applied predictor is equivalent to the minimum

mean square error estimator (MMSE) for $\mathbf{f}_{\mathcal{P}}$ given $\mathbf{f}_{\mathcal{U}}$, and therefore is optimal in terms of energy compaction.

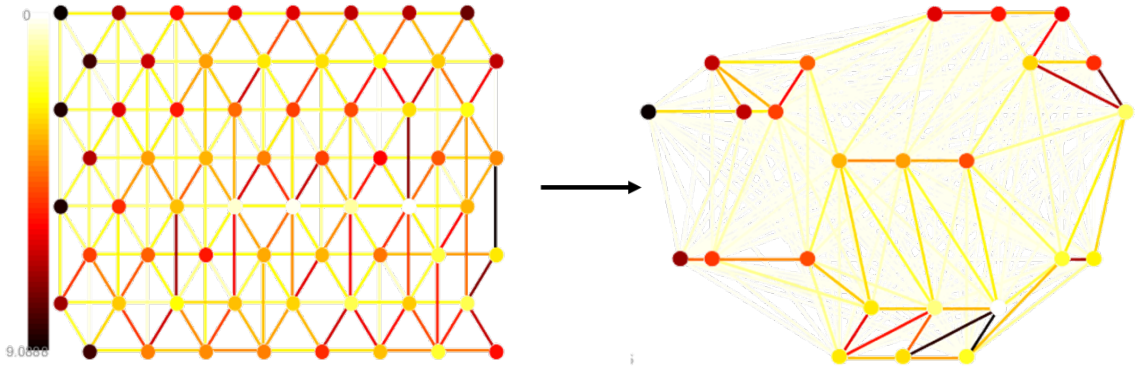
For Max Cut bipartition, we apply the algorithm proposed in [59] using MST algorithm. The update filter is calculated through orthogonalization, and the construction of graphs for lifting transform in levels ≥ 2 is based on Kron reduction. For entropy coding, we apply the Amplitude and Group Partitioning (AGP) algorithm [66]. Note that the graph-based lifting transform applied is the same as in Section 4.5 (*reconn-MaxCut*), but with graphs designed in this chapter.

5.6 Experiments

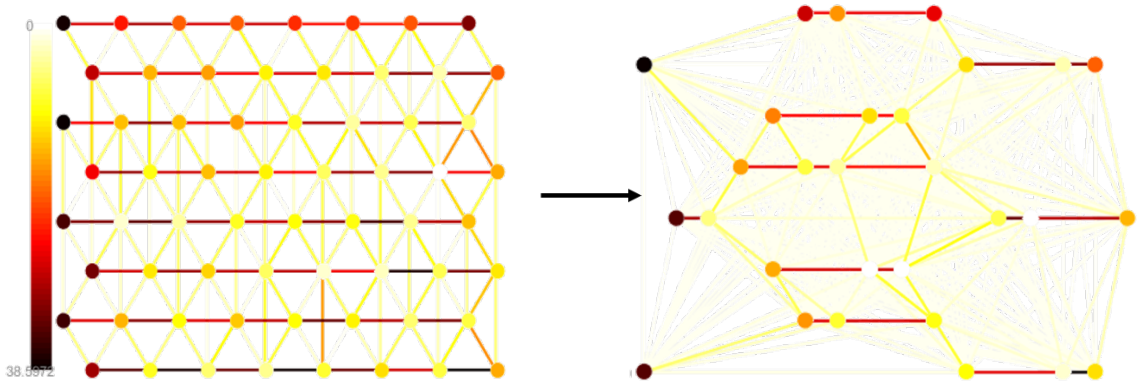
5.6.1 Experimental Setting

For archival purpose, one should assess the quality of reconstructed lenselet image in the original RGB pattern. However, current state of the art schemes using HEVC discard underexposed pixels at the boundary of macro-pixels during the conversion to sub-aperture image array, so it is difficult to recover the lenselet image on the decoder side. Hence, for evaluation, we compare performances on the reconstructed full color sub-aperture images. The full color sub-aperture image before compression, generated from the raw lenselet data using the demosaicking and calibration pipeline described in [17, 47], is taken as the ground truth. As a baseline, we consider the HEVC (HM 16.9) encoding of sub-aperture images in original 4:4:4 RGB (*intraHEVC-RGB444*) and 4:2:0 YUV (*intraHEVC-YUV420*) formats. Since we consider a coding scheme that allows efficient random access, the configuration used in HEVC is All-Intra coding, which allows intra-prediction but no inter-prediction. For fair comparison, the post filters in HEVC, including deblocking filter and SAO are disabled. In our proposed scheme, the same demosaicking and calibration will be applied on the decoder side to the reconstructed lenselet image, in order to generate the reconstructed sub-aperture images for evaluation. Images from the proposed and baseline schemes are compared in RGB format without sub-sampling. For *intraHEVC-YUV420*, the reconstructed sub-aperture images are translated back to 4:4:4 RGB format before evaluation. The up-sampling for U and V components is based on nearest neighbor interpolation.

In our method, each sub-aperture image is divided into non-overlapping 8×8 blocks. The light field images we consider in the experiments are obtained from the EPFL database [46]. We consider three different scenarios for the graph based coding scheme:



(a) $\theta = \frac{3\pi}{8}$



(b) $\theta = 0$ (horizontal)

FIGURE 5.10: Graph structure optimized for classes corresponding to intra-prediction angle (1) $\theta = \frac{3\pi}{8}$ (nearly vertical) and (2) $\theta = 0$ (horizontal). The link color indicates the associated weight and the node color indicates the associated self loop weight (darker: larger weight)

1. *DGLT*: geometric *D*istance based graph construction (Section 5.5.1) in *G*raph *L*ifting *T*ransform without intra-prediction, which has been use in [8].
2. *intraDGLT*: geometric *D*istance based graph construction in *G*raph *L*ifting *T*ransform with the proposed *intra*-prediction (Section 5.4)
3. *intraLGLT*: graph *L*earning based graph construction (Section 5.5.2) in *G*raph *L*ifting *T*ransform with the proposed *intra*-prediction

For graph learning in scenario 3, we select 4 sub-aperture images from each of the following LF images from EPFL dataset: *Ankylosaurus_&_Stegosaurus*, *Ceiling_Light*, *ISO_Chart_16*, *Perforated_Metal_1*, *Sophie_&_Vincent_3*, and *Yan_&_Krios_standing* to form the training set. The parameter δ in (5.12) is set as 0.9, and the parameters σ, p_1, p_2 in (5.15) and (5.17) for data-adaptive Gaussian kernels are chosen to be 1.6, 0.001 and 0.001. The threshold value

T in (5.28) for training block classification is set as 1.5. We apply 2 level lifting transform for transform coding for pixels in each block.

In Fig. 5.10, two illustrative examples are shown for weighted graph optimized for set $\mathcal{M}_{\theta=\frac{3\pi}{8}}$ and $\mathcal{M}_{\theta=0}$. The color on links and nodes indicates the associated link weight and self loop weights. It can be seen that the orientation of links with strong weights (links with darker color) well matches the intra-prediction direction, which is also the edge direction of the block. The nodes with large self loop weights concentrate around the boundaries close to the reference blocks, which have better prediction. Therefore, the associated residuals on those pixels tend to have lower variance. The observation matches our interpretation in Section 5.5 that self loop weights model the similarity of observed values on a node to their expected value.

The raw data in [46] are captured with Lytro Illum camera [64]. Each test image is of size 5368×7728. In the baseline scheme, the raw data will be converted into 15×15 full color sub-aperture images. Each sub-aperture image is of size 434×625. Therefore for each test image, there are a total of $91546875 = 15 \times 15 \times 434 \times 625 \times (1 + \frac{1}{4} + \frac{1}{4})$ pixels that need to be encoded by HEVC when using 4:2:0 YUV format. In our scheme, on the other hand, the compression is applied on the original raw data without demosaicking, and therefore only $41483904 = 5368 \times 7728$ pixels are required, saving more than 55% in input data size.

5.6.2 Results

Fig. 5.11 shows the PSNR comparisons for images *Friends_1*, *Bikes*, *Flowers*, and *Ankylosaurur_&_Diplodocus* from EPFL dataset. The considered QP values range from 4 to 36. For applications such as archiving and instant storage on cameras, images are typically stored in very high quality. Therefore, in the evaluation, we consider mainly the high bitrate region. It can be seen that for higher bit rates (bpp > 2), graph based coding schemes significantly outperform the conventional approach using HEVC. This is because as the bit rate increases, indicating a smaller quantization step, more high frequency components will be kept in the transform domain after quantization. Using conventional approach to encode will incur a large cost to encode all the coefficients, while in the proposed method, only around half as many coefficients are encoded. For low bit rate region, since most of the high frequency coefficients will be discarded after quantization, the number of coefficients that need to be encoded in HEVC are much smaller than the total number of coefficients. Therefore, the advantage of having a smaller amount of data is not as significant in this case. With the proposed intra-prediction, correlations between neighbouring blocks are utilized to reduce redundancies before compression. As a result, performance is improved by about 5dB over

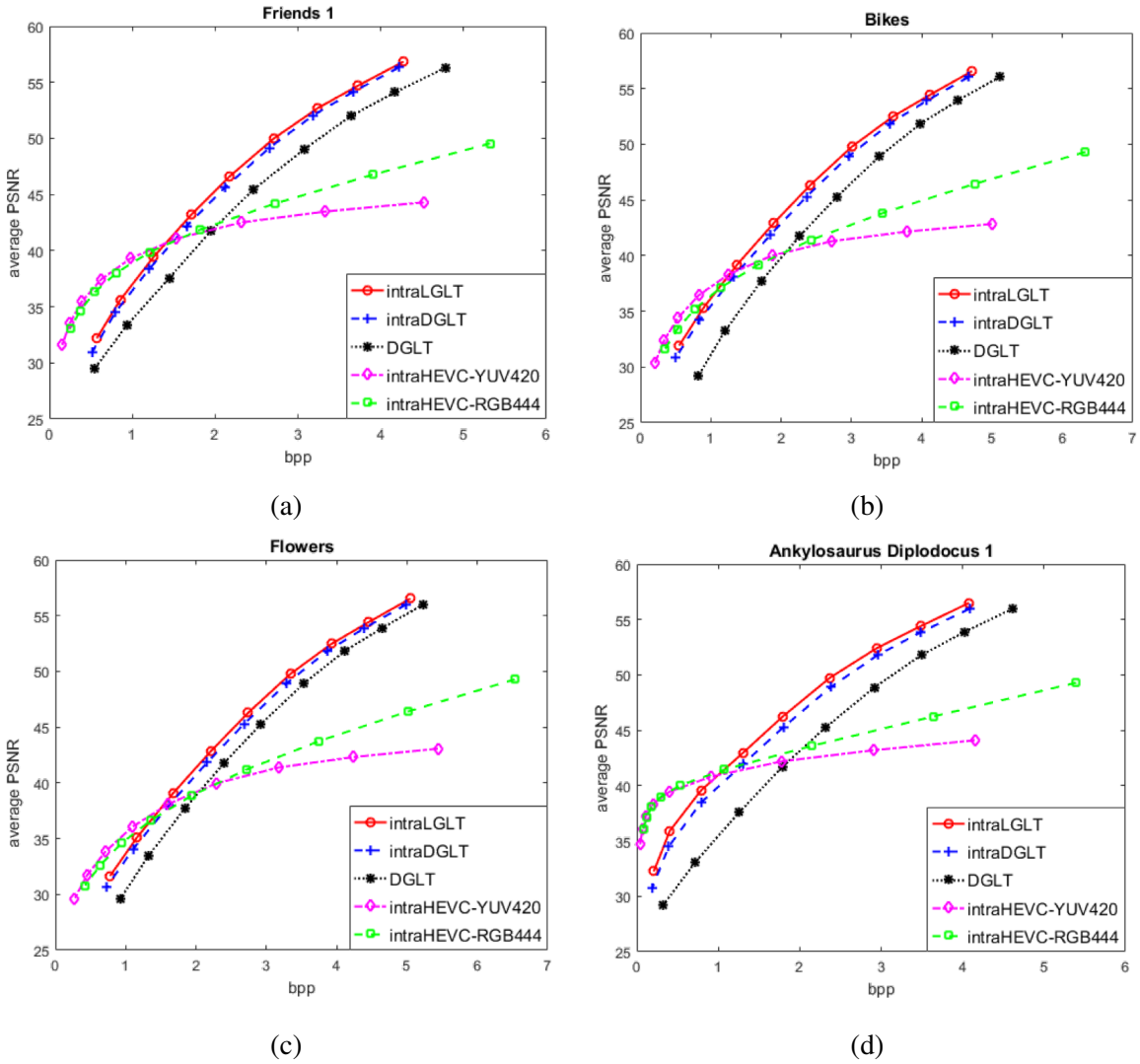


FIGURE 5.11: Average PSNR over R, G, and B components for test images (a) Friends1 (b) Bikes, (c) Flowers, and (d) Ankylosaurur & Diplodocus

our results without intra-prediction. Moreover, with graph learning, directional local characteristics are exploited in selecting graph link weights, which provides more accurate modeling of the similarity between pixels as compared to the link weight selection based solely on Euclidean distance. In the results, using graph learning leads to around 0.5dB gain over graph construction based on distance. For baseline method using 4:2:0 YUV format, PSNR will mostly saturate near 43dB, which is mainly caused by the color conversion. During the conversion, some details are lost when rounding floating point values, and resolution is reduced as down-sampling is performed. In the proposed method, since the compression is performed on the raw RGB data, the performance will not be degraded by distortion of color conversion and downsampling.

5.7 Summary

In this chapter, we have described a novel coding scheme for light field images based on graph based lifting transform. The scheme is able to encode the original raw data without introducing redundancies from demosaicking and calibration and distortion from color conversion and downsampling. Moreover, we proposed an intra-prediction and graph learning algorithm for pixels in sub-aperture images that are sparsely distributed. The pixels are then connected as graphs and encoded with low complexity graph based lifting transform. The coding results at high bitrates using the proposed method outperform the widely applied HEVC based approach.

Chapter 6

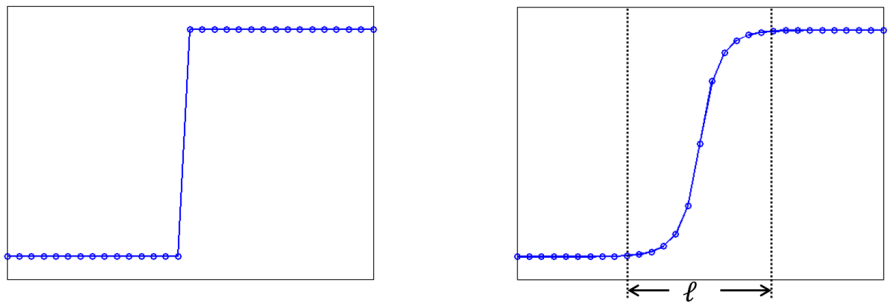
Edge Adaptive Graph-Based Transforms

Step/Ramp Edge Models for Video Compression

6.1 Introduction

In this chapter, we study edge models for edge adaptive graph-based transforms (EA-GBTs) in video compression. In particular, we consider step and ramp edge models to design graphs used for defining transforms, and compare their performance on coding intra and inter predicted residual blocks. EA-GBTs are special types of Graph Fourier Transforms (GFTs), described in Section 2.2, where the graphs are constructed based on the edge information. The first example of an EA-GBT was proposed in [28] for depth-map coding. By adjusting the graph weights based on the edge information, *i.e.*, smaller weights are assigned for links across edges, an EA-GBT was defined for each block. Such adaptation provides better compression, since the resulting transforms avoid filtering across the discontinuities which could have created high frequency coefficients. Recently, Hu *et.al* [76] proposed an EA-GBT to capture both strong and weak edges for depth-map coding. In our previous work [32, 80], we show that EA-GBTs can also be used to improve inter and intra predicted residual coding performance over the DCT. However, prior work only considers the step edge model for the EA-GBT design by implicitly assuming that all edges can be represented with ideal step functions, as shown in Fig. 6.1(a). As pointed out in [61], this is usually not the case, especially for high resolution natural images where edges are mostly ramps, as illustrated in Fig. 6.1(b). In this work, we show that such property also holds for intra predicted residual blocks generated in video coding and that better compression can be achieved by using the proposed ramp-edge model. In contrast, for inter predicted coding, we obtain better compression performance using the step edge model. Most of this work was published in [81].

In order to optimize EA-GBTs whose associated graphs are designed based on a ramp edge model, we propose a new probabilistic model for ramp edges and estimate the parameters



(a) (b)

FIGURE 6.1: (a) The step function and (b) ramp function for edge modeling

by training on residual block data. The optimized model parameters are used to design EA-GBTs by adjusting the graphs' weights according to the detected ramp edges within blocks. For compression, we employ a block-adaptive coding scheme, i.e., different EA-GBTs are designed for blocks with different edge positions, which requires sending edge information to the decoder for reconstruction. For efficient edge coding, we propose a new method for ramp edges, called *arithmetic ramp edge coding (AREC)*, extending AEC proposed in [19]. In our experiments, we compare EA-GBT with step and ramp models for inter and intra predicted residual videos. The results show that the proposed ramp edge model performs better than step edge model for intra predicted residuals, and both models outperform DCT-based encoding.

6.2 Edge Models for Residual Signals

6.2.1 Ramp-Edge Model

The derivation of the optimal graph for EA-GBT in [76] is based on the assumption that 1-D signals with an edge can be modeled as auto-regressive (AR) processes with a step transition. However, edges with step transition rarely exist in natural images. The edges mostly have a smoother (ramp-like) transition as a result of image capture and digitization, which is particularly true for high resolution images. Therefore, in our work, we model the 1-D signals $[x_1, x_2, \cdots x_N]^t$ with an edge in a block as AR processes with a sloped transition from pixels x_i to $x_{i+\ell}$, where ℓ denotes the ramp width, immersed in an independent and identically distributed (i.i.d.) Gaussian noise $e_k \sim \mathcal{N}(0, {\sigma_e}^2)$. The model is written as

$$
\begin{aligned}
x_1 &= \rho(y+\epsilon) + e_1 \\
x_2 &= \rho x_1 + e_2 \\
&\cdots \\
x_i &= \rho x_{i-1} + e_i \\
x_{i+1} &= \rho x_i + e_{i+1} + t_1 \\
&\cdots \\
x_{i+\ell} &= \rho x_{i+(\ell-1)} + e_{i+\ell} + t_\ell \\
x_{i+(\ell+1)} &= \rho x_{i+\ell} + e_{i+(\ell+1)} \\
&\cdots \\
x_N &= \rho x_{N-1} + e_N,
\end{aligned}
\tag{6.1}
$$

where y is the reference sample in the neighboring block, and $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$ is the associated distortion. A sloped transition is denoted as an *i.i.d.* random gap $t_p \sim \mathcal{N}(m, {\sigma_t}^2)$. Note that the model is equivalent to the model in [76] for $\ell = 1$ and $y = 0$, and thus can be seen as a generalization of that work. We can express (6.1) in matrix form as $\mathbf{Fx} = \mathbf{b} + \mathbf{y}$, where $\mathbf{y} = [\rho y, 0, 0, \cdots 0]^t$ and

$$
\mathbf{F} = \begin{bmatrix}
1 & 0 & 0 & \cdots & \cdots & 0 \\
-\rho & 1 & 0 & \ddots & \ddots & \vdots \\
0 & -\rho & 1 & \ddots & \ddots & \vdots \\
\vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\
\vdots & \ddots & \ddots & -\rho & 1 & 0 \\
0 & \cdots & \cdots & 0 & -\rho & 1
\end{bmatrix}, \mathbf{b} = \begin{bmatrix}
e_1 + \rho\epsilon \\ e_2 \\ e_3 \\ \vdots \\ \vdots \\ \vdots \\ e_N
\end{bmatrix} + \begin{bmatrix}
0 \\ \vdots \\ t_1 \\ \vdots \\ t_\ell \\ \vdots \\ 0
\end{bmatrix}. \tag{6.2}
$$

Since $\mathbf{F}$ is invertible, the signal can be written as $\mathbf{x} = \mathbf{F}^{-1}\mathbf{b} + \mathbf{F}^{-1}\mathbf{y}$, where $\mathbf{F}^{-1}\mathbf{y}$ is the optimal prediction for $\mathbf{x}$, and $\mathbf{r} = \mathbf{F}^{-1}\mathbf{b}$ is the resulting residual signal. The optimal transform for compressing $\mathbf{r}$ can be derived by computing the KLT. In order to do it, we first compute the

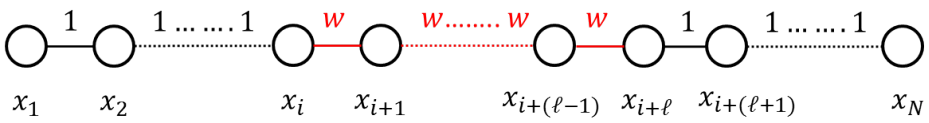


FIGURE 6.2: 1-D line graph with weak link weights w for the ramp spanned from x_i to $x_{i+\ell}$

covariance matrix of $\mathbf{r}$ as

$$\mathbf{C} = {\sigma_e}^2 \mathbf{F}^{-1} \begin{bmatrix} 1+\beta & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & 1 & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 1+\alpha_t & 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 & \ddots & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 & 1+\alpha_t & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & 1 \end{bmatrix} (\mathbf{F}^{-1})^t, \tag{6.3}$$

where $\alpha_t = \frac{{\sigma_t}^2}{{\sigma_e}^2}$ and $\beta = \frac{(\rho\sigma_\epsilon)^2}{{\sigma_e}^2}$. In this chapter, we only consider the case when $\rho = 1$. Then, the precision matrix, defined as $\mathbf{Q} = \mathbf{C}^{-1}$, can be written as

$$\mathbf{Q} = \frac{1}{{\sigma_e}^2} \cdot \begin{bmatrix} 1+\frac{1}{1+\beta} & -1 & & & & & & & & \\ -1 & 2 & -1 & & & & & & & \\ & \ddots & \ddots & \ddots & & & & & & \\ & & -1 & 2 & -1 & & & & & \\ & & & -1 & 1+\frac{1}{1+\alpha_t} & -\frac{1}{1+\alpha_t} & & & & \\ & & & & -\frac{1}{1+\alpha_t} & \frac{2}{1+\alpha_t} & -\frac{1}{1+\alpha_t} & & & \\ & & & & & \ddots & \ddots & \ddots & & \\ & & & & & -\frac{1}{1+\alpha_t} & 1+\frac{1}{1+\alpha_t} & -1 & & \\ & & & & & & -1 & 2 & -1 & \\ & & & & & & & \ddots & \ddots & \ddots \\ & & & & & & & & -1 & 1 \end{bmatrix}. \tag{6.4}$$

Given the 1-D signal $\mathbf{f} = [f_1, f_2, \cdots f_N]^t$ of length N with the sloped edge between location x_i and $x_{i+\ell}$, we can represent the signal using a line graph shown in Fig.6.2. It can be shown that if we assign the weights w across the ramp to be $\frac{1}{1+\alpha_t} = \frac{1}{1+\sigma_t^2/\sigma_e^2}$, while others are set to 1, the graph Laplacian $\mathbf{L}$ is approximately equivalent to $\mathbf{Q}$ in (6.4) if the distortion of the

reference sample is large. Since precision matrix $\mathbf{Q} \approx \mathbf{L}$ and the covariance matrix $\mathbf{C}$ share the same eigenvector set, the EA-GBT basis $\mathbf{U}$ defines the KLT. In our simulation, we assume the noise variance σ_e^2 is 1. The parameter σ_t^2, can be estimated from the sample variance $\hat{\sigma}_t^2$ of pixel gradients $\{|f_i - f_{i+1}|, |f_{i+1} - f_{i+2}|, \cdots, |f_{i+\ell-1} - f_{i+\ell}|\}$ extracted from the detected ramps.

6.2.2 Experimental Justification of the Edge Models

We justify our proposed model experimentally by learning a graph from the real inter/intra predicted residuals. We employ one of the graph learning methods proposed in [24]. The residual signal $\mathbf{f} \in \mathbb{R}^N$ is first modeled as a Gaussian Markov Random Field (GMRF) defined as follows.

$$p(\mathbf{f}|\mathbf{Q}) = \frac{1}{(2\pi)^{N/2}\det(\mathbf{Q})^{-1/2}} \exp\left(-\frac{1}{2}\mathbf{f}^t\mathbf{Q}\mathbf{f}\right), \tag{6.5}$$

where $\mathbf{Q}$ is the precision matrix to be estimated. Note that the AR model described in the previous subsection is one special case of a parametric GMRF model. The optimal precision matrix in (6.5) is computed by solving the maximum likelihood problem:

$$\mathbf{Q} = \underset{\mathbf{Q}\in\Gamma}{\arg\max}\, \log\det(\mathbf{Q}) - \mathrm{Tr}\,(\mathbf{Q}\mathbf{S}), \tag{6.6}$$

where $\mathbf{S}$ is the sample covariance of residual signal $\mathbf{f}$ and Γ defines the matrix type and graph connectivity constraint for $\mathbf{Q}$. In our case, we constrain the matrix to be a combinatorial Laplacian of a 2-connected line graph. In (6.6), the objective function is derived by taking the natural logarithm of likelihood term in (6.5).

In order to form the training set, we apply *Sobel* detector to identify step edges, and the edge detector proposed in [61] to identify ramp edges of width $\ell = 2$ on unquantized residuals obtained from HEVC intra and inter prediction for 8 video sequences, with block size fixed as 8×8. The training set is composed of row/column residual vectors $[x_1, x_2, \cdots, x_8]^t$ with a *step edge* detected between pixel x_4 and x_5, and a *ramp edge* detected between pixel x_3 and x_5. By solving for the graph Laplacian $\mathbf{L} \approx \mathbf{Q}$ in (6.6) for the training data, which includes residuals where both *step* and *ramp* edges were identified, the resulting model gives an estimate of the *most efficient* representation for the edge structure. The results are shown in Figs. 6.3 and 6.4, where the maximum link weights are normalized to 1. It can be seen that the graph derived for **INTRA** predicted residuals (Fig. 6.4) has a similar structure to the model in (6.1) with $\ell = 2$, which offers some justification for the application of ramp edge models. On the other hand, the step edge model, (6.1) with $\ell = 1$, is better for representing the **INTER** predicted

residuals (Fig. 6.3). This is because edges in inter predicted residuals tend to have sharper transitions due to motion estimation mismatches for some pixels with respect to the reference block.

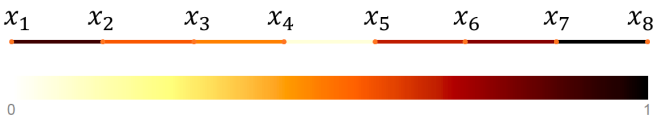


FIGURE 6.3: The optimal line graph for **INTER** predicted residuals

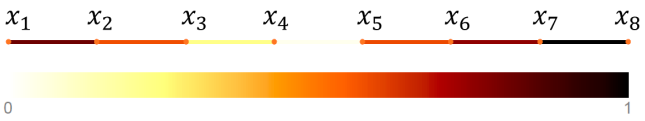


FIGURE 6.4: The optimal line graph for **INTRA** predicted residuals

6.3 Ramp Coding and Graph Construction

6.3.1 Arithmetic Ramp Edge Coding (AREC)

The ramp edge detection for each transform block is based on the work in [61]. Similar to *Canny* edge detector, the algorithm can be implemented in two steps: pre-filtering and differentiation. The optimal pre-filter coefficients are listed in [61] for ramps with different widths ℓ (ℓ is restricted to be even). For differentiation, a pixel with gradient larger than a threshold T in the pre-filtered image is detected as a ramp center and stored in a binary map B. The information of B will be used to define the graphs in EA-GBT, and thus needs to be signaled to the decoder. We extend the idea of arithmetic edge coding (AEC) [19], which was proposed for step edge coding, for encoding the positions of ramp centers. The proposed method is called arithmetic ramp edge coding (AREC).

Given the ramp positions $p_1, p_2, \cdots, p_n$ in B, as shown in Fig.6.5, we create an ordered chain code $\{c_{1,2}, c_{2,3}, \cdots, c_{n-1,n}\}$ by traversing through the ramp pixels, where each element in the chain code denotes a directed link connecting two adjacent ramp components. Note that there are 8 possible directions that $c_{i-1,i}$ can take: {N, NE, E, SE, S, SW, W, NW}, as shown in Fig. 6.6(a). Given the direction of link $c_{i-1,i}$ from node p_{i-1} to p_i, there are only 5 possible directions for $c_{i,i+1}$ from node p_i to p_{i+1}: {*forward, slight right, right, slight left, left*}, as denoted in Fig. 6.6(b), assuming the ramp edge detected contains no sharp corners. As in AEC, the chain code is encoded using arithmetic coding. However, the chain code in AEC is composed of boundaries detected for step edges, and for each boundary, only 4 directions:

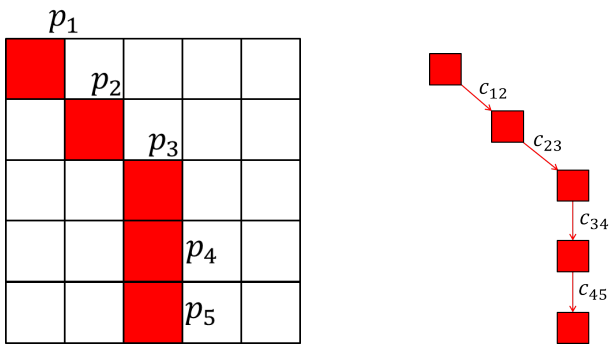


FIGURE 6.5: (a) An example of binary ramp map with pixels $p_1, p_2, \cdots p_5$ indicating the ramp centers, and (b) the chain code formed by traversing though the consecutive ramp pixels.

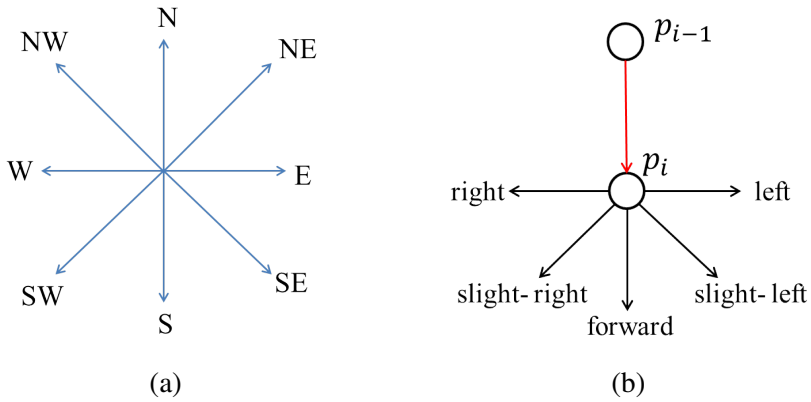
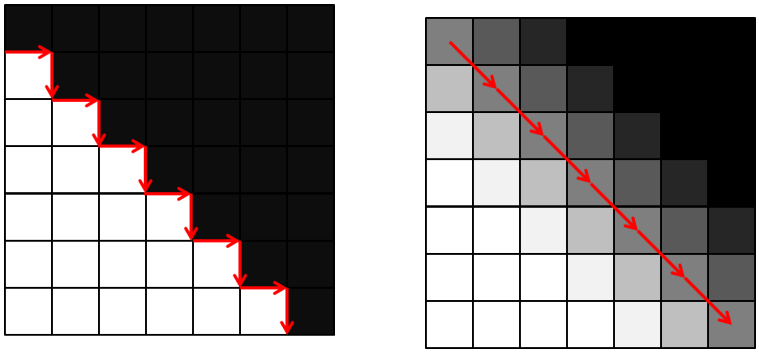


FIGURE 6.6: (a) The 8 directions that can be taken by c_i. (b) The potential traversing directions from p_i to p_{i+1} given the direction from p_{i-1} to p_i.

{N, E, W, S} can be taken, as shown in Fig. 6.7. AREC details are summarized in Algorithm 5, where the prediction for $\tilde{c}_{i,i+1}$ is computed using linear regression on k previously traversed pixels $p_{i-k-1}, \cdots p_{i-1}, p_i$. k is set to 4 in our simulation. The top most uncoded ramp center is chosen as p_1 with S (south) as the initial traversing direction.



(a) (b)

FIGURE 6.7: (a) An example of AEC coding on step edges and (b) AREC on ramp edges

In order to evaluate the coding efficiency of AREC, we compare AREC for ramp edge coding and AEC for step edge coding on the same set of 8×8 residual blocks taken from 8 video sequences without rate distortion optimization. The step edges are found using *Sobel* detector. The average bits per pixel (BPP) results are shown in Table 6.1. It can be seen that AREC achieves around 11.5% bitrate reduction with respect to AEC for both inter and intra predicted residuals.

	AEC for Step model	AREC for Ramp model	
	BPP ($\times 10^{-2}$)	BPP ($\times 10^{-2}$)	ΔBPP (%)
INTER predicted	6.50	5.71	-12.2
INTRA predicted	6.95	6.17	-11.2

TABLE 6.1: Bitrate comparison between AEC and AREC. ΔBPP indicates the bitrate gain of AREC over AEC

6.3.2 Graph Construction from the Edge Map

After applying ramp detector, a graph can be constructed based on the positions of detected ramp centers. A residual block is first represented as a 4-connected grid graph, where each link has weight 1, as shown in Fig. 6.8. Then for each ramp center detected, the neighbors along 4 directions: $\{top, bottom, left, right\}$ are inspected. If the neighbor along direction m is not a ramp component, the estimated weight $w < 1$ is assigned onto the $\frac{\ell}{2}$ links along

Algorithm 5 Arithmetic Ramp Edge Coding (AREC)

Input Binary map B with one ramp contour $\{p_1, p_2, \cdots p_n\}$

1: Initialize p_1 and traversing direction $c_{0,1}$
2: **for** $i = 1 : n-1$ **do**
3: Search for p_{i+1} from the 5 possible directions d_j with the priority ordered as {*forward, slight right, slight left, right, left*}
4: **if** $i \leq k$ **then**
5: Assign equal probability $\frac{1}{5}$ for d_j.
6: **else**
7: Predict the direction of $c_{i,i+1}$ as $\tilde{c}_{i,i+1}$
8: Compute the angle α_j between each d_j and $\tilde{c}_{i,i+1}$.
9: Compute the von Mises distribution $\varphi(\alpha_j)$ of angle α_j
10: Assign the probability for d_j to be $\frac{\varphi(\alpha_j)}{\sum_{r=1}^{5}\varphi(\alpha_r)}$
11: **end if**
12: Encode $c_{i,i+1}$, with one of the 5 possible directions, using the arithmetic coding with the assigned probability
13: **end for**

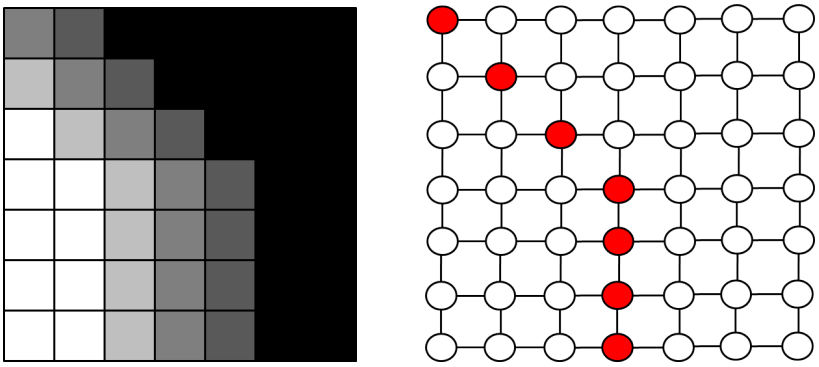


FIGURE 6.8: An example of residual block and the corresponding 4-connected grid graph (red nodes indicate the detected ramp centers)

direction m. An example is shown in Fig. 6.9(a), where the red nodes indicate the ramp centers detected and the dashed red links denote the links with the small weight. For ramp center c, $\frac{\ell}{2}$ links along the $right$ direction, where the neighbor n_2 is a non-ramp pixel, are assigned to be weak. In Fig. 6.9(b), we show an example of graph construction for $\ell = 2$.

6.4 Experimental Results

We apply the proposed transform to 6 test sequences: *BQMall*, *BasketballDrill*, *City*, *Crew*, *Harbour*, and *Soccer*. The transform block size is fixed as either 8×8 or 16×16. The encoder flow chart is shown in Fig. 6.10. The unquantized residual blocks for both inter and intra

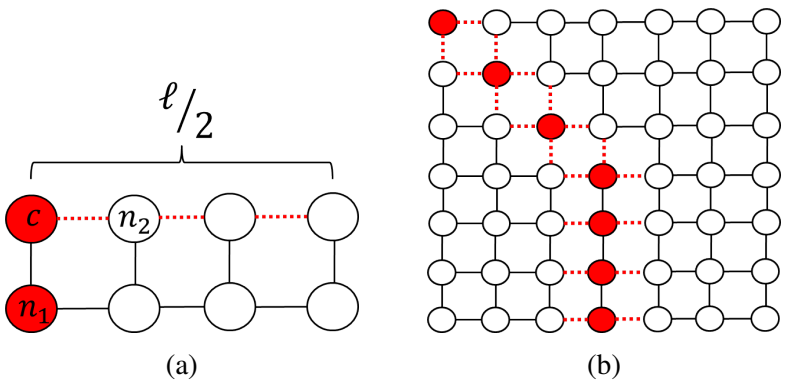


FIGURE 6.9: Examples of weak link assignment based on ramp positions

prediction are generated with HEVC (HM-14.0) at $QP = 32$. Each block is represented using a 4-connected grid graph. For transform coding, we apply a hybrid scheme, EA-GBT and DCT, of which the encoder will compare the rate distortion cost, defined as *Sum of Squared Error* + $\lambda \cdot$ *bitrate*. The transform with the lower cost will be selected as the final approach for the associated block. In our experiment, we consider two hybrid schemes, including the EA-GBT/DCT with step edge model and the EA-GBT/DCT with ramp edge model. The parameter λ is defined as $\lambda = 0.85 \cdot 2^{(QP-12)/3}$, where $QP = 24, 26, 28, 30, 32, 34$. For step edge detection, we use the *Sobel* operator, while for ramp edge detection, the method proposed in [61] is applied. For blocks using EA-GBT, the edge positions are encoded and signaled as an overhead. For step model, we use AEC proposed in [19]. For ramp model, we use AREC. In order to reduce the overhead, only one contour is allowed in each block. The ramp width is fixed as 2, chosen empirically, and therefore no signaling is required. The transformed coefficients for both EA-GBT and DCT are uniformly quantized and encoded

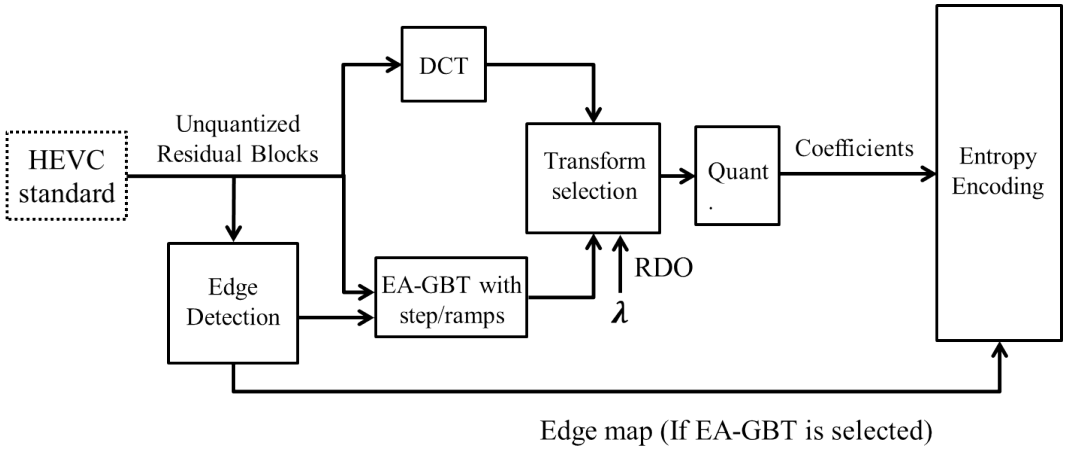


FIGURE 6.10: The video encoder with hybrid EA-GBT/DCT mode selection based on rate-distortion optimization (RDO)

using arithmetic coding.

The average PSNR and bitrate gain for the inter and intra predicted residual videos over pure DCT based encoder are shown in tables 6.2 and 6.3. For the intra predicted videos, EA-GBT/DCT with ramp edge model performs slightly better than EA-GBT/DCT with step edge model. For inter predicted residuals, even though AEC is not as efficient as AREC (shown in Table 6.1), EA-GBT/DCT with step model achieves better overall efficiency, indicating better edge representation for this model. The results coincide with our justifications discussed in Section 6.2.2. For both step and ramp models, EA-GBT/DCT outperforms the DCT based encoder. Note that as the size of transform block increases, the performance of EA-GBT improves, since larger blocks are more likely to have edges.

Methods	Size	Ramp 8×8		Step 8×8	
		Δ PSNR (dB)	**Δ rate (%)**	**Δ PSNR (dB)**	**Δ rate (%)**
BQmall	832×480	0.23	-4.04	0.24	-4.20
Basketball Drill	832×480	0.18	-3.82	0.18	-3.87
City	704×576	0.21	-3.37	0.23	-3.60
Crew	704×576	0.10	-2.04	0.11	-2.33
Harbour	704×576	0.21	-2.84	0.24	-3.26
Soccer	704×576	0.20	-2.88	0.24	-3.37
Average		**0.19**	**-3.17**	**0.21**	**-3.44**
Methods	Size	Ramp 16×16		Step 16×16	
		Δ PSNR (dB)	**Δ rate (%)**	**Δ PSNR (dB)**	**Δ rate (%)**
BQmall	832×480	0.35	-6.20	0.37	-6.65
Basketball Drill	832×480	0.18	-4.24	0.22	-5.11
City	704×576	0.25	-4.35	0.27	-4.91
Crew	704×576	0.14	-3.41	0.15	-3.64
Harbour	704×576	0.24	-3.46	0.24	-3.51
Soccer	704×576	0.21	-3.10	0.22	-3.34
Average		**0.23**	**-4.13**	**0.25**	**-4.53**

TABLE 6.2: Bjontegaard Delta Criterion for **INTER** predicted videos: PSNR and bitrate gain of EA-GBT-step and EA-GBT-ramp over DCT

6.5 Summary

In this chapter, we proposed a new edge model in EA-GBT based on ramp edges, which is justified experimentally for intra-predicted residuals using graph learning. Arithmetic ramp edge coding (AREC) is proposed to encode the detected ramp positions. Experimental results for EA-GBT with both step and ramp models demonstrate improved performance over DCT-based video coding. Moreover, for intra-predicted residuals, EA-GBT with the new ramp edge models performs better than EA-GBT with step edge models.

Methods	Size	Ramp 8×8		Step 8×8	
		Δ PSNR (dB)	**Δ rate (%)**	**Δ PSNR (dB)**	**Δ rate (%)**
BQmall	832×480	0.29	-3.40	0.27	-3.16
Basketball Drill	832×480	0.16	-2.69	0.15	-2.58
City	704×576	0.20	-2.13	0.14	-1.48
Crew	704×576	0.08	-1.52	0.06	-1.26
Harbour	704×576	0.21	-2.21	0.15	-1.58
Soccer	704×576	0.20	-1.98	0.13	-1.32
Average		**0.19**	**-2.32**	**0.15**	**-1.90**
Methods	Size	Ramp 16×16		Step 16×16	
		Δ PSNR (dB)	**Δ rate (%)**	**Δ PSNR (dB)**	**Δ rate (%)**
BQmall	832×480	0.48	-5.43	0.40	-4.61
Basketball Drill	832×480	0.15	-2.57	0.13	-2.22
City	704×576	0.24	-2.55	0.23	-2.40
Crew	704×576	0.13	-2.80	0.11	-2.38
Harbour	704×576	0.29	-2.96	0.26	-2.63
Soccer	704×576	0.22	-2.25	0.21	-2.15
Average		**0.25**	**-3.09**	**0.22**	**-2.73**

TABLE 6.3: Bjontegaard Delta Criterion for **INTRA** predicted videos: PSNR and bitrate gain of EA-GBT-step and EA-GBT-ramp over DCT

Chapter 7

Conclusions and Future Work

7.1 Conclusion

In this dissertation, we proposed several graph based algorithms for efficient compression images and video. In our first work, with theory and application discussed in Chapter 3 and 4, a low complexity graph based lifting transform is proposed. A problem of optimal bipartition, which divides nodes into a *Prediction set* and *Update set*, is defined in terms of energy compaction of the transformed coefficients. We applied a greedy algorithm for selecting the most representative nodes to be included in the *Update set* in each lifting level assuming the signal can be well modelled as GMRF. This statistical model has been widely used in the image and video processing literature. The results for the proposed bipartition outperform related work in terms of the mean square error of the high frequency coefficients stored int the *Prediction set*. Moreover, since for lifting transforms, graph links connecting nodes in the same sets cannot be utilized for filtering, we propose a bipartite graph reconnection based on Kron reduction, which is able to capture similarity between *Prediction Set* and *Update set* more accurately than the conventional approaches. The results in intra-predicted video coding show outstanding performance as compared to the state of the art DCT coding. In addition, comparable performance to the high complexity GFT can be achieved using the proposed lifting scheme with optimized bipartition and bipartite graph approximation.

In the second work in Chapter 5, we presented an application of graph based transform in light field image compression for random access. We proposed a novel coding scheme for light field images which is able to encode the original raw data without introducing redundancies from demosaicking and calibration. An intra-prediction algorithm is developed that explores the correlation between the sparsely distributed pixels between each block and its decoded neighbouring blocks within each sub-aperture images. The residual pixels are then connected as graphs and encoded with a graph based lifting transform. A learning algorithm for graph structure is also proposed using Maximum Likelihood (ML) estimation of GMRF model

parameters based on the observations with incomplete data. The results show very significant gains in coding efficiency against the All intra HEVC in high bit rates, which is commonly considered in archival scenario.

Finally, we discuss an edge model for video residuals and the construction of graphs in GFT based on different edge modeling. Each column and row from images signal is modeled as an Autoregressive Regressive (AR) process with edges modeled as *i.i.d* noise. We consider step edge and ramp edge models in our design, and derive the optimal GFTs for decorrelation. The discussion and justification of the two edge model on different types of predicted video residuals, *i.e.* the intra and inter-predicted residuals, is presented. The experiment on intra-predicted residuals shows promising performance using the proposed ramp edge model. Moreover, we developed a novel signalling method, call *Arithmetic Ramp Edge Coding*, for graph geometry based on ramp edges.

7.2 Future Work

There are several question we would like to address in future work. For lifting bipartition described in Chapter 3, currently we are using a greedy algorithm for selecting nodes to be included in the *Update set*. However, this has high complexity due to calculating and comparing the variance on each node. It will be interesting to develop a fast heuristic with the desired properties in the current algorithm, which include:

1. More nodes in areas with high variance texture should be included in the *Update set*.
2. Nodes with more local neighbours of large similarity, namely nodes that can provide better prediction for neighbouring nodes, should have higher likelihood to be selected into the *Update set*
3. Nodes having very few neighbours with large similarity, *i.e.* nodes that are nearly isolated on graphs, should have high likelihood to be included in the *Update set*, since they cannot be predicted well by any other nodes.

In the current light field coding scheme described in Chapter 5, we target a situation requiring efficient random access and therefore ignore the prediction and transform across different sub-aperture images. In the future work, we would like to consider a more general coding scheme considering both intra and inter view correlation, which may include block matching algorithm for sparsely distributed pixels and the possible edge connection between blocks in different sub-aperture images. In the last work in Chapter 6, currently we only consider the 1D AR process in modelling statistics in video residuals. Therefore the graph assignment for the 2D grid graph in our experiment is still an approximation. The extension

to a 2D model can be helpful in analysis. Also, it would be interesting to consider more edge models, *e.g.*, line edge, in future work.

Bibliography

[1] Nasir Ahmed, T Natarajan, and Kamisetty R Rao. “Discrete cosine transform”. In: *IEEE transactions on Computers* 100.1 (1974), pp. 90–93.

[2] Albert Cohen, Ingrid Daubechies, and J-C Feauveau. “Biorthogonal bases of compactly supported wavelets”. In: *Communications on pure and applied mathematics* 45.5 (1992), pp. 485–560.

[3] Alexandre Vieira, Helder Duarte, Cristian Perra, Luis Tavora, and Pedro Assuncao. “Data formats for high efficiency coding of lytro-illum light fields”. In: *Image Processing Theory, Tools and Applications (IPTA), 2015 International Conference on*. IEEE. 2015, pp. 494–497.

[4] An-Chao Tsai, Anand Paul, Jia-Ching Wang, and Jhing-Fa Wang. “Intensity gradient technique for efficient intra-prediction in H. 264/AVC”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 18.5 (2008), pp. 694–698.

[5] Ashok Veeraraghavan, Ramesh Raskar, Amit Agrawal, Ankit Mohan, and Jack Tumblin. “Dappled photography: Mask enhanced cameras for heterodyned light fields and coded aperture refocusing”. In: *ACM Trans. Graph.* 26.3 (2007), p. 69.

[6] Tom E Bishop and Paolo Favaro. “Full-resolution depth map estimation from an aliased plenoptic light field”. In: *Asian Conference on Computer Vision*. Springer. 2010, pp. 186–200.

[7] Changil Kim, Henning Zimmer, Yael Pritch, Alexander Sorkine-Hornung, and Markus H Gross. “Scene reconstruction from high spatio-angular resolution light fields.” In: *ACM Trans. Graph.* 32.4 (2013), pp. 73–1.

[8] Yung-Hsuan Chao, Gene Cheung, and Antonio Ortega. “Pre-demosaic light field image compression usinf graph lifting transform”. In: *to Image Processing (ICIP), 2017 IEEE International Conference on*. IEEE. 2017 forthcoming.

[9] Charilaos Christopoulos, Athanassios Skodras, and Touradj Ebrahimi. “The JPEG2000 still image coding system: an overview”. In: *IEEE transactions on consumer electronics* 46.4 (2000), pp. 1103–1127.

[10] Rama Chellappa and Shankar Chatterjee. “Classification of textures using Gaussian Markov random fields”. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 33.4 (1985), pp. 959–963.

[11] Chih-Chieh Chen, Yi-Chang Lu, and Ming-Shing Su. “Light field based digital refocusing using a DSLR camera with a pinhole array mask”. In: *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*. IEEE. 2010, pp. 754–757.

[12] Chin Chye Koh, Jayanta Mukherjee, and Sanjit K Mitra. “New efficient methods of image compression in digital cameras with color filter array”. In: *IEEE Transactions on Consumer Electronics* 49.4 (2003), pp. 1448–1456.

[13] King-Hong Chung and Yuk-Hee Chan. “A lossless compression scheme for Bayer color filter array images”. In: *IEEE Transactions on Image Processing* 17.2 (2008), pp. 134–144.

[14] Roger J Clarke. “Transform coding of images”. In: *Astrophysics* 1 (1985).

[15] Caroline Conti, Paulo Nunes, and Luís Ducla Soares. “HEVC-based light field image coding with bi-predicted self-similarity compensation”. In: *Multimedia & Expo Workshops (ICMEW), 2016 IEEE International Conference on*. IEEE. 2016, pp. 1–4.

[16] Caroline Conti, Paulo Nunes, and Luis Ducla Soares. “New HEVC prediction modes for 3D holoscopic video coding”. In: *Image Processing (ICIP), 2012 19th IEEE International Conference on*. IEEE. 2012, pp. 1325–1328.

[17] Donald G Dansereau, Oscar Pizarro, and Stefan B Williams. “Decoding, calibration and rectification for lenselet-based plenoptic cameras”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 1027–1034.

[18] Ismael Daribo, Gene Cheung, and Dinei Florencio. “Arithmetic edge coding for arbitrarily shaped sub-block motion prediction in depth video compression”. In: *2012 19th IEEE International Conference on Image Processing*. IEEE. 2012, pp. 1541–1544.

[19] Ismael Daribo, Dinei Florencio, and Gene Cheung. “Arbitrarily shaped motion prediction for depth video compression using arithmetic edge coding”. In: *Image Processing, IEEE Transactions on* 23.11 (2014), pp. 4696–4708.

[20] David I Shuman, Mohammad Javad Faraji, and Pierre Vandergheynst. “A multiscale pyramid transform for graph signals”. In: *IEEE Transactions on Signal Processing* 64.8 (2016), pp. 2119–2134.

[21] Dong Liu, Lizhi Wang, Li Li, Zhiwei Xiong, Feng Wu, and Wenjun Zeng. "Pseudo-sequence-based light field image compression". In: *Multimedia & Expo Workshops (ICMEW), 2016 IEEE International Conference on*. IEEE. 2016, pp. 1–4.

[22] Florian Dorfler and Francesco Bullo. "Kron reduction of graphs with applications to electrical networks". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 60.1 (2013), pp. 150–163.

[23] Eduardo Martinez-Enriquez, Jesus Cid-Sueiro, Fernando Diaz-De-Maria, and Antonio Ortega. "Directional Transforms for Video Coding Based on Lifting on Graphs". In: *IEEE Transactions on Circuits and Systems for Video Technology* (2016).

[24] Eduardo Pavez, Hilmi E Egilmez, Yongzhe Wang, and Antonio Ortega. "GTT: Graph template transforms with applications to image coding". In: *Picture Coding Symposium (PCS), 2015*. IEEE. 2015, pp. 199–203.

[25] Feng Dai, Jun Zhang, Yike Ma, and Yongdong Zhang. "Lenselet image compression scheme based on subaperture images streaming". In: *Image Processing (ICIP), 2015 IEEE International Conference on*. IEEE. 2015, pp. 4733–4737.

[26] Feng Pan, Xiao Lin, SUSANTO Rahardja, Keng Pang Lim, and ZG Li. "A directional field based fast intra mode decision algorithm for H. 264 video coding". In: *Multimedia and Expo, 2004. ICME'04. 2004 IEEE International Conference on*. Vol. 2. IEEE. 2004, pp. 1147–1150.

[27] Giulia Fracastoro and Enrico Magli. "Predictive graph construction for image compression". In: *Image Processing (ICIP), 2015 IEEE International Conference on*. IEEE. 2015, pp. 2204–2208.

[28] Godwin Shen, Woo-Shik Kim, Sunil K Narang, Antonio Ortega, Jaejoon Lee, and Hocheon Wey. "Edge-adaptive transforms for efficient depth map coding". In: *Picture Coding Symposium (PCS), 2010*. IEEE. 2010, pp. 566–569.

[29] Hae-Gon Jeon, Jaesik Park, Gyeongmin Choe, Jinsun Park, Yunsu Bok, Yu-Wing Tai, and In So Kweon. "Accurate depth map estimation from a lenslet light field camera". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 1547–1555.

[30] Harini Priyadarshini Hariharan, Tobias Lange, and Thorsten Herfet. "Low complexity light field compression based on pseudo-temporal circular sequencing". In: *Broadband Multimedia Systems and Broadcasting (BMSB), 2017 IEEE International Symposium on*. IEEE. 2017, pp. 1–5.

[31] Henrique S Malvar, Li-Wei He, and Ross Cutler. "High-quality linear interpolation for demosaicing of Bayer-patterned color images". In: *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*. Vol. 3. IEEE. 2004, pp. iii–485.

[32] Hilmi E Egilmez, Amir Said, Yung-Hsuan Chao, and Antonio Ortega. "Graph-based transforms for inter predicted video coding". In: *Image Processing (ICIP), 2015 IEEE International Conference on*. IEEE. 2015, pp. 3992–3996.

[33] Hilmi E Egilmez, Eduardo Pavez, and Antonio Ortega. "Graph learning from data under structural and laplacian constraints". In: *arXiv preprint arXiv:1611.05181* (2016).

[34] Hilmi E Egilmez, Yung-Hsuan Chao, Antonio Ortega, Bumshik Lee, and Sehoon Yea. "GBST: Separable transforms based on line graphs for predictive video coding". In: *Image Processing (ICIP), 2016 IEEE International Conference on*. IEEE. 2016, pp. 2375–2379.

[35] Hiroyuki Takeda, Sina Farsiu, and Peyman Milanfar. "Kernel regression for image processing and reconstruction". In: *IEEE Transactions on image processing* 16.2 (2007), pp. 349–366.

[36] Wei Hu, Gene Cheung, and Antonio Ortega. "Intra-prediction and generalized graph Fourier transform for image coding". In: *IEEE Signal Processing Letters* 22.11 (2015), pp. 1913–1917.

[37] Chiuan Hwang, ShinShan Zhuang, and Shang-Hong Lai. "Efficient intra mode selection using image structure tensor for H. 264/AVC". In: *Image Processing, 2007. ICIP 2007. IEEE International Conference on*. Vol. 5. IEEE. 2007, pp. V–289.

[38] Jean-Luc Starck, Emmanuel J Candès, and David L Donoho. "The curvelet transform for image denoising". In: *IEEE Transactions on image processing* 11.6 (2002), pp. 670–684.

[39] Ming Ji and Jiawei Han. "A Variance Minimization Criterion to Active Learning on Graphs." In: *AISTATS*. 2012, pp. 556–564.

[40] Wei Jiang, Hanjie Ma, and Yaowu Chen. "Gradient based fast mode decision algorithm for intra prediction in HEVC". In: *Consumer Electronics, Communications and Networks (CECNet), 2012 2nd International Conference on*. IEEE. 2012, pp. 1836–1840.

[41] Mladen Kolar and Eric P Xing. "Consistent covariance selection from data with missing values". In: *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*. 2012, pp. 551–558.

[42] Erwan Le Pennec and Stephane Mallat. "Bandelet image approximation and compression". In: *Multiscale Modeling & Simulation* 4.3 (2005), pp. 992–1039.

[43] Sang-Yong Lee and Antonio Ortega. "A Novel Approach for Compression of Images Captured using Bayer Color Filter Arrays". In: *arXiv preprint arXiv:0903.2272* (2009).

[44] Sang-Yong Lee and Antonio Ortega. "A novel approach of image compression in digital cameras with a Bayer color filter array". In: *Image Processing, 2001. Proceedings. 2001 International Conference on*. Vol. 3. IEEE. 2001, pp. 482–485.

[45] Marc Levoy and Pat Hanrahan. "Light field rendering". In: *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM. 1996, pp. 31–42.

[46] *Light-Field Image Dataset*. http://mmspg.epfl.ch/EPFL-light-field-image-dataset.

[47] *Light Field Toolbox v0.4*. https://www.mathworks.com/matlabcentral/fileexchange/49683-light-field-toolbox-v0-4.

[48] Li Li, Zhu Li, Bin Li, Dong Liu, and Houqiang Li. "Pseudo Sequence Based 2-D Hierarchical Coding Structure for Light-Field Image Compression". In: *Data Compression Conference (DCC), 2017*. IEEE. 2017, pp. 131–140.

[49] Karim Lounici et al. "High-dimensional covariance matrix estimation with missing observations". In: *Bernoulli* 20.3 (2014), pp. 1029–1058.

[50] *Lytro Illum*. https://illum.lytro.com/.

[51] Makan Fardad, Fu Lin, and Mihailo R Jovanović. "Algorithms for leader selection in large dynamical networks: Noise-free leaders". In: *2011 50th IEEE Conference on Decision and Control and European Control Conference*. IEEE. 2011, pp. 7188–7193.

[52] Eduardo Martínez Enríquez. "Lifting transforms on graphs and their application to video coding". PhD dissertation. Universidad Carlos III de Madrid, 2013.

[53] Eduardo Martínez-Enríquez, Fernando Díaz-de María, and Antonio Ortega. "Video encoder based on lifting transforms on graphs". In: *2011 18th IEEE International Conference on Image Processing*. IEEE. 2011, pp. 3509–3512.

[54] Eduardo Martínez-Enríquez and Antonio Ortega. "Lifting transforms on graphs for video coding". In: *2011 Data Compression Conference*. IEEE. 2011, pp. 73–82.

[55] Mozhdeh Seifi, Neus Sabater, Valter Drazic, and Patrick Perez. "Disparity-guided demosaicking of light field images". In: *Image Processing (ICIP), 2014 IEEE International Conference on*. IEEE. 2014, pp. 5482–5486.

[56] Sunil K Narang and Antonio Ortega. "Lifting based wavelet transforms on graphs". In: *Proceedings: APSIPA ASC 2009: Asia-Pacific Signal and Information Processing Association, 2009 Annual Summit and Conference*. 2009, pp. 441–444.

[57] Sunil K Narang and Antonio Ortega. "Local two-channel critically sampled filter-banks on graphs". In: *2010 IEEE International Conference on Image Processing*. IEEE. 2010, pp. 333–336.

[58] Sunil K Narang and Antonio Ortega. "Perfect reconstruction two-channel wavelet filter banks for graph structured data". In: *IEEE Transactions on Signal Processing* 60.6 (2012), pp. 2786–2799.

[59] Ha Q Nguyen and Minh N Do. "Downsampling of signals on graphs via maximum spanning trees". In: *IEEE Transactions on Signal Processing* 63.1 (2015), pp. 182–191.

[60] Eduardo Pavez and Antonio Ortega. "Generalized Laplacian precision matrix estimation for graph signal processing". In: *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE. 2016, pp. 6350–6354.

[61] Maria Petrou and Josef Kittler. "Optimal edge detectors for ramp edges". In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 5 (1991), pp. 483–491.

[62] *Raytrix Camera*. https://www.raytrix.de/.

[63] Ren Ng, Marc Levoy, Mathieu Brédif, Gene Duval, Mark Horowitz, and Pat Hanrahan. "Light field photography with a hand-held plenoptic camera". In: *Computer Science Technical Report CSTR* 2.11 (2005), pp. 1–11.

[64] Martin Řeřábek and Touradj Ebrahimi. "New light field image dataset". In: *8th International Conference on Quality of Multimedia Experience (QoMEX)*. EPFL-CONF-218363. 2016.

[65] Ricardo Monteiro, Luís Lucas, Caroline Conti, Paulo Nunes, Nuno Rodrigues, Sérgio Faria, Carla Pagliari, Eduardo da Silva, and Luís Soares. "Light field HEVC-based image coding using locally linear embedding and self-similarity compensated prediction". In: *Multimedia & Expo Workshops (ICMEW), 2016 IEEE International Conference on*. IEEE. 2016, pp. 1–4.

[66] Amir Said and William 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.

[67] Godwin Shen and Antonio Ortega. “Optimized distributed 2D transforms for irregularly sampled sensor network grids using wavelet lifting”. In: *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2008, pp. 2513–2516.

[68] Godwin Shen and Antonio Ortega. “Tree-based wavelets for image coding: Orthogonalization and tree selection”. In: *Picture Coding Symposium, 2009. PCS 2009*. IEEE. 2009, pp. 1–4.

[69] Shengyang Zhao, Zhibo Chen, Kun Yang, and Hongru Huangi. “Light field image coding with hybrid scan order”. In: *Visual Communications and Image Processing (VCIP), 2016*. IEEE. 2016, pp. 1–4.

[70] Jianbo Shi and Jitendra Malik. “Normalized cuts and image segmentation”. In: *IEEE Transactions on pattern analysis and machine intelligence* 22.8 (2000), pp. 888–905.

[71] Nicolas Städler and Peter Bühlmann. “Missing values: sparse inverse covariance estimation and an extension to sparse regression”. In: *Statistics and Computing* 22.1 (2012), pp. 219–235.

[72] Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. “The lumigraph”. In: *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM. 1996, pp. 43–54.

[73] David Taubman and Michael Marcellin. *JPEG2000 image compression fundamentals, standards and practice: image compression fundamentals, standards and practice*. Vol. 642. Springer Science & Business Media, 2012.

[74] Bryan Usevitch. “Optimal bit allocation for biorthogonal wavelet coding”. In: *Data Compression Conference, 1996. DCC’96. Proceedings*. IEEE. 1996, pp. 387–395.

[75] Wei-Chao Chen, Jean-Yves Bouguet, Michael H Chu, and Radek Grzeszczuk. “Light field mapping: efficient representation and hardware rendering of surface light fields”. In: *ACM Transactions on Graphics (TOG)* 21.3 (2002), pp. 447–456.

[76] Wei Hu, Gene Cheung, Antonio Ortega, and Oscar C Au. “Multiresolution graph fourier transform for compression of piecewise smooth images”. In: *IEEE Transactions on Image Processing* 24.1 (2015), pp. 419–433.

[77] Woo-Shik Kim, Sunil K Narang, and Ortega, Antonio. "Graph based transforms for depth video coding". In: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2012, pp. 813–816.

[78] Xiaowen Dong, Dorina Thanou, Pascal Frossard, and Pierre Vandergheynst. "Learning laplacian matrix in smooth graph signal representations". In: *IEEE Transactions on Signal Processing* 64.23 (2016), pp. 6160–6173.

[79] Shan Xu, Zhi-Liang Zhou, and Nicholas Devaney. "Multi-view Image Restoration from Plenoptic Raw Images". In: *Asian Conference on Computer Vision*. Springer. 2014, pp. 3–15.

[80] Yung-Hsuan Chao, Antonio Ortega, and Sehoon Yea. "Graph-based lifting transform for intra-predicted video coding". In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2016, pp. 1140–1144.

[81] Yung-Hsuan Chao, Hilmi E Egilmez, Antonio Ortega, Sehoon Yea, and Bumshik Lee. "Edge adaptive graph-based transforms: Comparison of step/ramp edge models for video compression". In: *Image Processing (ICIP), 2016 IEEE International Conference on*. IEEE. 2016, pp. 1539–1543.

[82] Yunsu Bok, Hae-Gon Jeon, and In So Kweon. "Geometric calibration of micro-lens-based light field cameras using line features". In: *IEEE transactions on pattern analysis and machine intelligence* 39.2 (2017), pp. 287–300.

[83] Bing Zeng and Jingjing Fu. "Directional discrete cosine transforms—a new framework for image coding". In: *IEEE transactions on circuits and systems for video technology* 18.3 (2008), pp. 305–313.

[84] Cha Zhang and Dinei Florêncio. "Analyzing the optimality of predictive transform coding using graph-based models". In: *IEEE Signal Processing Letters* 20.1 (2013), pp. 106–109.

[85] Cha Zhang, Dinei Florêncio, and Philip A Chou. "Graph signal processing–a probabilistic framework". In: *Microsoft Res., Redmond, WA, USA, Tech. Rep. MSR-TR-2015-31* (2015).

[86] Fuzhen Zhang. *The Schur complement and its applications*. Vol. 4. Springer Science & Business Media, 2006.

Appendix A

Reconnection using Kron Reduction

In this appendix we will prove that the prediction of any node v in the *Prediction set* ($\mathcal{P}$) using the proposed predictor scheme, namely applying the generalized CDF5/3 filterbanks after graph reconnection, is equivalent to applying the maximum a posteriori estimation (MAP) for v using $\mathcal{U}$ assuming the signal can be modeled as GMRF defined by the graph structure.

Define m and n as the size of $\mathcal{P}$ and $\mathcal{U}$, and sets $\mathcal{P}^- = \mathcal{P}/\{v\}$ and $\mathcal{U}^+ = \mathcal{U} \cup \{v\}$. Without loss of generality, the indices of nodes in $\mathcal{U}^+$ are ordered as $[v, \mathcal{U}]$, and the indices of nodes in $\mathcal{P}$ are ordered as $[v, \mathcal{P}^-]$. The MAP estimation of $\mathcal{P}$ given $\mathcal{U}$ is calculated as

$$\mathbf{P}_{\mathcal{P}|\mathcal{U}}\mathbf{f}_{\mathcal{U}} = -\mathbf{L}_{\mathcal{P},\mathcal{P}}^{-1}\mathbf{L}_{\mathcal{P},\mathcal{U}}\mathbf{f}_{\mathcal{U}}. \tag{A.1}$$

We can rewrite $\mathbf{L}_{\mathcal{P},\mathcal{P}}$ and $\mathbf{L}_{\mathcal{P},\mathcal{U}}$ in block matrix forms:

$$\mathbf{L}_{\mathcal{P},\mathcal{P}} = \left[\begin{array}{c|c} \deg(v) + h_v & \mathbf{L}_{v,\mathcal{P}^-} \\ \hline \mathbf{L}_{\mathcal{P}^-,v} & \mathbf{L}_{\mathcal{P}^-,\mathcal{P}^-} \end{array}\right] \tag{A.2}$$

$$\mathbf{L}_{\mathcal{P},\mathcal{U}} = \left[\begin{array}{c} \mathbf{L}_{v,\mathcal{U}} \\ \hline \mathbf{L}_{\mathcal{P}^-,\mathcal{U}} \end{array}\right], \tag{A.3}$$

where $\deg(v)$ and h_v are the degree and self loop of node v. Define $\mathbf{c} = \mathbf{L}_{\mathcal{P}^-,v}$, $\mathbf{b}^T = \mathbf{L}_{v,\mathcal{P}^-}$, and $s_v = \deg(v) + h_v$, the matrix in (A.2) can be re-written as

$$\mathbf{L}_{\mathcal{P},\mathcal{P}} = \left[\begin{array}{c|c} s_v & \mathbf{b}^T \\ \hline \mathbf{c} & \mathbf{L}_{\mathcal{P}^-,\mathcal{P}^-} \end{array}\right]. \tag{A.4}$$

The inverse can be calculated using block-wise matrix inversion:

$$\mathbf{L}^{-1}_{\mathcal{P},\mathcal{P}} = \left[\begin{array}{c|c} (s_v - \mathbf{b}^T\mathbf{L}^{-1}_{\mathcal{P}^-,\mathcal{P}^-}\mathbf{c})^{-1} & -s_v^{-1}\mathbf{b}^T(\mathbf{L}_{\mathcal{P}^-,\mathcal{P}^-} - \mathbf{c}s_v^{-1}\mathbf{b}^T)^{-1} \\ \hline -\mathbf{L}^{-1}_{\mathcal{P}^-,\mathcal{P}^-}\mathbf{c}(s_v - \mathbf{b}^T\mathbf{L}^{-1}_{\mathcal{P}^-,\mathcal{P}^-}\mathbf{c})^{-1} & (\mathbf{L}_{\mathcal{P}^-,\mathcal{P}^-} - \mathbf{c}s_v^{-1}\mathbf{b}^T)^{-1} \end{array}\right]. \tag{A.5}$$

The MAP estimated of v given $\mathcal{U}$, which corresponds to the first row in $\mathbf{P}_{\mathcal{P}|\mathcal{U}}$, is therefore expressed as below.

$$\begin{aligned} \mathbf{P}_{\mathcal{P}|\mathcal{U}}(1,:) &= -\mathbf{L}^{-1}_{\mathcal{P},\mathcal{P}}(1,:)\cdot\mathbf{L}_{\mathcal{P},\mathcal{U}} \\ &= -\left[\begin{array}{cc} (s_v - \mathbf{b}^T\mathbf{L}^{-1}_{\mathcal{P}^-,\mathcal{P}^-}\mathbf{c})^{-1} & -s_v^{-1}\mathbf{b}^T(\mathbf{L}_{\mathcal{P}^-,\mathcal{P}^-} - \mathbf{c}s_v^{-1}\mathbf{b}^T)^{-1} \end{array}\right] \left[\begin{array}{c} \mathbf{L}_{v,\mathcal{U}} \\ \mathbf{L}_{\mathcal{P}^-,\mathcal{U}} \end{array}\right] \\ &= -(s_v - \mathbf{b}^T\mathbf{L}^{-1}_{\mathcal{P}^-,\mathcal{P}^-}\mathbf{c})^{-1}\mathbf{L}_{v,\mathcal{U}} + s_v^{-1}\mathbf{b}^T(\mathbf{L}_{\mathcal{P}^-,\mathcal{P}^-} - \mathbf{c}s_v^{-1}\mathbf{b}^T)^{-1}\mathbf{L}_{\mathcal{P}^-\mathcal{U}}. \end{aligned} \tag{A.6}$$

Define constant q as

$$q = -(s_v - \mathbf{b}^T\mathbf{L}^{-1}_{\mathcal{P}^-,\mathcal{P}^-}\mathbf{c}), \tag{A.7}$$

(A.6) can be further simplified as

$$
\begin{aligned}
&q^{-1}\mathbf{L}_{v,\mathcal{U}} + s_v^{-1}\mathbf{b}^T(\mathbf{L}_{\mathcal{P}^-,\mathcal{P}^-} - \mathbf{c}s_v^{-1}\mathbf{b}^T)^{-1}\mathbf{L}_{\mathcal{P}^-,\mathcal{U}} \\
&= q^{-1}\mathbf{L}_{v,\mathcal{U}} + \underline{q^{-1}}qs_v^{-1}\mathbf{b}^T(\mathbf{L}_{\mathcal{P}^-,\mathcal{P}^-} - \mathbf{c}s_v^{-1}\mathbf{b}^T)^{-1}\mathbf{L}_{\mathcal{P}^-\mathcal{U}} \\
&= \underline{q^{-1}}(\mathbf{L}_{v,\mathcal{U}} + qs_v^{-1}\mathbf{b}^T(\mathbf{L}_{\mathcal{P}^-,\mathcal{P}^-} - \mathbf{c}s_v^{-1}\mathbf{b}^T)^{-1}\mathbf{L}_{\mathcal{P}^-\mathcal{U}}) \\
&= q^{-1}(\mathbf{L}_{v,\mathcal{U}}\underline{-(s_v - \mathbf{b}^T\mathbf{L}^{-1}_{\mathcal{P}^-,\mathcal{P}^-}\mathbf{c})}s_v^{-1}\mathbf{b}^T(\mathbf{L}_{\mathcal{P}^-,\mathcal{P}^-} - \mathbf{c}s_v^{-1}\mathbf{b}^T)^{-1}\mathbf{L}_{\mathcal{P}^-\mathcal{U}}) \\
&= q^{-1}(\mathbf{L}_{v,\mathcal{U}} - \underline{(\mathbf{b}^T - s_v^{-1}\mathbf{b}^T\mathbf{L}^{-1}_{\mathcal{P}^-,\mathcal{P}^-}\mathbf{c}\mathbf{b}^T)}(\mathbf{L}_{\mathcal{P}^-,\mathcal{P}^-} - \mathbf{c}s_v^{-1}\mathbf{b}^T)^{-1}\mathbf{L}_{\mathcal{P}^-\mathcal{U}}) \\
&= q^{-1}(\mathbf{L}_{v,\mathcal{U}} - \underline{\mathbf{b}^T}(\mathbf{I} - s_v^{-1}\mathbf{L}^{-1}_{\mathcal{P}^-,\mathcal{P}^-}\mathbf{c}\mathbf{b}^T)(\mathbf{L}_{\mathcal{P}^-,\mathcal{P}^-} - s_v^{-1}\mathbf{c}\mathbf{b}^T)^{-1}\mathbf{L}_{\mathcal{P}^-\mathcal{U}}) \\
&= q^{-1}(\mathbf{L}_{v,\mathcal{U}} - \mathbf{b}^T(\mathbf{I} - s_v^{-1}\mathbf{L}^{-1}_{\mathcal{P}^-,\mathcal{P}^-}\mathbf{c}\mathbf{b}^T)(\mathbf{L}_{\mathcal{P}^-,\mathcal{P}^-} - \underline{\mathbf{L}_{\mathcal{P}^-,\mathcal{P}^-}\mathbf{L}^{-1}_{\mathcal{P}^-,\mathcal{P}^-}}s_v^{-1}\mathbf{c}\mathbf{b}^T)^{-1}\mathbf{L}_{\mathcal{P}^-\mathcal{U}}) \\
&= q^{-1}(\mathbf{L}_{v,\mathcal{U}} - \mathbf{b}^T(\mathbf{I} - s_v^{-1}\mathbf{L}^{-1}_{\mathcal{P}^-,\mathcal{P}^-}\mathbf{c}\mathbf{b}^T)(\underline{(\mathbf{L}_{\mathcal{P}^-,\mathcal{P}^-})}(\mathbf{I} - s_v^{-1}\mathbf{L}^{-1}_{\mathcal{P}^-,\mathcal{P}^-}\mathbf{c}\mathbf{b}^T))^{-1}\mathbf{L}_{\mathcal{P}^-\mathcal{U}}) \\
&= q^{-1}(\mathbf{L}_{v,\mathcal{U}} - \mathbf{b}^T(\mathbf{I} - s_v^{-1}\mathbf{L}^{-1}_{\mathcal{P}^-,\mathcal{P}^-}\mathbf{c}\mathbf{b}^T)\underline{(\mathbf{I} - s_v^{-1}\mathbf{L}^{-1}_{\mathcal{P}^-,\mathcal{P}^-}\mathbf{c}\mathbf{b}^T)^{-1}\mathbf{L}^{-1}_{\mathcal{P}^-,\mathcal{P}^-}}\mathbf{L}_{\mathcal{P}^-\mathcal{U}}) \\
&= q^{-1}(\mathbf{L}_{v,\mathcal{U}} - \mathbf{b}^T\mathbf{L}^{-1}_{\mathcal{P}^-,\mathcal{P}^-}\mathbf{L}_{\mathcal{P}^-\mathcal{U}}).
\end{aligned}
\tag{A.8}
$$

In our design of prediction transform, we apply the generalized CDF5/3 after reconnecting nodes $v \in \mathcal{P}$ to $\mathcal{U} = [v_1, v_2, \cdots, v_n]$. The reconnection is derived with Kron reduction by removing nodes in $\mathcal{P}^- = \mathcal{P}/\{v\}$. We define the the weights on links between v and $[v_1, v_2, \cdots, v_n]$ after reconnection as $[w_{v,v1}, w_{v,v2}, \cdots, w_{v,vn}]$. The graph Laplacian $\mathbf{L}_{\text{kron}}$ after removing nodes in $\mathcal{P}^-$ can be written as

$$
\mathbf{L}_{\text{kron}} = \mathbf{L}_{\mathcal{U}^+,\mathcal{U}^+} - \mathbf{L}_{\mathcal{U}^+,\mathcal{P}^-}\mathbf{L}^{-1}_{\mathcal{P}^-,\mathcal{P}^-}\mathbf{L}_{\mathcal{P}^-,\mathcal{U}^+}. \tag{A.9}
$$

The first row of $\mathbf{L}_{\text{kron}}$, provides the information for connecting v and $\mathcal{U}$ and the associated weight on each link. The matrix $\mathbf{L}_{\mathcal{U}^+,\mathcal{U}^+}$, $\mathbf{L}_{\mathcal{U}^+,\mathcal{P}^-}$, and $\mathbf{L}_{\mathcal{P}^-,\mathcal{U}^+}$, can be expressed in block

matrix forms as

$$\mathbf{L}_{\mathcal{U}^+,\mathcal{U}^+} = \left[\begin{array}{c|c} s_v & \mathbf{L}_{v,\mathcal{U}} \\ \hline \mathbf{L}_{\mathcal{U},v} & \mathbf{L}_{\mathcal{U},\mathcal{U}} \end{array}\right] \tag{A.10}$$

,

$$\mathbf{L}_{\mathcal{U}^+,\mathcal{P}^-} = \left[\begin{array}{c} \mathbf{L}_{v,\mathcal{P}^-} \\ \hline \mathbf{L}_{\mathcal{U},\mathcal{P}^-} \end{array}\right] = \left[\begin{array}{c} \mathbf{b}^T \\ \hline \mathbf{L}_{\mathcal{U},\mathcal{P}^-} \end{array}\right] \tag{A.11}$$

, and

$$\mathbf{L}_{\mathcal{P}^-,\mathcal{U}^+} = \left[\begin{array}{c|c} \mathbf{L}_{\mathcal{P}^-,v} & \mathbf{L}_{\mathcal{P}^-,\mathcal{U}} \end{array}\right] = \left[\begin{array}{c|c} \mathbf{c} & \mathbf{L}_{\mathcal{P}^-,\mathcal{U}} \end{array}\right]. \tag{A.12}$$

The first row of $\mathbf{L}_{\text{kron}}$ can therefore be written as

$$\begin{aligned} \mathbf{L}_{\text{kron}}(1,:) &= \left[\begin{array}{c|c} s_v & \mathbf{L}_{v,\mathcal{U}} \end{array}\right] - \mathbf{b}^T \mathbf{L}^{-1}_{\mathcal{P}^-,\mathcal{P}^-} \left[\begin{array}{c|c} \mathbf{c} & \mathbf{L}_{\mathcal{P}^-,\mathcal{U}} \end{array}\right] \\ &= \left[\begin{array}{c|c} s_v - \mathbf{b}^T \mathbf{L}^{-1}_{\mathcal{P}^-,\mathcal{P}^-}\mathbf{c} & \mathbf{L}_{v,\mathcal{U}} - \mathbf{b}^T \mathbf{L}^{-1}_{\mathcal{P}^-,\mathcal{P}^-}\mathbf{L}_{\mathcal{P}^-,\mathcal{U}} \end{array}\right]. \end{aligned} \tag{A.13}$$

The first term $s_v - \mathbf{b}^T \mathbf{L}^{-1}_{\mathcal{P}^-,\mathcal{P}^-}\mathbf{c}$ is the summation of degree and self loop of node v after the reconstruction, which is also equivalent to the negative of constant q defined in (A.7). The negative of the $1 \times n$ vector $\mathbf{L}_{v,\mathcal{U}} - \mathbf{b}^T \mathbf{L}^{-1}_{\mathcal{P}^-,\mathcal{P}^-}\mathbf{L}_{\mathcal{P}^-,\mathcal{U}}$, on the other hand, stores the weights $[w_{v,v1}, w_{v,v2}, \cdots, w_{v,vn}]$ on links between v to $\mathcal{U}$. As mentioned in Section 2.4.1 and 5.5.3,

the prediction of node v, denoted as $\hat{f}_v$, using CDF5/3 is expressed as

$$\begin{aligned}\hat{f}_v &= \frac{1}{\deg(v) + h_v} \sum_{v_j \in \mathcal{U}} w_{v,vj} f_{v_j} \\ &= \frac{1}{\deg(v) + h_v} [w_{v,v1}, w_{v,v2}, \cdots, w_{v,vn}] \mathbf{f}_{\mathcal{U}}\end{aligned}, \tag{A.14}$$

where $\deg(v)$, h_v, and $w_{v,vj}$ are the degree, self loop, and link weights on the links connecting v in the graph after reconnection. Replace the variables with the representation derived in (A.13), (A.14) can be written as

$$\begin{aligned}\hat{f}_v &= -(s_v - \mathbf{b}^T \mathbf{L}^{-1}_{\mathcal{P}^-,\mathcal{P}^-} \mathbf{c})^{-1} (\mathbf{L}_{v,\mathcal{U}} - \mathbf{b}^T \mathbf{L}^{-1}_{\mathcal{P}^-,\mathcal{P}^-} \mathbf{L}_{\mathcal{P}^-,\mathcal{U}}) \\ &= q^{-1} (\mathbf{L}_{v,\mathcal{U}} - \mathbf{b}^T \mathbf{L}^{-1}_{\mathcal{P}^-,\mathcal{P}^-} \mathbf{L}_{\mathcal{P}^-,\mathcal{U}})\end{aligned}, \tag{A.15}$$

which is equivalent to the MAP estimation in (A.8). We therefore prove that the proposed prediction with the generalized CDF5/3 after reconnection is equivalent to the MAP estimation of the underlying GMRF.