DPTO. DE TEORÍA DE LA SEÑAL Y COMUNICACIONES
UNIVERSIDAD CARLOS III DE MADRID

LIFTING TRANSFORMS ON GRAPHS AND THEIR APPLICATION TO VIDEO CODING

by

Eduardo Martinez Enriquez

December 2013

Thesis Title:
LIFTING TRANSFORMS ON GRAPHS AND THEIR APPLICATION TO VIDEO CODING

Author:
EDUARDO MARTÍNEZ ENRÍQUEZ

Advisors:
DR. FERNANDO DÍAZ DE MARÍA
DR. ANTONIO ORTEGA DIEGO

Dissertation Committee:

DR. NARCISO GARCÍA SANTOS

DRA. CARMEN PELÁEZ MORENO

DR. PASCAL FROSSARD

A mi familia, María y Nico...

ABSTRACT

Compact representations of data are very useful in many applications such as coding, denoising or feature extraction. "Classical" transforms such as Discrete Cosine Transforms (DCT) or Discrete Wavelets Transforms (DWT) provide sparse approximations of smooth signals, but lose efficiency when they are applied to signals with large discontinuities. In such cases, *directional transforms*, which are able to adapt their basis functions to the underlying signal structure, improve the performance of "classical" transforms.

In this PhD Thesis we describe a general class of **lifting transforms on graphs** that can be seen as **N-dimensional directional transforms**. Graphs are constructed so that every node corresponds to a specific sample point of a discrete N-dimensional signal and links between nodes represent correlation between samples. Therefore, non-correlated samples (e.g., samples across a large discontinuity in the signal) should not be linked.

We propose a lifting-based directional transform that can be applied to any undirected graph. In this transform, filtering operations are performed following high-correlation directions (indicated by the links between nodes), thus avoiding filtering across large discontinuities that give rise to large high-pass coefficients in those locations. In this way, the transform efficiently exploits the correlation that exists between data on the graph, leading to a more compact representation.

We mainly focus on the design and optimization of these lifting transforms on graphs, studying and discussing the three main steps required to obtain an invertible and critically sampled transform: (i) graph construction, (ii) design of "good" graph bipartitions, and (iii) filter design. We also explain how to extend the transform to J levels of decomposition, obtaining a multiresolution analysis of the original N-dimensional signal.

The proposed transform has many desirable properties, such as perfect reconstruction, critically-sampled, easy generalization to N-dimensional domains, non-separable and one-dimensional filtering operations, localization in frequency and in the original domain, and the ability to choose any filtering direction.

As an application, we develop a graph-based video encoder where the goal is to obtain a compact representation of the original video sequence. To this end, we first

propose a graph-representation of the video sequence and then design a 3-dimensional (spatio-temporal) non-separable directional transform. This can be viewed as an extension of wavelet transform-based video encoders that operate in the spatial and in the temporal domains independently. Our transform yields better compaction ability (in terms of non-linear approximation) than a state of the art motion-compensated temporal filtering transform (which can be interpreted as a temporal wavelet transform) and a comparable hybrid Discrete Cosine Transform (DCT)-based video encoder (which is the basis of the latest video coding standards).

In order to obtain a complete video encoder, the transform coefficients and the side information (needed to obtain an invertible scheme) should be entropy coded and sent to the decoder. Therefore, we also propose a coefficient-reordering method based on the information of the graph which allows to improve the compression ability of the entropy encoder. Furthermore, we design two different low-cost approaches which aim to reduce the extensive computational complexity of the proposed system without causing significant losses of compression performance. The proposed complete system leads to an efficient encoder which significantly outperforms a comparable hybrid DCT-based encoder in rate-distortion terms. Finally, we investigate how rate-distortion optimization can be applied to the proposed coding scheme.

RESUMEN

La representación compacta de señales resulta útil en diversas aplicaciones, tales como compresión, reducción de ruido, o extracción de características. Transformadas "clásicas" como la Transformada Discreta del Coseno (DCT) o la Transformada Wavelet Discreta (DWT) logran aproximaciones compactas de señales suaves, pero pierden su eficiencia al ser aplicadas sobre señales que contienen grandes discontinuidades. En estos casos, las *transformadas direccionales*, capaces de adaptar sus funciones base a la estructura de la señal a analizar, mejoran la eficiencia de las transformadas "clásicas".

En esta tesis nos centramos en el diseño y optimización de **transformadas "lifting" sobre grafos**, las cuales pueden ser interpretadas como **transformadas direccionales N-dimensionales**.

Los grafos son construidos de manera que cada nodo se corresponde con una muestra específica de una señal discreta N-dimensional, y los enlaces entre los nodos representan correlación entre muestras. Así, muestras no correlacionadas (por ejemplo, muestras que se encuentran a ambos lados de una discontinuidad) no deberían estar unidas. Sobre el grafo formado aplicaremos transformadas basadas en el esquema "lifting", en las que las operaciones de filtrado se realizan siguiendo las direcciones indicadas por los enlaces entre nodos (direcciones de alta correlación). De esta manera, evitaremos filtrar cruzando a través de largas discontinuidades (lo que resultaría en coeficientes con alto valor en dichas discontinuidades), dando lugar a una transformada direccional que explota la correlación que existe entre las muestras de la señal en el grafo, obteniendo una representación compacta de dicha señal.

En esta tesis nos centramos, principalmente, en investigar los tres principales pasos requeridos para obtener una transformada direccional basada en el esquema "lifting" aplicado en grafos: (i) la construcción del grafo, (ii) el diseño de biparticiones del grafo, y (iii) la definición de los filtros. El buen diseño de estos tres procesos determinará, entre otras cosas, la capacidad para compactar la energía de la transformada. También explicamos cómo extender este tipo de transformadas a J niveles de descomposición, obteniendo un análisis multi-resolución de la señal N-dimensional original. La transformada propuesta tiene muchas propiedades deseables, tales como reconstrucción perfecta, muestreo crítico, fácil generalización a dominios N-dimensionales, operaciones

de filtrado no separables y unidimensionales, localización en frecuencia y en el dominio original, y capacidad de elegir cualquier dirección de filtrado.

Como aplicación, desarrollamos un codificador de vídeo basado en grafos donde el objetivo es obtener una versión compacta de la señal de vídeo original. Para ello, primero proponemos una representación en grafos de la secuencia de vídeo y luego diseñamos transformadas no separables direccionales 3-dimensionales (espacio-tiempo). Nuestro codificador puede interpretarse como una extensión de los codificadores de vídeo basados en "wavelets", los cuales operan independientemente (de forma separable) en el dominio espacial y en el temporal. La transformada propuesta consigue mejores resultados (en términos de aproximación no lineal) que un método del estado del arte basado en "wavelets" temporales compensadas en movimiento, y un codificador DCT comparable (base de los últimos estándares de codificación de vídeo).

Para conseguir un codificador de vídeo completo, los coeficientes resultantes de la transformada y la información secundaria (necesaria para obtener un esquema invertible) deben ser codificados entrópicamente y enviados al decodificador. Por ello, también proponemos en esta tesis un método de reordenación de los coeficientes basado en la información del grafo que permite mejorar la capacidad de compresión del codificador entrópico. El esquema de codificación propuesto mejora significativamente la eficiencia de un codificador híbrido basado en DCT en términos de tasa-distorsión. Sin embargo, nuestro método tiene la desventaja de su gran complejidad computacional. Para tratar de paliar este problema, diseñamos dos algoritmos que tratan de reducir dicha complejidad sin que ello afecte en la capacidad de compresión. Finalmente, investigamos como realizar optimización tasa-distorsión sobre el codificador basado en grafos propuesto.

Agradecimientos

Me gustaría escribir estos agradecimientos dejando la literatura a un lado, sin cuidar las palabras demasiado, sin interponer obstáculos entre los pensamientos más primarios y el papel.

En primer lugar quería dar las gracias a mis tutores, Fernando Díaz y Antonio Ortega. A Fernando Díaz, por su confianza en mí, su optimismo y sus ánimos. Sus sabios consejos, tanto técnicos como no técnicos, su generosidad, y la calma que me ha transmitido en épocas tempestuosas, han sido imprescindibles para la realización de esta tesis. A Antonio Ortega por todo lo que he aprendido de él, su genial punto de vista científico/técnico, que ha ayudado a forjar el mío; su inestimable ayuda, y porque, a pesar de nunca tener tiempo para él, siempre lo ha tenido para mí... y finalmente, a ambos, por su gran bondad.

También quiero agradecer especialmente a Jesús Cid por regalarme desinteresadamente su tiempo y sus conocimientos, y por estar siempre dispuesto a debatir sin prisas. Sin su ayuda, parte de esta tesis no hubiera sido posible.

Sin duda, dar las gracias a mis compañeros del Departamento de Teoría de la Señal de la Universidad Carlos III; muy especialmente, a la gente del Grupo de Procesado Multimedia, GPM. Desde la más profunda sinceridad, creo que siempre ha estado formado por personas maravillosas. Entre ellas, mis amigos: Iván González, Sergio Sanz, Manuel de Frutos, Darío García, Sara Pino, Luis Azpicueta, Amanda Garci, Elena Jiménez, Irene Moreno, Óscar del Ama, Ana Belén Mejía, Tomás Martínez, Fernando de la Calle y Fernando Fernández. Citar especialmente a Rubén Solera, por responderme con paciencia infinita una pregunta cada día, a Chelus González de Suso, por su impagable trabajo psicológico en torno a mi persona, y con tremendo amor a mi hermana, Amaya Jiménez, que me lo ha dado todo. También me han ayudado en mi día a día Rosa Mª Barrio, Rocío Arroyo, Carmen Peláez y Ascensión Gallardo, así como Fidela, Encarna, Paqui y Loli.

Fuera del ámbito del trabajo, quería mencionar a mis grandes amigos, los del barrio, los de la escuela (en especial a Ángel Fuertes y Fernando García), y los de mi eterna banda (en especial a Jorge Rodríguez). A todos ellos por aguantarme y darme ánimos en este bonito y duro proceso.

Agradecer también a Mari Ángeles Hernández y Gerardo Casado, por su ayuda en momentos difíciles... sin dicha ayuda, realizar esta tesis hubiera sido una tarea inabarcable.

Agradecimientos especiales a mi familia, por apoyarme incondicionalmente y confiar siempre en mí. Y concretamente a mis padres, por darme tantas oportunidades y por su forma de criarme, repleta siempre de felicidad. A ellos con infinito amor.

Finalmente, quería agradecer a Nico... que aún sin saber hablar, me inspira, y me ayuda a situar las cosas en el lugar adecuado y a darle sentido a la ciencia,

y a María... por su incansable amor, apoyo y comprensión, y por la gran felicidad que me aporta.

Muchas gracias a todos, de corazón.

Table of Contents

List of Figures

List of Tables

Acronyms

AVC	Advanced Video Coding
$\mathcal{C}$	Set-Cover
CDF	Cohen-Deaubechies-Feauveau
DCT	Discrete Cosine Transform
DWT	Discrete Wavelet Transform
HEVC	High Efficiency Video Coding
JPEG	Joint Photographic Experts Group
LIMAT	Lifting-based Invertible Motion Adaptive Transform
LPF	Low-Pass Filter
MA	Moving Average Model
MC	Maximum Cut Problem
MCTF	Motion-Compensated Temporal Filtering
ME	Motion Estimation
MPEG	Moving Picture Experts Group
MRA	Multiresolution Analysis
$m\mathcal{C}$	Minimum-Cardinality Set-Cover

MV	Motion Vector
NM	Noisy Model
RDO	Rate-Distortion Optimization
PSNR	Peak Signal to Noise Ratio
RLE	Run-Length Encoding
SC	Set Covering Problem
SPL	Sound Pressure Level
STM	Spatio-Temporal Model
W	Weight of the Cut
WMC	Weighted Maximum Cut Problem
WSN	Wireless Sensor Networks

Chapter 1

Introduction

There are many applications in which it is useful to achieve a sparse signal representation. Discrete Cosine Transforms (DCT) and "classical" Discrete Wavelets Transforms (DWT) usually obtain an efficient representation of smooth functions, and have been widely used in image and video coding standards (e.g., DCT has been used in JPEG, H.261/H.263, MPEG1/2/4 and H.264/AVC, and DWT in JPEG2000). However, for functions with large discontinuities, good DCT approximations require high energy coefficients in numerous cosine basis functions, and DWT expansions need large magnitude coefficients in the wavelet basis functions, leading to non-sparse representations of the signal.

Figure 1.1 illustrates this behaviour in a one-dimensional signal. In particular, it shows the absolute value of DCT and DWT coefficients of a smooth signal (left column) and a signal with large discontinuities (right column). Note that coefficients are sorted in decreasing order of their absolute value.

The same concept applies to N-dimensional signals such as images or video sequences, where large discontinuities (e.g., contours[1] on images) produce many high energy coefficients. In such cases, *directional transforms* are useful since they allow us to obtain a more sparse representation of multidimensional signals with large discontinuities (note that directional information is a unique feature of multidimensional signals).

In this thesis we describe and optimize a general class of **lifting transforms on graphs** that can be interpreted as **N-dimensional directional transforms**.

This chapter is organized as follows. We first provide some motivation for the construction of new N-dimensional directional transforms in Section 1.1. In Section 1.2 we give a brief overview of some selected state-of-the-art directional transforms which

[1] To avoid confusion we call image "contours" edges that appear in the image, between sets of pixels of different intensities, while we reserve the term "edge", for the links between nodes in a graph.

can be considered the inspiration for this thesis. Given that the proposed transforms are graph-based, we briefly describe graph-representations of N-dimensional signals in Section 1.3 and introduce lifting transforms on general graphs in Section 1.4. This will be useful to describe the starting point and summarize the contributions of this thesis in Section 1.5. Finally, the contents of this thesis are outlined in Section 1.6.

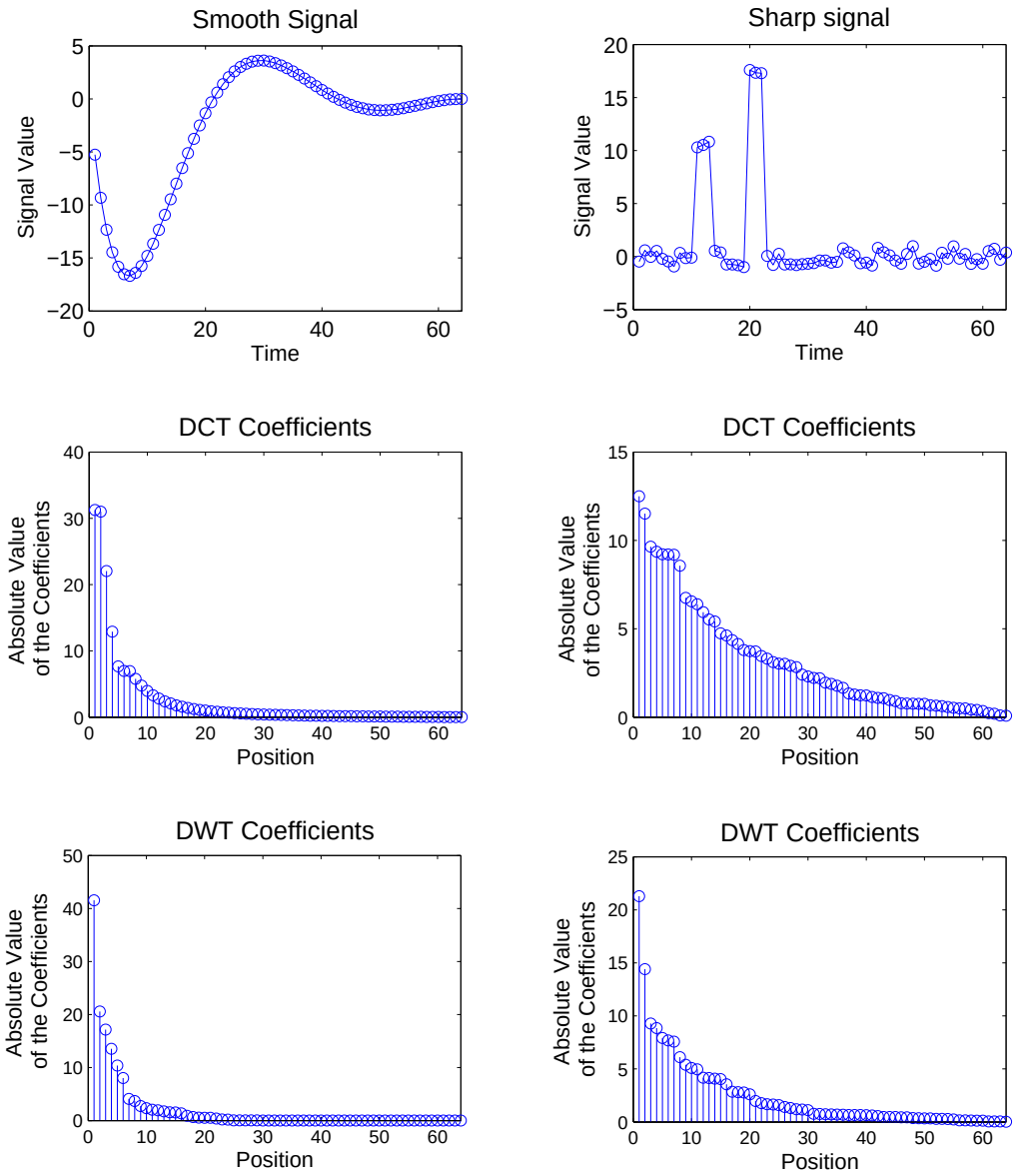


Figure 1.1: DCT and DWT coefficients of a smooth signal (left column) and a signal with large discontinuities (right column).

1.1 Motivation

In several applications such as coding, denoising, or feature extraction, it is useful to achieve a sparse representation of the signal of interest, compacting most of the information into a small number of coefficients (e.g., in an image or video coding application, overall bit rates can be reduced if the selected transform compacts the signal into a smaller number of large coefficients).

Standard separable DWT and DCT transforms have been widely used for compact signal representation. Nevertheless, applying these transforms to signals with large discontinuities (e.g., contours in images) may give rise to many large high-frequency coefficients to represent these discontinuities, thus reducing the sparsity of the representation (e.g., this can be costly in terms of rate in a coding application).

This observation has motivated the interest in *directional transforms*, which are able to adapt their basis functions in order to filter along high-correlation paths (e.g., directions of low variation in pixel intensity in an image or video sequence), avoiding filtering across large discontinuities, and resulting in smaller high frequency coefficients in those locations. In general, the construction of such *directional transforms* is mathematically complex, and their generalization to N-dimensional domains is not easy. Furthermore, most of these transforms are restricted to a given number of feasible directions, limiting their adaptation to the specific signal structure.

When conventional transforms are applied to N-dimensional domains, they usually work in a separable way, filtering independently in each direction (e.g., when conventional wavelets are applied to image processing, they usually operate independently in rows and columns; or when wavelets are applied to video coding, they firstly filter in the temporal domain and then in the spatial one, or vice versa). Some of the directional transforms proposed in the literature are separable too. Separable wavelets (e.g., 2-dimensional wavelets constructed as a separable extension of 1-dimensional bases) have the disadvantage that they do not "see" the smoothness along contours, thus poorly capturing directional information of multidimensional signals.

Finally, it is interesting to have a critically-sampled transform. This means that the transform generates as many coefficients as samples in the original signal, avoiding

redundancy in the representation. Some of the directional transforms proposed in the literature are not critically-sampled (e.g., *Contourlets* [1]).

The motivation of this thesis is to **design and optimize non-separable and critically-sampled graph-based *Directional transforms* constructed by means of the lifting scheme** [2]. Relying on the versatility and simplicity of the lifting scheme and on the graph-based representation of data, we design **directional transforms in N-dimensional domains** that are **intuitive**, **can easy being interpreted**, and that **can adapt their filtering operations to any possible direction**. Finally, we test their performance in a **video coding application**.

1.2 Directional Transforms

The main advantage of directional transforms with respect to conventional ones is their ability to compact the fundamental information of a signal into a smaller number of coefficients [3]. For image coding, directional transforms have been proposed in order to filter following the directions of low variation in pixel intensity, avoiding filtering crossing contours [4], [5]. For video coding, filtering along the motion trajectories, e.g., motion-compensated temporal filtering (MCTF), has been investigated [6], [7], [8]. Side information (e.g., motion vectors or contour locations) is typically transmitted so that the decoder can identify the selected directional transform.

Some of these works [6], [7], [8], [5] are based on the lifting scheme, which is an intuitive and structurally invertible approach to construct multiresolution signal representations [2]. In [6], lifting is applied in the temporal domain, using motion compensated lifting steps to implement the transform. In [5], lifting wavelet transforms in trees are applied to image coding. [9] extends the application of lifting transforms to graphs in the Euclidean space and [10] to general undirected graphs.

These works can be considered as the **starting point and the inspiration for this thesis**. To understand the main contributions of our work (outlined in Section 1.5), in the next sections we provide a brief overview of graph representations of data and lifting transforms on graphs.

1.3 Graph-Based Representation of Data

A graph-based representation of data allows us to generalize standard signal processing operations, such as filtering or transforms, to a broad class of N-dimensional signals [11], [12], [13], [14], [15].

In this way, there are many scenarios in which one can construct a graph which represents N-dimensional signals and that reflects some relationships among data (e.g., correlation, geometric distance or connectivity). For example, in video data each node on the graph can represent a pixel and links between nodes may capture similarity between luminance values; or in Wireless Sensor Networks (WSN), each node on the graph can correspond to a point in Euclidean space containing data read by each sensor and links between nodes can capture the connectivity between sensors. In this thesis, we assume that every node on the graph represents a specific sample point of a discrete signal, while links between nodes capture the correlation between samples.

1.4 Lifting Transforms on Graphs

The design of the proposed directional transforms is based on applying the lifting scheme to general graph signal representations.

The lifting scheme makes it possible to easily construct wavelets and multiresolution signal representations. Basically, lifting on graphs is specified by three main stages, namely: (i) a *split stage*, which finds a bipartition of the graph so that the input data at each specific level of decomposition j is split into *prediction* ($\mathcal{P}_j$) and *update* ($\mathcal{U}_j$) disjoint sets; (ii) a *prediction stage*, where the data of the $\mathcal{P}_j$ set is predicted from the data of the $\mathcal{U}_j$ set using the prediction $\mathbf{p}_j$ filters, yielding the *detail coefficients*; (iii) and an *update stage*, where the data of the $\mathcal{U}_j$ set is filtered with detail coefficients of the $\mathcal{P}_j$ set using the update $\mathbf{u}_j$ filters, giving rise to the *smooth coefficients*.

Transform invertibility is guaranteed for arbitrary $\mathcal{U}_j/\mathcal{P}_j$ disjoint splittings, and $\mathbf{p}_j$ and $\mathbf{u}_j$ filters definitions; thus the lifting scheme is a very versatile way to construct *perfect reconstruction transforms*.

1.5 Contributions

In this thesis we **describe and optimize lifting transforms on graphs**, and, based on these transforms, we present a new **framework that allows to easily generalize the construction of directional transforms to N-dimensional domains**.

We first investigate the graph construction, which leads to a weighted graph in which the filtering directions are defined by means of the links on the graph. Regarding the transform optimization, we investigate new split designs to obtain optimal $\mathcal{U}_j/\mathcal{P}_j$ graph bipartitions (using optimization criteria to minimize the expected energy of the detail coefficients) under two different approaches. Afterwards, we describe a way to design optimal prediction filters $\mathbf{p}_j$ given an arbitrary weighted graph and an $\mathcal{U}_j/\mathcal{P}_j$ bipartition, and considering F different kinds of links between nodes to construct the prediction (e.g., in a video sparse representation, we consider two different kinds of links, related to the spatial and the temporal domains). The update filters $\mathbf{u}_j$ are designed to be orthogonal to the prediction filters of their prediction neighbors as explained in [16]. Once we have constructed a graph, found a bipartition of the graph, and specified the update and predict filters, we have completely defined an N-dimensional directional transform that is invertible and critically-sampled. The optimized lifting transform can be performed in J levels of decomposition to obtain a multiresolution representation of the original signal which compacts the energy in the low-frequency subbands.

In terms of applications, we use these lifting transforms for video coding. First, we construct a graph in which any pixel could be linked to an arbitrary number of spatial and temporal correlated neighbors (i.e., avoiding linking pixels of very different luminance values) thus defining the filtering directions. Next, we apply the proposed transform to this graph.

Initially, our approach can filter following any 3-dimensional direction of the spatio-temporal domain, giving rise to a directional non-separable transform that allows spatial and temporal correlation to be jointly exploited, in contrast to existing techniques in video coding that can be seen as separable transforms. This transform shows improvements in terms of energy compaction ability when compared to the LIMAT method [6] and to a motion compensated DCT-based video encoder. Moreover, our proposed scheme can avoid problems due to occlusions and uncovered areas that appear in the

LIMAT method (and in general in MCTF approaches), leading to a simple critically-sampled invertible transform.

Furthermore, we propose new coefficient reordering techniques, leading to an efficient encoder which improves the performance of a comparable motion compensated DCT video encoder in rate-distortion terms. Given that the graphs created to represent the video information can be very large, we consider two different low-complexity versions of the proposed transform: (i) one that can operate on subgraphs and (ii) another that operates in a distributed manner. Note that the two contributions proposed for video coding, that is, the low complexity approach and the new reordering techniques, are general contributions and can be used in other applications for similar purposes.

Finally, we investigate how to perform rate-distortion optimization in our graph-based video encoder.

Summarizing, our *contributions* are:

1. **Optimization of lifting transforms on graphs:**
   - Description of a general framework to construct N-dimensional directional transforms based on lifting on graphs.
   - Graph weighting.
   - New $\mathcal{U}/\mathcal{P}$ graph-partition techniques based on two different approaches.
   - Novel prediction filter design for lifting transforms on arbitrary graphs.
2. **Video coding application of the transform:**
   - Extension of lifting transforms on graphs to J levels of decomposition.
   - New coefficient reordering technique.
   - Low-complexity versions of the transform.
   - Rate-distortion optimization of the proposed scheme.

1.6 Thesis Outline

This thesis is organized as follows. First, we provide an overview of lifting transforms on graphs and directional transforms in Chapter 2. We then describe our contributions to lifting transforms on graphs and propose a general framework to obtain N-dimensional directional transforms in Chapter 3. In Chapter 4 we apply these transforms to video coding and propose techniques to improve the coding performance and to reduce the computational cost of the transform. Furthermore, we investigate rate-distortion optimization of the encoder. Finally, some concluding remarks and directions for future work are discussed in Chapter 5.

Chapter 2

Overview of Lifting and Directional Transforms

In this chapter we present the necessary background to understand the main contributions of this thesis. These contributions can be summarized as: (i) optimization of **lifting transforms on graphs**; (ii) proposal of a new framework to construct **N-dimensional directional transforms** based on **lifting on graphs**; and (iii) **application** of this kind of transforms **to video coding**, leading to 3-dimensional directional transforms. Therefore, the overview is focused on lifting transforms on graphs and directional transforms.

In Section 2.1 we present the **lifting scheme** [2], which allows to easily obtain a multiresolution analysis (MRA) of a given signal. In every step[1] j of the transform, lifting provides subsampled low-pass (*smooth coefficients*) and high-pass (*detail coefficients*) versions of the signal at the immediately lower level $j-1$. If detail coefficients are close to zero, the main information of the signal is kept in the smooth coefficients, thus obtaining a more compact representation. Applying this process iteratively leads to a MRA [17] of the original signal. The lifting scheme can be interpreted as a cascade of filter banks, or as the projection of the signal at level $j-1$ onto the approximation ($\mathbf{V}_j$) and detail ($\mathbf{W}_j$) subspaces at level j. Section 2.2 is devoted to the construction of lifting transforms on arbitrary graphs [10], where every sample can have a different number of neighbors and, thus, the subsampling and filtering operations become complicated.

Finally, Section 2.3 gives an overview and describes the main properties of some **directional transforms** presented in the literature. In this way, we will be able to understand the general features and the main advantages of the proposed directional transforms as compared to the ones of the state of the art.

[1] Step, level of decomposition, or resolution refer to the same concept.

2.1 Lifting Transforms

In this section we describe the lifting scheme from a practical and intuitive point of view [18]. Details about the relation between lifting, second generation wavelets, and MRA can be found in [2], [19] and [20].

Given a digital signal at a specific resolution, the lifting transform can lead to a compact representation of this signal with some interesting properties.

Consider a signal s_{j-1} at resolution (i.e., level of decomposition) $j-1$ which we would like to transform into a coarser[2] signal s_j and a detail signal d_j. This can be easily achieved applying the lifting transform. It consists of three stages: split, predict and update.

- **Split**[3]: This stage basically consists on splitting the signal into two disjoint sets of samples that we will call *Predict* ($\mathcal{P}_j$) and *Update* ($\mathcal{U}_j$) sets throughout this thesis.

- **Predict**: The predict stage aims to remove the local correlation of the signal. To do that, each sample of the $\mathcal{P}_j$ set is predicted from samples of the $\mathcal{U}_j$ set. If the local correlation of the signal is high, it should be possible to obtain a **reasonably accurate prediction**, giving rise to small residual information (small **detail coefficients**) and, therefore, **compaction of the information**. We will characterize this stage with the predict filter **p**.

- **Update**: The Update stage can be designed with different objectives in mind. One of these objectives could be to keep the average value of the coefficients across multiple levels of decomposition, thus reducing the aliasing and obtaining a better frequency localization [18]. Another goal would be to design the update stage so that low-pass and high-pass equivalent filters[4] are orthogonal, which would

[2] In this thesis we follow the convention of using smaller j indices to represent finer approximations, as in Mallat's book [21] and as opposite to other works as Mallat's MRA original paper [17] or Sweldens's papers.

[3] Note that the split stage of the transform will be referred to as $\mathcal{U}/\mathcal{P}$ splitting, $\mathcal{U}/\mathcal{P}$ assignment or graph bipartition problem throughout this thesis.

[4] We refer to the filtering that results from the predict and update stages as equivalent filters. It can be seen that, generally, the equivalent filter of the predict stage is a *high-pass filter* (giving rise to *detail coefficients*), and the equivalent filter of the update stage is a *low-pass filter* (giving rise to *smooth coefficients*).

minimize the reconstruction distortion due to quantization of the transform coefficients [22]. In the update stage, data at nodes $u \in \mathcal{U}_j$ are filtered using detail coefficients of the $\mathcal{P}_j$ set, leading to the **smooth coefficients**. We will characterize this stage with the update filter **u**.

Following these stages, the data s_{j-1} at level of decomposition $j-1$ should be split into prediction ($\mathcal{P}_j$) and update ($\mathcal{U}_j$) disjoint sets (**split stage**), and the predict ($\mathbf{p}_{m,j}(m \in \mathcal{P}_j)$) and update ($\mathbf{u}_{n,j}(n \in \mathcal{U}_j)$) filters should be specified. Then, the m-th detail coefficient at level j, $d_{m,j}$, can be computed from $h \in \mathcal{U}_j$ update neighbors using the predict filter (**predict stage**), and the n-th smooth coefficient $s_{n,j}$ can be computed from $l \in \mathcal{P}_j$ prediction neighbors using the update filter (**update stage**). Mathematically, we can write:

$$\begin{aligned} d_{m,j} &= s_{m,j-1} - \sum_{h \in \mathcal{U}_j} \mathbf{p}_{m,j}(h) s_{h,j-1}, \\ s_{n,j} &= s_{n,j-1} + \sum_{l \in \mathcal{P}_j} \mathbf{u}_{n,j}(l) d_{l,j}. \end{aligned} \tag{2.1}$$

This way, the smooth coefficients at the $(j-1)$-th decomposition level (s_{j-1}) are projected onto the approximation ($\mathbf{V}_j$) and detail ($\mathbf{W}_j$) subspaces, yielding, respectively, the smooth ($s_{n,j}$) and detail ($d_{m,j}$) coefficients at the j-th decomposition level. Applying this process iteratively gives rise to a multiresolution decomposition. In Figure 2.1 the lifting structure for two levels of decomposition is illustrated, where data at level $j = 0$ is the original raw data, denoted as x_k (i.e., $x_k = s_{n,j=0}$).

Therefore, given a signal at scale j_0, a lifting transform representation of this signal is composed of detail coefficients at scales $j_0 < j \leq J$, plus the smooth coefficients at the largest scale J:

$$\left[d_{j_{(j_0 < j \leq J)}}, s_J \right]. \tag{2.2}$$

Generally, this transformation can be interpreted as a filter bank decomposition, where the s_j coefficients are the low-pass version (smooth projection) and the d_j coefficients the high-pass version (detail projection) of coefficients s_{j-1} at $(j-1)$-th level.

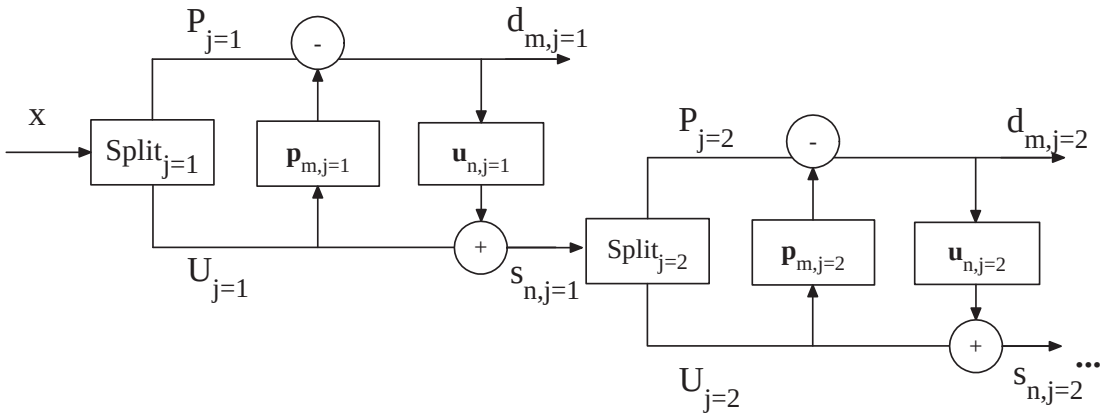


Figure 2.1: Lifting scheme. Two levels of decomposition of the forward transform.

Figure 2.2 shows an illustrative example of the application of the lifting transform to a 1-dimensional signal. The left column shows the corresponding stages of the transform at decomposition level $j = 1$, and right column at $j = 2$. Finally, the last row of the figure shows the subband decomposition for each level. Note that in the example, the splitting process is trivial: every odd (resp. even) sample belongs to the $\mathcal{U}$ (resp. $\mathcal{P}$) set at each level of decomposition. Then, every sample $p \in \mathcal{P}$ is predicted from its two adjacent $\mathcal{U}$ neighbors, and every sample $u \in \mathcal{U}$ is updated from its two adjacent $\mathcal{P}$ neighbors. The signal at level $j = 2$ is composed of the resulting $\mathcal{U}$ samples at $j = 1$, and $\mathcal{U}_1 = \mathcal{U}_2 \cup \mathcal{P}_2$. If one chooses the filters $\mathbf{p}$ so that, at each level j, every detail coefficient $m \in \mathcal{P}$ is calculated as the difference between the value of node m and the average of its $h \in \mathcal{U}$ adjacent neighbors ($\mathbf{p}_m(h = m - 1) = \mathbf{p}_m(h = m + 1) = 1/2$), and $\mathbf{u}$ so that for every update coefficient $n \in \mathcal{U}$, $\mathbf{u}_n(l = n-1) = \mathbf{u}_n(l = n+1) = 1/4$, this scheme leads to the $5/3$ biorthogonal wavelet transform of Cohen-Deaubechies-Feauveau (CDF) [23], [18].

One of the main advantages of the lifting scheme is that the inverse transform is immediately obtained by inverting the operations of the forward transform. Again, we have three stages:

- **Undo Update**: Given $d_{m,j}$ and $s_{n,j}$, we can recover the **update** $s_{n,j-1}$ samples simply subtracting the update information.

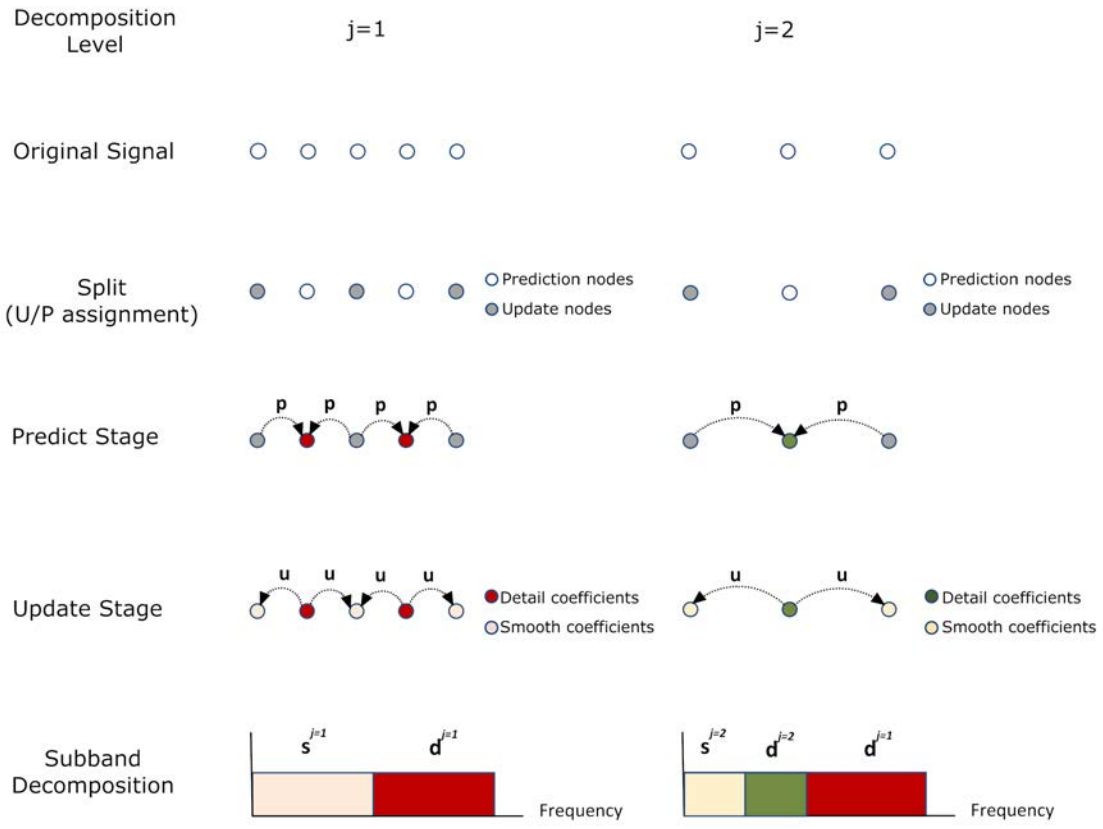


Figure 2.2: Example of the lifting scheme applied to a 1-dimensional signal.

- **Undo Predict**: Given $d_{m,j}$ and $s_{n,j-1}$, we can recover the **predict** $s_{m,j-1}$ samples subtracting the predict information.
- **Merge**: Given the update and predict samples obtained before, we put them together in their corresponding location to recover the original signal.

Mathematically, we can write:

$$\begin{aligned} s_{n,j-1} &= s_{n,j} - \sum_{l \in P_j} \mathbf{u}_{n,j}(l) d_{l,j}, \\ s_{m,j-1} &= d_{m,j} + \sum_{h \in U_j} \mathbf{p}_{m,j}(h) s_{h,j-1}. \end{aligned} \tag{2.3}$$

An inverse transform with two levels is illustrated in Figure 2.3.

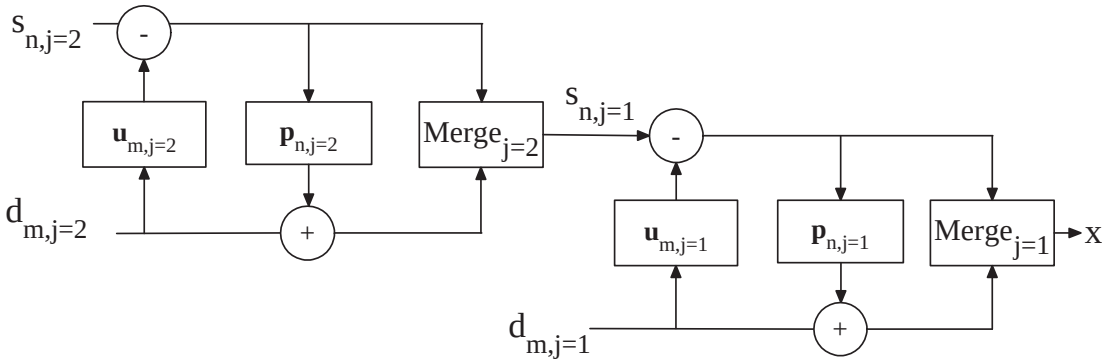


Figure 2.3: Lifting scheme. Two levels of decomposition of the inverse transform.

Finally, Figure 2.4 shows the multiresolution approximations $s_j[n]$ (left column) and details $d_j[m]$ (right column) of a specific discrete signal, computed with the $5/3$ lifting transform. Three levels of decomposition are shown.

Some remarks extracted from this example can be good for summarize the previously explained concepts:

- ***Smooth* coefficients** at a coarse level j, $s_j[n]$, can be obtained by low-pass filtering and subsampling the smooth coefficients of a finer level $j-1$, $s_{j-1}[n]$ (2.1). Therefore, as illustrated in the left column of Figure 2.4, each level j is a low-pass version of the approximation at level $j-1$ and contains half of the samples. Furthermore, the smooth coefficients at level of decomposition j ($s_j[n]$) characterize the projection of the original signal into the approximation subspaces $\mathbf{V}_j$.

- ***Detail* coefficients** at a coarse level j, $d_j[m]$, can be obtained by high-pass filtering and subsampling the smooth coefficients of a finer level $j-1$, $s_{j-1}[n]$ (2.1) . Detail coefficients at level j characterize the projection of the original signal into the detail subspaces $\mathbf{W}_j$. See right column of Figure 2.4.

- Given that filtering operations are local, coefficients are localized in time. Note for example that the position of large magnitude detail (high-frequency) coefficients $d_{m,j=1}$ are in agreement with positions of large variation in the original signal. Frequency localization is given by the cascade of low-pass and high-pass filters.

- Detail coefficients d_j can be thought of as the difference between consecutive approximations of the signal at resolutions $j-1$ and j.

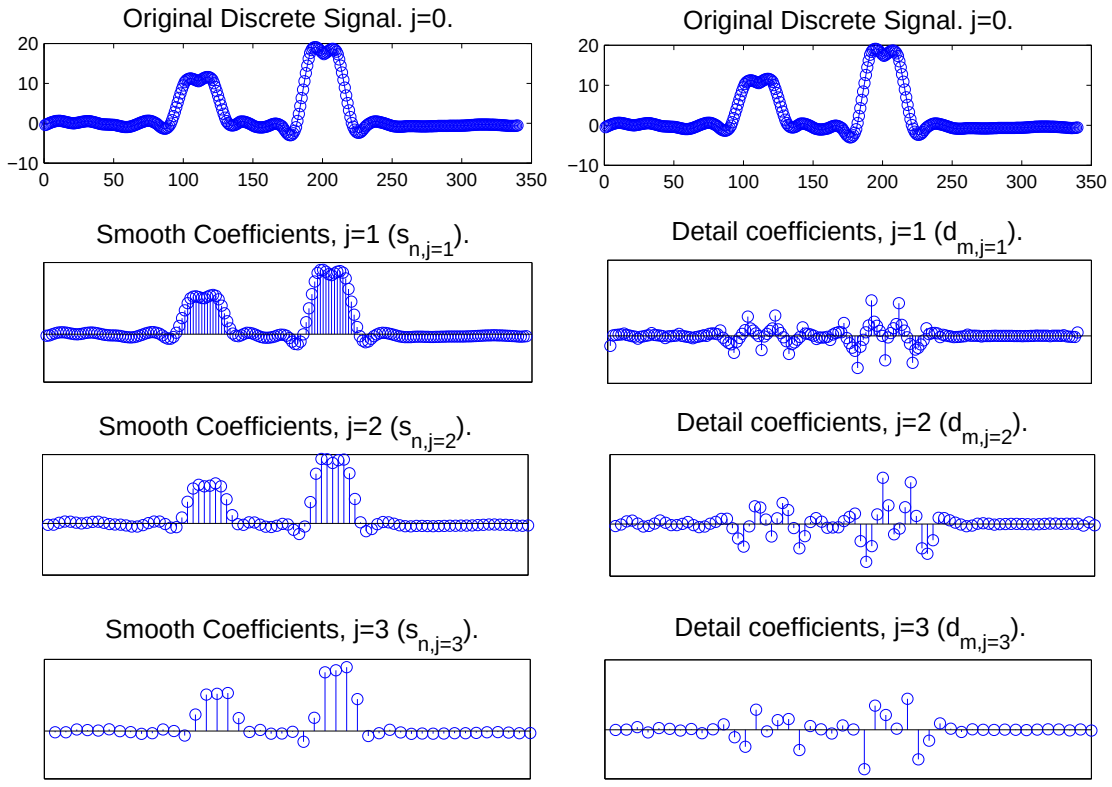


Figure 2.4: Smooth and detail coefficients.

- The lifting representation of the signal is $\left[d_{j\,(j_0<j\leq J)}, s_J\right]$, where J is the coarsest considered level. Thus, in the example of Figure 2.4 in which three levels of decomposition are shown, $[d_{j=1}, s_{j=1}]$, $[d_{j=1,2}, s_{j=2}]$, and $[d_{j=1,2,3}, s_{j=3}]$ are perfect reconstruction and critically sampled representations.

- Note that the sparse representation of the signal is obtained because coefficients are nearly zero at fine scales (high-pass subbands). In fact, where the signal is locally smooth, there will be high correlation between low-pass coefficients at different levels of the transform (i.e., coefficients will be nearly zero at various subbands). Therefore, in the example, it will usually be more efficient to transmit $[d_{j=1,2,3}, s_{j=3}]$, where three levels of the transform have been decorrelated, than any other of the above representations (e.g., $[d_{j=1,2}, s_{j=2}]$).

- One can interpret the process shown in the example as the decomposition of the original signal in a set of independent frequency channels or the projection of the original signal on successive smooth and detail subspaces.

Thanks to these interesting features of the lifting scheme, it has been used in many applications achieving a very good performance, e.g., image and video compression [24], [25], [5], [26], [6], [27], data denoising [10], distributed data gathering [28], [29], [30], feature extraction in brain-computer interface [31], [32], or audio coding [33].

2.2 Lifting Transforms on Graphs

Lifting transforms on arbitrary graphs were initially proposed by [9] and [10]. The main concepts are those mentioned above, and the stages and equations of Section 2.1 hold. Nevertheless, some important peculiarities and open questions arise. This section presents these peculiarities, which will be handled in this thesis.

Assume we are given an arbitrary undirected graph. At this point, we do not make assumptions about the structure of the graph or the links between nodes, so that every node can have a different number of neighbors.

If $\mathcal{P}_j$ and $\mathcal{U}_j$ are **disjoint sets**, the lifting transform on the graph is critically sampled (i.e., the number of samples of the original signal $\mathbf{x} = s_{j=0}$ is equal to the number of coefficients of any decomposition $\left[d_{j_{(j=0<j\leq J)}}, s_J\right]$). Besides, [22] showed that if $\mathcal{P}_j$ and $\mathcal{U}_j$ are **disjoint sets**, lifting transforms on graphs are invertible by construction. Thus, **arbitrary $\mathcal{P}_j/\mathcal{U}_j$ disjoint splittings and p and u filter designs can be used without compromising the perfect reconstruction and critically sampled properties** of the transform. This implies that lifting transforms on graphs are very flexible (in the sense that they can operate with different splittings and filters), which is an important feature to design the proposed N-dimensional directional transforms, but, at the same time, they open some questions that need be solved in order to obtain an efficient transformation. Some of them are:

- **How should the graphs be constructed to capture the correlation of the signal?**

One has total freedom to construct the graph representation of a signal deciding which nodes should be linked to which ones. Therefore, a signal can give rise to different graph representations, and as a function of the selected representation, the correlation between samples can be better or worse captured.

The graph construction will influence the compaction ability of the transform as well as the support of the filters (higher number of neighbors, higher support), which affects the overlapping of the filtering operations. Besides, the graph construction will influence the hop length of the filters (one node can be linked to n-hops neighbors), and thus the localization of the transform in frequency and in the original domain (i.e., spatial or temporal).

This question is discussed in Section 3.1.

- **How should the $\mathcal{U}/\mathcal{P}$ splitting be performed?**

  The $\mathcal{U}/\mathcal{P}$ splitting process requires assigning a label ($\mathcal{U}$ or $\mathcal{P}$) to each node of the graph, which is usually trivial when one works with 1-dimensional signals (as in the example of Figure 2.2) or when working with regular grids in which the local topology of the graph is the same around each vertex (same number of neighbors, same relative position for the neighbors), such as the quincunx grid [21]. Nevertheless, it becomes a complex problem in arbitrary graphs, in which every node can have a different number (and location) of neighbors. Besides, due to the arbitrary structure of the graph, it may not be possible to assign a different label to each pair of connected nodes in the graph[5]. Given that $\mathcal{P}$ nodes (resp. $\mathcal{U}$ nodes) are predicted (resp. updated) from $\mathcal{U}$ neighbors (resp. $\mathcal{P}$ neighbors), links between same-label neighbors are not useful to perform the transform, and are discarded.

  Some properties of the transformation, such as the energy compaction ability, depend on this assignation. The prediction residuals and thus the detail coefficients $d_{m,j}$ will be small if $\mathcal{U}$ and $\mathcal{P}$ are chosen so that they are correlated, thus obtaining a compact representation of the underlying signal.

[5] In graph theory literature, this is a graph coloring problem, and the sentence means that the graph may not be 2-colorable.

The formulation of this problem and some proposed $\mathcal{U}/\mathcal{P}$ splitting techniques are described in Sections 3.2 and 3.3.

- **How should the p and u filters be defined?**

  p and **u** filters should be designed for each node of the graph taking into account that each $\mathcal{P}$ (resp. $\mathcal{U}$) node could be predicted (resp. updated) from a different number of $\mathcal{U}$ (resp. $\mathcal{P}$) neighbors, in contrast to standard lifting approaches as the $5/3$ CDF or the quincunx wavelets [21].

  The compaction ability of the transform also depends on the design of **p** filters because as the predictors are better, the average energy of detail coefficients $d_{m,j}$ is smaller; **u** filters can be designed to be orthogonal to the arbitrary number of **p** filters of $\mathcal{P}$ neighbors. Besides, given a graph, the support of the filters depends on their definition (higher number of nodes used in the filters, higher support).

  This problem is studied in Section 3.4.

- **How should the graphs be constructed at decomposition levels $j > 1$?**

  Once we have the graph at level of decomposition $j = 1$, we should decide how to construct the graph at subsequent levels of decomposition in order to retain the correlation between linked samples at different levels and thus further decorrelate the signal. Again, this operation is not difficult in standard lifting approaches as $5/3$ or quincunx, but becomes harder in arbitrary graphs.

  One approach to construct the graph at decomposition levels $j > 1$ is proposed in Section 4.1.3.

Finally, note that as a function of the graph construction, the selection of the $\mathcal{U}/\mathcal{P}$ splitting, and the design of **p** and **u** filters, the transform leads to different equivalent low-pass and high-pass filters, which determine the subband representation obtained.

Figure 2.5 shows an example of the application of lifting transform on graphs. Left column shows the corresponding stages of the transform at decomposition level $j = 1$, and right column at $j = 2$. Note that in the example, the splitting process is not trivial, because every node of the original graph at $j = 1$ and at $j = 2$ can have a different number of neighbors. Therefore, every sample $m \in \mathcal{P}$ is predicted from an arbitrary

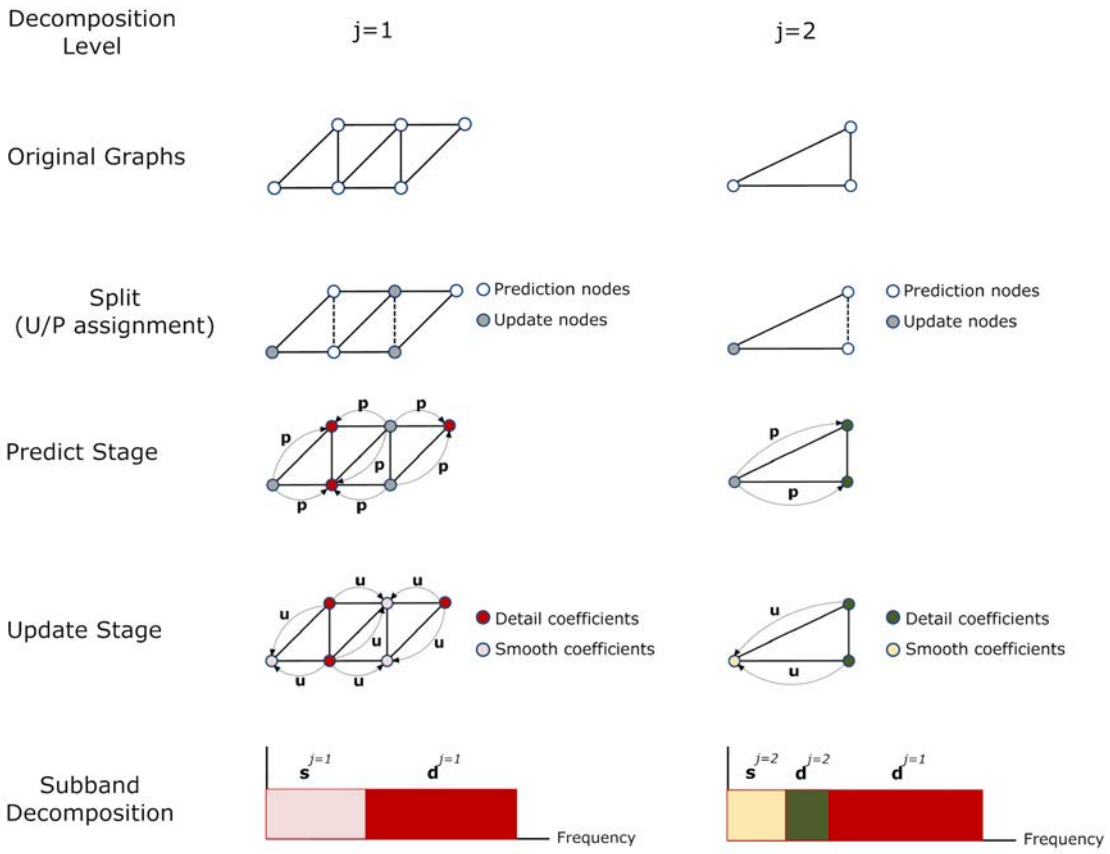


Figure 2.5: Example of the lifting scheme applied to a graph. Discarded links are indicated with dashed lines in the split process.

number of $\mathcal{U}$ neighbors, and every sample $j \in \mathcal{U}$ is updated from its $\mathcal{P}$ neighbors. Note that the signal at $j = 2$ level consists of the $\mathcal{U}$ samples at $j = 1$, and $\mathcal{U}_1 = \mathcal{U}_2 \cup \mathcal{P}_2$. Nevertheless, one should decide how to link nodes at $j = 2$ to construct the graph at this level using the information of the graph at $j = 1$.

Finally, we summarize some important ideas explained so far:

- If $\mathcal{P}_j$ and $\mathcal{U}_j$ are **disjoint sets**, lifting transforms are invertible and critically-sampled by construction. Thus, arbitrary $\mathcal{P}_j/\mathcal{U}_j$ disjoint splittings and **p** and **u** filter designs can be used, which implies that lifting transforms are highly versatile.

- Lifting transforms *decorrelate* the data, obtaining a more compact representation than the original one. This means than we can obtain an accurate approximation of the original signal by only using a small fraction of coefficients.

- Lifting transforms can be designed to be localized in the original domain (i.e., time or space) by building filters with compact support (e.g., only allowing nodes to use neighbor data in the filtering operations), what implies that the filtering operations are local.

- Generally, lifting decomposition gives rise to a MRA and thus can be interpreted as a filter bank decomposition, where the s_j coefficients are the low-pass version and the d_j coefficients the high-pass version of the signal s_{j-1}.

- The frequency localization comes from the interpretation of the transform as, on the one hand, the low-pass filtered and downsampled version of the signal (*smooth* coefficients) and, on the other hand, the band-pass filtered and downsampled version of the signal (*detail* coefficients).

2.3 Directional Transforms

Directional transforms can filter along directional paths, in order to avoid crossing large discontinuities, which leads to a sparser representation of the original signal than that obtained with non-directional transforms. This would be useful in several applications such as coding, denoising or feature extraction.

In general, directional transforms can be classified into adaptive (i.e., which use knowledge of the intrinsic structure of the object and adapts the basis to that structure) or non-adaptive (i.e., the representation is constructed without using the knowledge of the underlying object). Note that, in some cases, adaptation requires side information (overhead).

2.3.1 Directional Transforms for Sparse Image Representation

Candès and Donoho [3] quantified how well different transforms compact the energy of a function $f \in \mathbb{R}^2$, which has a discontinuity and which is otherwise smooth, into

a few coefficients (i.e., the performance of different expansions from an asymptotic point of view). They show that wavelets outperform Fourier representations and that directional approaches are better than wavelets. Furthermore, they conclude that non-adaptive methods can achieve similar performance to that of adaptive methods, and propose a non-adaptive transform called *Curvelet*. To be more precise and justify the use of directional transforms, we present their results below.

Suppose that there is an object supported in $[0,1]^2$, which has a discontinuity across a curve Γ, and which is otherwise smooth. If one approximates f with $\tilde{f}$ built from the best m non zero coefficients using different transforms, one obtains:

- Fourier Representation: $\left\|f-\tilde{f}\right\|^2 \approx m^{-1/2}, m \to \infty$
- Wavelet Representation: $\left\|f-\tilde{f}\right\|^2 \approx m^{-1}, m \to \infty$
- Adaptive method: $\left\|f-\tilde{f}\right\|^2 \approx m^{-2}, m \to \infty$
- Non-adaptive Curvelets: $\left\|f-\tilde{f}\right\|^2 \leq Cm^{-2}(\log m)^3, m \to \infty$.

The quadratic error between the original function f and the reconstruction $\tilde{f}$ as a function of the number of coefficients m decays faster as the α value in $m^{-\alpha}$ is higher. Therefore, directional methods (adaptive or non-adaptive) can reconstruct the original signal with the same quality than wavelets and Fourier transforms using less coefficients.

In [1] Do and Vetterli proposed another non-adaptive transform, the *Contourlet* transform. Unlike the *Curvelet*, *Contourlet* transforms work directly in the discrete domain. *Contourlets* have elongated supports at various scales, directions and aspects ratios, allowing to efficiently approximate a smooth contour at multiple resolutions (see Figure 2.6). *Contourlet* is a non-separable (i.e., it performs non-separable filter operations) and non-critically sampled (it has a redundancy of about 33%) transform. Some other directional wavelets proposed in the literature for image processing are adaptive methods and thus are based on information of the underlying object (e.g., the contours).

Candès and Donoho non-adaptive *Curvelets* achieved a significant improvement over wavelets for typical images with smooth contours and similar performance compared to adaptive methods. Nevertheless, Le Pennec and Mallat observed [34] that the

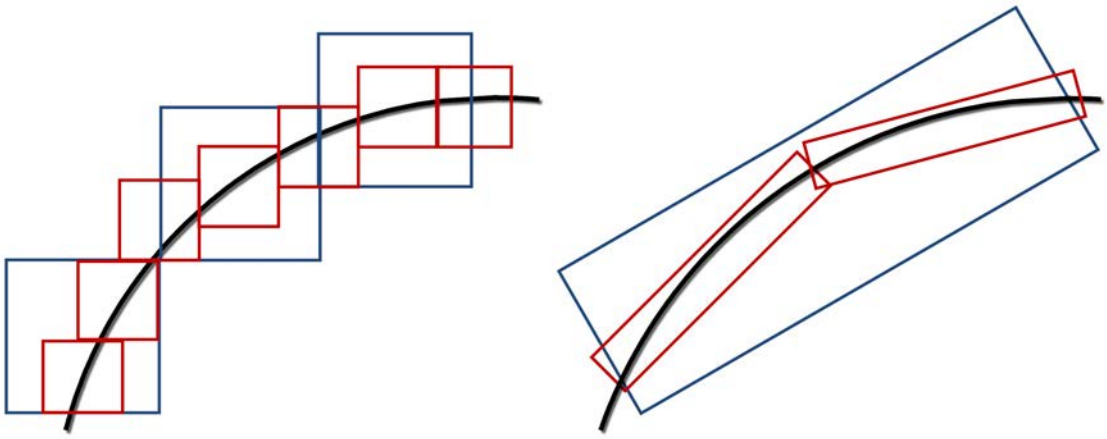


Figure 2.6: Wavelets (left side) and *Contourlets* (right side) support at two different resolutions.

curvelets approximation lose their near optimal properties if contours are along irregular curves of finite length. The authors presented the so-called *Bandelets* (mathematical details appear in [35]) and applied them to image compression and noise reduction. The *Bandelets* are adaptive wavelet basis that are warped along the geometric flow. For their construction, the authors partition the image into small square regions so that each region includes at most one contour. Then, bandelets are constructed in those regions by warping separable wavelet basis so that they follow the lines of the geometric flow, taking advantage of the regularity along it.

Velisavljevic *et al.* proposed the *Directonlets* transforms in [4]. *Directionlets* are critically sampled, perfect reconstruction and discrete transforms that retain the separable filtering, subsampling, computations and filter design from the standard 2-dimensional wavelet transform. They are "separable" transforms based on independent operations on one dimensional wavelets, but allowing directionality and anisotropy. Additionally, directional DCT basis for image coding have been proposed in [36].

The transforms that we propose in this thesis are perfect reconstruction and critically-sampled, which are generally desirable properties. They operate in N-dimensional domains by construction (i.e., without needing complex operations to be generalized), and perform non-separable filtering operations following directions of high correlation. Besides, our proposed transform provides a great freedom to choose these directions, which

are defined by means of the links between nodes on the graph. Finally, another important property is that our proposal is useful for irregularly spaced sample grids.

As was explained in Section 2.1, the lifting approach allows us to construct wavelets adapted to the domain in a simple way. Therefore, some directional transforms for image processing based on lifting have been proposed in the literature. [37] and [38] incorporated adaptivity via lifting by choosing the prediction filter **p** based on the local properties of the image. Wavelets with large support generally work very well away from the contours, exhibiting a fast decay of the coefficients value. Nevertheless, this large support leads to a larger set of coefficients affected by the contours. Therefore, the authors basically proposed to choose larger predictors (which correspond to smoother basis functions) away from the contours and to reduce the order of the predictor (and thus the support) near the contours, so that the neighborhood they used to predict never overlaps the contour.

Similarly, [26] proposed adaptive non-separable lifting transforms for image compression which use prediction filters that are sensitive to directional information, exploiting local orientation at contour boundaries. Another example of compact image representation using lifting was proposed in [5]. The key novelty in this paper is that the authors define critically sampled separable transforms that operate in arbitrary trees. These trees can be constructed to follow the geometric flow of an image, capturing the directional information and thus obtaining a directional transform.

The proposed transforms can be considered a generalization of these previous works. By means of the graph construction and the filter design, our transforms allow to choose any predictors length at any point of the N-dimensional domain. These predictors perform non-separable filtering operations.

2.3.2 Directional Transforms for Sparse Video Representation

Directional transforms for video representation are usually constructed via lifting, which is applied in the temporal domain. The main multiresolution decomposition structures in wavelet-based video coding are referred to as "$t+2D$" and "$2D+t$". In the former, the video sequence is first filtered in the temporal direction along the motion trajectories (MCTF) and then a 2-dimensional wavelet transform is carried out in the spatial

domain [39]. In the latter, each frame is first wavelet transformed in the spatial domain, followed by MCTF. Focusing on the temporal domain, representative examples of MCTF implementations are [6] and [7], which use motion-compensated lifting steps to implement the temporal wavelet transform, filtering along a set of motion trajectories described by a specific motion model. These approaches can be described as "separable" because spatial and temporal filtering are applied in separate steps. Side information (e.g., motion vectors) is typically transmitted so that the decoder can identify the directional transform that was selected. Therefore, we can consider these approaches as adaptive methods.

In all of these works, in order to perform the prediction and update steps of the lifting scheme, the input sequence is split into update (even frames) and prediction (odd frames) subsequences (see Figure 2.7), and for each level of the transform, the prediction subsequence is predicted from the update subsequence giving rise to the high-pass subband sequence, and the update subsequence is updated by using a filtered version of the prediction one, thus obtaining the low-pass subband sequence. In cases in which the motion model cannot accurately capture the real motion of the scene, this kind of splitting into even and odd frames will lead to the linking of update and prediction pixels with very different luminance values. In this way, prediction frames will be poorly predicted from update frames, leading to significant energy in the high pass subband sequence, and thus relatively low energy compaction. Moreover, when using MCTF, problems arise due to occlusions and uncovered areas (pixels that are filtered several times or are not filtered at all). Some authors handle this problem by identifying unconnected and multiple connected pixels and adapting the predict and update operators accordingly (e.g., [27]).

Finally, graph-based transforms are used to coding depth maps for view synthesis in multi-view video coding in [40] and [41].

When we apply the proposed transforms to video coding, they generalize wavelets approaches, which usually work in a separable way (first in the spatial and then in the temporal domain or vice versa). Thanks to the versatility of the proposed scheme, $\mathcal{U}$ and $\mathcal{P}$ nodes and filters can be arbitrarily chosen, solving some problems that arise in the MCTF approaches, such as those described above, or the needed of several frames

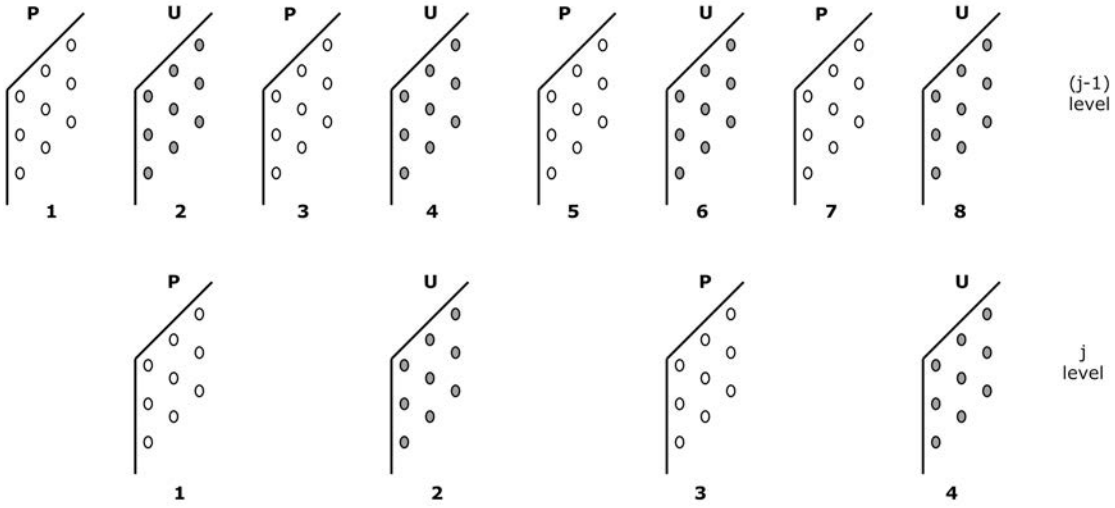


Figure 2.7: Update-Predict assignment in typical MCTF approaches.

to obtain a MRA. Furthermore, this versatility allows the transform to adapt to the video content, thus improving its performance.

Chapter 3

Lifting Transforms on Graphs

In order to perform lifting transforms on graphs one needs three essential elements: (i) a given **graph**, (ii) an $\mathcal{U}/\mathcal{P}$ **splitting**, and (iii) a **definition of p and u filters**. In this chapter we discuss the design and optimization of these three elements.

We first focus on the graph construction, explaining how to obtain a **graph representation of N-dimensional signals including directional information** in order to adapt the filtering operations to the domain. Then, we discuss the $\mathcal{U}/\mathcal{P}$ splitting and the **p** and **u** filter construction considering **a given arbitrary graph**, obtaining general results on the design and optimization of **lifting transforms on arbitrary undirected graphs** that represent a generic signal. At that point, we will have all the necessary ingredients to obtain **N-dimensional directional transforms** based on lifting transforms on graphs with some particular properties that will be discussed at the end of the chapter.

A correct graph construction is crucial to obtain an efficient transform. Intuitively, if linked nodes are not correlated, the prediction of a node from its neighbors will usually be inaccurate, leading to large detail coefficients and thus, low energy compaction. Besides, it will be useful to weight the links of the graph in order to capture the different correlations that exist between linked nodes. This is discussed in Section 3.1.

Lifting transforms on graphs are invertible for any $\mathcal{U}/\mathcal{P}$ disjoint splitting. Nevertheless, the appropriate choice of these sets on arbitrary graphs is not an easy problem, and greatly influence the performance and properties of the transform (e.g., if the chosen $\mathcal{U}$ and $\mathcal{P}$ sets are not correlated, the prediction of $\mathcal{P}$ from $\mathcal{U}$ will be inaccurate). Our goal when performing the $\mathcal{U}/\mathcal{P}$ assignment is to obtain a sparse representation of the original signal, so that we search partitions that minimize the detail coefficient energy. With this goal in mind, various $\mathcal{U}/\mathcal{P}$ assignment procedures based on different philosophies are discussed, namely: (i) techniques that only depend on the weighted graph previously defined (Section 3.2), (ii) techniques that assume a signal model and a predictor and minimize the expected value of the quadratic prediction error (Section 3.3).

The filter design is studied in Section 3.4. Given an arbitrary weighted graph and an $\mathcal{U}/\mathcal{P}$ splitting, the prediction **p** filters are designed as a function of the weights of the graph. If these weights were "correctly" chosen in the weighting process, the prediction filters will be close to minimize the detail coefficient energy. We also briefly described the update **u** filters used in this thesis, which are designed to make the low-pass and high-pass filters orthogonal [16].

3.1 Graph construction

This section discusses the graph construction, which includes the graph-based signal representation (that defines the links between nodes -samples- of a generic signal), and the graph weighting (which tries to characterize the correlation between different connected nodes). Section 3.1.1 presents the definition of graph-based signal representation, and shows two illustrative examples of video and N-channel audio graph representations. Note that there are no restrictions associated with the graph construction, so that any node could be linked to an arbitrary set of nodes, and therefore the graph representation of a signal is not unique. The graph weighting affects different processes of the transform, and therefore, its performance. Section 3.1.2 proposes two different approaches to weight the graph.

Given that the filtering operations are performed using neighboring (linked) nodes, the graph representation defines the filtering directions, and the weighting process allows to give, more or less "importance", to the different defined directions. The graph construction at levels of the transform $j > 1$ will be discussed in Chapter 4.

3.1.1 Graph-Based Representation of a Generic Signal

Graph-based representation of data allows us to generalize standard signal processing operations to a broad class of signals. In this thesis we focus on the lifting transform, which can be defined on arbitrary graphs. As discussed in Chapter 2, some properties of the transform depend on the suitable construction of the graph, in the sense of accurately representing correlation between signal samples.

Definition 3.1. Graph-based signal representation of a signal

Let $\mathbf{x} = \{x_k\}_{k=1}^{M}$ be an N-dimensional digital signal sampled in an N-dimensional grid. Assume that data is organized in a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \ldots, M\}$ is a set of ordered nodes associated with the samples $\{x_k\}_{k=1}^{M}$, and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ is a set of edges between nodes that in some way represent the correlation between them (i.e., nodes a and b are linked if they are correlated). Note that node $a \in \mathcal{V}$, associated to x_a, can be linked to any subset of nodes $\mathcal{F} \subset \{\mathcal{V} \setminus \{a\}\}$ without restrictions (i.e., graph can represent correlations between multiple nodes in the different domains). This leads to a graph-based signal representation (or for short a graph representation) of $\mathbf{x}$.

Observe that there exist several graph representations of the same **x** depending on the way in which the correlation between nodes is defined. Therefore, the main challenge when constructing the graph representation of a signal **x** is how to link the nodes of the graph in order to accurately capture the correlation between samples. As a first approximation to construct the graph, one could link all the pixels together (all-connected graph) and remove those links between nodes with very different signal values. Another approach could consist on making some assumptions about the correlation between nodes. For example, it seems reasonable that closer data in a WSN, or in an image, are more correlated, so that one could decide to link together closer nodes (i.e., one-hop or in general n-hop neighbors) and not link the farther ones.

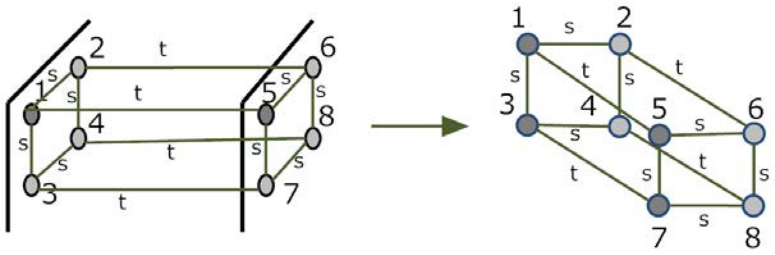


Figure 3.1: Graph representation of video data.

In a coding application, the encoder should send some side information to allow the decoder to correctly construct the same graph. Therefore, a trade-off exists between accuracy in the graph description and side information to be sent. Next, two different graph representation examples are described, namely, the graph representation of a video signal, and of an N-channel audio signal.

3.1.1.1 Graph-Based Representation of a Video Signal

Some examples and experimental results throughout this chapter are based on real video data. To make easier the explanation and understanding of subsequent sections, we briefly introduce now the graph representation of video data. More details about video representations are given in Chapter 4.

Let $\{x_k\}_{k=1}^{M}$ be a given video sequence where x_a refers to the luminance value of the pixel a belonging to a specific frame and spatial position. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be its graph representation, so that pixel a can be linked to several correlated pixels $\mathcal{F} \subset \{\mathcal{V} \backslash \{a\}\}$ without restrictions. Consider that edges (links) between nodes (pixels) can be spatial ($\mathcal{S}$) or temporal ($\mathcal{T}$) with differentiated statistical dependencies (correlations). Every edge belongs to $\mathcal{S}$ or $\mathcal{T}$ so that $\mathcal{S} \cup \mathcal{T} = \mathcal{E}$.

An example of the graph representation of a video signal is shown in Figure 3.1, where, in this case, every pixel is linked (i) to one temporal neighbor (i.e., a pixel of frame in time instant t linked to a pixel in frame $t + 1$) following a motion estimation (ME) process and (ii) to some one-hop spatial neighbors (i.e., pixels of the same frame), assuming that spatial neighboring pixels will have similar luminance values. Therefore, spatio-temporal pixel correlation is jointly considered.

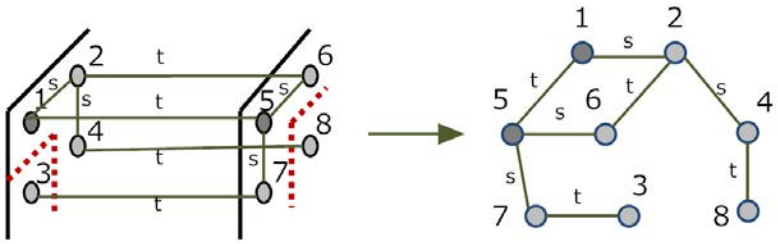


Figure 3.2: Graph representation of video data removing spatial links that cross the contours of a frame.

Moreover, a reasonable approach to improve the spatial correlation could be to remove links between spatial neighboring pixels that cross contours of an image (frame) assuming that they will have very different luminance value. This gives rise to the graph representation shown in Figure 3.2, where red dashed lines represent contours of a frame. Finally note that, in a video coding application, the accuracy-side information trade-off holds (e.g., using lower block sizes in the ME, such as blocks of 4×4 pixels, leads to more accurate graphs, but higher side information to be sent).

3.1.1.2 Graph-Based Representation of an N-Channel Digital Audio Signal

Digital audio signals usually exhibit a high level of correlation among neighboring samples. These correlations are exploited by means of linear prediction in predictive coders. Therefore, the encoder generates an estimate $\hat{x}_k$ of the current sample x_k from previous samples. Then, the encoder subtracts the prediction from the input sample to generate a residual signal, $d_k = x_k - \hat{x}_k$, which in general has smaller amplitude than x_k. This is called the *short-term* prediction.

Moreover, most audio signals have long-term correlations due to the harmonic nature of speech or musical instruments. Some audio encoders such as the MPEG-4 ALS have a dedicated *long-term* prediction scheme. This way, the residual $d(k)$ is predicted from long-term residuals as $\tilde{d}(k) = d(k) - \left(\sum_{j=-s}^{s} \gamma d(k - \tau + j)\right)$, where γ and τ are the *gain* and *lag* parameters, respectively, and s indicates the number of long-term residuals that are used to predict $d(k)$.

Finally, for stereo or more generally multi-channel audio signals, there exist an inherent correlation between every pair of channels. Then, a residual channel $\tilde{d}(k)^c$ can

be predicted from one reference residual channel $\tilde{d}(k)^r$, and send the difference signal $\tilde{e}(k)^c = \tilde{d}(k)^c - \tilde{d}(k)^r$. This is referred to as *inter-channel* prediction.

Figure 3.3 illustrate the *long-term* (red-solid line) and *inter-channel* (blue-dashed line) correlations in a stereo audio signal.

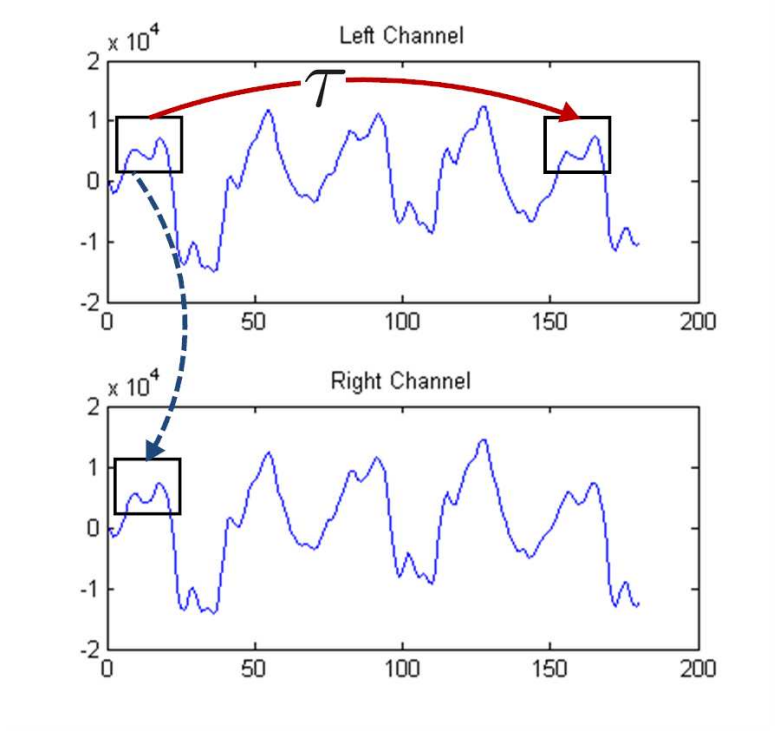


Figure 3.3: *Long-term* (red-solid line) and *inter-channel* (blue-dashed line) correlations in a stereo audio signal.

Now, we present a graph representation of multi-channel audio signals. Every specific sample a can be linked to a set of "short-term", "long-term" and "inter-channel" samples in order to exploit the inherent correlations that arise in multi-channel audio signals. Therefore, every audio sample could be simultaneously linked to an arbitrary number of "short-term", "long-term" and "inter-channel" samples, depending on the graph construction strategy followed.

Figure 3.4 shows an illustrative example in which every node is linked to its immediately previous and subsequent samples ("short-term" correlation, grey solid lines), to one "inter-channel" sample (blue dashed lines), and to one "long-term" sample (red solid lines) following the displacement indicated by the lag parameter τ. Any other

graph construction (e.g., considering N-“short-term” or “long-term” neighbors, considering a lag parameter in the “inter-channel” correlation, or breaking the “short-term” links between samples of very different sound pressure level values of each channel) can be made.

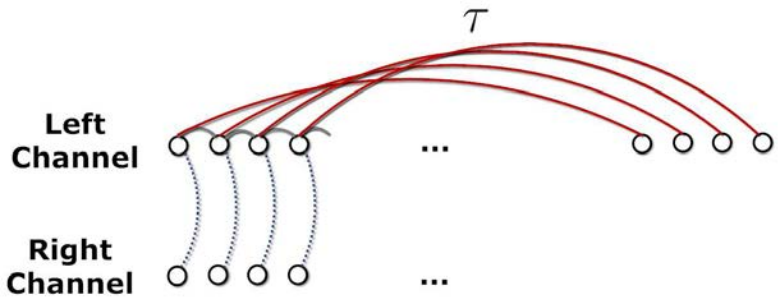


Figure 3.4: Stereo audio graph construction example.

Note that, as in the video coding example, there exists a trade-off between side information to be sent to the decoder and accurate graph representation (and thus better prediction). For example, *lag parameters* can be calculated every W samples (using windows of size W). When W is lower, the correlation between nodes is more accurately captured, but the side information is larger.

Given this graph representation of the N-channel audio signal, lifting transforms can be applied on this graph, obtaining a directional transformation of the data in which the different correlations are jointly exploited, and which is localized in frequency (which is very important in audio coding to consider the perceptual models) and spatio-temporal domains. This procedure can establish an interesting new framework for audio coding, since nowadays state of the art systems are generally based on transforms that remove the different redundancies separately. Therefore, it could be an interesting future research line.

3.1.2 Graph Weighting

Statistical dependencies (correlations) between nodes depends on the nature of links between them and on the specific graph representation used for the signal at hand. In order to take these features into account, it is useful to assign a specific weight to every link of the same nature (e.g., in a video representation, every spatial (resp. temporal)

link is associated with a specific spatial (resp. temporal) weight). This weight selection influences the $\mathcal{U}/\mathcal{P}$ assignment, the **p** and **u** filters design, and the reordering of the coefficients, to be discussed in Sections 3.2, 3.4, and Chapter 4 respectively.

In this section we propose two approaches to obtain the weights of the graph: (i) assuming fixed weights (as a function of the a-priori knowledge of the signal and its graph representation), and (ii) optimizing the weights in order to minimize the prediction error when using one-hop prediction filters (as a function of the signal at hand). In the second case, we first show how to calculate the optimal weights in a video representation example, and then we extend this result to F different kinds of links. Previous work on this topic was introduced by the authors in [42].

3.1.2.1 Fixed Weighting

As a starting point, weights (w) of the graph can be chosen fixed as a function of the a-priori knowledge of the signal and its graph representation. For example, in the graph representation of a video signal illustrated in Figure 3.1, temporal links are identified using an explicit search that minimizes a distortion measure (i.e., the standard ME) and spatial links are constructed by linking every node to its one-hop spatial neighbors. Therefore, in general, temporal links in the graph are more reliable than spatial links, that is, the expected correlation between temporal-linked pixels is higher than that between spatial-linked pixels. In this way, it is reasonable to assign higher weights to temporal connections. On the other hand, one can construct a graph representation of a video signal in which every pixel is linked to its spatial neighbors that do not cross contours as in Figure 3.2, and to its co-located pixels in the temporal domain (i.e., without using an explicit motion model to identify temporal correlated pixels). In this representation, temporal connections in the graph will be less correlated (and spatial connections more correlated) than in the previous example.

Given a graph representation, if one does not have any knowledge about the signal on the graph and how the representation was obtained, the weights can be fixed to $w = 1$, leading to an unweighted graph (i.e., non-connected neighbors can be considered to have $w = 0$).

3.1.2.2 Optimal Weighting for a Video Representation Example

The local correlation between neighboring nodes on the graph depends on the underlying signal. For example, in a video signal, the correlation between temporal and spatial neighbors changes with the video content, and thus the value of graph weights would change as well. Fixed weights are not suitable to handle these situations.

We now find the optimal graph weights that minimize the quadratic prediction error (**assuming one-hop prediction filters defined below**) for a given graph representation $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The optimal weights can be computed for any subgraph $\mathcal{H} \subset \mathcal{G}$ (i.e., their value can change for every subgraph) and at any level of decomposition j. In a video representation case, optimal weights can be computed, for example, in a frame-by-frame or in a block-by-block basis, as we will see in Section 4.2.2.

First, we consider a graph video representation example and derive the optimal weighting. In the next section, we extend the result to F general kinds of edges with different correlations.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph, where $\mathcal{V} = \{1, \ldots, N\}$ is a set of nodes and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ a set of edges. Let $\mathcal{S}, \mathcal{T}$ the set of spatial and temporal edges, respectively, with $\mathcal{S} \cup \mathcal{T} = \mathcal{E}$. Let $\mathcal{N}_i^{\mathcal{S}} = \{j : ij \in \mathcal{S}\}$ denote one-hop spatial neighborhood of i, for all nodes $i \in \mathcal{V}$.

Thus, the mean value of the spatial neighbors of node i is defined as

$$\bar{x}_i^s = \frac{1}{|\mathcal{N}_i|^{\mathcal{S}}} \sum_{j \in \mathcal{N}_i^{\mathcal{S}}} x_j, \tag{3.1}$$

where $\left|\mathcal{N}_i^{\mathcal{S}}\right|$ is the number of spatial neighbors of i. The mean value of the temporal neighbors is calculated similarly. Let us assume that every node i is linearly predicted from its spatial and temporal neighbors as:

$$\hat{x}_i = w_s \bar{x}_i^s + w_t \bar{x}_i^t. \tag{3.2}$$

Then, we seek the weights w_s and w_t that minimize the quadratic prediction error over all the nodes $i \in \mathcal{V}$:

$$\min_{w_s,w_t} \sum_{i\in\mathcal{V}} \left(x_i - w_s\bar{x}_i^s - w_t\bar{x}_i^t\right)^2. \tag{3.3}$$

Differentiating with respect to w_s and w_t we obtain:

$$\mathbf{w}^* = [w_s^*, w_t^*] = \mathbf{R^{-1}r}, \tag{3.4}$$

where

$$\mathbf{R} = \begin{bmatrix} \sum_{i\in\mathcal{V}} \bar{x}_i^s\bar{x}_i^s & \sum_{i\in\mathcal{V}} \bar{x}_i^s\bar{x}_i^t \\ \sum_{i\in\mathcal{V}} \bar{x}_i^t\bar{x}_i^s & \sum_{i\in\mathcal{V}} \bar{x}_i^t\bar{x}_i^t \end{bmatrix} \tag{3.5}$$

and

$$\mathbf{r} = \sum_{i\in\mathcal{V}} x_i \begin{bmatrix} \bar{x}_i^s \\ \bar{x}_i^t \end{bmatrix} \tag{3.6}$$

are the correlation matrices.

Usually, the graph topology is defined by means of its adjacency matrix. Next, we express the optimal weights as a function of the adjacency matrix of the graph. Let $\overline{\mathbf{A}}_\mathbf{s} = \left[\overline{a}_{s_{i,j}}\right]$ and $\overline{\mathbf{A}}_\mathbf{t} = \left[\overline{a}_{t_{i,j}}\right]$ be the adjacency matrices of the subgraphs containing only the spatial and temporal edges, respectively, where each column is normalized (i.e., $\overline{a}_{s_{i,j}} = 1/\left|\mathcal{N}_j^{\mathcal{S}}\right|$ if $ij \in \mathcal{S}$; $\overline{a}_{s_{i,j}} = 0$ if $ij \notin \mathcal{S}$). Vectorizing the sequence into a $1 \times (L \times H \times K)$ row vector $\mathbf{x}$, where $L \times H$ is the frame size and K the number of frames considered, we can write:

$$\mathbf{w}^* = \begin{bmatrix} \mathbf{x}\overline{\mathbf{A}}_\mathbf{s}\overline{\mathbf{A}}_\mathbf{s}^\mathbf{T}\mathbf{x}^\mathbf{T} & \mathbf{x}\overline{\mathbf{A}}_\mathbf{s}\overline{\mathbf{A}}_\mathbf{t}^\mathbf{T}\mathbf{x}^\mathbf{T} \\ \mathbf{x}\overline{\mathbf{A}}_\mathbf{t}\overline{\mathbf{A}}_\mathbf{s}^\mathbf{T}\mathbf{x}^\mathbf{T} & \mathbf{x}\overline{\mathbf{A}}_\mathbf{t}\overline{\mathbf{A}}_\mathbf{t}^\mathbf{T}\mathbf{x}^\mathbf{T} \end{bmatrix}^{-1} \cdot \begin{bmatrix} \mathbf{x}\overline{\mathbf{A}}_\mathbf{s}\mathbf{x}^\mathbf{T} \\ \mathbf{x}\overline{\mathbf{A}}_\mathbf{t}\mathbf{x}^\mathbf{T} \end{bmatrix}. \tag{3.7}$$

Note that, in a coding application, the weights should be sent to the decoder as side information. Therefore, a trade-off exists between accuracy in the weights selection

(lower subgraph sizes in which the weights are computed) and side information to be sent. Once $\mathbf{w}^*$ has been calculated, we assign w_s^* (resp. w_t^*) to every spatial (resp. temporal) link, leading to a weighted graph that accurately captures the correlation between nodes.

3.1.2.3 Optimal Weighting for F Different Kinds of Links

We now generalize the result in previous section to the case of F different kinds of links with different correlations.

Let us define the mean value of the class f neighbors of node i as:

$$\bar{x}_i^f = \frac{1}{|\mathcal{N}_i|^f} \sum_{j \in \mathcal{N}_i^f} x_j, \tag{3.8}$$

where $\mathcal{N}_i^f = \{j : ij \in f\}$ is the number of one-hop neighbors of i that belong to the class f. Assuming that every node in $\mathcal{V}$ is linearly predicted from its F types of neighbors, we would like to find the weights $w_1, w_2, \ldots, w_F$ that minimize the quadratic prediction error over all the nodes $\in \mathcal{V}$:

$$\min_{w_1,w_2,\ldots,w_F} \sum_{i \in \mathcal{V}} (x_i - \hat{x}_i)^2 = \min_{w_1,w_2,\ldots,w_F} \sum_{i \in \mathcal{V}} \left(x_i - w_1 \bar{x}_i^1 - w_2 \bar{x}_i^2 - \ldots - w_F \bar{x}_i^F\right)^2. \tag{3.9}$$

Extending (3.7), is straightforward to obtain the optimal weight vector, $\mathbf{w}^* = [w_1\,, w_2\,\ldots\,, w_F]$, as:

$$\mathbf{w}^* = \begin{bmatrix} \mathbf{x}\overline{\mathbf{A}}_\mathbf{1}\overline{\mathbf{A}}_\mathbf{1}^\mathbf{T}\mathbf{x}^\mathbf{T} & \mathbf{x}\overline{\mathbf{A}}_\mathbf{1}\overline{\mathbf{A}}_\mathbf{2}^\mathbf{T}\mathbf{x}^\mathbf{T} & \cdots & \mathbf{x}\overline{\mathbf{A}}_\mathbf{1}\overline{\mathbf{A}}_\mathbf{F}^\mathbf{T}\mathbf{x}^\mathbf{T} \\ \mathbf{x}\overline{\mathbf{A}}_\mathbf{2}\overline{\mathbf{A}}_\mathbf{1}^\mathbf{T}\mathbf{x}^\mathbf{T} & \mathbf{x}\overline{\mathbf{A}}_\mathbf{2}\overline{\mathbf{A}}_\mathbf{2}^\mathbf{T}\mathbf{x}^\mathbf{T} & \cdots & \mathbf{x}\overline{\mathbf{A}}_\mathbf{2}\overline{\mathbf{A}}_\mathbf{F}^\mathbf{T}\mathbf{x}^\mathbf{T} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}\overline{\mathbf{A}}_\mathbf{F}\overline{\mathbf{A}}_\mathbf{1}^\mathbf{T}\mathbf{x}^\mathbf{T} & \mathbf{x}\overline{\mathbf{A}}_\mathbf{F}\overline{\mathbf{A}}_\mathbf{2}^\mathbf{T}\mathbf{x}^\mathbf{T} & \cdots & \mathbf{x}\overline{\mathbf{A}}_\mathbf{F}\overline{\mathbf{A}}_\mathbf{F}^\mathbf{T}\mathbf{x}^\mathbf{T} \end{bmatrix}^{-1} \cdot \begin{bmatrix} \mathbf{x}\overline{\mathbf{A}}_\mathbf{1}\mathbf{x}^\mathbf{T} \\ \mathbf{x}\overline{\mathbf{A}}_\mathbf{2}\mathbf{x}^\mathbf{T} \\ \vdots \\ \mathbf{x}\overline{\mathbf{A}}_\mathbf{F}\mathbf{x}^\mathbf{T} \end{bmatrix}. \tag{3.10}$$

3.1.3 Discussion

The explained graph construction (i.e., graph representation of a signal and graph weighting) leads to a weighted graph that characterizes the correlation between samples of an

N-dimensional signal. Therefore, it can be useful to perform different signal processing operations.

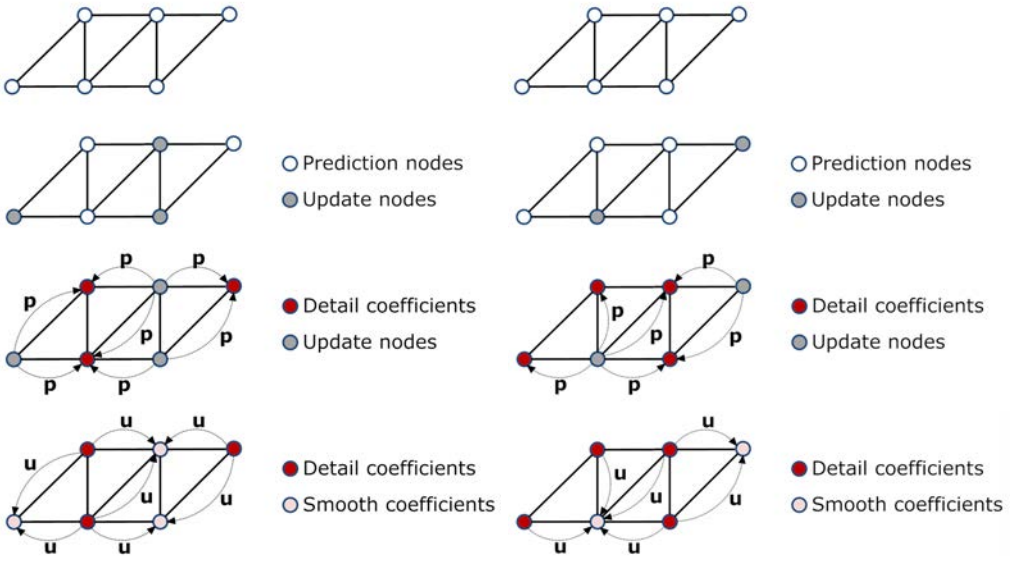
The weighting process can be made over any given undirected graph. In this thesis we focus on the lifting transform, which needs a bipartion of the graph to apply the filtering. Therefore, given a graph and a bipartition (i.e., $\mathcal{U}/\mathcal{P}$ assignment), one can find the optimal weights of the graph that minimize the detail coefficient energy of the predict stage when using one-hop prediction filters. To do that, the label of each node has to be taken into account in the optimization process described in Sections 3.1.2.2 and 3.1.2.3. We experimentally observed that the weight values do not significantly change if they are calculated before (minimizing over all the nodes of the graph) or after the $\mathcal{U}/\mathcal{P}$ assignment (minimizing over the prediction nodes, using only the update neighbors to obtain the prediction), so that our graph weighting will be near optimal in the sense of minimizing the detail coefficient energy. The formulation to obtain the optimal weights that minimize the detail coefficient energy using a given $\mathcal{U}/\mathcal{P}$ assignment, which is quite similar to the one presented below, is described in Appendix C.

Given the total freedom in the graph representation, one could link n-hop temporal or spatial neighbors (i.e., temporal links between nodes that are n-frames away or spatial links between nodes that are n-spatial hops away), and assign n different temporal and spatial weights as a function of the distance between linked nodes.

3.2 Graph-Based $\mathcal{U}/\mathcal{P}$ Assignment Methods

As discussed in Chapter 2, the first stage to compute lifting transforms on graphs consists on splitting nodes into Update ($\mathcal{U}$) and Prediction ($\mathcal{P}$) disjoint sets. One has great freedom to select which nodes will belong to the $\mathcal{U}$ set (and thus will be the low pass coefficients) and which ones will belong to the $\mathcal{P}$ set (and thus will be the high pass coefficients). Besides, the number of $\mathcal{U}/\mathcal{P}$ nodes for each level of the transform j can be arbitrarily chosen without compromising the invertibility of the transform, in contrast to the dyadic decomposition of classical wavelets (where one half of samples are low-pass coefficients and the other half high-pass coefficients in each level of the transform j, thus having subsamplings of $\frac{1}{2^j}$).

Figure 3.5 shows an illustrative example of two different $\mathcal{U}/\mathcal{P}$ disjoint assignments for the same original graph. Note that each one gives rise to different number and location of detail (high-pass) and smooth (low-pass) coefficients, implying different filtering operations and thus different (invertible) transformations of the original data. Besides, each assignment is a bipartition and leads to a different number of links being used in the transform.



(a) Example 1. (b) Example 2.

Figure 3.5: Two different transformations of the same original graph.

In this section we study different **graph-based** $\mathcal{U}/\mathcal{P}$ assignment techniques. Given a weighted graph, these methods find bipartitions without making any assumption about the graph signal[1].

The graph-based $\mathcal{U}/\mathcal{P}$ splitting is essentially a 2-color[2] graph-partition problem in which one color, chosen from the two available ones, is assigned to each node of the graph following some design criterion that does not depend on the signal at hand. These graph-partition problems have been widely studied in the graph-theory literature for

[1] Note that, if the weights on the graph correctly capture correlation between nodes, some information of the signal is implicitly considered in these weights.

[2] Terms "color"(white/grey) , "label"(1/0), "set"($\mathcal{P}$/$\mathcal{U}$) or "parity" will refer to the same concept.

different design criteria, and will be referred to as "classical" graph-partition problems throughout this thesis. Section 3.2.1 describes the "classical" *maximum-cut* and *set-covering* problems.

In Section 3.2.2 we summarize various graph-based $\mathcal{U}/\mathcal{P}$ assignment methods for lifting transforms on graphs that have been proposed in the literature, and their connection to "classical" graph-partition problems described in Section 3.2.1. In a coding application, it is natural to search graph-partitions in which a high number of $\mathcal{P}$ nodes can be well predicted from $\mathcal{U}$ neighbor nodes, obtaining many small detail coefficients. Intuitively, a "good" $\mathcal{U}/\mathcal{P}$ assignment to achieve this goal is the solution to the "classical" *weighted maximum-cut* problem, as the authors point out in [43] and [44]. This $\mathcal{U}/\mathcal{P}$ assignment method is discussed in Section 3.2.3.

3.2.1 Some "Classical" Graph-Partition Problems

In this section we describe two of the most well-known "classical" graph-partition problems: the *maximum-cut* and the *set-covering* problems. These two problems and the different solutions proposed in the literature will be useful in the graph $\mathcal{U}/\mathcal{P}$ splitting process for lifting transforms.

Note that we use the convention of representing $\mathcal{P}$ and $\mathcal{U}$ nodes as white and grey nodes, respectively, in the figures of this thesis.

3.2.1.1 Set-Covering Problem

A set-cover is a partition of the node (vertex) set of a graph into two disjoint subsets so that every node of one of the subsets has, at least, one neighbor in the other subset (i.e., every node in one subset is linked to the other subset). The *set-covering* problem can be defined as the problem of finding a set-cover in which one of the sets has the minimum possible number of elements.

Let us formally define the *set-covering* (SC) problem[3].

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph. Denote one-hop neighborhood of k, $\mathcal{N}_k = \{u \in \mathcal{V} : ku \in \mathcal{E}\}$, and closed neighborhood of k, $\mathcal{N}_{[k]} = \mathcal{N}_k \cup k$, for all nodes $k \in \mathcal{V}$. Given a collection $\mathcal{M}$ of all sets $\mathcal{N}_{[k]}$, a set-cover $\mathcal{C} \subseteq \mathcal{M}$ is a subcollection of the sets

[3] For simplicity, hereafter we refer to the solution of the SC problem as SC solution.

whose union is $\mathcal{V}$. The *set-covering* problem is, given $\mathcal{M}$, to find a minimum-cardinality set-cover $m\mathcal{C}$. Once we obtain a minimum-cardinality set-cover $m\mathcal{C} = \left\{\mathcal{N}_{[k_h]}\right\}_{h\in 1,2,\dots i}$, we can denote set $\{k_h\}_{h\in 1,2,\dots i}$ as $\mathcal{U}$ nodes and the remaining as $\mathcal{P}$ nodes ($\text{SC}_{\mathcal{U}}$) or vice-versa ($\text{SC}_{\mathcal{P}}$).

3.2.1.2 Maximum-Cut Problem

The *maximum-cut* problem can be viewed as finding the partition of the node set of a graph into two disjoint subsets which has maximum number of edges (or more generally, if the edges are weighted, the maximum sum of the weights) between elements of both subsets.

Let us formally define the *maximum-cut* (MC) and *weighted maximum-cut* (WMC) problems.

Consider an undirected **edge-weighted** graph $(\mathcal{G}, w)$, where $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is the graph and w is the weight function. A **cut** is defined as a partition of the node set into two disjoint subsets $\mathcal{U}$ and $\mathcal{P} := \mathcal{V}\backslash\mathcal{U}$. The **weight of the cut** (W) is given by the function

$$W(\mathcal{U}, \mathcal{P}) = \sum_{i\in\mathcal{P}, j\in\mathcal{U}} w_{ij}, \tag{3.11}$$

where w_{ij} is the weight of the link (edge) between nodes i and j.

A *weighted maximum-cut* is a cut of maximum weight, and is defined as:

$$WMC(\mathcal{G}, w) = \max_{\forall\mathcal{U}\subseteq\mathcal{V}} W(\mathcal{U}, \mathcal{P}). \tag{3.12}$$

The *maximum-cut* problem is defined similarly for an unweighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, that is, an edge-weighted graph defining the weight function as:

$$w_{ij} = \begin{cases} 1, & \text{if } ij \in \mathcal{E}, \\ 0, & \text{if } ij \notin \mathcal{E}. \end{cases} \tag{3.13}$$

Figure 3.6 shows different graph-partition solutions to the problems explained above and the weight of the cut, W, obtained for each solution, for a given graph. Note that

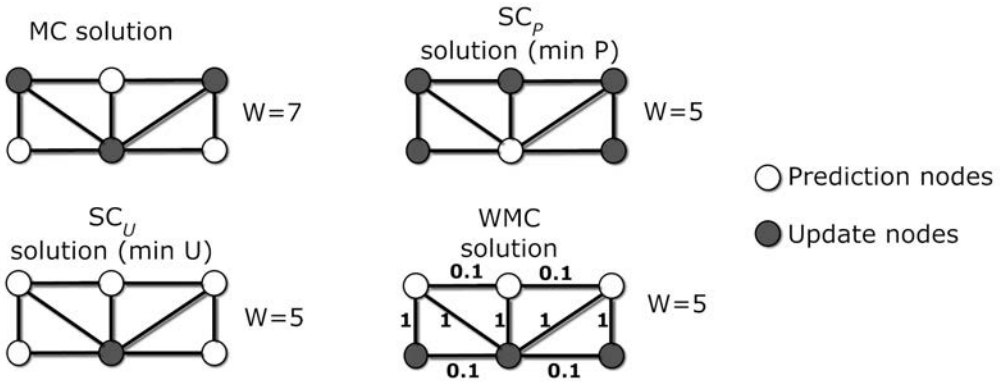


Figure 3.6: "Classical" graph-partition strategies.

in the example each color (white and black) has been associated with a specific set of nodes ($\mathcal{P}$ and $\mathcal{U}$ sets respectively).

3.2.2 $\mathcal{U}/\mathcal{P}$ Assignment Methods for Lifting Transforms on Graphs

In this section we summarize some graph-based $\mathcal{U}/\mathcal{P}$ assignment techniques proposed in the literature to perform the graph transformation and their link to the described "classical" graph-partition problems.

The $\mathcal{U}/\mathcal{P}$ assignment proposed in [45] attempts to optimize the total energy consumption in a wireless sensor network minimizing the number of nodes that have to transmit raw (not decorrelated) data in the network (the updates nodes $j \in \mathcal{U}$). In addition, in order to reduce the energy in prediction nodes $i \in \mathcal{P}$, every i must have at least one $\mathcal{U}$ neighbor to compute its detail coefficient. The authors show that this is equivalent to the $\text{SC}_{\mathcal{U}}$ problem defined in Section 3.2.1.

As explained in Chapter 2, for an arbitrary $\mathcal{U}/\mathcal{P}$ assignment, nodes that are neighbors in the graph are not guaranteed to have opposite parity. Connected nodes of the same parity cannot use each other's data to perform the transform, and edges connecting nodes of the same parity are considered "discarded" edges. As a solution to this problem, techniques that minimize the number of "discarded" edges (i.e., the percentage of direct neighbors in the graph that have the same parity) have been proposed [10]. A similar idea was presented in [46], where the authors proved that the $\mathcal{U}/\mathcal{P}$ graph-partition that minimizes the error between the transform in the original graph and in the

simplified graph (i.e., after edges discard) correspond to the solution to the classical MC problem of the graph $\mathcal{G}$ defined in Section 3.2.1.

3.2.3 Proposed Splitting Solution for a Coding Application

In a coding application it will be of interest to maximize energy compaction, which means storing the maximum amount of energy in the smallest number of coefficients (i.e., obtaining a large number of small detail coefficients).

One criterion to do that will be to maximize the reliability with which update nodes can predict prediction neighbors. Intuitively, a good approach to achieve this goal is the well known WMC, which we proposed in [43] and in [44] for a video coding application. Next, we analyze this intuition more in depth.

Let us assume that the edges $\mathcal{E}$ between nodes are weighted with specific values $w_{\mathcal{E}} \in \mathbb{R}$, which in some way are a similarity measure (e.g., similar luminance or correlation in an image) between nodes. Intuitively, a $\mathcal{P}$ node is better predicted (i.e., the detail coefficient is smaller) if it has a higher number of similar (with a high w value) $\mathcal{U}$ neighbor nodes. If we impose that every node must have at least one different-parity neighbor in order to make it possible to perform the update and predict stages of the transform, the solution to this problem is close to the solution of the $\text{SC}_{\mathcal{P}}$ problem (which maximizes the number of $\mathcal{U}$ neighbors that nodes $\mathcal{P}$ have). Nevertheless, the $\text{SC}_{\mathcal{P}}$ gives rise to a low number of $\mathcal{P}$ nodes, and we would like to obtain a high number of $\mathcal{P}$ nodes in which the data is decorrelated. Alternatively, we could find the $\text{SC}_{\mathcal{U}}$ solution, thus obtaining a large number of $\mathcal{P}$ nodes (detail coefficients) but with a low number of $\mathcal{U}$ neighbors (and thus not well estimated).

A good trade-off in this problem is thus to maximize the total weight (similarity) of the edges between the $\mathcal{P}$ and the $\mathcal{U}$ sets, which will give rise to a large number of $\mathcal{P}$ nodes with many reliable (correlated) $\mathcal{U}$ neighbors. Besides, this usually leads to balanced $\mathcal{P}$ and $\mathcal{U}$ sets, similar to the dyadic decomposition in classical wavelets. This problem is the WMC problem defined in Section 3.2.2.

Note that the WMC usually has a closed solution that gives rise to a specific location and number of $\mathcal{U}$ ($|\mathcal{U}_{WMC}|$) and $\mathcal{P}$ ($|\mathcal{P}_{WMC}|$) nodes. Therefore, for a given $|\mathcal{P}_{WMC}|$ nodes, WMC maximizes the weight between $\mathcal{U}$ an $\mathcal{P}$ sets, leading to accurate predictions, as we will show in experimental results of Section 3.3.4.

To compute the WMC solution we use the greedy approach of [47], leaving for future work the study of alternative methods. Note that, if the given graph is unweighted, the algorithm provides the MC solution. The algorithm is described in Algorithm 1, where $\mathcal{U}_j$ and $\mathcal{P}_j$ form a bipartition of the node set $\mathcal{U}_{j-1}$, and we consider *gain* of a node to be the sum of weights (i.e., the number of neighbors if the graph is unweighted) of all its incident edges.

Algorithm 1 *Weighted Maximum-Cut* Algorithm

Require: $\mathcal{U}_j = \{\varnothing\}$, $\mathcal{P}_j = \{\mathcal{U}_{j-1}\}$
1: Calculate the $Gain$ of the $\mathcal{U}_{j-1}$ node set
2: Select the node a with largest $Gain$, $a = max(Gain)$
3: **while** $Gain > 0$ **do**
4: Let $\mathcal{U}_j \leftarrow \mathcal{U}_j \cup \{a\}$
5: Let $\mathcal{P}_j \leftarrow \mathcal{P}_j \setminus \{a\}$
6: Change the sign of the incident edge weights
7: Update $Gains$ of adjacent nodes
8: Select the node a with largest $Gain$, $a = max(Gain)$
9: **end while**
10: **return** $\mathcal{U}_j$ and $\mathcal{P}_j$

Note that, even though the WMC has a closed solution, the algorithm could stop at any iteration, given a near optimal solution to problem of finding the maximum number of $\mathcal{U}$ neighbors for a given $|\mathcal{P}|$ nodes.

3.3 Signal Model-Based $\mathcal{U}/\mathcal{P}$ Assignment Methods

In this section we propose $\mathcal{U}/\mathcal{P}$ assignment criteria based on minimizing the expected value of the quadratic prediction error assuming a signal model and a predictor. In Section 3.3.1 we formally formulate our $\mathcal{U}/\mathcal{P}$ assignment problem, which is focused on minimize the detail coefficients energy. Sections 3.3.2, 3.3.3 and 3.3.4 present three different approaches to design partition algorithms that rely on model-based approaches. To do so, we assume a data generation model and a predictor, and find an expression of the mean squared prediction error under the assumed model and predictor. Then, we minimize it using different greedy algorithms, thus finding a near-optimal solution to the problem defined in Section 3.3.1.

The first model assumes that data in each node is a noisy version of some constant. Despite of its simplicity, some interesting conclusions regarding the split criteria can be extracted from its analysis. The second model assumes smooth variations between neighbor nodes values and therefore implicitly considers some correlation between them. This second model is further extended to consider that different links between nodes can be more or less correlated as a function of the nature of the link.

3.3.1 Proposed Signal Model-Based $\mathcal{U}/\mathcal{P}$ Assignment Problem Formulation for Lifting Transforms on Graphs

As discussed before, to obtain a sparse representation of the original signal, it is interesting to have small detail coefficients so that the signal energy is compacted on the smooth coefficients. Therefore, our goal is to find the $\mathcal{U}/\mathcal{P}$ assignment that minimizes the expected value of the detail coefficient energy (i.e., the expected value of the squared prediction error) **for a given number of $\mathcal{P}$ nodes**[4].

Assuming a signal model and a predictor, the problem can be stated as follows:

Problem 3.1. ***$\mathcal{U}/\mathcal{P}$ Assignment Problem Formulation.***

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a given undirected graph, where $\mathcal{V} = \{1, \ldots, N\}$ is a set of nodes and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ a set of edges. Let $\mathcal{X}$ be a set of N random variables, such that x_i represents the data value associated with node i in the graph. For each node $i \in \mathcal{P}$, consider the predictor $\hat{x}_i$. Define the total prediction error (E_{tot}) as the sum of the expected value of the squared error over the $\mathcal{P}$ nodes.

Find the $\mathcal{U}/\mathcal{P}$ assignment that minimizes E_{tot} ***for a given number of $\mathcal{P}$ nodes,*** *$|\mathcal{P}|$:*

$$\min_{\mathcal{U}/\mathcal{P}} E_{tot} = \min_{\mathcal{U}/\mathcal{P}} \sum_{i \in \mathcal{P}} \mathbb{E}\{(x_i - \hat{x}_i)^2\}. \tag{3.14}$$

Fixing $|\mathcal{P}|$ in the problem formulation is important because E_{tot} is minimized by minimizing the size of $\mathcal{P}$. Thus, solving (3.14) is practical only if some constraint on the size of $\mathcal{P}$ is introduced.

[4] Note that given that $|\mathcal{U}| = N - |\mathcal{P}|$ (where N is the number of nodes on the graph), fixing the number of $\mathcal{P}$ nodes is equivalent to fixing the number of $\mathcal{U}$ nodes.

Note that the brute-force solution would be unfeasible because one should try every possible $\mathcal{U}/\mathcal{P}$ assignment and calculate the squared error over all the $\mathcal{P}$ nodes for each of these assignments.

3.3.2 Noisy Model (NM)

In this section we assume a simplistic signal model in which the value of each node of the graph is a noisy version of some constant. Under the assumed model and predictor, we obtain and analyze an expression of the expected value of the quadratic prediction error. Then, we minimize it for a given $|\mathcal{P}|$ using a greedy algorithm, thus solving Problem 3.1.

This model is generally not realistic (e.g., if the graph represents a video signal, it would be a noisy version of a sequence of constant frames). Nevertheless, it can be taken as locally true (e.g., if the graph only represents subregions of the video sequence with similar pixels, the NM can be a more reasonable approximation of the real signal).

Definition 3.2. Noisy Model

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected unweighted graph. Let $\mathcal{X}$ be a set of N random variables, such that $x_i \in \mathcal{X}$ represents the data value associated with node i in the graph.

Let us assume that x_i is a noisy version of some constant c, in such a way that

$$x_i = c + \eta_i, \tag{3.15}$$

where η_i are independent noise variables with zero mean and variance v_i.

Definition 3.3. Unweighted Predictor

Let $\mathcal{N}_i = \{j \in \mathcal{V} : ij \in \mathcal{E}\}$ be the set of neighbors of node i. For each node $i \in \mathcal{P}$, consider the predictor

$$\hat{x}_i = \frac{1}{m_i} \sum_{j \in \mathcal{N}_i \cap \mathcal{U}} x_j, \tag{3.16}$$

where $m_i = |\mathcal{N}_i \cap \mathcal{U}|$.

Proposition 3.1. Noisy Model Prediction Error

Let x_i and $\hat{x}_i$ satisfy Definition 3.2 and Definition 3.3 respectively. Consider that $v_j = v$ for any j. The total prediction error over all nodes $i \in \mathcal{P}$ is given by

$$E_{totNM} = \sum_{i \in \mathcal{P}} \mathbb{E}\{(x_i - \hat{x}_i)^2\} = v \left(|\mathcal{P}| + \sum_{i \in \mathcal{P}} \frac{1}{m_i} \right). \tag{3.17}$$

Proof. $\hat{x}_i$ is an unbiased estimate of c, i.e.,

$$\mathbb{E}\{\hat{x}_i\} = c = \mathbb{E}\{x_i\}, \tag{3.18}$$

with variance

$$\text{var}(\hat{x}_i) = \frac{1}{m_i^2} \sum_{j \in \mathcal{N}_i \cap \mathcal{U}} v_j. \tag{3.19}$$

Our aim is to use $\hat{x}_i$ as a prediction for x_i. The mean square prediction error is

$$\begin{aligned} \mathbb{E}\{(x_i - \hat{x}_i)^2\} &= \mathbb{E}\{(x_i - c + c - \hat{x}_i)^2\} \\ &= \mathbb{E}\{(x_i - c)^2\} + \mathbb{E}\{(c - \hat{x}_i)^2\} + 2\mathbb{E}\{(x_i - c)(c - \hat{x}_i)\} \\ &= v_i + \text{var}(\hat{x}_i) + 2\mathbb{E}\{(x_i - c)(c - \hat{x}_i)\} \\ &= v_i + \frac{1}{m_i^2} \sum_{j \in \mathcal{N}_i \cap \mathcal{U}} v_j, \end{aligned} \tag{3.20}$$

where we have used (3.15), (3.16) and the independence of the noise variables.

The total prediction error is

$$\begin{aligned} E_{\text{totNM}} &= \sum_{i \in \mathcal{P}} \mathbb{E}\{(x_i - \hat{x}_i)^2\} \\ &= \sum_{i \in \mathcal{P}} v_i + \sum_{i \in \mathcal{P}} \left(\frac{1}{m_i^2} \sum_{j \in \mathcal{N}_i \cap \mathcal{U}} v_j \right). \end{aligned} \tag{3.21}$$

If $v_j = v$ for all j,

$$E_{\text{totNM}} = v \left(|\mathcal{P}| + \sum_{i \in \mathcal{P}} \frac{1}{m_i} \right). \tag{3.22}$$

□

For a fixed $|\mathcal{P}|$, some interesting conclusions can be extracted from (3.17):

1. E_{totNM} increases with the variance of the nodes v (e.g., in a video coding application, E_{totNM} will be higher in sequences with complex textures or motions).

2. The first term of the right side of (3.17), $v|\mathcal{P}|$, represents the intrinsic variance of nodes $i \in \mathcal{P}$. The second term, $v \sum_{i\in\mathcal{P}} \frac{1}{m_i}$, refers to the variance of the prediction, which decreases with the number of $\mathcal{U}$ neighbors of each node $i \in \mathcal{P}$, m_i.

3. For a fixed $|\mathcal{P}|$ and weight of the cut $W = \sum_{i\in\mathcal{P}} m_i = W_c$ (note that given that the graph is unweighted, W is the number of links between $\mathcal{U}$ and $\mathcal{P}$), the best graph-partition that one can construct in order to minimize E_{totNM} (under the assumed NM and predictor) is to equally distribute these links amongst all the $i \in \mathcal{P}$ nodes. This is formally stated in Corollary 3.1.

Corollary 3.1. Optimal Criteria for Fixed $|\mathcal{P}|$ and $W = W_c$

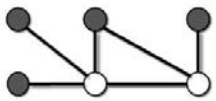
For a fixed $|\mathcal{P}|$ and $W = W_c$, the optimal criteria to minimize E_{totNM} over these $\mathcal{P}$ nodes is that all of them have the same number of $\mathcal{U}$ neighbor nodes.

Proof. We want to seek the $m_i \; \forall i \in \mathcal{P}$ that minimizes E_{totNM} for a fixed $|\mathcal{P}|$ and $W = W_c$:

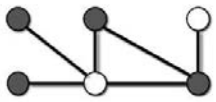
$$\min_{m_i} \left\{ v \left(|\mathcal{P}| + \sum_{i\in\mathcal{P}} \frac{1}{m_i} \right) \right\}$$
$$\text{s. t. W} = \sum_{i\in\mathcal{P}} m_i = W_c, \tag{3.23}$$

which, for a fixed $|\mathcal{P}|$, is equivalent to minimizing

$$\min_{m_i} \sum_{i\in\mathcal{P}} \frac{1}{m_i}$$
$$\text{s. t. W} = \sum_{i\in\mathcal{P}} m_i = W_c. \tag{3.24}$$



W=5 $\quad E_{\text{tot}} = v\left(|\mathcal{P}| + \sum_{i\in\mathcal{P}}^{|\mathcal{P}|}\frac{1}{m_i}\right) = v\left(2+\frac{1}{3}+\frac{1}{2}\right) = v\left(2+\frac{5}{6}\right).$



W=5 $\quad E_{\text{tot}} = v\left(|\mathcal{P}| + \sum_{i\in\mathcal{P}}^{|\mathcal{P}|}\frac{1}{m_i}\right) = v\left(2+\frac{1}{4}+1\right) = v\left(2+\frac{5}{4}\right).$

Figure 3.7: MC and NM $\mathcal{U}/\mathcal{P}$ assignment strategies.

Defining the Lagrangian:

$$\mathcal{L} = \sum_{i\in\mathcal{P}}\frac{1}{m_i} + \lambda\left(\sum_{i\in\mathcal{P}} m_i - W_c\right) \tag{3.25}$$

and minimizing with respect to each m_i:

$$\frac{\partial\mathcal{L}}{\partial m_i} = -\frac{1}{m_i^2} + \lambda = 0 \implies m_i = \frac{1}{\sqrt{\lambda}} = K. \tag{3.26}$$

Furthermore,

$$\sum_{i\in\mathcal{P}} m_i = K|\mathcal{P}| = W_c \implies K = \frac{W_c}{|\mathcal{P}|}. \tag{3.27}$$

So, given a weight for the cut W_c and $|\mathcal{P}|$, we obtain the minimum of $\mathcal{L}$ when every node $i \in \mathcal{P}$ has the same number of $\mathcal{U}$ neighbors $K = \frac{W_c}{|\mathcal{P}|}$. □

Figure 3.7 shows an example of two $\mathcal{U}/\mathcal{P}$ graph-partitions with the same W and the same number of $\mathcal{U}$ (dark) and $\mathcal{P}$ (white) nodes. Note that in the upper partition the number of $\mathcal{U}$ neighbors of $\mathcal{P}$ nodes is more balanced, giving rise to a smaller prediction error.

Hereafter we refer to the $\mathcal{U}/\mathcal{P}$ assignment strategy that aims to minimize (3.17) for a given $|\mathcal{P}|$ as NM solution.

3.3.2.1 Proposed NM Greedy Solution

To obtain the **NM solution and thus solve Problem 3.1, we design a greedy algorithm** that locally minimizes (3.17) in each iteration. First, the algorithm finds the $\text{SC}_{\mathcal{U}}$ solution (SC solution with minimum number of $\mathcal{U}$ nodes) in order to guarantee that every node $\in \mathcal{P}$ has at least one neighbor $\in \mathcal{U}$ and can be predicted. Then, **in each iteration** t, the algorithm changes to $\mathcal{U}$ the node $c \in \mathcal{P}$ that, when is changed, minimizes E_{totNM} (3.17) (this c is referred to as c^*). Note that each iteration t corresponds with a specific $|\mathcal{P}|_t$, so that the process should end when the given $|\mathcal{P}|$ defined in Problem 3.1 is reached.

It would be very computationally expensive to calculate E_{totNM} for every of the possible $c \in \mathcal{P}$ candidates to be changed to $\mathcal{U}$, in each iteration t. Actually, we are interested in finding c^* and not in the total cost E_{totNM} at each iteration t of the algorithm. Therefore, we focus on finding the $c \in \mathcal{P}$ that, when is changed to $\mathcal{U}$, **maximizes the** E_{totNM} **difference between iterations** $t-1$ **and** t.

Let $E_{totNM}^{\{t,c\}}$ be the E_{totNM} obtained if node $c \in \mathcal{P}$ is changed to $\mathcal{U}$ at iteration t. We want to minimize $E_{totNM}^{\{t,c\}}$ over all possible candidates $c \in \mathcal{P}$:

$$\min_{c \in \mathcal{P}} E_{totNM}^{\{t,c\}}. \tag{3.28}$$

Note that $E_{totNM}^{\{t,c\}}$ can be written as a function of the cost at the past iteration $t-1$ ($E_{totNM}^{\{t-1\}}$):

$$E_{totNM}^{\{t,c\}} = E_{totNM}^{\{t-1\}} + \Theta^{\{t,c\}}, \tag{3.29}$$

where $\Theta^{\{t,c\}}$ represents the changes in E_{totNM} that occur, from iteration $t-1$ to t, if we move node c from $\mathcal{P}$ to $\mathcal{U}$. Note that, as E_{totNM} decreases when we incorporate more $\mathcal{U}$ nodes, we have that $E_{totNM}^{\{t-1\}} > E_{totNM}^{\{t,c\}}$, and so $\Theta^{\{t,c\}} < 0$.

Given that $E_{totNM}^{\{t-1\}}$ does not depend on the node that will be moving at iteration t, thus (3.28) can be written, using (3.29), as:

$$\min_{c\in\mathcal{P}} E_{totNM}^{\{t,c\}} = \min_{c\in\mathcal{P}} \Theta^{\{t,c\}}. \tag{3.30}$$

The advantage of minimizing $\Theta^{\{t,c\}}$ instead of $E_{totNM}^{\{t,c\}}$ is that now we can find c^* just by observing how E_{totNM} changes in the neighborhood of every candidate node c (specifically, looking at two-hop neighbors), and thus without having to evaluate E_{totNM} summing over all nodes $i \in \mathcal{P}$ for every c.

The intrinsic variance $v\left(|\mathcal{P}|\right)$ in E_{totNM} (3.17) does not depend on the node that is changed, so that, using (3.17), (3.30) can be written:

$$\min_{c\in\mathcal{P}} \Theta^{\{t,c\}} = \min_{c\in\mathcal{P}} \sum_{k\in\mathcal{N}_c\cap\mathcal{P}} \left(\frac{1}{(m_k+1)} - \frac{1}{m_k}\right) - \frac{1}{m_c}, \tag{3.31}$$

where m_c and m_k are the number of $\mathcal{U}$ neighbors of candidate node c, and of its $k \in \mathcal{P}$ neighbors, respectively. The terms in the sum in (3.31) indicates that, if c is moved to $\mathcal{U}$, its $k \in \mathcal{P}$ neighbors will have one more $\mathcal{U}$ neighbor to be predicted. Last term in (3.31) reflects that E_{totNM} is also reduced because node c becomes $\mathcal{U}$, and therefore it should not be taken into account in the calculation of E_{totNM}. Note that now, in order to find c^*, we just have to sum over its neighbors.

Figure 3.8 illustrates an example. The left side shows the $\mathcal{U}/\mathcal{P}$ of the graph at iteration $t-1$, and right side at iteration t, assuming that candidate node 3 is changed to $\mathcal{U}$. Note that changing node 3 implies that nodes 2, 4, and 6, go from having one $\mathcal{U}$ neighbors to two. So that, $\Theta^{\{t,c=3\}} = (1/2 - 1 + 1/2 - 1 + 1/2 - 1) - 1$.

Finally, note that $\Theta^{\{t,c\}}$ can be calculated for every c with a few simple matrix operations[5], obtaining the vector $\mathbf{\Theta}^{\{t,c\}} = [\Theta^{\{t,c=1\}}\Theta^{\{t,c=2\}} \dots \Theta^{\{t,c=|\mathcal{P}|_t\}}]$. The complete NM greedy approach is summarized in Algorithm 2.

[5] Source code related to this thesis will be available online at www.tsc.uc3m.es / ∼ emenriquez/.

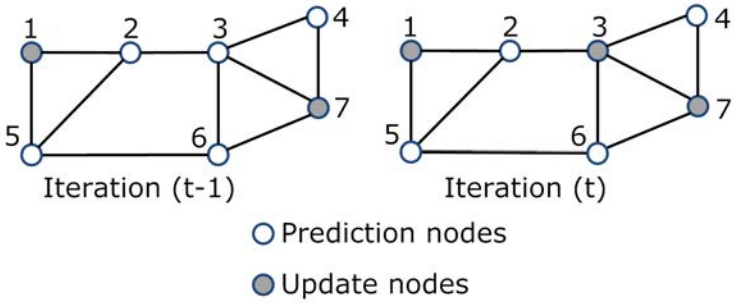


Figure 3.8: Greedy algorithm for the NM.

Algorithm 2 NM Greedy Algorithm

Require: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $|\mathcal{P}|$ nodes

1: Calculate the $\text{SC}_{\mathcal{U}}$ solution
2: **while** $|\mathcal{P}|_t > |\mathcal{P}|$ **do**
3: Calculate $\mathbf{\Theta}^{\{t,c\}}$
4: Select the node c^* with minimum $\Theta^{\{t,c\}}$, $c^* = \min \mathbf{\Theta}^{\{t,c\}}$
5: Let $\mathcal{U} \leftarrow \mathcal{U} \cup \{c^*\}$
6: Let $\mathcal{P} \leftarrow \mathcal{P} \backslash \{c^*\}$
7: **end while**
8: **return** $\mathcal{U}/\mathcal{P}$ assignment

3.3.2.2 Experimental Results

In this subsection we present some experimental results in terms of the quadratic prediction error (i.e., the energy of the detail coefficients) in the first level of the transform ($j = 1$) when applying different $\mathcal{U}/\mathcal{P}$ assignment "classical" strategies studied in Section 3.2.1 and the proposed one in Section 3.3.2 (NM).

The experiments have been carried out in the context of video representation, using subgraphs of real data obtained from different standard test sequences (i.e., the subgraphs contain data from a specific area and number of frames of each video). The graph representation of the video follows the philosophy of Figure 3.1, in which any node represents the luminance value of a pixel and can have some spatial and temporal neighbors simultaneously. In this case, every node is connected to its 8-one hop spatial neighbors and to an arbitrary number of temporal neighbors following a ME model.

Note that the NM does not distinguish between spatial or temporal neighbors, so that the links between different nodes are not weighted and the nature of each link is not taken into account. Therefore, we have an unweighted graph (3.13) in which every $\mathcal{U}$ neighbor of a $\mathcal{P}$ node has the same importance in the prediction (3.16).

To evaluate the performance of each approach, we measure the **average prediction error** over all nodes in $\mathcal{P}$ as:

$$E_{\text{av-meas}} = \frac{1}{|\mathcal{P}|} \sum_{i \in \mathcal{P}}^{|\mathcal{P}|} (x_i - \hat{x}_i)^2 = \frac{1}{|\mathcal{P}|} \sum_{i \in \mathcal{P}}^{|\mathcal{P}|} (d_i)^2, \tag{3.32}$$

where x_i is the luminance value of the pixels, and $\hat{x}_i$ the prediction, defined as in (3.16). To obtain the NM $\mathcal{U}/\mathcal{P}$ assignment solution and thus solve Problem 3.1 we use Algorithm 2.

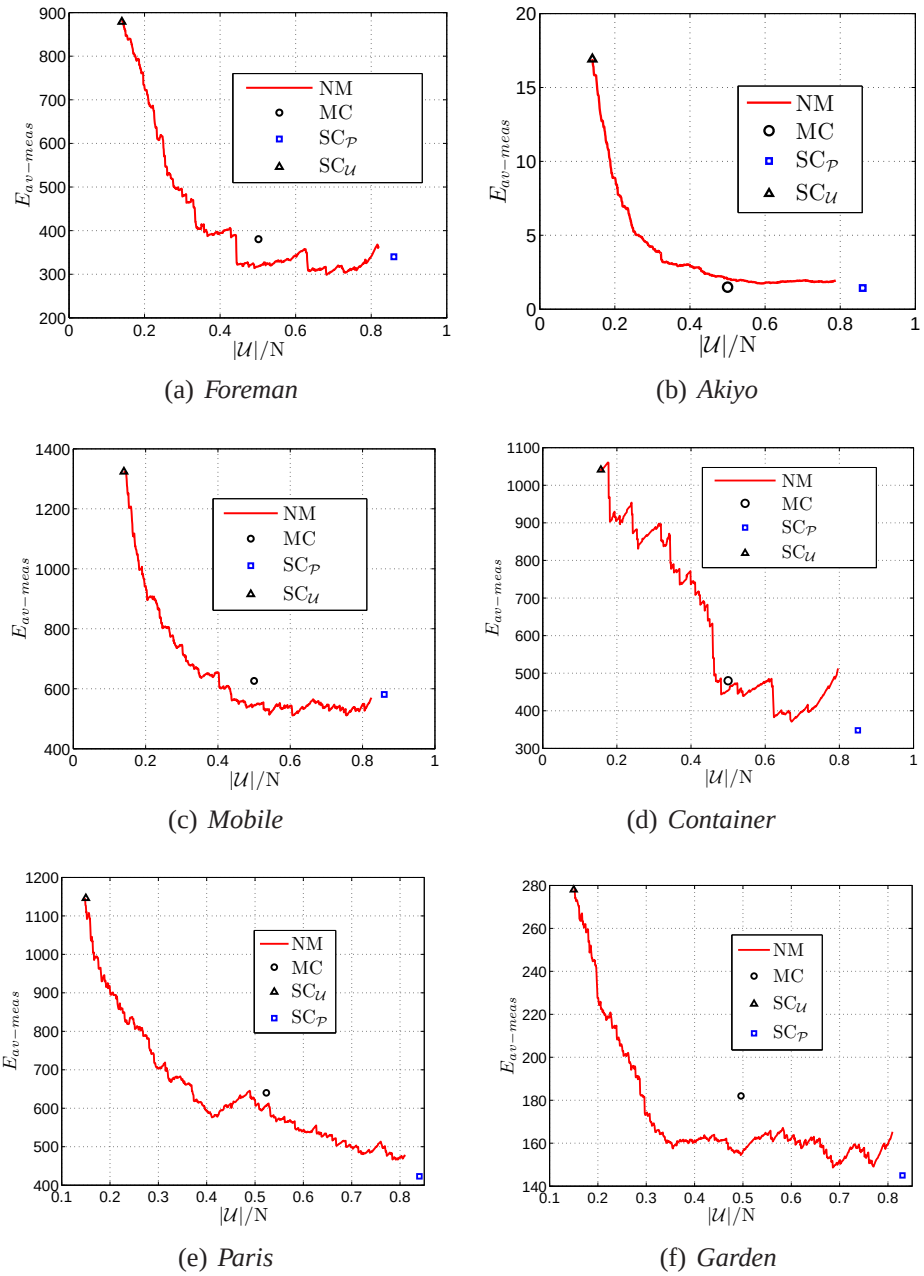
Figure 3.9 shows experimental results where $E_{\text{av-meas}}$ in the first level of the transform ($j = 1$) is plotted as a function of the relation $|\mathcal{U}|/N$ selected by the different $\mathcal{U}/\mathcal{P}$ strategies (MC, SC$_{\mathcal{U}}$, SC$_{\mathcal{P}}$ and the proposed NM solution)[6].

Note that the number of $\mathcal{U}/\mathcal{P}$ nodes is fixed for the MC, SC$_{\mathcal{U}}$ and SC$_{\mathcal{P}}$ solutions, and can vary in the NM approach (by letting the given $|\mathcal{P}|$ in Problem 3.1 vary). The reason is that the MC, SC$_{\mathcal{U}}$ and SC$_{\mathcal{P}}$ have closed solutions that give rise to a specific $\mathcal{U}/\mathcal{P}$ bipartition (and thus a specific number and location of $\mathcal{U}$ and $\mathcal{P}$ nodes), while the NM aims to minimize (3.17) for any given number of $\mathcal{U}$ and $\mathcal{P}$ nodes.

Some conclusions can be extracted from the experimental results shown in Figure 3.9:

- If we compare the proposed NM optimization method and the MC in the $|\mathcal{U}|/N$ that is solution to the MC, in general, NM achieves lower $E_{\text{av-meas}}$ than the MC method. The reason is that, in spite of the fact that MC gives rise to a higher average number of $\mathcal{U}$ neighbors (because for that $|\mathcal{U}|/N$, MC maximizes the cut and thus the number of $\mathcal{U}$ neighbors that $\mathcal{P}$ nodes have), NM generates a similar number of $\mathcal{U}$ neighbors for all the $\mathcal{P}$ nodes. This is illustrated in Figure 3.10,

[6] The greedy algorithm used to find the SC$_{\mathcal{U}}$ (or SC$_{\mathcal{P}}$) solution is given in Appendix A.



(a) *Foreman* (b) *Akiyo*

(c) *Mobile* (d) *Container*

(e) *Paris* (f) *Garden*

Figure 3.9: $E_{\text{av-meas}}$ for different sequences. A comparison of NM with several "classic" solutions to the $\mathcal{U}/\mathcal{P}$ assignment problem.

which shows, for the NM and the MC, the mean number ($\mu_{\mathcal{U}}$) and the standard deviation ($\sigma_{\mathcal{U}}$) of $\mathcal{U}$ neighbors that $\mathcal{P}$ nodes have as a function of $|\mathcal{U}|/N$, in the sequence *Foreman*.

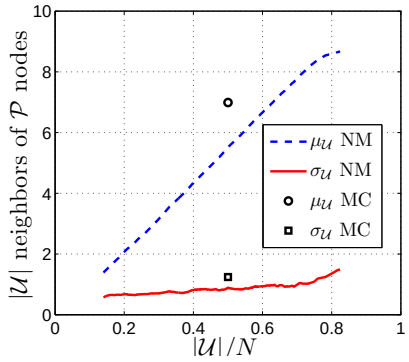


Figure 3.10: $\mu_{\mathcal{U}}$ and $\sigma_{\mathcal{U}}$ for the sequence *Foreman*.

- Note that, as expected, $E_{\text{av-meas}}$ generally decreases as the proposed algorithm chooses a higher number of $\mathcal{U}$ nodes, because the obtained predictions are better.

- The $\text{SC}_{\mathcal{U}}$ solution involves obtaining the minimum number of $\mathcal{U}$ nodes (and therefore low pass coefficients) that guarantees that every $\mathcal{P}$ node has at least one $\mathcal{U}$ neighbor and thus can be predicted. This implies that we will have a large number of nodes $\mathcal{P}$ in which the data is decorrelated (giving rise to the detail coefficients). Nevertheless, minimizing the number of $\mathcal{U}$ nodes implies that $\mathcal{P}$ nodes would have, in general, a low number of $\mathcal{U}$ neighbors to calculate the detail coefficient and thus the prediction of this detail coefficients will not usually be so accurate. Therefore, the mean energy of detail coefficients $E_{\text{av-meas}}$ will be large as is shown in Figure 3.9.

- On the other hand, the $\text{SC}_{\mathcal{P}}$ solution implies that we will have accurate predictions (e.g., the $E_{\text{av-meas}}$ will be low, as shown in Figure 3.9) but a low number of detail coefficients in which data is predicted.

- In sequences with low pixel variance (v), such as the fragment of *Akiyo*, which is quite homogeneous and stationary , the $\mathcal{P}$ nodes do not need many $\mathcal{U}$ neighbors to be correctly predicted. Thus, once the algorithm reaches a reasonable value of $|\mathcal{U}|/N$, increases in this value do not improve the prediction, and the $E_{\text{av-meas}}$ remains almost constant.

Note that, as it was discussed before, in a coding application there exists a trade-off between obtaining a low number of low pass coefficients (i.e., low number of $\mathcal{U}$ nodes) and having small detail coefficients (i.e., high number of $\mathcal{U}$ neighbor nodes). Furthermore, this trade-off depends on the video content (i.e., in sequences such as *Akiyo*, one does not need too many $\mathcal{U}$ nodes to obtain "good" predictions and thus low detail coefficients; while in sequences such as *Coastguard* every new $\mathcal{U}$ node makes the prediction better, decreasing $E_{\text{av-meas}}$). Therefore, it would be of interest to have an encoder procedure that tells the decoder the optimal number of $\mathcal{U}$ and $\mathcal{P}$ nodes depending on the video content and the available resources (i.e., a target quality or bit rate).

We have drawn some interesting conclusions regarding the $\mathcal{U}/\mathcal{P}$ assignment problem based on the analysis of NM. In the next section we introduce the *Moving Average Model*, which considers smooth noise variations between neighbor nodes.

3.3.3 Moving Average Model (MA)

Generally, data across nearby sample points present some correlation (e.g., nearby pixels in an image or video usually have similar luminance values; adjacent sample points in audio data generally present similar sound pressure level (SPL) values; or neighboring data in a WSN tend to be correlated). In this section we propose a data generation model that considers **smooth noise variations** between neighbors on the graph. Specifically, we consider that data in node i is generated as is defined below.

Definition 3.4. Moving Average Model

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph. Let us assume that x_i is generated as the mean noise ϵ_j value of the closed neighborhood of node i plus an independent noise η_i as:

$$x_i = \frac{1}{|\mathcal{N}_{[i]}|} \sum_{j \in \mathcal{N}_{[i]}} \epsilon_j + \alpha \eta_i, \tag{3.33}$$

where ϵ_j and η_i are zero-mean independent random variables, with variances v_{ϵ_j} and v_{η_i}, respectively; $\mathcal{N}_{[i]}$ is the closed neighborhood set of node i ($\mathcal{N}_{[i]} = \mathcal{N}_i \cup i$), and α is an arbitrary nonnegative real constant.

Thus, this model can be viewed as a low-pass filtered noisy (every node with noise ϵ_i) random graph signal plus an additive independent random noise (every node with noise η_i). Figure 3.11 illustrates the data generation following this model, which will be referred to as Moving Average (MA) model.

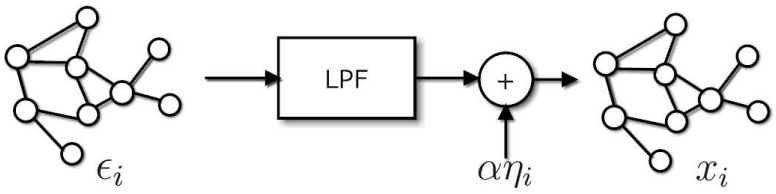


Figure 3.11: MA data generation model.

Despite its simplicity, similar models have been employed in the literature for image texture representation [48], and for image [49] [50], audio and speech modeling. Define the *clustering degree* of nodes m and n on graph $\mathcal{G}$ as

$$c(m,n) = \frac{|\mathcal{N}_{[m]} \cap \mathcal{N}_{[n]}|}{|\mathcal{N}_{[m]}||\mathcal{N}_{[n]}|}. \tag{3.34}$$

Next, we calculate the expected value of the prediction error assuming the MA data generation model (3.33) and using the unweighted predictor defined in Definition 3.3.

Proposition 3.2. Moving Average Model Prediction Error

Let x_i and $\hat{x}_i$ satisfy Definition 3.4 and Definition 3.3 respectively. Consider that $v_{\eta_i} = v_\eta$ and that $v_{\epsilon_i} = v_\epsilon$ for any $i \in \mathcal{V}$. The prediction error of a node $i \in \mathcal{P}$ is given by

$$\begin{aligned} E_{MA_i} &= \mathbb{E}\{(x_i - \hat{x}_i)^2\} = \mathbb{E}\{(x_i)^2\} + \mathbb{E}\{(\hat{x}_i)^2\} - 2\mathbb{E}\{x_i\hat{x}_i\} \\ &= \underbrace{\alpha^2 v_\eta + \frac{v_\epsilon}{|\mathcal{N}_{[i]}|}}_{\mathbf{A}} + \underbrace{\frac{\alpha^2 v_\eta}{m_i} + \frac{v_\epsilon}{m_i^2} \sum_{j \in |\mathcal{N}_i \cap \mathcal{U}|} \sum_{k \in |\mathcal{N}_i \cap \mathcal{U}|} c(j,k)}_{\mathbf{B}} \\ &\quad \underbrace{- 2\,\frac{v_\epsilon}{m_i} \sum_{k \in |\mathcal{N}_i \cap \mathcal{U}|} c(i,k)}_{\mathbf{C}}. \end{aligned} \tag{3.35}$$

The proof is in Appendix B.1.

Let us now analyze (3.35). The terms inside **A** represent the **variance of the observation**, $\mathbb{E}\{(x_i)^2\}$ (given that $\mathbb{E}\{(x_i)\} = 0$, var$(x_i) = \mathbb{E}\{(x_i)^2\}$). Note that the first term inside A is due to the independent noise η_i of each node, and does not depend on the node at hand (because of our assumption of $v_{\eta_i} = v_\eta$). The second factor is due to the smooth variation of the noise ϵ_j (the low-pass filtered noisy graph signal). Note that as the number of neighbors of node i ($|\mathcal{N}_{[i]}|$) increases, this second factor (and thus the variance of the observation) is lower, which is reasonable because x_i is obtained by averaging with a higher number of samples. In this way, from an $\mathcal{U}/\mathcal{P}$ strategy point of view, will be of interest to choose as $\mathcal{P}$ nodes (nodes to be predicted) those nodes that have more neighbors and thus can be more easily predicted (e.g., assuming that we are given a graph representation of an image which links neighbor nodes that do not cross contours of the image, it will be better to choose as $\mathcal{P}$ those nodes that are far away from the contours and thus have more neighbors because their value will be smoother and thus more easily predicted).

The terms inside **B** represent the **variance of the predictor**, $\mathbb{E}\{(\hat{x}_i)^2\}$. It is composed of two factors, which decrease as the number of $\mathcal{U}$ neighbors of node i (m_i) increases. Therefore, as expected, the variance of the predictor is lower as we have more data to perform the prediction. Furthermore, the variance decreases as the factor $c(j,k)$ (i.e., the clustering degree **between $\mathcal{U}$ neighbors of node** i) decreases. Note that $c(j,k)$ indicates the proportion of shared neighbors between nodes j and k. Thus, it is of interest to have uncorrelated $\mathcal{U}$ neighbors of i (i.e., $\mathcal{U}$ neighbors of i that do not share many neighbors/information) to perform its prediction.

To give some insight into the behaviour of the $\mathbb{E}\{(\hat{x}_i)^2\}$ term, Figure 3.12 shows the values that it takes for different situations and graph topologies. In this example we consider $v_\eta = v_\epsilon = v$ and $\alpha = 1$.

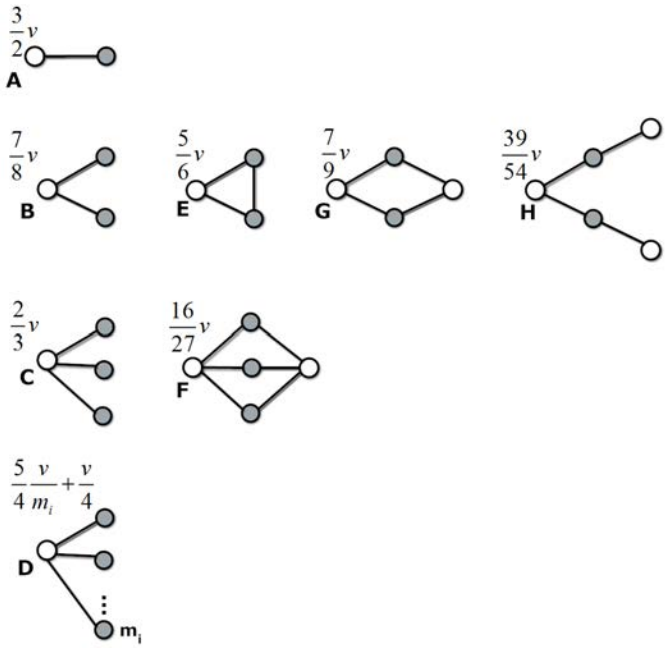


Figure 3.12: $\mathbb{E}\{(\hat{x}_i)^2\}$ for different graph topologies.

Note that, in each row of the figure, the node i to be predicted (the most left white node in every example) has the same number of $\mathcal{U}$ (dark) neighbors, m_i. First column shows that, considering the same topology, variance $\mathbb{E}\{(\hat{x}_i)^2\}$ is lower as m_i is higher ($\mathbb{E}\{(\hat{x}_A)^2\} > \mathbb{E}\{(\hat{x}_B)^2\} > \mathbb{E}\{(\hat{x}_C)^2\}$, and in general, $\mathbb{E}\{(\hat{x}_i)^2\} = \frac{5}{4}\frac{v}{m_i} + \frac{v}{4}$. In the second row of the figure, the node i to be predicted has $m_i = 2$ in all the examples. Nevertheless, $\mathbb{E}\{(\hat{x}_i)^2\}$ decreases as the $\mathcal{U}$ neighbors of node i have more neighbors that they do not share with each other (i.e., as the $c(j,k)$ factor is lower).

Finally, the term inside **C** represents the **cross-correlation between the estimate and the observation**, $\mathbb{E}\{x_i\hat{x}_i\}$. Note that this term increases as m_i is lower and as the clustering degree **between node** i **and its** $\mathcal{U}$ **neighbors**, $c(i,k)$, is higher (i.e., the observation x_i and the predictor $\hat{x}_i$ will be more correlated as they share more neighbors/information).

Once we have explained (3.35), we calculate the total prediction error just by summing over all nodes $i \in \mathcal{P}$. After reordering the equation we get the following expression:

$$E_{\text{totMA}} = \sum_{i \in \mathcal{P}} \mathbb{E}\{(x_i - \hat{x}_i)^2\} \tag{3.36}$$

$$= \underbrace{\alpha^2 v_\eta \sum_{i \in \mathcal{P}} \left(1 + \frac{1}{m_i}\right)}_{\mathbf{A}} + \underbrace{v_\epsilon \sum_{i \in \mathcal{P}} \left(\frac{1}{|\mathcal{N}_{[i]}|} + \frac{1}{m_i^2} \sum_{j \in |\mathcal{N}_i \cap \mathcal{U}|} \sum_{k \in |\mathcal{N}_i \cap \mathcal{U}|} c(j,k) - \frac{2}{m_i} \sum_{k \in |\mathcal{N}_i \cap \mathcal{U}|} c(i,k) \right)}_{\mathbf{B}}.$$

Note that the terms in **A** are related to the independent noise of each node, and thus are equivalent to the total prediction error of the NM (3.17). The terms in **B** are due to the smooth variations of the low-pass filtered noisy graph signal.

Some interesting remarks have been extracted from the analysis of (3.35) and (3.36). Nevertheless, as in the NM, the total prediction error (3.36) is minimized by minimizing the size of $\mathcal{P}$, and thus some constraint on the size of $\mathcal{P}$ is needed.

In next section we propose a Greedy algorithm that minimizes (3.36) for a given $|\mathcal{P}|$ and thus solves Problem 3.1.

3.3.3.1 Proposed MA Greedy Solution

To obtain the MA solution, we design a greedy algorithm that, as in the NM approach, first finds the $\text{SC}_{\mathcal{U}}$ solution and then, in each iteration, moves to $\mathcal{U}$ the node in $\mathcal{P}$ that minimizes E_{totMA}.

The philosophy of the algorithm is the same to that of the NM, explained in Section 3.3.2.1. Therefore, the algorithm finds the candidate $c \in \mathcal{P}$ that minimizes E_{totMA} by minimizing the changes in E_{totMA} from iteration $t-1$ to t (i.e., minimizing $\Theta^{\{t,c\}}$).

In the MA, the problem of minimizing $\Theta^{\{t,c\}}$ is defined as follows:

$$\min_{c \in \mathcal{P}} \Theta^{\{t,c\}} = \min_{c \in \mathcal{P}} \left(A^{\{t,c\}} - B^{\{t,c\}} - C^{\{t,c\}} \right), \tag{3.37}$$

where $A^{\{t,c\}}$ indicates the E_{totMA} in the neighborhood of candidate $c \in \mathcal{P}$ if it is changed to $\mathcal{U}$, $B^{\{t,c\}}$ the E_{totMA} before changing c, and $C^{\{t,c\}}$ the E_{totMA} reduction due to node c becomes $\mathcal{U}$ and thus is not taken into account in the E_{totMA} calculation.

Using (3.35), and considering that $v_\eta = v_\epsilon = v$ and $\alpha = 1$, $A^{\{t,c\}}$, $B^{\{t,c\}}$ and $C^{\{t,c\}}$ can be written in the minimization problem (3.37) as:

$$A^{\{t,c\}} = \sum_{k \in \mathcal{N}_c \cap \mathcal{P}} \left(\frac{1}{(m_k+1)} + \frac{1}{\mathcal{N}_{[k]}} + \frac{1}{(m_k+1)^2} \sum_{j \in |\mathcal{N}_k \cap \mathcal{U}|} \sum_{h \in |\mathcal{N}_k \cap \mathcal{U}|} c(j,h) - \frac{2}{m_k+1} \sum_{h \in |\mathcal{N}_k \cap \mathcal{U}|} c(k,h) \right), \tag{3.38}$$

$$B^{\{t,c\}} = \sum_{k \in \mathcal{N}_c \cap \mathcal{P}} \left(\frac{1}{(m_k)} + \frac{1}{\mathcal{N}_{[k]}} + \frac{1}{(m_k)^2} \sum_{j \in |\mathcal{N}_k \cap \mathcal{U}|} \sum_{h \in |\mathcal{N}_k \cap \mathcal{U}|} c(j,h) - \frac{2}{m_k} \sum_{h \in |\mathcal{N}_k \cap \mathcal{U}|} c(k,h) \right), \tag{3.39}$$

$$C^{\{t,c\}} = \frac{1}{(m_c)} + \frac{1}{\mathcal{N}_{[c]}} + \frac{1}{(m_c)^2} \sum_{j \in |\mathcal{N}_c \cap \mathcal{U}|} \sum_{h \in |\mathcal{N}_c \cap \mathcal{U}|} c(j,h) - \frac{2}{m_c} \sum_{h \in |\mathcal{N}_c \cap \mathcal{U}|} c(c,h). \tag{3.40}$$

Note that the number of $\mathcal{U}$ neighbors of every $k \in \mathcal{N}_c \cap \mathcal{P}$ increases from $B^{\{t,c\}}$ to $A^{\{t,c\}}$, because c becomes a new $\mathcal{U}$ neighbor for its $\mathcal{P}$ neighbors. Nevertheless, $\mathcal{N}_{[k]}$ does not change, because it just depends on the graph topology and not on the $\mathcal{U}/\mathcal{P}$ assignment. The same applies for $c(j,k)$, so that it can be calculated just once for every combination of nodes (j,k), obtaining the clustering degree matrix $\mathbf{C} = [c_{j,k}]$. As in the NM, $\Theta^{\{t,c\}}$ can be computed for every c with some matrix operations without using loops, obtaining the vector $\mathbf{\Theta}^{\{t,c\}} = [\Theta^{\{t,c=1\}} \Theta^{\{t,c=2\}} \dots \Theta^{\{t,c=|\mathcal{P}|_t\}}]$.

Details about the proposed MA greedy approach are in Algorithm 3.

Algorithm 3 MA Greedy Algorithm

Require: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $|\mathcal{P}|$ nodes
1: Calculate the clustering degree matrix $\mathbf{C} = [c_{j,k}]$
2: Calculate the $\text{SC}_{\mathcal{U}}$ solution
3: **while** $|\mathcal{P}|_t > |\mathcal{P}|$ **do**
4: Calculate $\mathbf{\Theta}^{\{t,c\}}$
5: Select the node c^* with minimum $\Theta^{\{t,c\}}$, $c^* = \min \mathbf{\Theta}^{\{t,c\}}$
6: Let $\mathcal{U} \leftarrow \mathcal{U} \cup \{c^*\}$
7: Let $\mathcal{P} \leftarrow \mathcal{P} \backslash \{c^*\}$
8: **end while**
9: **return** $\mathcal{U}/\mathcal{P}$ assignment

3.3.3.2 Experimental Results

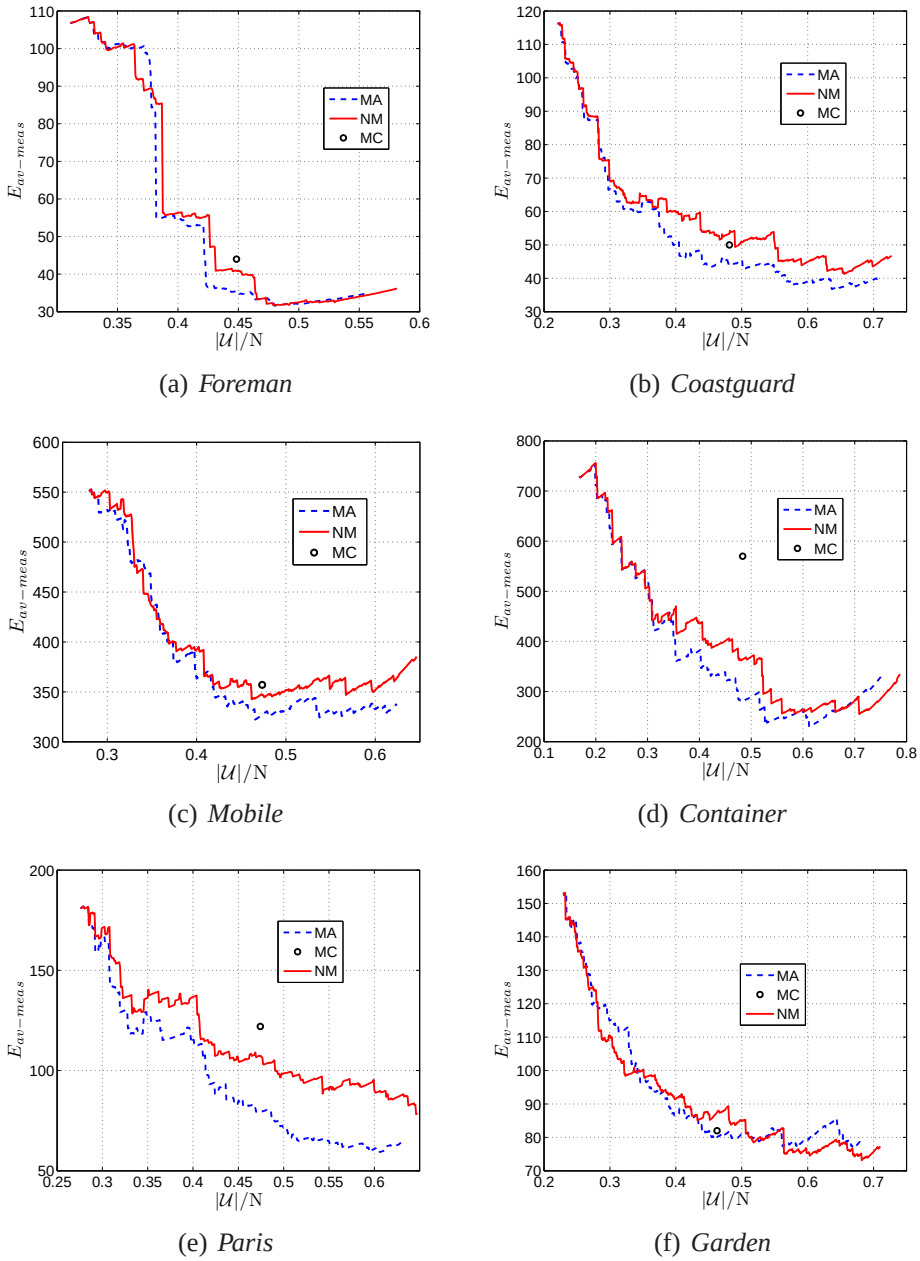
Next we present some experimental results in terms the quadratic prediction error in the first level of the transform ($j = 1$) when applying different $\mathcal{U}/\mathcal{P}$ assignment strategies, comparing MC, NM and MA.

The experiments have been carried out using the same video segments of Section 3.3.2. Nevertheless, in this case links of the graph between neighboring nodes with very different luminance values (i.e., links between neighbor nodes that cross contours) are removed, as illustrated in Figure 3.2. In this manner, assuming that the contours are well defined, **we have smooth luminance variations, which is the main hypothesis of the MA model**.

Summarizing, we have an unweighted graph as defined in (3.13) in which every node i could have very different number of neighbors ($|\mathcal{N}_i|$).

To evaluate the performance of each approach, we measure $E_{\text{av-meas}}$ as in (3.32), and to obtain the MA $\mathcal{U}/\mathcal{P}$ assignment solution we employ the algorithm described in previous section.

Figure 3.13 shows achieved $E_{\text{av-meas}}$ values as a function of the proportion of $\mathcal{U}$ nodes on the graph ($|\mathcal{U}|/N$) selected by each method. In the experiments, $v_\eta = v_\epsilon$ and $\alpha = 1$.



(a) *Foreman* (b) *Coastguard*

(c) *Mobile* (d) *Container*

(e) *Paris* (f) *Garden*

Figure 3.13: $E_{\text{av-meas}}$ for different sequences. MA Vs NM.

Some conclusions can be derived from the results shown in Figure 3.13:

- Considering smooth noise variations between neighboring pixels leads to a more realistic model and thus the $E_{\text{av-meas}}$ for the MA model is usually lower than for the NM and the MC solutions.

- MA considerably outperforms NM in regions where the luminance value of neighboring pixels changes smoothly (in both the spatial and the temporal domains) because the model is accurate (e.g., *Paris* or *Container*, after removing the high frequencies). On the other hand, in noisy areas, the results are similar to the ones achieved with NM (e.g., *Garden*).

- $E_{\text{av-meas}}$ for the NM approach is consistently lower in Figure 3.13 than in Figure 3.9 for the same video sequences and test conditions. This is due to the fact that, in the former, spatial links between neighbors that cross contours are broken (graph representation of Figure 3.2) while in the latter do not (graph representation of Figure 3.1). In other words, including the directional information helps to improve the prediction and thus to decrease the detail coefficient energy.

- As expected, $E_{\text{av-meas}}$ usually decreases as the number of $\mathcal{U}$ nodes on the graph increases.

As discussed before, in a video representation, temporal correlation will usually be stronger than spatial correlation, and thus pixels linked by means of "temporal links" will usually be more correlated than pixels linked by means of "spatial links". NM and MA models do not consider this fact.

3.3.4 Spatio-Temporal Model (STM)

The MA model can be extended to take into account that spatial and temporal neighbor pixels may have differentiated correlations. Next, we focus on a video representation, obtaining the expected value of the prediction error assuming that data is generated from temporal and spatial neighbors. This model will be referred to as Spatio-Temporal model (STM). The result could be generalized to the case where F different kinds of links, with different correlation values, are considered.

Definition 3.5. Spatio-Temporal Model

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph. Let x_i be a random variable that represents the data value associated to node i in the graph. Let us assume that

$$x_i = \left(\frac{w_s}{|\mathcal{N}_{[i]}^s|} \sum_{j \in \mathcal{N}_{[i]}^s} \epsilon_j + \frac{w_t}{|\mathcal{N}_{[i]}^t|} \sum_{j \in \mathcal{N}_{[i]}^t} \epsilon_j \right) + \alpha \eta_i, \tag{3.41}$$

where $\mathcal{N}_{[i]}^s$ and $\mathcal{N}_{[i]}^t$ are the closed sets of spatial and temporal neighbors, respectively, of node i; w_s is an arbitrary constant in $[0, 1]$, with $w_t = 1 - w_s$; ϵ_j and η_i are zero-mean independent random variables with variances v_{ϵ_j}, and v_{η_i}, respectively; and α is an arbitrary nonnegative real constant.

Definition 3.6. Weighted Predictor

Consider the predictions given by

$$\hat{x}_i = \frac{w_s}{m_i^s} \sum_{j \in \mathcal{N}_i^s \cap \mathcal{U}} x_j + \frac{w_t}{m_i^t} \sum_{j \in \mathcal{N}_i^t \cap \mathcal{U}} x_j, \tag{3.42}$$

where $m_i^s = |\mathcal{N}_i^s \cap \mathcal{U}|$ and $m_i^t = |\mathcal{N}_i^t \cap \mathcal{U}|$.

Let the *clustering degree* $c(m, n)$ be defined as in (3.34).

Define

$$D_{ab}^{cd}(i) = \sum_{j \in \mathcal{N}_i^a \cap \mathcal{U}} \sum_{k \in \mathcal{N}_i^b \cap \mathcal{U}} \frac{|\mathcal{N}_{[j]}^c \cap \mathcal{N}_{[k]}^d|}{|\mathcal{N}_{[j]}^c||\mathcal{N}_{[k]}^d|} \tag{3.43}$$

and

$$D_a^{cd}(i) = \sum_{j \in \mathcal{N}_i^a \cap \mathcal{U}} \frac{|\mathcal{N}_{[j]}^c \cap \mathcal{N}_{[i]}^d|}{|\mathcal{N}_{[j]}^c||\mathcal{N}_{[i]}^d|} \tag{3.44}$$

for a, b, c, d equal to "s" or "t".

Proposition 3.3. Spatio-Temporal Model Prediction Error

Let x_i and $\hat{x}_i$ satisfy Definition 3.5 and Definition 3.6 respectively. Consider that $v_{\eta_i} = v_\eta$ and that $v_{\epsilon_i} = v_\epsilon$ for any $i \in \mathcal{V}$. The prediction error of a node $i \in \mathcal{P}$ is given by

$$
\begin{aligned}
E_{ST_i} &= \mathbb{E}\{(x_i - \hat{x}_i)^2\} = \mathbb{E}\{(x_i)^2\} + \mathbb{E}\{(\hat{x}_i)^2\} - 2\mathbb{E}\{x_i \hat{x}_i\} \\
&= \underbrace{\alpha^2 v_\eta + v_\epsilon \left(\frac{w_s^2}{|\mathcal{N}_{[i]}^s|} + \frac{w_t^2}{|\mathcal{N}_{[i]}^t|} + \frac{2 w_s w_t}{|\mathcal{N}_{[i]}^s||\mathcal{N}_{[i]}^t|} \right)}_{\mathbf{A}} \\
&+ \underbrace{\alpha^2 v_\eta \left(\frac{w_s^2}{m_i^s} + \frac{w_t^2}{m_i^t} \right) + v_\epsilon \left(\frac{w_s^2}{(m_i^s)^2} G + \frac{w_t^2}{(m_i^t)^2} H + \frac{2 w_s w_t}{m_i^t m_i^s} I \right)}_{\mathbf{B}} \\
&- \underbrace{2\, v_\epsilon \left(\frac{w_s}{m_i^s} J + \frac{w_t}{m_i^t} K \right)}_{\mathbf{C}},
\end{aligned}
\tag{3.45}
$$

where

$$
\begin{aligned}
G &= w_s^2 D_{ss}^{ss} + w_t^2 D_{ss}^{tt} + 2 w_s w_t D_{ss}^{st}, \\
H &= w_s^2 D_{tt}^{ss} + w_t^2 D_{tt}^{tt} + 2 w_s w_t D_{tt}^{st}, \\
I &= w_s^2 D_{st}^{ss} + w_t^2 D_{st}^{tt} + w_s w_t (D_{st}^{st} + D_{st}^{ts}), \\
J &= w_s^2 D_{s}^{ss} + w_t^2 D_{s}^{tt} + w_s w_t (D_{s}^{st} + D_{s}^{ts}), \\
K &= w_s^2 D_{t}^{ss} + w_t^2 D_{t}^{tt} + w_s w_t (D_{t}^{st} + D_{t}^{ts}).
\end{aligned}
\tag{3.46}
$$

The proof is in Appendix B.2.

The terms in **A** consider the variance of the model; the terms in **B** represent the variance of the predictor; and the terms in **C** represent the correlation between model and predictor.

Expression (3.45) is quite similar to that of the MA model (3.35). The main difference is that in STM all the factors are weighted by spatio-temporal terms, thus taking into account the different statistical dependencies of temporal and spatial links. Therefore, for example, $\frac{1}{m_i}$ in (3.35) becomes $\frac{w_s^2}{m_i^s} + \frac{w_t^2}{m_i^t}$ in (3.45), or $\frac{1}{\mathcal{N}_{[i]}}$ in (3.35) becomes $\frac{w_s^2}{|\mathcal{N}_{[i]}^s|} + \frac{w_t^2}{|\mathcal{N}_{[i]}^t|} + \frac{2 w_s w_t}{|\mathcal{N}_{[i]}^s||\mathcal{N}_{[i]}^t|}$ in (3.45).

Finally, summing E_{ST_i} (3.45) over all nodes $i \in \mathcal{P}$, the total prediction error is:

$$E_{\text{totST}} = \sum_{i \in \mathcal{P}}^{|\mathcal{P}|} \mathbb{E}\{(x_i - \hat{x}_i)^2\} = \sum_{i \in \mathcal{P}}^{|\mathcal{P}|} E_{\text{ST}_i}. \tag{3.47}$$

Some other important remarks about expressions (3.45) and (3.47) are outlined below:

- Note that now E_{ST_i} strongly depends on the nature of the links between nodes. This way, for example, if $w_t >> w_s$ it will be useful to partition the graph so that m_i^t and $|\mathcal{N}_{[i]}^t|$ are large, in order to reduce the variance of the predictor and the observation, respectively.

- Likewise, observe that if one of the weights is much higher than the other, STM tends to be similar to the MA model. Therefore, if for example $w_t \to 1$ (and thus $w_s \to 0$), E_{ST_i} (3.45) $\to E_{\text{MA}_i}$ (3.35), ignoring the spatial links.

- Optimizing E_{totST} ignoring the terms G, H, I, J and K, gives rise to a similar result to Corollary 3.1 for the NM case, but now taking into account the different weights of spatial and temporal neighbors. That is, in order to minimize the detail coefficient energy, every node should have the same proportion of temporal and spatial update neighbors, and the right proportion depends on w_t and w_s (the higher w_t, the higher proportion of temporal update neighbors).

3.3.4.1 Proposed STM Greedy Solution

Next, we present a greedy algorithm that finds the STM solution. To that end, the algorithm moves to $\mathcal{U}$ the node in $\mathcal{P}$ that minimizes E_{totST} (3.47) in each iteration. For simplicity, we assume that the terms G, H, J, and K in (3.45) are insignificant[7].

Therefore, setting $v_\eta = v_\epsilon = v$ and $\alpha = 1$, the function to be minimized becomes:

[7] Note that, in our video representation examples, this is a reasonable assumption because some of the intersection between sets in the numerator of terms D_{ab}^{cd} and D_{ab}^{cd} are empty or small, and thus the terms are zero or close to zero.

$$\sum_{i\in\mathcal{P}}^{|\mathcal{P}|}\left(\frac{w_s^2}{|\mathcal{N}_{[i]}^s|}+\frac{w_t^2}{|\mathcal{N}_{[i]}^t|}+\frac{2w_sw_t}{|\mathcal{N}_{[i]}^s||\mathcal{N}_{[i]}^t|}+\frac{w_s^2}{m_i^s}+\frac{w_t^2}{m_i^t}\right). \tag{3.48}$$

As in the NM and MA greedy solutions, the algorithm finds the candidate $c \in \mathcal{P}$ that minimizes $\Theta^{\{t,c\}}$ (instead of (3.48)), defined as in (3.37), where $A^{\{t,c\}}$, $B^{\{t,c\}}$ and $C^{\{t,c\}}$ are conceptually the same as in the MA algorithm. Nevertheless, now, if c is moved to $\mathcal{U}$, its $\mathcal{P}$ neighbors have one more $\mathcal{U}$ neighbor to be used in the prediction, but it can be spatial or temporal (i.e., m_i^s and m_i^t do not increase their value in one from $B^{\{t,c\}}$ to $A^{\{t,c\}}$). As in the MA, $\Theta^{\{t,c\}}$ can be computed for every c with some matrix operations without using loops.

Details about the proposed STM greedy approach are in Algorithm 4.

Algorithm 4 STM Greedy Algorithm

Require: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $|\mathcal{P}|$ nodes, w_s, w_t
1: Calculate the SC$_{\mathcal{U}}$ solution
2: **while** $|\mathcal{P}|_t > |\mathcal{P}|$ **do**
3: Calculate $\mathbf{\Theta}^{\{t,c\}}$
4: Select the node c^* with minimum $\Theta^{\{t,c\}}$, $c^* = \min \mathbf{\Theta}^{\{t,c\}}$
5: Let $\mathcal{U} \leftarrow \mathcal{U} \cup \{c^*\}$
6: Let $\mathcal{P} \leftarrow \mathcal{P} \setminus \{c^*\}$
7: **end while**
8: **return** $\mathcal{U}/\mathcal{P}$ assignment

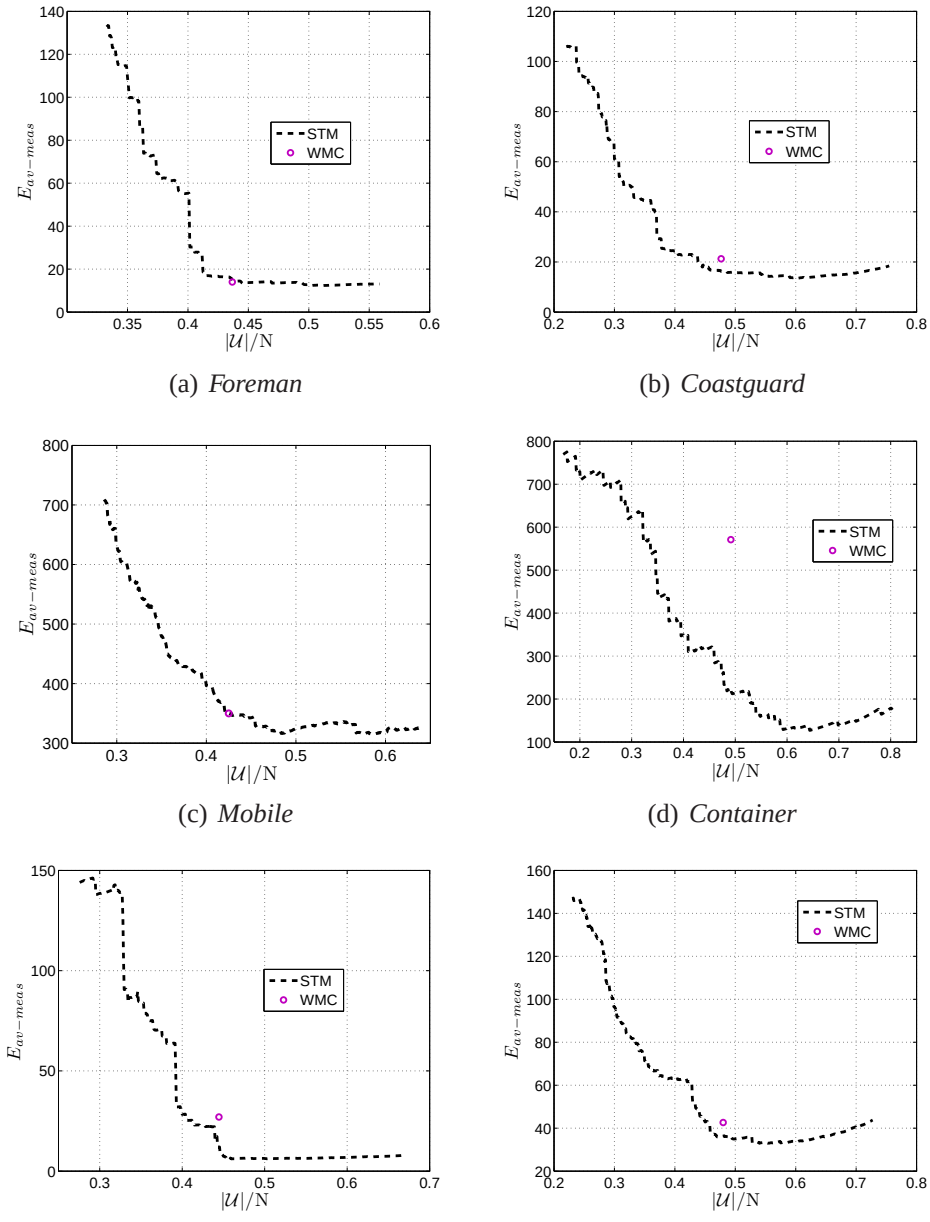
3.3.4.2 Experimental Results

In this section we compare $E_{\text{av-meas}}$ achieved when applying different $\mathcal{U}/\mathcal{P}$ assignment techniques to the same fragments of video sequences used in previous sections. Graph-based representations of these sequences are defined after disconnecting links between nodes across contours (Figure 3.2) as in Section 3.3.3.

In this case, we have an edge-weighted graph (with weights w_s and w_t for the spatial and temporal neighbors respectively) in which every node i may have a different number of spatial and temporal neighbors. The predictions are thus obtained following (3.42) (i.e., given different importance to temporal and spatial linked neighbors). The w_s and w_t weights used in the experiments were chosen using the method described in

Section 3.1.2.2. We compare the proposed WMC and STM solutions. To obtain the STM solution we employ the algorithm explained in Section 3.3.4.1.

Figure 3.14 shows some results of $E_{\text{av-meas}}$ as a function of the proportion of $\mathcal{U}$ nodes on the graph ($|\mathcal{U}|/N$) selected by WMC and STM solutions.



(a) *Foreman* (b) *Coastguard*

(c) *Mobile* (d) *Container*

(e) *Paris* (f) *Garden*

Figure 3.14: $E_{\text{av-meas}}$ for different sequences. STM.

Next, some conclusions extracted from Figure 3.14 are outlined:

- For the same number of $\mathcal{U}$ selected nodes, STM gives generally rise to lower $E_{\text{av-meas}}$ than WMC. This can be viewed as indicating that STM leads to $\mathcal{U}$ and $\mathcal{P}$ sets with better prediction ability (i.e., $\mathcal{P}$ set is better predicted from the $\mathcal{U}$ set) than WMC.

- WMC tries to maximize the cut between $\mathcal{U}$ and $\mathcal{P}$ sets. This generally leads to a certain number of $\mathcal{P}$ nodes having a large number of $\mathcal{U}$ correlated neighbors, while other nodes may not have any correlated neighbor, giving rise to good and bad predictions respectively. STM tries to obtain a solution in which every node has a balanced number of correlated neighbors, thus improving the mean prediction error when considering all nodes in $\mathcal{P}$.

- WMC obtains reasonably good results (except for *Container*), that are close to the STM solution. Thus, it can be considered a good heuristic.

- $E_{\text{av-meas}}$ obtained with STM (Figure 3.14) is greatly lower than the one obtained with the MA model (Figure 3.13), so it can be concluded that it is very important to take into account that temporal and spatial linked neighbors usually have different correlations. Specifically, this will influence the $\mathcal{U}/\mathcal{P}$ assignment and the predictors (filters) used to calculate the detail coefficients.

3.3.5 Discussion

In the NM, x_i is modeled as a noisy version of some constant c (3.15), which represents the mean luminance value. MA (3.33) and STM (3.41) do not consider any constant in their models for simplicity. Nevertheless, it can be proven that this fact does not affect the $\mathbb{E}\{(x_i - \hat{x}_i)^2\}$ calculation (i.e., the MSE expression obtained in Propositions 3.2 and 3.3 would be exactly the same if constant c is considered in the models).

Note that, for simplicity, we are assuming the same arbitrary constants (i.e., w_s and w_t) in the STM model generation (3.41) and in the predictors (3.42). A practical way to choose these constants is to use the optimal weight values of the graph as explained in Section 3.1.2.2.

When considering more than one decomposition level in the transform, the $\mathcal{U}/\mathcal{P}$ assignment solution at level $j = 1$ will influence the $\mathcal{U}/\mathcal{P}$ assignment at levels $j > 1$ and thus the global performance of the transform. Therefore, it would be useful to find optimal $\mathcal{U}/\mathcal{P}$ assignment designs by considering jointly several levels of the transform. This is an interesting research question, which is left for future work.

3.4 Filter Design

As discussed before, in the predict stage of the transform the data is decorrelated. To do that, each $\mathcal{P}$ node is linearly predicted from its $\mathcal{U}$ neighbors as $\hat{x}_i = \sum_{h \in \mathcal{U}_j} \mathbf{p}_i(h) x_h$, and the detail coefficient is obtained as $d_i = x_i - \hat{x}_i$. To obtain an efficient representation of the original data, it would be desirable that $x_i \approx \hat{x}_i$ and thus $d_i \approx 0$. Therefore, given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and two disjoint sets of $\mathcal{U}$ and $\mathcal{P}$ nodes, choosing a good prediction filter $\mathbf{p}$ is crucial to obtain accurate predictions of $\mathcal{P}$ nodes and thus a compact representation of the original data.

Finally, it is also necessary to define the update $\mathbf{u}$ filters to perform the update stage of the transform. In Section 3.4.2 we briefly describe the update filter design, which is based on the method proposed by [16].

Note that through all this section we assume a given weighted graph for which a bipartition of the graph (i.e., $\mathcal{U}/\mathcal{P}$ assignment), has been chosen.

3.4.1 Prediction Filter Design

In this section we outline some prediction filter designs for lifting transforms proposed in the literature and the peculiarities that arise in the context of **graph** lifting transforms. Then, we propose the design of prediction filters based on the given weighted graph.

3.4.1.1 Filter Design for Lifting Transforms on Graphs

The problem of optimizing prediction filters in lifting transforms has been considered by several authors, typically based on optimization criteria that seek to minimize the expected energy of the detail coefficients. In this way, [51] obtained the optimal predictors of an arbitrary lifting scheme and applied them to lossless image compression. [52]

minimized the energy of the detail coefficients through an additional predict stage, improving the compression performance of the transform. [25] designed a predict stage that minimized the expected energy of the detail signal in a generalized lifting scheme, and [53] proposed to jointly find the forward and backward motion vectors that minimized the energy of detail coefficients in a motion compensated 5/3 transform. [54] proposed to use adaptive filters to estimate the optimal prediction filters. This approach has the advantage that no side information is required to be sent to the decoder in a coding application. Nevertheless, in order for the decoder to reproduce the same prediction filters used at the encoder, both must have the same prediction errors and initial prediction filters. Note that, if quantization is used (e.g. in a lossy coding application), encoder and decoder must use the same quantized prediction errors to update the filters.

We focus on the design of prediction filters **in the context of graph lifting transforms**, in which every node can have an arbitrary number of neighbors of different classes (e.g., spatial or temporal neighbors in video representation). Therefore, calculating a different weight for each relative location as in $5/3$ or quincunx wavelets (e.g., w_1 for the left-side neighbor of every i, w_2 for the upper-side neighbor of every i, and so on) is not possible since every node i has an arbitrary number and location of its neighbors. Furthermore, is important to note that the proper choice of $\mathbf{p}$ depends on how data is correlated across nodes. With these observations in mind, we design filters that are based on the graph weights, which, in our case, represent an estimate of the correlation between nodes.

3.4.1.2 Graph-Based Filter Design

In Section 3.1.2, links on the graph were optimized in order to minimize the one-hop prediction error. Thus, prediction filters constructed from the graph weights should lead to accurate predictions. To obtain the prediction of pixel $i \in \mathcal{P}$, we define filters that weight its $\mathcal{U}$ neighboring pixels taking into account the weights of their respective connections to i.

Let us define the prediction filter for node $i \in \mathcal{P}$ as:

$$\mathbf{p}_i = \frac{[p_1\ , p_2\ , \ldots , p_k\ , \ldots , p_{m_i}]}{\sum_{k=1}^{m_i} p_k}, \tag{3.49}$$

where p_k is the prediction value associated to node $k \in \mathcal{N}_i \cap \mathcal{U}$ and m_i is the number of $\mathcal{U}$ neighbors of i (i.e., $|\mathcal{N}_i \cap \mathcal{U}|$). The normalization factor $\sum_{k=1}^{m_i} p_k$ is important to obtain a normalized predictor when fixed weights are used in the graph or to define prediction filters in higher levels of the transform[8].

In a video representation example, every link can be spatial ($\mathcal{S}$) or temporal ($\mathcal{T}$), with weights w_s and w_t, respectively. Let us define $m_i^{\mathcal{S}}$ (resp. $m_i^{\mathcal{T}}$) as the number of $\mathcal{U}$ spatial (resp. temporal) neighbors of i. Normalizing the weights by $m_i^{\mathcal{S}}$ and $m_i^{\mathcal{T}}$, respectively, p_k is obtained as:

$$p_k = \begin{cases} w_s/m_i^{\mathcal{S}}, & \text{if } ik \in \mathcal{S}, \\ w_t/m_i^{\mathcal{T}}, & \text{if } ik \in \mathcal{T}. \end{cases} \tag{3.50}$$

Note that this design leads to the prediction filters used in the STM (3.42). Also note that, if the nature of the links is not taken into account in the graph weighting, we have an unweighted graph and thus, $p_k = 1/m_i$, leading to predictors used in NM and MA (3.16). Chapter 4 shows experiments about the energy compaction achieved comparing filters constructed from fixed and optimal weights.

Considering F different kinds of links, p_k is defined analogously to the video representation example, so that, for the f-th kind of link, $p_k = w_f/m_i^f$ if $ik \in f$.

3.4.2 Update Filter Design

So far we have completely defined different $\mathcal{U}/\mathcal{P}$ assignments on the graph and the prediction filter design. In this section we focus on the design of the update filters, completely determining the graph transform. Our update filters are designed based on the method proposed by [16]. We briefly outline the **u** filter that we employ for the sake of completeness.

For each update node we design an update filter that is orthogonal to the prediction filters of its neighboring prediction neighbors. While the resulting update filters are not orthogonal to all the prediction filters, this solution reduces the impact of the "worst-case" coherence, because the prediction filters centered in prediction nodes that

[8] In higher levels of the transform $j > 1$, p_k at j will be calculated as the product of the weights in the path between connected nodes at $j-1$, as explained in Section 4.1.3.

are not neighbors have little or no common support with the given update filter. Other approaches for update filter design can be found in the literature [55], [56].

3.4.3 Discussion

Note that the predictors defined in (3.50) are very similar to that used for optimizing the weights of the graph in Section 3.1.2.2, and, therefore, provide a near optimal solution in the sense of minimizing the detail coefficient energy. The difference between both cases is that in (3.50) we use the $\mathcal{U}/\mathcal{P}$ bipartition information. Actually, the graph weights could be recalculated after the $\mathcal{U}/\mathcal{P}$ assignment as is discussed in Section 3.1.3, leading to optimal filters. Nevertheless, it is not worthy because weight values do not significantly change.

3.5 Summary of the Properties of the Transform

We now focus on analyzing the main features of the proposed N-dimensional directional transform. Some of these features, such as invertibility (perfect reconstruction) are derived by the lifting-based construction of the transform (i.e., these features are inherent to the lifting scheme). Other characteristics, such as energy compaction and frequency and spatio-temporal (original domain) localization, depend on the $\mathcal{U}/\mathcal{P}$ assignment, the **u** and **p** filters design, and the graph construction.

Next, we outline some properties of the proposed transform:

- **Perfect Reconstruction Transform.**

  Since the transform is based on the lifting scheme, it is guaranteed to be invertible if the $\mathcal{U}$ and $\mathcal{P}$ sets are disjoint sets, as it was explained in Chapter 2. Note that all the $\mathcal{U}/\mathcal{P}$ assignment methods discussed in this thesis give rise to $\mathcal{U}/\mathcal{P}$ disjoint sets.

- **Easy generalization to N-dimensional domains.**

  The transform can be applied to any arbitrary graph. Besides, one can easily reflect correlations of N-dimensional signals using the graph representation of data

explained in 3.1.1. Therefore, obtaining an N-dimensional transform is straightforward, just constructing the graph representation of the N-dimensional signal and applying the lifting transform on this graph. Furthermore, it gives rise to a simple process in which the formulation and the conceptual idea do not become complicated as the dimensionality of the input signal is higher.

- **Useful for irregularly spaced sample grids.**

  The graph representation of data is versatile enough to cope with data sampled in an irregular grid, which is common to many applications such as WSNs. Then, given the graph, the transform operates on it in the usual way.

- **Any feasible filtering direction.**

  Thanks to the total freedom in the graph construction (i.e., one can link any node (sample point) to any other node (or set of nodes)), the proposed transform allows filtering operations in any direction with no restrictions.

- **Non-separable filtering operations.**

  In the graph representation of a signal, one node can have an arbitrary number of different kinds of neighbors (e.g., spatial and temporal neighbors in a video representation). Then, the filtering operations are performed using the available neighbors of every node, giving rise to non-separable filtering operations in which all types of neighbors are jointly considered (e.g., in a video representation, this gives rise to spatio-temporal filtering), in contrast to the "separable" way, in which filtering operations are performed separately in each direction.

- **One-dimensional filtering operations.**

  Independently of the dimensionality of the original signal, once we obtain its graph representation, the resulting predict and update filtering operations to perform the transform are one-dimensional operations (2.1).

- **Critically-sampled transform.**

  Given two $\mathcal{U}/\mathcal{P}$ disjoint sets, the proposed transform is critically sampled (independently of the number of levels of decomposition of the transform J) in the sense that it generates the same number of coefficients than samples of the original signal, avoiding redundancy in the representation.

3.6 Conclusions

In this chapter we have discussed different strategies for the optimization of lifting transforms on graphs.

First, we have explained the graph construction, which involves the graph representation of an N-dimensional signal and the graph weighting. The directionality of the transform is determined by the graph representation as long as the filtering operations

are performed through linked nodes. Regarding the graph weighting, we have proposed two methods: (i) assuming fixed weights; and (ii) optimizing the weights in order to minimize the quadratic prediction error when using one-hop predictors and considering F kinds of links with differentiated statistical properties.

In this chapter we have also investigated the $\mathcal{U}/\mathcal{P}$ assignment process, discussing two different approaches to find a suitable bipartition of the graph in order to minimize the detail coefficient energy: (i) based on the given weighted graph and (ii) based on signal models.

Graph-based $\mathcal{U}/\mathcal{P}$ assignment methods find bipartitions without making any assumption about the graph signal. In this way, we have proposed a solution which relies on the next intuition: if weights of the graph represent similarity between nodes (i.e., similar luminance value), the WMC maximizes the similarity between $\mathcal{U}$ and $\mathcal{P}$ node sets. Signal model-based $\mathcal{U}/\mathcal{P}$ assignment methods are optimal in the sense that, given an arbitrary graph and a data generation model, the average detail coefficient energy is minimized. Three data generation models have been proposed, namely: (i) the NM, which assumes that the value of each node on the graph is a noisy version of a constant; (ii) the MA model, which considers smooth variations between neighbor nodes; and (iii), the STM model, that considers different statistical properties for spatial and temporal neighbors. We have experimentally shown that the WMC is a good method for coding applications, since it reaches near optimal solutions with less complexity than signal-model based approaches.

We have also described the update and prediction filter design, which is based on the weights of the graph. Finally, the main properties of the proposed N-dimensional transform have been summarized.

Chapter 4

Video Coding Application

In this chapter we describe the application of our proposed graph-based lifting transforms to video coding. As discussed in Chapter 1, the key novelty in our approach is describing the video sequence as a weighted graph of connected pixels and applying the lifting transform on this graph.

The connections in the graph are constructed in such a way that pixels expected to have similar luminance tend to be connected. These connections can be temporal or spatial, and the number of neighbors that one pixel can have in the graph can vary locally. Therefore, we can have flexibility in designing the corresponding spatio-temporal filtering operations, which can be selected to follow spatio-temporal directions of high correlation. To achieve a more accurate prediction, the connection between any pair of pixels is weighted as a function of estimates of correlation between the pixels.

Our work could be considered as a generalization of wavelet-based video coding. In particular, our proposal gives rise to a more versatile solution where spatial and temporal operations are no longer separable. The transform requires that some side information be sent to the decoder, so that the same graph can be constructed at both encoder and decoder. Specifically, temporal information (motion vectors) and spatial information (contours) have to be sent. Most of the work described in this chapter was published in [43], [44] and [42].

This chapter is organized as follows. In Section 4.1 we present our proposed graph-based transform for video coding and evaluate its energy compaction ability in comparison with other schemes. Furthermore, we show how the proposed transform can overcome classical problems that arise in MTCF approaches (e.g., LIMAT), such as their poor performance in uncovered areas. Once we experimentally prove the efficiency of our scheme in terms of energy compaction, we move towards a complete encoder in Section 4.2, describing a new reordering approach to sort the coefficients before they are entropy coded, and discussing low-complexity versions of the transform. Finally, in Section 4.3, we present how to apply rate-distortion optimization to our coding scheme.

4.1 Graph-Based Transform for Video Coding

The processes needed to perform lifting transform on graphs (graph construction, $\mathcal{U}/\mathcal{P}$ assignment, and filter design) were studied in Chapter 3. In this section we give some details about these processes applied to video coding. Graph construction is defined in Section 4.1.1, while $\mathcal{U}/\mathcal{P}$ assignment and filter design are described in Section 4.1.2. In Section 4.1.3, we discuss how to obtain a MRA of the original signal extending the transform to J decomposition levels. In Section 4.1.4, we evaluate its performance in non-linear approximation terms (which allows to estimate the energy compaction performance of the transform, and does not depend on other typical encoders processes such as quantization or entropy coding) and compare it with the LIMAT approach [6] and with a simple DCT based video encoder (which is the basis of the latest video coding standards). Finally, we compare the performance in uncovered areas of the proposed scheme and the LIMAT in Section 4.1.5.

4.1.1 Graph Construction

The goal in the construction of the graph at the j-th level of decomposition is to link pixels with similar luminance values, so that detail coefficients $d_{m,j}$ in (2.1) are very close to zero. In this manner, the energy of the high pass subband at this level j will be low, achieving an efficient representation of the data. First, we explain how to form the graph at the $j = 1$ level of decomposition from the original video sequence. Then, in successive levels $j > 1$, we construct the graph at level j from the graph at level $j - 1$ as explained in Section 4.1.3.

Consider a video sequence of V frames of size $L \times H$ and a subsequence of K frames ($K \leq V$). We will employ a new graph representation for every subset of K frames, until all the V frames in the sequence are coded. Let $\{x_k\}_{k=1}^{L\times H\times K}$ be the luminance value of pixels $k \in \mathcal{O} = \{1, 2, ..., L \times H \times K\}$, whose graph representation is $\mathcal{G} = (\mathcal{O}, \mathcal{E})$ so that any pixel $k \in \mathcal{O}$ can be linked to any subset of pixels $\mathcal{H} \subset \{\mathcal{O} \backslash \{k\}\}$, following criteria to be described next. Since we exploit the spatial and temporal correlation jointly, a pixel g can be linked to spatial and temporal neighbors at the same time.

With respect to the spatial correlation, the criterion for graph construction is very similar to that employed in [5] for image compression. Pixels that are close to each other and, in general, pixels that belong to the same object, will tend to have correlated luminance values. In contrast, when filtering across contours, there can be a significant amount of energy in the high pass subbands, because the value of neighboring pixels can be very different. Thus, if we avoid filtering across the contours, we are more likely to obtain a more compact representation of the data. Following this reasoning, we link those pixels that are one-hop neighbors in any direction and do not cross any contour. To do that, we need to estimate the contours and send this information to the decoder. To reduce the resulting overhead, we note that if there are no occlusions and the motion model captures object motion accurately, it is possible to estimate the contours of the current frame using contour data obtained from the reference frame along with motion information. Thus, in practice we only need to explicitly send contour information to the decoder once every K frames.

Regarding the temporal correlation, we link those pixels that are related by means of a motion model. In our case, block matching is used, and every pixel belonging to a block is linked to the corresponding pixel belonging to the best block match in the reference frame. Motion vectors (MV) need to be sent to the decoder in order to describe the motion. Finally, note that motion mappings are estimated using the original video frames, that is, the reference frame is not a reconstruction from a previously encoded frame as in the latest video coding standards such as H.264/AVC and H.265/HEVC (High Efficiency Video Coding). An example of graph construction and contour information transmission is shown in Figure 4.1 for two frames, where it can be seen that links between pixels follow the motion direction and avoid crossing contours within a frame.

4.1.1.1 Graph Weighting

As discussed in Section 3.1.2, the weights of the graph are used in the design of the $\mathcal{U}/\mathcal{P}$ assignment process and the **p** and **u** filters, and in the construction of the graph in successive levels of decomposition, thus helping improve prediction at all levels. Furthermore, this weighting will be useful to reorder the coefficients before they are arithmetically coded.

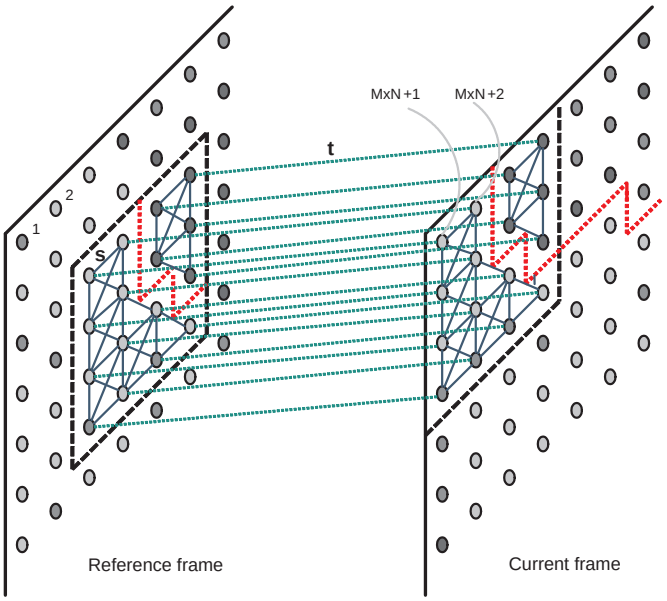


Figure 4.1: Spatio-temporal graph construction. The grey level represents the luminance value of each pixel; the red-thick dashed lines are the object contours; the green-fine dashed lines represent temporal connections, and the blue solid lines spatial connections. Finally, the black dashed lines represent the block size.

As a starting point, fixed weights are used as described in Section 3.1.2.1. Given that temporal links are identified using ME, the expected correlation between temporal-linked pixels is higher than that between spatial-linked pixels. In particular, we experimentally set $w_t = 10$ for temporal connections and $w_s = 2$ for spatial connections.

4.1.2 $\mathcal{U}/\mathcal{P}$ Assignment and Filter Design

We have proposed in Section 3.3 some $\mathcal{U}/\mathcal{P}$ assignment strategies that minimize detail coefficient energy under certain data generation models, and we have compared them to graph-based $\mathcal{U}/\mathcal{P}$ assignment strategies described in Section 3.2. As we concluded, using model-based solutions lead to lower detail coefficient energy for a given number of $|\mathcal{P}|$ nodes. Nevertheless, these solutions are more computationally expensive than MC and WMC because they need more complex greedy algorithms. Another relevant conclusion extracted from analysis in Section 3.3.4 is that it is very important to include spatial and temporal information to perform the $\mathcal{U}/\mathcal{P}$ assignment. Furthermore, we

concluded that the WMC solution is a good approximation to the optimal solution under the assumed spatio-temporal data generation model.

Summarizing, given the lower computational cost and the near-optimal performance of the WMC, we use it as criterion to assign a label to each pixel in every level of the transform j, obtaining the $\mathcal{P}_j$ and the $\mathcal{U}_j$ disjoint sets. To compute the WMC solution we use the greedy approach described in Algorithm 1. An example of the $\mathcal{U}/\mathcal{P}$ assignment for two levels of decomposition is shown in Figure 4.2. Note that the $\mathcal{U}$ nodes are usually connected by means of reliable links to $\mathcal{P}$ nodes, so we can obtain an accurate prediction of these $\mathcal{P}$ nodes from the $\mathcal{U}$ nodes. Discarded links (same label connected pixels) are indicated as broken links.

Finally, to obtain the detail coefficient in a prediction pixel $i \in \mathcal{P}$, we define the filters as in Section 3.4.1, thus obtaining robust prediction filters that weight the $\mathcal{U}$ neighbor pixels taking into account the reliability of each of their connections to i. The update **u** filters are designed as was explained in Section 3.4.2.

4.1.3 Extending the Transform to Multiple Levels of Decomposition

In order to carry out a multiresolution analysis, the low pass coefficients are successively projected in different transformation levels onto smooth and detail subspaces. To obtain the graph at transformation level j from the graph at level $j-1$, we connect those $\mathcal{U}$ nodes that are directly connected or at two-hop of distance in the graph at level $j-1$, so that the simplified graph continues to capture the correlation between pixels. If the link exists at level $j-1$ then the corresponding link at level j inherits the same weight. Alternatively, if two nodes are linked that were two hops away at level $j-1$ then the corresponding link weight is the product of the weights in the path between connected nodes at level $j-1$. Once we have constructed the graph at level j, we should split the nodes again into prediction ($\mathcal{P}_j$) and update ($\mathcal{U}_j$) disjoint sets in order to perform the transform. Figure 4.2 shows an example of graph construction at level j from a graph at level $j-1$, and the $\mathcal{U}/\mathcal{P}$ assignment at both transformation levels.

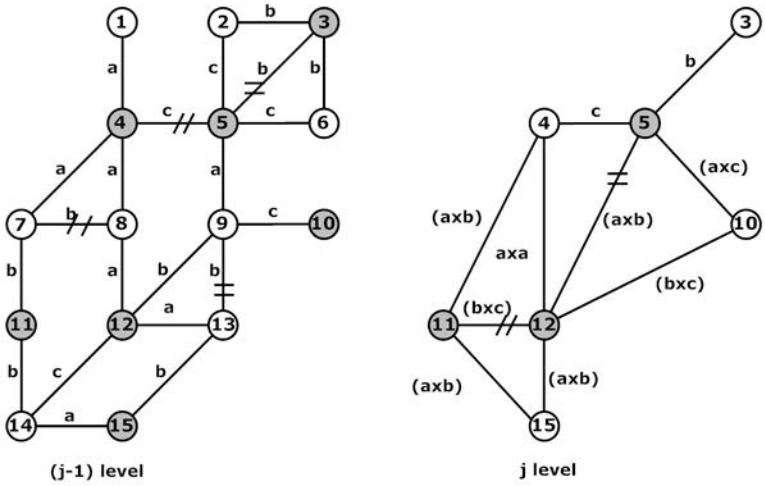


Figure 4.2: Graph construction for consecutive levels of decomposition. $a = 10$, $b = 5$ and $c = 3$ are the different weight values. Grey nodes are $\mathcal{U}$ nodes, and white ones are $\mathcal{P}$ nodes.

4.1.4 Experimental Results

To evaluate the performance of the proposed transform, we employ the ***k*** term non-linear approximation (outlined in [4]), which consists of keeping the ***k*** largest coefficients of the transform and setting the rest to zero. This is a good indicator of energy compaction ability of the transform (and thus of the potential coding performance). We compute the average PSNR of each sequence consisting of $V = 100$ frames as a function of the percentage of retained coefficients.

In our experiments, five levels of decomposition of the transform are performed on the constructed graphs. Our method is compared to the Haar version of the MCTF approach described in [6] (the LIMAT method), and to a motion-compensated discrete cosine transform (DCT)-based video coder. In the DCT-based coder, the residual image, obtained after block motion estimation (ME) and compensation processes, is transformed by a 8×8 DCT. This scheme is the basis of the latest video coding standards such as H.264/AVC or H.265/HEVC.

Given that our purpose is to evaluate the compaction ability of the different transforms keeping its ***k*** largest coefficients and measuring the quality of the reconstructed signals, side information is not taken into account in these first results. Nevertheless,

note that in the proposed method we will have an overhead associated with the temporal and spatial information needed to construct the graph at the decoder. Regarding the temporal overhead, the same motion model is employed in *all* compared methods, i.e., a standard motion vector on 8×8 pixel blocks is assumed (only one reference frame), and thus this overhead does not need to be considered in the comparison. Regarding the spatial information overhead, we choose $K = 20$ and assume that a binary contours map (obtained using Roberts' gradient operators) is sent to the decoder once every K frames, so that the spatial side information will be very low, as we will see in the rate-distortion experimental results provided in Section 4.2.5.

Note that, as discussed in Section 3.1, there exists a trade-off between how accurately the graph captures correlation information and the side information needed to construct the graph. A higher rate to describe the spatial and temporal information (e.g., very small block sizes for motion) means that the correlation between linked pixels is also better captured by the graph, leading to potential compression gains. The weights used in these experiments are $w_t = 10$ and $w_s = 2$.

Figure 4.3 shows PSNR as a function of percentage of retained coefficients of three different QCIF sequences, *Mobile*, *Carphone* and *Foreman*. The proposed method outperforms the DCT and the LIMAT transforms. In the *Mobile* sequence, when 40 percent of coefficients are retained, our method is 7 dB and 4 dB better than the DCT and the LIMAT, respectively. However, the LIMAT is better than the proposed one when a very small percentage of coefficients are retained for the *Mobile* sequence. One possible reason could be that we may have to chosen spatio-temporal filtering directions worse than the temporal-only ones chosen by the LIMAT.

For subjective evaluation, Figure 4.4 shows the original version of the frame number 12 of the sequence *Mobile* (upper-left part) and the reconstruction obtained from the DCT transform applied on the residual (upper-right part), LIMAT (lower-left part), and the proposed method (lower-right part). The reconstruction is carried out from the 20 % of retained coefficients. It can be seen that our transform achieves significantly better perceptual quality than the DCT, and slight improvements over LIMAT (see for example the three animals of the upper-left part of the frames).

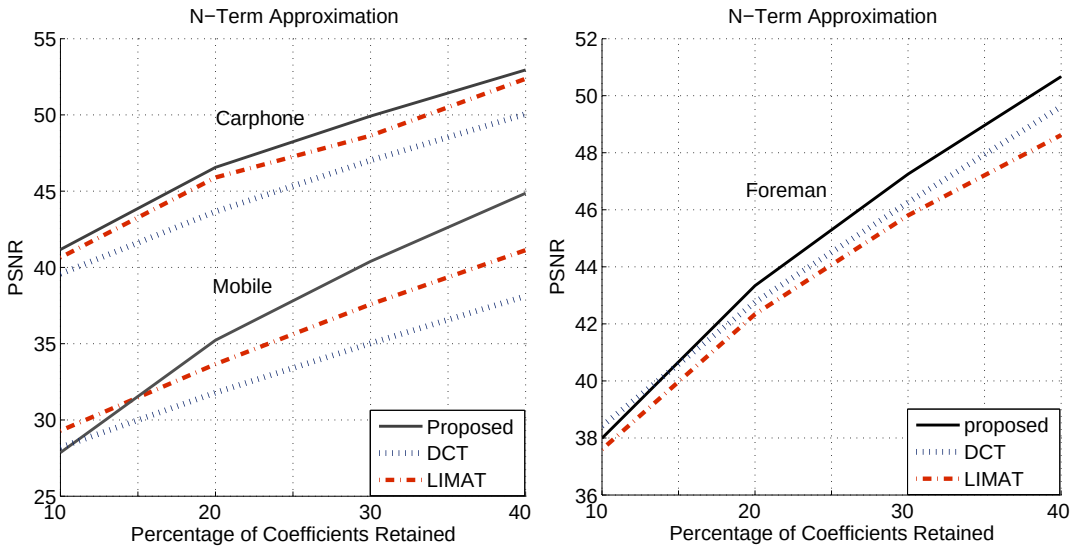


Figure 4.3: PSNR versus percentage of retained coefficients.

Table 4.1: Comparison of LIMAT and the proposed transform coding different areas.

	PSNR(dB) in Area 1	**PSNR(dB) in Area 2**
Proposed	43.1	36
LIMAT	42.4	33.3
Δ	0.7	2.7

4.1.5 Performance in Uncovered Areas

To further explain the advantages of the proposed scheme we now consider in more detail situations involving uncovered areas. Figure 4.5 shows the motion mappings used by the Haar version of the LIMAT approach with two levels of decomposition. Prediction frames (P) will be filtered following the directions indicated by the MV, and update frames (U) will be updated using inverse mappings MV^{-1}. Grey pixels represent non-updated pixels in the $j-1$ level of decomposition, that is, pixels that have not been low-pass filtered and thus contain high frequency energy. This high frequency content will not be removed using the smooth coefficients at j level, giving rise to inefficiency. The black pixel represents a pixel that has not been decorrelated at any level, so that the coefficient after both levels of decomposition will be the "raw" original pixel, instead

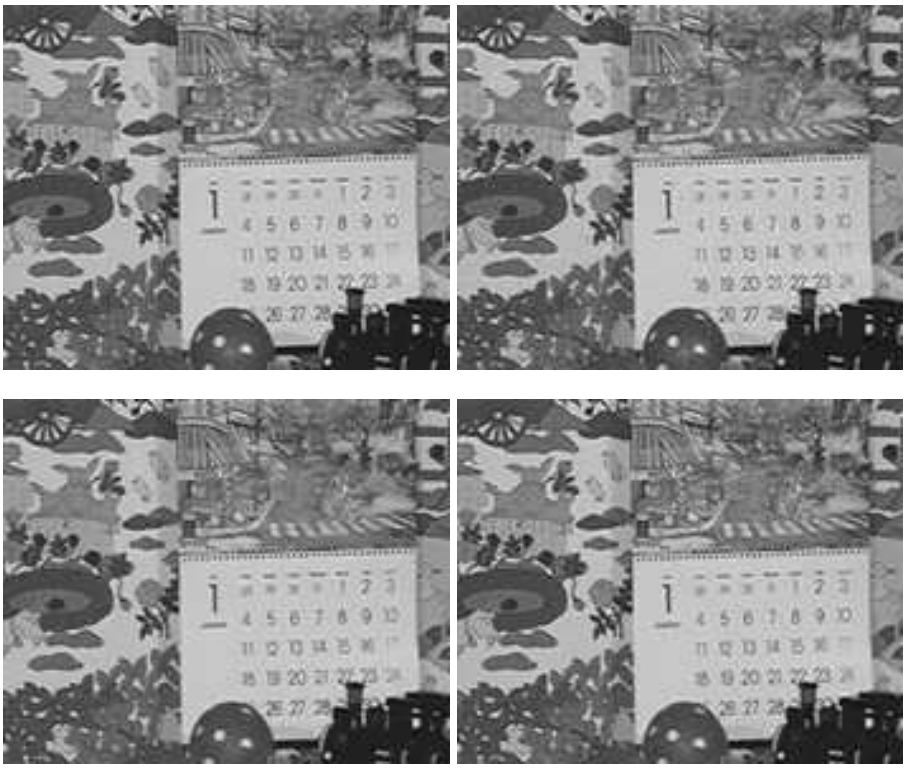


Figure 4.4: Original (upper-left) and reconstruction with 20 % of the transform coefficients from the DCT applied on the residual image (upper-right), LIMAT (lower-left) and the proposed method (lower-right).

of a transform coefficient. The proposed method can solve this problem by representing video information as a graph (Figure 4.1) leading to a versatile $\mathcal{U}/\mathcal{P}$ assignment, in which $\mathcal{P}$ and $\mathcal{U}$ nodes can belong to the same frame. To illustrate this statement, we have encoded two different 32×32 pixel areas of the sequence *Foreman*. Area 1 starts at pixel (1,1), so that could be considered a fairly static area. Area 2 starts at pixel (80,80), corresponding to a very dynamic area (the face of the man). The results in terms of PSNR when the 20 % of the coefficients are preserved are given in Table 4.1. The proposed method obtains slightly better results than LIMAT in Area 1, while it significantly outperforms LIMAT in Area 2, where there is a lot of motion and the uncovered background problem manifests itself.

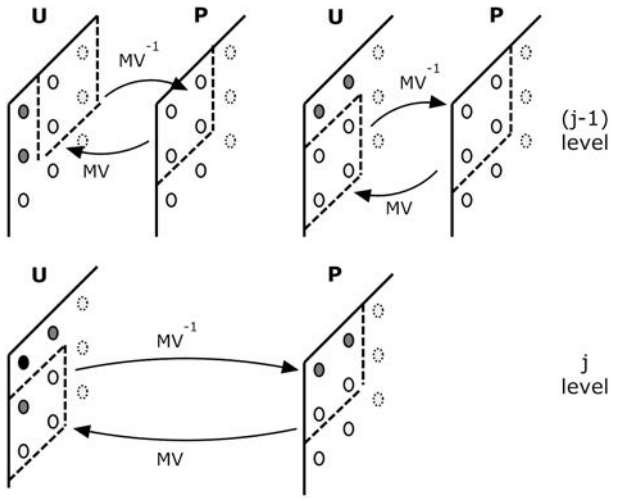


Figure 4.5: Uncovered areas in LIMAT.

4.2 Towards a Complete Encoder

So far we have evaluated the performance of the proposed graph-based transform for video coding, obtaining promising results in non-linear approximation terms. Nevertheless, in practical encoders, coefficients of the transform are reordered, quantized and entropy coded (together with the side information) thus obtaining a bitstream of specific rate R.

In this section a complete graph-based transform video encoder is proposed. To this end, in Section 4.2.1 we present a new reordering technique to be applied in our graph transform in order to sort the coefficients and thus increase the coding efficiency. Then, in Section 4.2.2, we obtain the optimal weights as a function of the video content as was discussed in Section 3.1.2.2, and compare the coding performance when using these optimal weights and the fixed weights of previous section. Furthermore, in order to reduce the high complexity of our encoder (especially of the $\mathcal{U}/\mathcal{P}$ assignment process), we design two low-complexity versions of this process that work (i) with sub-graphs formed from the original graph and (ii) in a distributed manner. This is presented in Section 4.2.4. Finally, rate-distortion results are provided to evaluate the performance of our coding scheme and compare it with a DCT-based encoder in Section 4.2.5.

4.2.1 Coefficient Reordering

In typical practical encoders, quantized transform coefficients are scanned in certain order before applying entropy coding. For example, in DCT-based encoders, the reordering is usually performed following a zigzag scanning order within each block, while in wavelet-based approaches, bitplane by bitplane scanning of transform coefficients has been a popular approach [57], [58]. We next propose two different approaches to re-order the coefficients generated by our graph-based transform: (i) inter-subband reordering, which implies sorting the coefficients as a function of the subband to which they belong; and (ii) intra-subband reordering, which sorts the coefficients of a subband as a function of the reliability with which they were predicted.

4.2.1.1 Inter-subband reordering

Because our transform achieves significant energy compaction, the energy in the middle-high frequency subbands tends to be very low, so that these sub-bands likely have a large number of zero coefficients after quantization. Based on this, we group coefficients that belong to the same subband, increasing the probability of having long strings of zero coefficients. Specifically, the coefficients are sorted as $\mathbf{coeffs}_{inter} = \left[\mathbf{s}^{j=J}, \mathbf{d}^{j=J}, \mathbf{d}^{j=J-1}, \ldots, \mathbf{d}^{j=1}\right]$, where $\mathbf{s}^{j=J}$ are the smooth coefficients at level of decomposition $j = J$ (the lower frequency subband), and $\mathbf{d}^{j}$ are the detail coefficients at a generic level of decomposition j. Refer to Figure 4.6 for an example of the effect of ordering on quantized coefficients from 20 frames of the sequence *Carphone*.

4.2.1.2 Intra-subband reordering

The graph is known at both encoder and decoder. Its edge weights provide an estimate of the reliability with which one $\mathcal{P}$ node is predicted from $\mathcal{U}$ neighbors. We make the assumption that the magnitude of detail coefficients in $\mathcal{P}$ nodes tend to be smaller if they have been predicted from more "reliable" $\mathcal{U}$ neighbors (i.e., prediction is better). Thus, we propose to reorder the coefficients in each subband according to the reliability of their links, grouping together the more reliably predicted nodes, which likely lead to smaller magnitude detail coefficients. An example of this reordering is shown in Figure 4.7. In the example, the detail coefficients (white nodes) of a generic subband j, $\mathbf{d}^{j} = [1, 2, 3, 5, 6, 7]$, have the following reliability values,

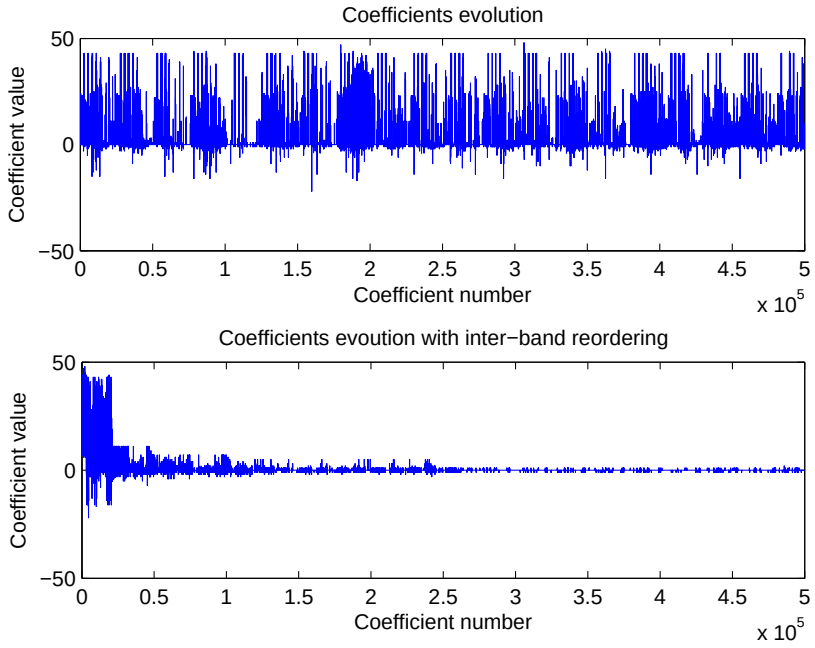


Figure 4.6: Inter-subband reordering example. Top: original coefficients. Bottom: reordered coefficients.

Table 4.2: Performance comparison using inter-subband and intra-subband reordering.

	Without reordering	**Inter reordering**	**Inter and Intra reordering**
Foreman	503 Kbps	404 Kbps	350 Kbps
Carphone	502 Kbps	425 Kbps	371 Kbps

$\mathbf{r}^j = [a, 2a, a, 3a/2, (a+b)/2, 3b/3]$, respectively, calculated as the average of the weights of all graph edges used to compute that coefficient. Assuming that $a > b$, this gives rise to the following intra-subband reordered coefficients: $\mathbf{d}^j_{intra} = [7, 6, 1, 3, 5, 2]$. Figure 4.8 shows a real example of the detail coefficients at decomposition level $j = 4$ in the sequence *Carphone*. The upper part of the figure shows the quantized coefficients vector without reordering, and the lower part shows the coefficients after the intra-subband reordering.

Table 4.2 shows bit rates (Kbps) after coding 20 frames of the sequences *Foreman* and *Carphone* at different qualities (32.9 dB and 36 dB, respectively) without reordering the coefficients, employing inter-reordering, and inter and intra reordering. The rate is obtained with an arithmetic coder as is explained in Section 4.2.5.

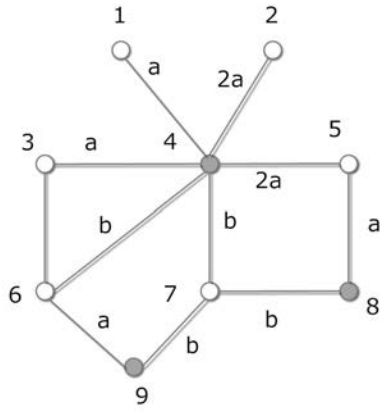


Figure 4.7: Intra-subband reordering example.

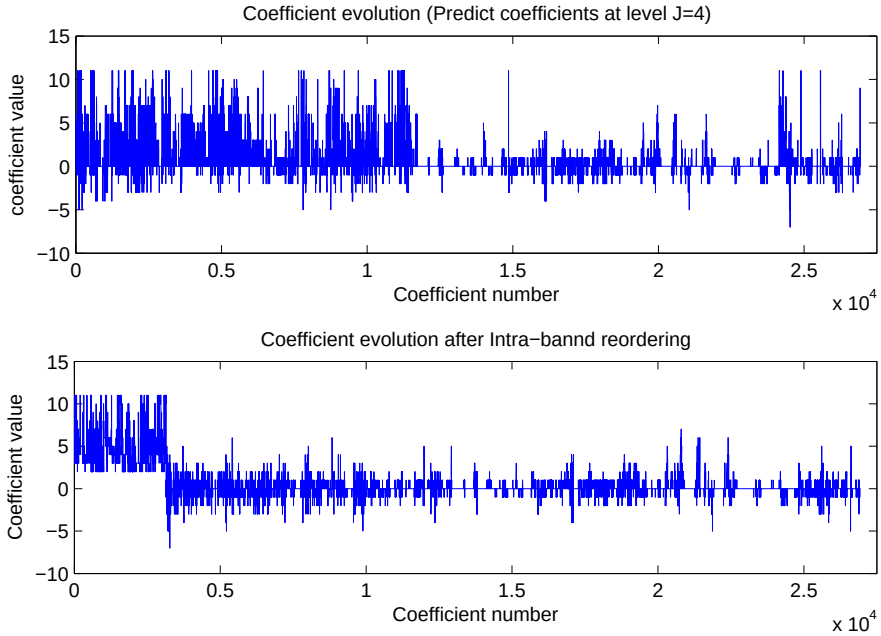


Figure 4.8: Intra-subband reordering. Predict coefficients at decomposition level j=4.

4.2.2 Optimal Weighting Vs. Fixed Weighting

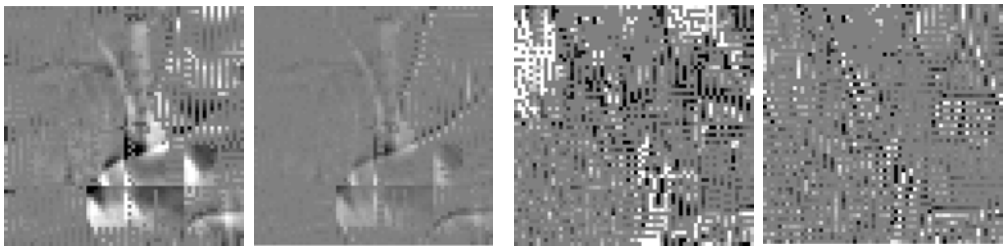
In the first approximation to the application of the transform to video coding (Section 4.1) the weights on the graph were experimentally fixed, with values $w_t = 10$ and $w_s = 2$ for temporal and spatial links, respectively. In this section we compare the detail coefficient energy obtained using these fixed weights and the optimal weights (calculated as discussed in Section 3.1.2.2). To perform the prediction stage of the transform, we use the prediction filters defined in Section 3.4.1.

Table 4.3: Comparison between different weightings. Detail coefficient energy per coefficient in $j = 1$: $E_{d_{j=1}}$.

	Carphone	***Mobile***	***Airshow*** **(scene cut)**	***Football*** **(fast motion)**
$E_{\textbf{av-meas},j=1}$ $\mathbf{w} = (w_s, w_t) = [2, 10]$	14	44	34	408
$E_{\textbf{av-meas},j=1}$ $\mathbf{w}^*$	12	37	17	240

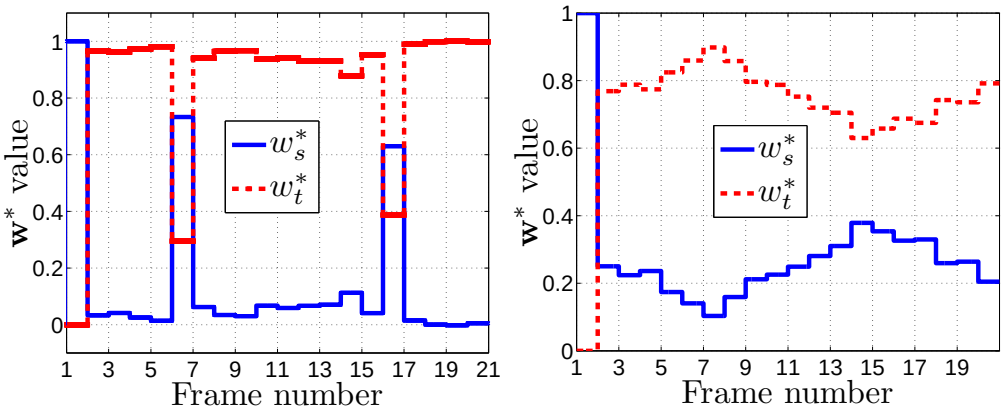
Table 4.3 shows examples of detail coefficient energy normalized by the number of $\mathcal{P}$ nodes in the first level of the transform $j = 1$ $\left(E_{\text{av-meas, j=1}} = \frac{1}{|\mathcal{P}_{j=1}|} \sum_{m \in \mathcal{P}_{j=1}}^{|\mathcal{P}_{j=1}|} d^2_{m,j=1}\right)$ obtained coding 20 frames using the optimal weights (**calculated in a frame-by-frame basis**) and using the fixed weights. Note that $E_{\text{av-meas, j=1}}$ is lower when the optimal weights are used for all the considered cases.

Figure 4.9 shows the detail coefficient values obtained using the optimal weights (right part of each subfigure) and the fixed weights (left part of each subfigure). The example corresponds to a region of a specific frame of *Airshow* (scene cut) and *Football* (fast motion). It can be seen that the absolute value of the detail coefficients is lower when using the optimal weights. Specifically, in the scene cut of *Airshow*, we have $(w_s^*, w_t^*) = (0.7, 0.3)$, and thus the filtering mainly follows the spatial directions, giving rise to better predictions and lower detail coefficients energy. The evolution of the (w_s^*, w_t^*) values is shown in Figure 4.10. Observe that, in *Airshow*, w_t^* is close to one (actually *Airshow* is a very static sequence) except in the scene cuts (frames number 6 and 16), where w_s^* becomes larger. Also note that pixels in the first frame do not have any temporal forward neighbors, and therefore $w_s^* = 1$ and $w_t^* = 0$.



(a) *Airshow* (b) *Football*

Figure 4.9: Detail coefficient values. Darker colors indicate higher negative coefficient values, while brighter colors mean higher positive coefficient values. Grey indicates coefficients close to zero.



(a) *Airshow* (b) *Football*

Figure 4.10: $\mathbf{w}^*$ evolution.

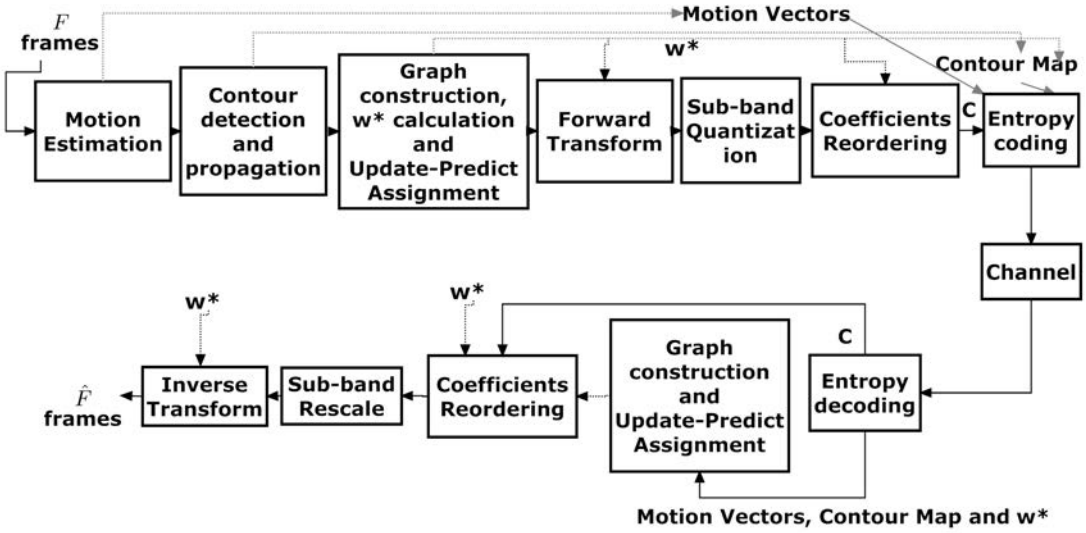


Figure 4.11: Encoder and decoder data flow.

4.2.3 Encoder and Decoder Data Flow

Figure 4.11 shows the encoder and the decoder data flow, assuming that the optimal weighting is selected to weight the graph.

First, ME and contour detection processes are performed, obtaining the MVs and contour map that are needed to construct the graph at level of decomposition $j = 1$ (Section 4.1.1). Once we have the graph, the encoder calculates the link weights, $\mathbf{w}^*$. At this point, the encoder performs the $\mathcal{U}_{j=1}/\mathcal{P}_{j=1}$ assignment process solving the WMC problem. Next, the weighted graphs at levels $j > 1$ are obtained as is explained in

Section 4.1.3, and the $\mathcal{U}_{j>1}/\mathcal{P}_{j>1}$ assignments are made. Once we have the graphs and $\mathcal{U}/\mathcal{P}$ assignments for all levels of decomposition, the encoder performs the transform, quantizes the coefficients, and reorders them (Section 4.2.1). Finally, an entropy coder is used to generate the definitive bitstream.

Note that, as it can be seen in Figure 4.11, the weight values are needed to perform the $\mathcal{U}/\mathcal{P}$ assignment, the filtering operations of the transform, and the reordering of the coefficients. Also note that, since the motion vectors, the contour map, and the weights are sent to the decoder, the process performed at the encoder is known at the decoder so that the system is invertible.

4.2.4 Low Complexity Approach

Low complexity is an important feature of practical encoders, especially for real-time applications. Therefore, there are many works that propose low-complexity approaches for video coders which aims to reduce the operations to be performed in the encoder without deteriorate the coding performance. Some examples in the context of the standard H.264/AVC have been proposed by the author in [59], [60], [61], [62], [63] or [64].

Next, we explain two different approaches to reduce the computational cost of the proposed transform. Specifically, we focus on the $\mathcal{U}/\mathcal{P}$ assignment process, which is the most time demanding subsystem of the encoder.

4.2.4.1 Low complexity Transform Using Subgraphs

The complexity of the graph-partition process ($\mathcal{U}/\mathcal{P}$ assignment) increases rapidly with the number of nodes N. In particular, the worse-case time complexity for the greedy WMC assignment algorithm used in our encoder is $O\left(N^3 \cdot logN\right)$ [47]. Therefore, it becomes the most complex process of the proposed encoder. Besides, another problem with operating in the whole graph is memory and delay. We now present a transform that operates on subgraphs of the original graph in order to reduce overall complexity with negligible loss of performance.

The goal is to divide the original graph node set $\mathcal{V}$ of size $L \times H \times K$ (where $L \times H$ is the frame size and K the number of frames considered in the graph construction) in I subsets A_i, so that in any of the subgraphs formed with the nodes of each subset

$\mathcal{S}_i\,(\mathcal{A}_i, \mathcal{E}_i)$ we can obtain a transform that is invertible and critically sampled, which takes into account the interactions between the nodes of different subgraphs. Critical sampling in this context means that the number of transform coefficients generated over all sub-graphs is the same as the original number of pixels. A necessary condition to achieve these objectives will be that the A_i node subsets have to be disjoint.

The proposed solution creates subgraphs based on disjoint subsets that contain linked pixels in K temporal hops , thus keeping the more reliable links of the subgraph in any level of the transform (under the assumption that temporal links are more reliable than spatial ones).

To do that, we divide each frame into blocks of size $P \times Q$. Then, we perform the motion estimation for each of these blocks in the K frames. With this information, a tree is generated in which the children of a given block a are the blocks of the reference frame that are linked to a by means of the motion model. Finally, each subgraph is composed of the pixels that belong to the blocks that are linked along the K frames. This is achieved using Algorithm 5, which given an initial set of n groups G_i, each composed by a block a and its children, constructs the subgraphs by joining groups that have common elements, and deleting those groups that have already been aggregated into a subgraph. An example of the subgraph construction is shown in Figure 4.12.

Algorithm 5 Subgraph Formation

Require: n Groups G_i
1: **while** $Flag \neq 0$ **do**
2: Set $Flag = 0$
3: **for** $i = 1$ **to** n **do**
4: **if** $G_i \neq deleted$ **then**
5: **for** $j = i + 1$ **to** n **do**
6: **if** $(G_j \neq deleted)$ and $(G_i \cap G_j \neq \phi)$ **then**
7: Set $Flag = 1$
8: Set $G_i = G_i \cup G_j$
9: Delete G_j
10: **end if**
11: **end for**
12: **end if**
13: **end for**
14: **end while**
15: **return** G_i and Index Vector of Non-deleted Groups

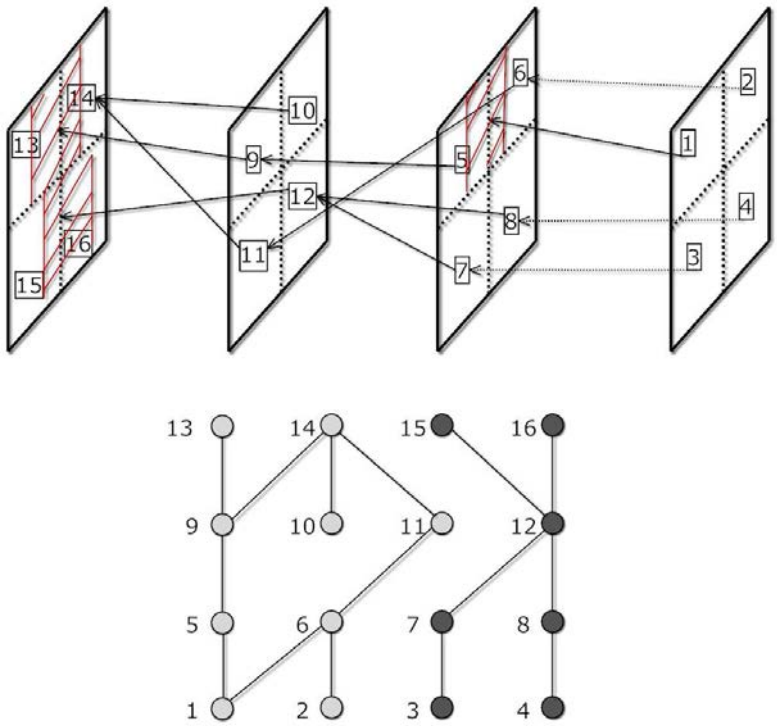


Figure 4.12: Subgraph construction from 4 frames. Top: motion dependency tree. Bottom: two subgraphs are formed corresponding to dark and light grey block pixels, respectively.

Table 4.4 provides experimental results for 20 frames of three QCIF sequences (*Mobile*, *Foreman* and *Carphone*). The table shows the number of subgraphs formed and the corresponding complexity reduction (CR), calculated as the ratio of encoding times when using the subgraphs and the original graph. It can be seen that the complexity reduction can be significant. Nevertheless, one drawback of this approach is that the final complexity depends on the motion content of the video sequence (faster motion sequences tend to lead to larger subgraphs). There are several approaches to mitigate this problem. For example, motion vectors could be constrained (e.g., motion vectors would have to point to co-located slices in previous frame). Alternatively, links between nodes in the bigger subgraphs could be removed leading to new smaller disjoint subgraphs until a required complexity restriction is achieved, or the maximum number of blocks that a subgraph can have in Algorithm 5 could be limited. Any of these approaches would lead to a simpler transform but would have an impact on performance.

Table 4.4: Subgraph approach performance.

	Number of Subgraphs	**CR**
Mobile	82	48
Foreman	14	4
Carphone	1	1

4.2.4.2 Distributed $\mathcal{U}/\mathcal{P}$ assignment

We now propose an approach for performing the $\mathcal{U}/\mathcal{P}$ assignment that works in a distributed manner, leading to a computational complexity almost independent of the video content. This method reduces the complexity of the WMC greedy algorithm used, from the $O\left(N^3 \cdot \log N\right)$ worse-case complexity of [47], to $O\left(\frac{N}{B}B^3 \cdot \log B\right)$, where B is the block size used in the algorithm. Note that for a fixed B, the complexity increases linearly with N in the distributed approach.

The idea consists in calculating the WMC solution in blocks of size B, making local $\mathcal{U}/\mathcal{P}$ decisions, and transferring this information to neighboring blocks. This is achieved by operating with overlapping blocks. Note that there exists a complexity-precision trade-off in the selection of B. The larger the block size B, the more complex and accurate the solution.

The proposed greedy solution is described in Algorithm 6, where $\mathcal{U}_j$ and $\mathcal{P}_j$ form a bipartition of the node set $\mathcal{U}_{j-1}$, $\mathcal{F}_i$ and $\mathcal{G}_i$ form a bipartition of $\mathcal{B}_i$, and we consider *Gain* of a node to be the sum of weights of all its incident edges. The algorithm requires $\mathcal{N}_\mathcal{B}$ blocks of size B so that $\bigcup_{i \in \mathcal{N}_\mathcal{B}} \mathcal{B}_i = \mathcal{V}$, covering all the nodes of the graph. Every block must "see" the decisions taken in neighboring blocks, which in the algorithm means that $\mathcal{B}_i \cap \mathcal{B}_j \neq \varnothing$, where i is the block to be processed and j is each one of the already processed neighboring blocks. The intersection is the information that they share, and must include the nodes in $\mathcal{B}_j$ that have edges that go from block j to block i. Figure 4.13 illustrates two iterations of the algorithm. In the first iteration (left part of the figure), a local WMC solution is found in block B_1. Then, in the second iteration, block B_2 includes the nodes of B_1 that have edges that go from B_1 to B_2 (boundary nodes). Therefore, the local WMC in B_2 is influenced by the already known colors (labels) of these boundary nodes, which means that the solution for block B_1 affects the solution

for block B_2. With this simple approach we get the speed-up benefits of operating with blocks, while guaranteeing a consistent solution across blocks.

Algorithm 6 Distributed *Weighted Maximum-Cut* Algorithm

Require: $\mathcal{U}_j = \{\varnothing\}$, $\mathcal{P}_j = \{\mathcal{U}_{j-1}\}$, $\mathcal{N}_\mathcal{B}$ blocks of size B

for $i = 1$ **to** $\mathcal{N}_\mathcal{B}$ **do**
  $\mathcal{F}_i = \{\varnothing\}$ and $\mathcal{G}_i = \mathcal{B}_i$
  $\mathcal{F}_i \leftarrow \mathcal{B}_i \cap \mathcal{U}_j$ and $\mathcal{G}_i \leftarrow \mathcal{G}_i \backslash \mathcal{F}_i$
  Change the sign of the incident edge weights to every node $f \subset \mathcal{F}_i$
  Calculate the $Gain$ of the nodes $\subset \mathcal{B}_i$
  Select the node a with largest $Gain$, $a = \max(Gain)$
  while $Gain > 0$ **do**
    Let $\mathcal{F}_i \leftarrow \mathcal{F}_i \cup \{a\}$
    Let $\mathcal{G}_i \leftarrow \mathcal{G}_i \backslash \{a\}$
    Change the sign of the incident edge weights to node a
    Update $Gains$ of adjacent nodes
    Select the node a with largest $Gain$, $a = \max(Gain)$
  end while
  $\mathcal{U}_j \leftarrow \mathcal{U}_j \cup \mathcal{F}_i$
  $\mathcal{P}_j \leftarrow \mathcal{P}_j \backslash \mathcal{F}_i$
end for
return $\mathcal{U}_j$ and $\mathcal{P}_j$

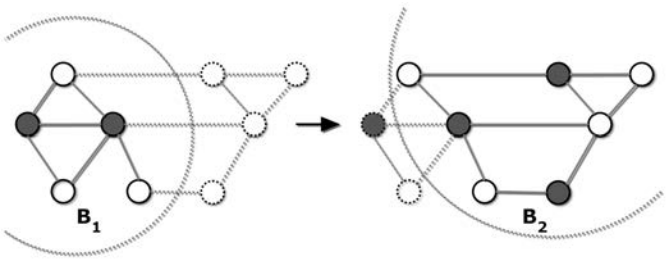


Figure 4.13: Distributed WMC.

The experimental results for the complexity reduction (CR), calculated as the ratio of encoding times when coding 20 frames using the centralized and the distributed approaches ($CR = \frac{time_{cent}}{time_{dist}}$), show the efficiency of the proposed method. Using a block size of $B = 512$, we experimentally obtain $CR = 228$ in *Carphone*, $CR = 203$ in *Mobile* and $CR = 197$ in *Container*, keeping the cut of the graph and the number of $\mathcal{U}$

and $\mathcal{P}$ nodes selected very similar to those chosen in the centralized approach, and thus causing a negligible loss in performance.

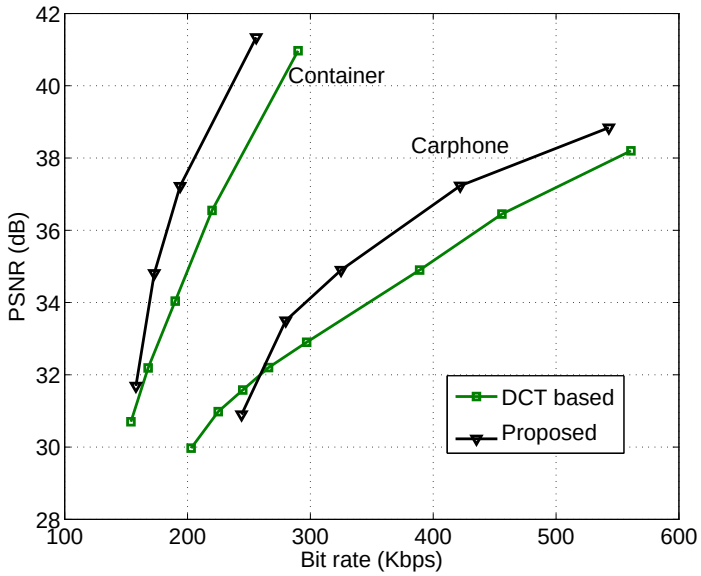
4.2.5 Experimental Results

To evaluate the coding performance of the proposed encoder, we compare it with a motion-compensated DCT video encoder in terms of rate-distortion for different test sequences. The coefficients are quantized using a uniform dead-zone quantizer in the DCT, and a subband dependent quantization in our encoder (i.e., the quantization step is lower in low frequency subbands and vice versa). These quantized coefficients are scanned as explained in Section 4.2.1 in our proposed method, and in the traditional zigzag scanning order in the DCT-based encoder. Note that this process is performed in scanning units of size S. Then, run-length encoding (RLE) is performed in both encoders, obtaining the symbols to be entropy coded. An end-of-block special symbol is used to indicate that all remaining coefficients in the scanning unit are quantized to zero. Finally, the bitstream is obtained coding the symbols using an adaptive arithmetic coder.

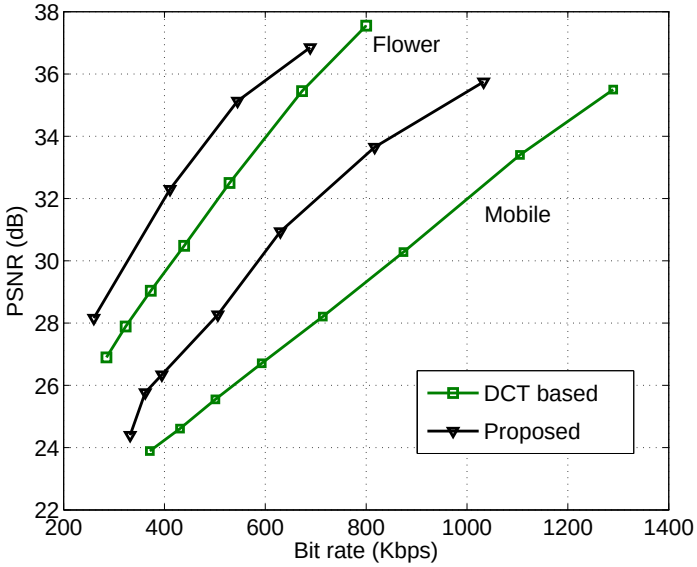
Regarding the side information, motion vectors are differentially encoded with respect to a predicted motion vector obtained from adjacent blocks. Then, a variable length code (VLC) is used to code the difference motion vector. Note that the motion vectors to transmit will be different in the proposed and in the DCT based encoders, since the matching is carried out in original frames in the former and in reconstructed reference frames in the latter. Nevertheless, the rate turns out to be similar in both cases. The proposed encoder has an extra overhead because it should send the contour information to the decoder once every K frames and the optimal weight values every frame, since they are calculated as in Section 4.2.2. Contour maps are encoded using JBIG, obtaining negligible rates of around 10 Kbps, and weights are coded using 9 bits per weight, giving rise to insignificant rates.

In the experiments, $K = 20$, $S = 256$, and five levels of decomposition of the proposed transform are performed. Block sizes of 16×16 and one reference frame are assumed in the motion estimation process. In the DCT encoder, we use 8×8 DCT. Finally, the block size used in the low cost approach is set to $B = 512$.

Figure 4.14 shows the rate-distortion curves for four different QCIF sequences, *Mobile*, *Carphone*, *Flower* and *Container*. In general, the proposed method outperforms the DCT-based approach. In *Mobile* sequence, our method is 4 dB better than the DCT in medium to high qualities. The gain is also significant in the rest of sequences (around 1-1.5 dB in *Carphone*, an 2 dB in *Container* and *Flower*). However, the efficiency of the encoder at low qualities gets worse, losing against the DCT based encoder in *Carphone* for qualities lower than 32 dB. The results are in agreement with the non-linear approximation results presented in Section 4.1.4.



(a) *Container* (QCIF) and *Carphone* (QCIF)



(b) *Flower* (QCIF) and *Mobile* (QCIF)

Figure 4.14: PSNR versus bit rate.

4.3 Rate-Distortion Graph Optimization

In Chapter 3 we discussed some optimal $\mathcal{U}/\mathcal{P}$ assignment strategies which minimize the detail coefficient energy under specific data generation models, and **for a given number of $\mathcal{P}$ nodes** ($|\mathcal{P}|$). Therefore, these models do not provide the optimal $|\mathcal{P}|$, which actually depends on the application.

In a typical coding application, the encoder performs rate-distortion optimization (RDO) to find the coding parameters that minimize the distortion under a rate constraint. Thus, for our proposed coding scheme, $|\mathcal{P}|$ (and some others parameters) should be chosen by solving the RDO. As a first approximation, one can assume that, as $|\mathcal{P}|$ increases, the distortion (D) increases (because worse predictions are obtained, as is shown in the experimental results of Section 3.2) and the rate (R) decreases (because detail coefficients need lower number of bits to be represented).

In this section we study how to apply RDO to our proposed graph-based video encoder. We first formulate the original RDO problem in Section 4.3.1, turning it to an unconstrained problem as in [65]. Then, in Sections 4.3.2 and 4.3.3 we provide, respectively, D and R models that depend on the $\mathcal{U}/\mathcal{P}$ assignment (which implicitly determines $|\mathcal{P}|$), and the quantization step Δ of smooth coefficients for decomposition level $j = 1$. Although some simplifying assumptions are needed to construct the models, they give useful intuition into how to apply RDO to predict node selection. In Section 4.3.4 we use the proposed R and D models to obtain analytically the λ parameter that balances the weight of the R and D terms in the unconstrained RDO problem. We provide a formula that relates λ and Δ, remaining only one parameter in the optimization process, which is described in Section 4.3.5.

It should be noted that the goal of this section is just to give some intuitions and illustrate how the RDO process could be done, so that we make some assumptions and simplifications explained in Section 4.3.1. Finally, in Section 4.3.6, we discuss how these simplifications affect the RDO, and the way it could be extended in order to obtain a more realistic process.

4.3.1 Rate-Distortion Optimization Problem for Lifting Transforms on Graphs

In this section we formulate the RDO problem in general sense for the lifting transform on graphs. Usually, a video encoder performs the RDO aiming to find the coding option that minimizes a D measure subject to a given R restriction.

Let $\boldsymbol{\theta}$ be a combination of the different coding options:

$$\boldsymbol{\theta} = \{\boldsymbol{\Delta}_{\mathbf{s}}, MVs, \text{contour map}, \mathcal{U}_{\mathbf{j}}/\mathcal{P}_{\mathbf{j}}, \mathbf{w_j}\}\,, \tag{4.1}$$

where $\boldsymbol{\Delta}_{\mathbf{s}}$ is the quantization step vector used to quantize each subband of the lifting representation given in (2.2)(i.e., $\boldsymbol{\Delta}_{\mathbf{s}} = \left[\Delta_{d_{j=1}} \dots \Delta_{d_J} \Delta_{s_J}\right]$), $\mathcal{U}_{\mathbf{j}}/\mathcal{P}_{\mathbf{j}}$ is the $\mathcal{U}_j/\mathcal{P}_j$ assignment for each level of the transform j, and $\mathbf{w_j}$ represents the weights of the links on the graph for each level j.

Thus, the problem can be formulated as:

Problem 4.1. ***RDO Problem Formulation.***

$$\min_{\boldsymbol{\theta}} \{D(\boldsymbol{\theta})\} \textit{ subject to } R(\boldsymbol{\theta}) \leq R_c, \tag{4.2}$$

where $D(\boldsymbol{\theta})$ represents the D between the original and the reconstructed coding unit; $R(\boldsymbol{\theta})$ is the R needed to encode it (the number of bits needed to encode headers, side information -MVs, contour map, weight and Δ values, ...- and transform coefficients); and R_c the maximum R allowed (the R constraint).

Using a Lagrange formulation, this constrained optimization problem can be converted into an unconstrained problem [65], [66], [67]:

$$\begin{aligned} &\min_{\boldsymbol{\theta}} \{J(\boldsymbol{\theta})\} \\ &\text{with } J(\boldsymbol{\theta}) = D(\boldsymbol{\theta}) + \lambda R(\boldsymbol{\theta}), \end{aligned} \tag{4.3}$$

where λ is the Lagrange multiplier that weights the relative importance between $D(\boldsymbol{\theta})$ and $R(\boldsymbol{\theta})$. A given value of λ yields a solution $\boldsymbol{\theta}^*(\lambda)$ that is also an optimal solution to

the original RDO problem (4.2) for a particular value of $R_c = R(\boldsymbol{\theta}^*)$. Therefore, given a R_c, one should find the λ multiplier so that $R(\boldsymbol{\theta}^*(\lambda)) = R_c$.

To solve Problem 4.1, we make some **assumptions and simplifications**, namely: (i) we only optimize the first level of the transform, $j = 1$; (ii) we employ two quantizers related by $\Delta_{d_{j=1}} = 2\Delta_{s_{j=1}}$; (iii) we do not take into account the side information in the optimization process; and (iv) we obtain the optimal weights $\mathbf{w}_{j=1}$ as explained in Section 3.1.2.2.

Given that we just optimize the first level of the transform, hereafter, for simplicity, we omit the subindex $j = 1$, so that Δ refers to $\Delta_{s_{j=1}}$ (and $\Delta_{d_{j=1}} = 2\Delta$), and $\mathcal{U}/\mathcal{P}$ refers to $\mathcal{U}_{j=1}/\mathcal{P}_{j=1}$.

Under these assumptions and simplifications, Problem 4.1 can be written, using the Lagrange formulation, as:

Problem 4.2. ***RDO Problem Formulation with Simplifications.***

$$\min_{\mathcal{U}/\mathcal{P},\Delta} J(\mathcal{U}/\mathcal{P},\Delta) = \min_{\mathcal{U}/\mathcal{P},\Delta} \left\{ D(\mathcal{U}/\mathcal{P},\Delta) + \lambda R(\mathcal{U}/\mathcal{P},\Delta) \right\}. \tag{4.4}$$

Note that optimizing the $\mathcal{U}/\mathcal{P}$ assignment we are optimizing the $|\mathcal{P}|$. Other parameters could be easily considered in the problem formulation, as the side information of the MVs and the contour map.

Next, we present $D(\mathcal{U}/\mathcal{P},\Delta)$ and $R(\mathcal{U}/\mathcal{P},\Delta)$ models and we derive the λ multiplier as a function of Δ, so that both parameters are tied together. This way, given a Δ value and a sequence, one can solve Problem (4.2) by finding the $\mathcal{U}/\mathcal{P}$ that minimizes J.

4.3.2 Distortion Model

In this section we propose a D model for lifting transforms on graphs, under the assumptions given in previous section. The distortion of the lifting transform on the graph can be expressed as:

$$D\left(\mathcal{U}/\mathcal{P},\Delta\right) = \sum_{u\in\mathcal{U}} \left(x_u - \tilde{x_u}\right)^2 + \sum_{p\in\mathcal{P}} \left(x_p - \tilde{x}_p\right)^2, \tag{4.5}$$

where $\tilde{x}_k$ represents the k-th reconstructed pixel after quantization and inverse transformation. Assuming that the transform does not perform the update stage and that the d_p coefficients are orthogonal to its s_u neighbors, D can be written as:

$$D\left(\mathcal{U}/\mathcal{P},\Delta\right) \approx \sum_{u\in\mathcal{U}} \left(s_u - \tilde{s}_u\right)^2 + \sum_{p\in\mathcal{P}} \left(\sum_{u\in\mathcal{N}_p\cap\mathcal{U}} \mathbf{p}\left(s_u - \tilde{s}_u\right) \right)^2 + \sum_{p\in\mathcal{P}} \left(d_p - \tilde{d}_p\right)^2, \tag{4.6}$$

where $\mathcal{N}_p$ is the set of one-hop neighbors of node $p \in \mathcal{P}$, m_p is the number of one-hop $\mathcal{U}$ neighbors of node p, and $\mathbf{p}$ is the prediction filter used.

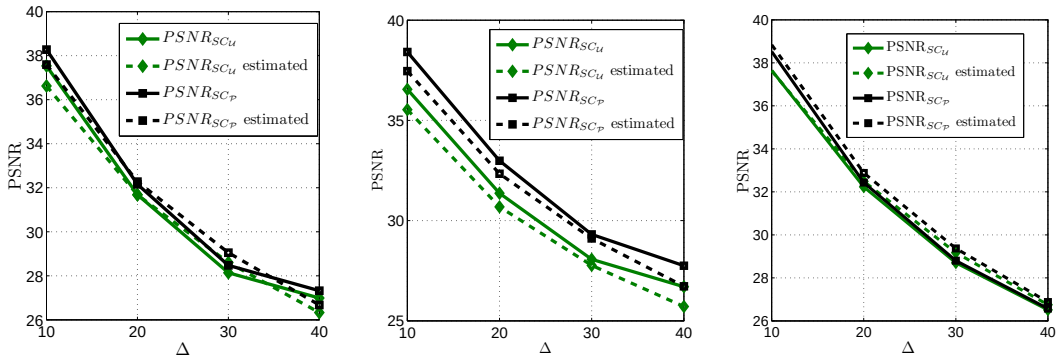
Now, considering that $(s_u - \tilde{s_u})$ is the same value for every $u \in \mathcal{N}_p \cap \mathcal{U}$, and assuming high-resolution quantization [68] of the s_u coefficients and the d_p non-zero quantized coefficients, we get

$$\begin{aligned} D\left(\mathcal{U}/\mathcal{P},\Delta\right) &\approx \frac{\Delta^2}{12}N + \sum_{p_{nz}\in\mathcal{P}} \left(d_p - \tilde{d}_p\right)^2 + \sum_{p_z\in\mathcal{P}} \left(d_p\right)^2 \\ &\approx \frac{\Delta^2}{12}N + \frac{\left(2\Delta\right)^2}{12}|\mathcal{P}_{nz}| + D_0, \end{aligned} \tag{4.7}$$

where p_z (resp. p_{nz}) are the $p \in \mathcal{P}$ nodes in which the corresponding d_p coefficients are quantized to zero (resp. are not quantized to zero), $|\mathcal{P}_z|$ (resp. $|\mathcal{P}_{nz}|$) is the number of detail coefficients quantized to zero (resp. not quantized to zero), and $D_0 = \sum_{p_z\in\mathcal{P}}^{|\mathcal{P}_z|} \left(d_p\right)^2$. Note that D depends on the $\mathcal{U}/\mathcal{P}$ assignment through $|\mathcal{P}_{nz}|$ and D_0. From (4.7) we can conclude that, in general, increasing the number of $\mathcal{U}$ nodes would imply decreasing $|\mathcal{P}_{nz}|$ and D_0, and thus D.

Figure 4.15 shows some examples of actual PSNR values and PSNR estimated using (4.7) for the SC solution with minimum $|\mathcal{U}|$ ($SC_{|\mathcal{U}|}$) and with minimum $|\mathcal{P}|$ ($SC_{|\mathcal{P}|}$). In the experimental results, we use three fragments of specific areas of the video sequences *Carphone*, *Mobile* and *Container*, and the video encoder described in Section 4.2, but working just in the first level of the transform.

Some observations can be made from the analysis of Figure 4.15:



(a) *Carphone* (b) *Mobile* (c) *Container*

Figure 4.15: PSNR estimation for different sequences.

- As it was to be expected, the $SC_{|\mathcal{P}|}$ achieves higher PSNR values than the $SC_{|\mathcal{U}|}$ in all the examples.
- The PSNR estimation is reasonably good for *Carphone* and *Container* (around -0.4 dB on average), and slightly worse for *Mobile* (-0.7 dB).
- *Carphone* and *Container* are examples of sequences with homogeneous and stationary areas, where a $\mathcal{P}$ node does not need too many $\mathcal{U}$ neighbors to be accurately predicted. Therefore, the PSNR obtained for the $SC_{|\mathcal{U}|}$ and $SC_{|\mathcal{P}|}$ solutions is similar. On the contrary, in *Mobile*, a video fragment with complex texture, $SC_{|\mathcal{P}|}$ significantly outperforms the $SC_{|\mathcal{U}|}$ solution.

4.3.3 Rate Model

Next, we propose a rate model for lifting transforms on graphs for video coding, under the assumptions given in Section 4.3.1. This model estimates R obtained when employing the RLE and arithmetic coders used in Section 4.2.5. We consider a parametric logarithmic model:

$$R\left(\mathcal{U}/\mathcal{P}, \Delta\right) = m \ln(\Delta) + G(M), \tag{4.8}$$

where m is a sequence-dependent negative constant that indicates the decay velocity of R with Δ (the higher $|m|$, the faster the decay); M is the number of non-zero quantized coefficients; and $G(M)$ is an unknown function[1] that increases with M.

It is important to highlight that we will employ the R model to derive λ as a function of Δ. Therefore, we focus on the dependence of R with Δ, and not in obtaining an accurate expression for $G(M)$.

Firstly, $G(M)$ depends on Δ through M, because as Δ decreases, M should increase. Nevertheless, one can prove that, given that $\Delta_{d_{j=1}} = 2\Delta_{s_{j=1}}$, in general, $|\mathcal{P}_{nz}|$ is low[2] and thus $M \approx |\mathcal{U}|$ for any Δ. This implies that R **increases with the number of** $\mathcal{U}$ **nodes, and depends on** Δ **just through** $m\ln(\Delta)$. Let us analyze this observation.

To obtain R, we employ a RLE over the quantized coefficients, and then the resulting symbols are arithmetically coded. Therefore, R depends on the number of symbols to be encoded and their entropy. **For a given** $\mathcal{U}/\mathcal{P}$ **assignment** (i.e., $|\mathcal{U}|$), the RLE leads to a similar number of symbols for every Δ. Nevertheless, R **decreases as** Δ **increases** because the variance (and thus the entropy) of the symbols decreases as Δ increases, and therefore they are arithmetically coded more efficiently. Now, we **fix the** Δ **value**. In this case, R increases with $|\mathcal{U}|$ because as $|\mathcal{U}|$ increases, the number of long strings of zeros decreases, and the RLE gives rise to a higher number of symbols to be encoded.

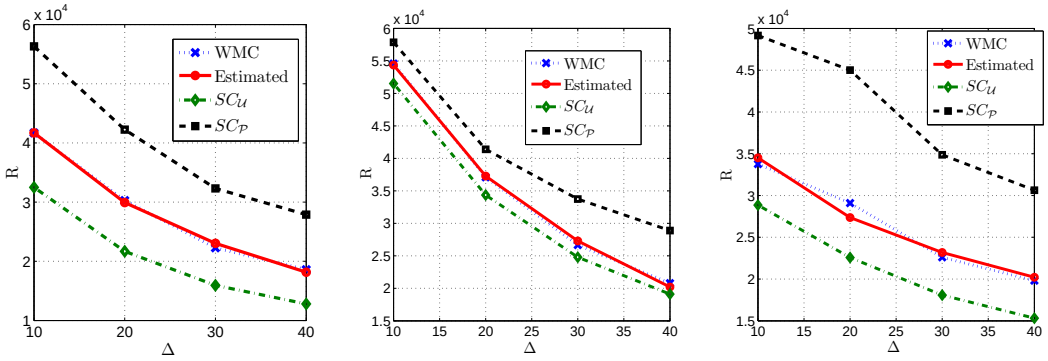
Figure 4.16 shows $R(\Delta)$ obtained with three different $\mathcal{U}/\mathcal{P}$ assignments (namely: WMC, $SC_{\mathcal{U}}$, and $SC_{\mathcal{P}}$) and our estimated R for the WMC case (i.e., for the WMC, we consider the model $R = m\ln(\Delta) + K$, and estimate the parameters m and K by regression using the Least Squares method). The experiments have been made using three fragments extracted from QCIF sequences *Carphone*, *Mobile* and *Container*, and employing the entropy coding of Section 4.2.

Some conclusions can be extracted from Figure 4.16:

- The model accurately fits the WMC solution.

[1] $G(M)$ could be modeled as $G(M) = Mc + d$, with c and d constants that depend on the sequence. In this way, R would increase linearly with M [69].

[2] This assumption is analyzed in Section 4.3.6.



(a) *Carphone* (b) *Mobile* (c) *Container*

Figure 4.16: Rate estimation for different sequences.

- As expected, R increases with $|\mathcal{U}|$. Therefore, the lowest R values in all the sequences are obtained for the SC$_\mathcal{U}$ solution, which minimizes the number of $\mathcal{U}$ nodes.

- For a given video, all curves are almost parallel, and thus R depends on Δ mostly through $m\ln(\Delta)$ and with only a small dependence in terms of $G(\mathcal{U})$. Therefore, the factor $G(\mathcal{U})$ just displaces the curve as a function of $|\mathcal{U}|$. This is especially true for the WMC and SC$_\mathcal{U}$ solutions.

- To estimate the rate at any $|\mathcal{U}|$, one should displace the estimation of WMC a factor indicated by $G(\mathcal{U})$ (e.g., an estimation for the R of the SC$_\mathcal{U}$ could be made by subtracting a constant value to the estimation of the WMC).

- In *Carphone*, the R of the SC$_\mathcal{U}$ is considerably lower than the R of the SC$_\mathcal{P}$, contrary to what occurs in *Mobile*. This is because the SC$_\mathcal{U}$ and the SC$_\mathcal{P}$ solutions are distant in *Carphone* (i.e., they lead to ratios of $|\mathcal{U}_{SC_\mathcal{U}}|/N = 0.26$ and $|\mathcal{U}_{SC_\mathcal{P}}|/N = 0.73$, respectively), and closer in *Mobile* ($|\mathcal{U}_{SC_\mathcal{U}}|/N = 0.36$ and $|\mathcal{U}_{SC_\mathcal{P}}|/N = 0.64$). These differences are due to the different graph topologies.

Finally, note that in a practical implementation of this rate model, its parameters should be estimated *on the fly*.

4.3.4 Lambda Calculation

In this section we derive the λ parameter that balances the weight of the R and D terms in the minimization Problem 4.2 as a function of the Δ used in the coding process.

Minimizing J as a function of Δ, we get:

$$\min_{\Delta} J \implies \frac{\partial J}{\partial \Delta} = \frac{\partial (D + \lambda R)}{\partial \Delta} = 0. \tag{4.9}$$

From (4.9), the λ parameter can be calculated as:

$$\frac{\partial D}{\partial \Delta} = -\lambda \frac{\partial R}{\partial \Delta}. \tag{4.10}$$

Given that $\Delta_{d_{j=1}} = 2\Delta_{s_{j=1}}$, $|\mathcal{P}_{nz}|$ is generally small and thus $|\mathcal{P}_{nz}| << N$, $M \approx |\mathcal{U}|$, and $|\mathcal{P}_z| \approx |\mathcal{P}|$ for any Δ. Therefore, for a fixed $|\mathcal{P}|$, we can assume that D_0 does not depend on Δ. Differentiating in (4.10) we obtain:

$$\lambda = -\frac{(2\Delta N)/12}{m/\Delta} = \frac{-N}{6m}\Delta^2. \tag{4.11}$$

This relation gives the Lagrange multiplier λ as a function of Δ and the negative parameter m, which depends on the sequence. Specifically, λ decreases when $|m|$ increases. Let us analyze this behaviour.

Lower $|m|$ values imply lower decay of R with Δ, and thus faster decay of D with R. λ can be interpreted as the slope of the line tangent to the operational $D(R)$ convex hull at the point $R(\lambda) = R_c$ [66]. Therefore, to reach an specific R-D trade-off, the optimal λ should be higher as $|m|$ is lower, as indicated in (4.11). Figure 4.17 shows an example to graphically illustrate this fact. Note that λ (Slope 1) is higher in *Container*, which has a low $|m|$, than in *Mobile* (Slope 2), in which $|m|$ is large, for the same Δ value. Finally, note that λ is proportional to Δ^2, which is a reasonable result as is shown in [70].

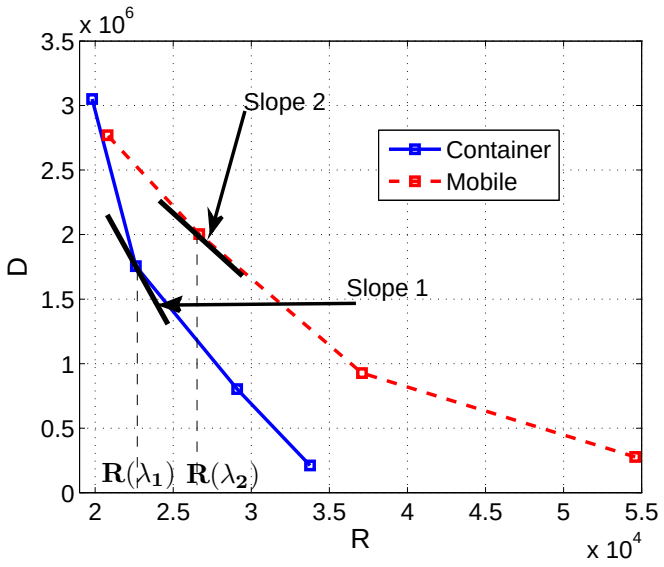


Figure 4.17: Relation between λ and the parameter m of the sequence.

4.3.5 Optimization Process

In this section we explain how the RDO can be performed using the results and assumptions discussed previously. Furthermore, we provide a greedy algorithm to solve the simplified RDO Problem 4.2 and give an experimental evaluation that shows the benefits of RDO.

In the simplified problem, the parameters to be optimized are $\boldsymbol{\theta} = (\mathcal{U}/\mathcal{P}, \Delta)$. Note that letting λ vary we are guaranteed to minimize D for a fixed $R_c = R(\boldsymbol{\theta}^*(\lambda))$. Equation (4.11) relates analytically λ and the optimal Δ. Therefore, we assume a given Δ value (which fixes λ), and search the $\mathcal{U}/\mathcal{P}$ assignment on the graph that minimizes J in Problem 4.2, obtaining the optimal parameters $\boldsymbol{\theta}^* = ((\mathcal{U}/\mathcal{P})^*, \Delta^*)$ which solve the problem for $R_c = R(\boldsymbol{\theta}^*(\lambda))$.

Using the models of (4.8), (4.7), and (4.11) in Problem 4.2 we can write:

$$\min_{\mathcal{U}/\mathcal{P}} J \approx \min_{\mathcal{U}/\mathcal{P}} \left\{ \frac{\Delta^2}{12} N + \frac{(2\Delta)^2}{12} |\mathcal{P}_{nz}| + D_0 \right. \\ \left. - \frac{N}{6m} \Delta^2 \left(m \ln(\Delta) + G(M) \right) \right\}. \tag{4.12}$$

Minimizing (4.12) is equivalent to finding the $\mathcal{U}/\mathcal{P}$ assignment that minimizes D for a given number of M nonzero quantized coefficients. Assuming that $|\mathcal{P}_{nz}|$ is low (i.e., $M \approx |\mathcal{U}|$), the minimization of (4.12) given $|\mathcal{U}|$ is achieved by finding the $\mathcal{U}/\mathcal{P}$ assignment that minimizes D_0 (because the rest of the terms do not depend on the $\mathcal{U}/\mathcal{P}$ assignment), that is, finding small d_p amplitude coefficients. Note that, considering one of the pixel generation models of Section 3.2, this is equivalent to solving Problem 3.3.1, which can be done using one of the greedy algorithms proposed in that section.

To solve Problem 4.2 given a Δ value and a sequence, one could start by searching the $\text{SC}_{\mathcal{U}}$ solution and calculating the cost $J = D + \lambda R$ using real D and R data for that $\mathcal{U}/\mathcal{P}$ assignment. Then, in each iteration, a $\mathcal{P}$ node should be "converted" to $\mathcal{U}$ following criteria given in algorithms of Section 3.2, thus minimizing the D_0 for that specific $|\mathcal{U}|$. Finally, the optimal $\mathcal{U}/\mathcal{P}$ assignment would be the one that minimizes J. The greedy approach of Algorithm 7 performs this process.

Algorithm 7 RDO process.

Require: Δ

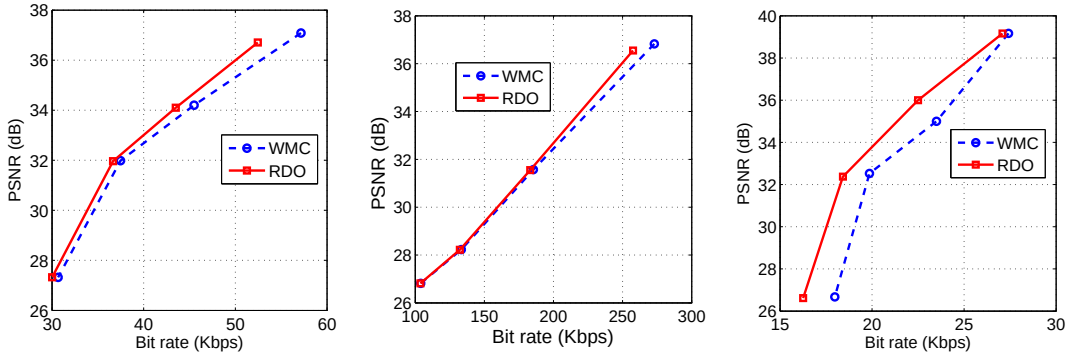
Find the *set-covering* solution with minimum $|\mathcal{U}|$, $\text{SC}_{\mathcal{U}}$
Estimate parameter m
Calculate λ
Calculate the cost $J((\mathcal{U}/\mathcal{P})_{SC_{\mathcal{U}}}) = D((\mathcal{U}/\mathcal{P})_{SC_{\mathcal{U}}}) + \lambda R((\mathcal{U}/\mathcal{P})_{SC_{\mathcal{U}}})$
$J_{opt} = J$
$(\mathcal{U}/\mathcal{P})^* = (\mathcal{U}/\mathcal{P})_{SC_{\mathcal{U}}}$
for $\forall i \in I$ **do**
    Convert in $\mathcal{U}$ the node $\mathcal{P}$ that solves Problem 3.3.1
    Calculate the cost $J((\mathcal{U}/\mathcal{P})_i) = D((\mathcal{U}/\mathcal{P})_i) + \lambda R((\mathcal{U}/\mathcal{P})_i)$
    if $J < J_{opt}$ **then**
        $J_{opt} = J$
        $(\mathcal{U}/\mathcal{P})^* = (\mathcal{U}/\mathcal{P})_i$
    end if
end for
return $(\mathcal{U}/\mathcal{P})^*$

Note that the algorithm should calculate the "real" D and R values for each $\mathcal{U}/\mathcal{P}$ assignment, so that it is not practical for a real implementation. In Section 4.3.6 we discuss how to perform the RDO process in a block-by-block basis, which considerably would reduce the complexity.

Table 4.5: Proportion of $\mathcal{U}$ nodes selected by the RDO.

	WMC	$\Delta = 10$	$\Delta = 20$	$\Delta = 30$	$\Delta = 40$
Carphone	0.44	0.30	0.43	0.43	0.44
Mobile	0.42	0.35	0.41	0.42	0.42
Container	0.46	0.46	0.24	0.36	0.36

Figure 4.18 shows some R-D experimental results obtained for three specific areas and fragments of QCIF sequences, *Carphone*, *Mobile* and *Container*, using the encoder described in Section 4.2 under the assumptions given in Section 4.3.1. The figure gives the R-D curves obtained using the RDO of Algorithm 7 and the WMC solution. Besides, Table 4.5 indicates the proportion of $\mathcal{U}$ nodes ($|\mathcal{U}|/N$) chosen by the RDO process for each Δ value, and the proportion for the WMC (which does not depend on Δ).



(a) *Carphone* (b) *Mobile* (c) *Container*

Figure 4.18: RDO Vs WMC for different sequences.

The RDO solution improves the WMC in all the examples, as expected. Specifically, in the fragment of *Container*, which is homogeneous and stationary, RDO clearly outperforms WMC. Note that, in this case, RDO selects a lower proportion of $\mathcal{U}$ nodes, because increasing this proportion does not significantly improve the D term, but, on the other hand, it increases the R. Therefore, the optimal trade-off is found by choosing a low $|\mathcal{U}|$. Unlike *Container*, the *Mobile* fragment has complex textures. This way, it is worth to select a high proportion of $\mathcal{U}$ nodes on the graph so that predictions are better and D is lower, despite of the R is increased. Therefore, the optimal $|\mathcal{U}|$ is close to the WMC solution.

4.3.6 Discussion

The RDO process explained above is derived under some assumptions and simplifications. Next, we give some ideas of how the RDO process could be extended to obtain a more realistic and practical process.

Working in a block by block basis is an important property of video encoders. Besides, in the case of the lifting transforms on graphs, performing the transform on graphs of size $N = M \times H \times F$, with F the number of frames and $M \times H$ the size of each frame, is computationally unapproachable.

Assume that we have B blocks and that we would like to optimize some parameters in order to minimize the total D (in the B blocks) subject to a global R constraint ($\sum_{b=1}^{B} R_b \leq R_c$). If we consider that the R-D curves are independent for each block b, and that $R = \sum_{b=1}^{B} R_b$ and $D = \sum_{b=1}^{B} D_b$, we can write [67]:

$$\min \left(\sum_{b=1}^{B} D_b + \lambda R_b \right) = \sum_{b=1}^{B} \min \left(D_b + \lambda R_b \right), \tag{4.13}$$

so that the minimum can be computed independently for each block b. To that end, the same λ should be used for every block, leading to a so called *constant slope optimization*.

To do that, we could use Algorithm 7, performing the $\mathcal{U}/\mathcal{P}$ assignment locally in each block b, and transferring the taken decisions to its neighboring blocks as in Section 4.2.4.2. The R-D values to be used in the algorithm for each block could be obtained by performing the transform with the information of two-hop distance neighbors (Figure 4.19). In this way, the algorithm would return the optimal $\mathcal{U}/\mathcal{P}$ assignment for each block b, minimizing the total D for the R_c constraint.

Finally, note that the encoder should send to the decoder the optimal number of $\mathcal{U}$ nodes for each block, but not the label of each node (the decoder knows the criteria to perform the $\mathcal{U}/\mathcal{P}$ assignment).

Other extensions of our proposed RDO could be to optimize all the levels of the transform. Given that the filtering operations are not orthogonal, but biorthogonal, they are not energy conserving, so that one should weight the D of each subband as a function of the closeness of the biorthogonal filters to the class of orthogonal filters to compute

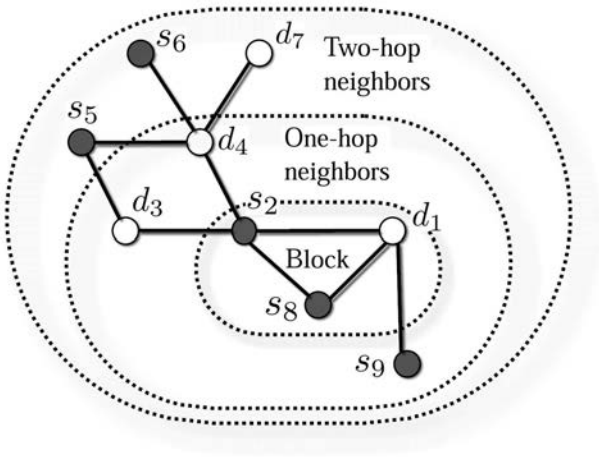


Figure 4.19: Transform by blocks.

the global D (i.e., the global D should be computed as a weighted sum of the D in each subband) ([71]). Once the weights are computed, they can be used to solve allocation problems using standard algorithms [72], [73].

In our experiments, the Δ value (and thus λ) is swept, obtaining, by means of the RDO, parameters that are optimum if the resulting $R(\lambda) = R_c$. Nevertheless, in a real application, one should find the desired optimal λ, which is not known a priori, in order to obtain the desired target budget R_c. This can be done using fast algorithms [66] or modeling the resulting R as a function of λ or Δ.

Assumption $\Delta_{d_{j=1}} = 2\Delta_{s_{j=1}}$ allows us to relate two of the parameters to be optimized, thus making easier the RDO process. Nevertheless, this relation could not be optimal. Specifically, one should optimize Δ for each subband ($\mathbf{\Delta_s}$), as is posted in Problem 4.1.

Finally, note that the hypothesis of the low $|\mathcal{P}_{nz}|$ (i.e., $M \approx |\mathcal{U}|$) used to derive λ is quite accurate in our framework. Specifically, for *Carphone*, the worse case (the case in which $|\mathcal{P}_{nz}|$ is higher) is $|\mathcal{P}_{nz}|/N = 0.03$, while in *Mobile* the worse case is $|\mathcal{P}_{nz}|/N = 0.08$, both for $\Delta = 10$. Different results between the two sequences can be explained by the fact that in *Carphone* predictions are more accurate.

4.4 Conclusions

In this chapter we have proposed a complete video encoder based on lifting transforms on graphs presented in Chapter 3. The proposed system gives rise to a non-separable 3-dimensional directional transform which is critically-sampled, versatile and of easy interpretation. Our transform outperforms a MCTF and DCT based transforms in energy compaction ability. Furthermore, it solves some typical problems inherent in temporal wavelet transforms (i.e., MCTF approaches).

Besides, we have described a new coefficient reordering method which is based on the graph information that improves the compression ability of the entropy encoder, leading to a system that outperforms a DCT based video encoder in R-D terms. Given that one drawback of our system is the computational complexity, we have investigated two low complexity approaches that reduce the computational cost of the $\mathcal{U}/\mathcal{P}$ assignment process.

Finally, we have described how the RDO process can be performed in our coding scheme under some simplifying assumptions.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

A general class of **graph-based transforms for N-dimensional signals** and their optimization have been proposed. These kind of transforms can be seen as **N-dimensional directional transforms** that avoid filtering across large discontinuities. They may be employed for compact representation of N-dimensional signals in many scenarios and for different applications such as coding, denoising or feature extraction.

To perform the proposed lifting transform, the first step consists in constructing a suitable graph. In Chapter 3 we discussed how to obtain a graph representation of a generic N-dimensional signal, giving examples of multichannel audio and video representations. To maximize energy compaction, graphs should be constructed so that they accurately capture correlation between samples. Given that filtering operations are performed using linked nodes, directional information is implicit in the graph representation. Graph weighting greatly influences the performance of the transform, because some processes are based on the graph weights. We discussed two approaches for weighting the graph in Chapter 3. At that point, we have a weighted graph that captures the correlation between samples, and which is useful to perform different signal processing operations.

Given an undirected graph, the lifting transform is guaranteed to be invertible and critically sampled by finding a graph bipartition ($\mathcal{U}/\mathcal{P}$ assignment) and defining the update **u** and prediction **p** filters. Therefore, we mainly focused on the optimization of these two processes in order to minimize the detail coefficient energy. Regarding the $\mathcal{U}/\mathcal{P}$ assignment, we proposed graph-based and signal model-based approaches. Graph-based designs use the information of the weighted graph in order to obtain the splitting, while signal model-based approaches assume different data generation models

and predictors, and assign a label to each node on the graph aiming to minimize the expected value of the squared prediction error.

Prediction **p** filters that provide "good" predictors for a given arbitrary weighted graph were also proposed in Chapter 3. Finally, the main properties of the proposed transform, some of them inherent to the lifting scheme and others related to the $\mathcal{U}/\mathcal{P}$ assignment, the **u** and **p** filters design, and the graph construction, were summarized.

In Chapter 4 we designed graph-based transform for use in video coding. These transforms follow 3-dimensional spatio-temporal high-correlation filtering paths, and can be considered a generalization of classical s+t or t+s MTCF wavelet encoders.

Specifically, we used the WMC splitting method and the filter design explained in Chapter 3, and provided a way to perform the transform at multiple levels obtaining a MRA of the original sequence. This led to more efficient representations than a MCTF and a DCT-based transforms for video coding. Also, we explained how the proposed transform is able to handle different problems that arise in MCTF approaches.

As a final contribution, we presented a complete video encoder in Section 4.2. In particular, we proposed an efficient way to reorder the coefficients before they are entropy coded, improving the compression performance of the proposed encoder. This led to very efficient video representations that outperform a comparable hybrid DCT based video encoder, which is the basis of the latest video coding standards. Besides, we proposed two low complexity approaches which allows to reduce the computational complexity of the proposed scheme incurring a negligible loss of performance. Finally, we investigated how to apply rate-distortion optimization to our proposed scheme.

5.2 Future Work

There are some interesting directions for future work.

The signal model-based $\mathcal{U}/\mathcal{P}$ assignment methods proposed in Chapter 3 provide an idea of how to assign a label to each node in order to minimize the detail coefficients energy in the first level of the transform. Nevertheless, this may be extended in order to jointly consider the optimization of the transform at all levels. This way, for example, one could perform the $\mathcal{U}/\mathcal{P}$ assignment at any arbitrary level taking into account how this assignment will influence the expected value of the coefficient energy of the final

transformed graph signal. Three signal model-based $\mathcal{U}/\mathcal{P}$ assignment designs were discussed in Section 3.3. It would be interesting to use more accurate models as Gaussian Markov random fields [74] and try to find optimal $\mathcal{U}/\mathcal{P}$ assignment methods under these models.

RDO process presented in Section 4.3 assumed some simplifications, discussed in that section. Given that experimental results obtained using RDO are promising, an interesting direction for future work could be to design a practical RDO process.

The flexibility of the transform and the good results obtained in a video coding application provides confidence that it may be successfully applied in a broad kind of signals and applications. For example, as discussed in Section 4.1.1, it may be used for multichannel-audio coding, trying to jointly exploit the different correlations that arise in audio signals, obtaining a frequency and time localized compact representation of the multiple channels, which is an important property in order to consider subjective models. Other applications could be image and video denoising, or biomedical signals compact representation, where one usually has multiple signals that present correlation in different domains (e.g., data extracted from the temporal evolution of different brain sensors present spatial and temporal correlation).

Appendix A

Greedy Algorithm for the $SC_{\mathcal{U}}$/$SC_{\mathcal{P}}$

This appendix contains the greedy algorithm used to obtain the $SC_{\mathcal{U}}$ solution. Note that the $SC_{\mathcal{P}}$ solution is equivalent, and thus can be found with the same algorithm, just by exchanging $\mathcal{P}$ and $\mathcal{U}$ sets.

$\mathcal{V}$ is the set of nodes of the graph, $\mathcal{M}$ is a collection of all sets $\mathcal{N}_{[k]}$, with $k \in \mathcal{V}$, and *Gain* of a node is the number of neighbors that a node has.

Algorithm 8 $SC_{\mathcal{U}}$ Algorithm

Require: $\mathcal{M} = \{\mathcal{N}_k\}_{k\in\mathcal{V}}, \mathcal{R} = \mathcal{V}, \mathcal{U} = \{\varnothing\}, \mathcal{P} = \{\varnothing\}$

Calculate the $Gain$ of the $\mathcal{R}$ node set
Select the node a with largest $Gain$, $a = max(Gain)$
while $\mathcal{R} \neq \{\varnothing\}$ **do**
    Let $\mathcal{U} \leftarrow \mathcal{U} \cup \{a\}$
    Let $\mathcal{P} \leftarrow \mathcal{P} \cup \mathcal{N}_a$
    Remove the incident edges to $\{a \cup \mathcal{N}_a\}$
    Update $Gain$
    Select the node a with largest $Gain$, $a = max(Gain)$
    $\mathcal{R} \leftarrow \mathcal{R} \backslash \{a \cup \mathcal{N}_a\}$
end while
return $\mathcal{U}$ and $\mathcal{P}$

Appendix B

Additional Proofs

This appendix contains additional proofs for Chapter 3.

B.1 Proof of Proposition 3.2

Proof. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph, where $\mathcal{V} = \{1, \ldots, N\}$ is a set of nodes and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ a set of edges. Let $\mathcal{X}$ be a set of N random variables, such that x_i represents the data value associated to node i in the graph.

Let us assume that x_i is generated as the mean noise value ϵ_j of the closed neighborhood of node i plus an independent noise η_i as:

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

where ϵ_j and η_i are zero-mean independent random variables, with variances v_ϵ and v_η, respectively; $\mathcal{N}_{[i]}$ is the closed neighborhood set of node i, and α is an arbitrary non-negative real constant. Note that we are considering that $v_{\eta_i} = v_\eta$ and that $v_{\epsilon_i} = v_\epsilon$ for any $i \in \mathcal{V}$.

For each node $i \in \mathcal{P}$, consider the estimator given by

$$\hat{x}_i = \frac{1}{m_i} \sum_{j \in \mathcal{N}_i \cap \mathcal{U}} x_j, \tag{B.2}$$

where $m_i = |\mathcal{N}_i \cap \mathcal{U}|$.

$\hat{x}_i$ is an unbiased estimate of x_i,

$$\mathbb{E}\{\hat{x}_i\} = \frac{1}{m_i} \sum_{j \in \mathcal{N}_i \cap \mathcal{U}} \mathbb{E}\{x_j\} = \frac{1}{m_i} \sum_{j \in \mathcal{N}_i \cap \mathcal{U}} \left(\frac{1}{|\mathcal{N}_{[j]}|} \sum_{k \in \mathcal{N}_{[j]}} \mathbb{E}\{\epsilon_k\} + \alpha \mathbb{E}\{\eta_j\} \right) = 0. \tag{B.3}$$

The expected value of the squared error of node i can be written as:

$$\mathbb{E}\{(x_i - \hat{x}_i)^2\} = \mathbb{E}\{(x_i)^2\} + \mathbb{E}\{(\hat{x}_i)^2\} - 2\mathbb{E}\{x_i\hat{x}_i\} = \text{var}(x_i) + \text{var}(\hat{x}_i) - 2\mathbb{E}\{x_i\hat{x}_i\}, \tag{B.4}$$

where we have used that $\mathbb{E}\{(x_i)\} = \mathbb{E}\{(\hat{x}_i)\} = 0$.

Let us first calculate the variance of the model, var(x_i) :

$$\begin{aligned}
\mathbb{E}\{(x_i)^2\} = \text{var}(x_i) &= \mathbb{E}\{(\frac{1}{|\mathcal{N}_{[i]}|}\sum_{j\in\mathcal{N}_{[i]}}\epsilon_j + \alpha\eta_i)^2\} \\
&= \frac{1}{|\mathcal{N}_{[i]}|^2}\mathbb{E}\{(\sum_{j\in\mathcal{N}_{[i]}}\epsilon_j)^2\} + \alpha^2\mathbb{E}\{\eta_i^2\} + \frac{2\alpha}{|\mathcal{N}_{[i]}|}\mathbb{E}\{\sum_{j\in\mathcal{N}_{[i]}}\epsilon_j\eta_i\} \\
&= \frac{1}{|\mathcal{N}_{[i]}|^2}\sum_{m\in\mathcal{N}_{[i]}}\sum_{n\in\mathcal{N}_{[i]}}\mathbb{E}\{\epsilon_m\epsilon_n\} + \alpha^2 v_\eta \\
&= \frac{1}{|\mathcal{N}_{[i]}|^2}\sum_{m\in\mathcal{N}_{[i]}}\sum_{n\in\mathcal{N}_{[i]}}\delta_{m,n} + \alpha^2 v_\eta \\
&= \frac{v_\epsilon}{|\mathcal{N}_{[i]}|} + \alpha^2 v_\eta,
\end{aligned} \tag{B.5}$$

where we have used that ϵ_j and η_i are zero-mean independent random variables, and that $\sum_{m\in\mathcal{N}_{[i]}}\sum_{n\in\mathcal{N}_{[i]}}\delta_{m,n} = |\mathcal{N}_{[i]}|$.

Next we obtain the variance of the estimator, var($\hat{x}_i$):

$$\mathbb{E}\{(\hat{x}_i)^2\} = \text{var}(\hat{x}_i) = \mathbb{E}\{\frac{1}{m_i^2}(\sum_{j\in\mathcal{N}_i\cap\mathcal{U}}x_j)^2\} = \frac{1}{m_i^2}\sum_{m\in\mathcal{N}_i\cap\mathcal{U}}\sum_{n\in\mathcal{N}_i\cap\mathcal{U}}\mathbb{E}\{x_m x_n\}. \tag{B.6}$$

Note that

$$\begin{aligned}
\mathbb{E}\{x_m x_n\} &= \mathbb{E}\{(\frac{1}{|\mathcal{N}_{[m]}|}\sum_{j\in\mathcal{N}_{[m]}}\epsilon_j + \alpha\eta_m)(\frac{1}{|\mathcal{N}_{[n]}|}\sum_{k\in\mathcal{N}_{[n]}}\epsilon_k + \alpha\eta_n)\\
&= \frac{1}{|\mathcal{N}_{[m]}||\mathcal{N}_{[n]}|}\sum_{j\in\mathcal{N}_{[m]}}\sum_{k\in\mathcal{N}_{[n]}}\mathbb{E}\{\epsilon_j\epsilon_k\} + \frac{\alpha}{|\mathcal{N}_{[m]}|}\mathbb{E}\{\sum_{j\in\mathcal{N}_{[m]}}\epsilon_j\eta_n\}\\
&+ \frac{\alpha}{|\mathcal{N}_{[n]}|}\mathbb{E}\{\sum_{k\in\mathcal{N}_{[n]}}\epsilon_k\eta_m\} + \alpha^2\mathbb{E}\{\eta_m\eta_n\}\\
&= \frac{v_\epsilon}{|\mathcal{N}_{[m]}||\mathcal{N}_{[n]}|}\sum_{m\in\mathcal{N}_{[j]}}\sum_{n\in\mathcal{N}_{[k]}}\delta_{j,k} + \alpha^2 v_\eta\delta_{m,n}\\
&= v_\epsilon\frac{|\mathcal{N}_{[m]}\cap\mathcal{N}_{[n]}|}{|\mathcal{N}_{[m]}||\mathcal{N}_{[n]}|} + \alpha^2 v_\eta\delta_{m,n}. \qquad \text{(B.7)}
\end{aligned}$$

where we have used that ϵ_j and η_i are zero-mean independent random variables.

From (B.6) and (B.7) we get

$$\begin{aligned}
\text{var}(\hat{x}_i) &= \frac{1}{m_i^2}\sum_{m\in\mathcal{N}_{[i]}\cap\mathcal{U}}\sum_{n\in\mathcal{N}_{[i]}\cap\mathcal{U}}\left\{v_\epsilon\frac{|\mathcal{N}_{[m]}\cap\mathcal{N}_{[n]}|}{|\mathcal{N}_{[m]}||\mathcal{N}_{[n]}|} + \alpha^2 v_\eta\delta_{m,n}\right\} \qquad \text{(B.8)}\\
&= \frac{v_\epsilon}{m_i^2}\sum_{m\in\mathcal{N}_{[i]}\cap\mathcal{U}}\sum_{n\in\mathcal{N}_{[i]}\cap\mathcal{U}}\frac{|\mathcal{N}_{[m]}\cap\mathcal{N}_{[n]}|}{|\mathcal{N}_{[m]}||\mathcal{N}_{[n]}|} + \frac{\alpha^2 v_\eta}{m_i},
\end{aligned}$$

where we have used that $\sum_{m\in\mathcal{N}_{[i]}\cap\mathcal{U}}\sum_{n\in\mathcal{N}_{[i]}\cap\mathcal{U}}\delta_{m,n} = m_i$.

Now, we obtain the correlation between the model and the estimator $\mathbb{E}\{x_i\hat{x}_i\}$:

$$\begin{aligned}
\mathbb{E}\{x_i\hat{x}_i\} &= \mathbb{E}\{x_i\frac{1}{m_i}\sum_{j\in\mathcal{N}_i\cap\mathcal{U}}x_j\} \qquad \text{(B.9)}\\
&= \frac{1}{m_i}\sum_{j\in\mathcal{N}_i\cap\mathcal{U}}\mathbb{E}\{x_i x_j\}\\
&= \frac{1}{m_i}\sum_{j\in\mathcal{N}_i\cap\mathcal{U}}\left\{v_\epsilon\frac{|\mathcal{N}_{[i]}\cap\mathcal{N}_{[j]}|}{|\mathcal{N}_{[i]}||\mathcal{N}_{[j]}|} + \alpha^2 v_\eta\delta_{i,j}\right\}\\
&= \frac{v_\epsilon}{m_i}\sum_{j\in\mathcal{N}_{[i]}\cap\mathcal{U}}\frac{|\mathcal{N}_{[i]}\cap\mathcal{N}_{[j]}|}{|\mathcal{N}_{[i]}||\mathcal{N}_{[j]}|},
\end{aligned}$$

where we have used that $\sum_{j\in\mathcal{N}_{[i]}\cap\mathcal{U}} v_\eta\delta_{i,j} = 0$.

From (B.6), (B.8), and (B.9), we obtain the mean squared prediction error of a node $i \in \mathcal{P}$:

$$\begin{aligned}\mathbb{E}\{(x_i - \hat{x}_i)^2\} &= \mathbb{E}\{(x_i)^2\} + \mathbb{E}\{(\hat{x}_i)^2\} - 2\mathbb{E}\{x_i\hat{x}_i\} \\ &= \alpha^2 v_\eta + \frac{v_\epsilon}{|\mathcal{N}_{[i]}|} + \frac{\alpha^2 v_\eta}{m_i} + \frac{v_\epsilon}{m_i^2} \sum_{j \in \mathcal{N}_{[i]} \cap \mathcal{U}} \sum_{k \in \mathcal{N}_{[i]} \cap \mathcal{U}} \frac{|\mathcal{N}_{[j]} \cap \mathcal{N}_{[k]}|}{|\mathcal{N}_{[j]}||\mathcal{N}_{[k]}|} \\ &\quad - 2\frac{v_\epsilon}{m_i} \sum_{j \in \mathcal{N}_{[i]} \cap \mathcal{U}} \frac{|\mathcal{N}_{[i]} \cap \mathcal{N}_{[j]}|}{|\mathcal{N}_{[i]}||\mathcal{N}_{[j]}|}.\end{aligned} \tag{B.10}$$

Define the *clustering degree* of nodes j and k on graph $\mathcal{G}$ as

$$c(j,k) = \frac{|\mathcal{N}_{[j]} \cap \mathcal{N}_{[k]}|}{|\mathcal{N}_{[j]}||\mathcal{N}_{[k]}|}. \tag{B.11}$$

From (B.10) and (B.11) we have:

$$\begin{aligned}\mathbb{E}\{(x_i - \hat{x}_i)^2\} &= \alpha^2 v_\eta + \frac{v_\epsilon}{|\mathcal{N}_{[i]}|} + \frac{\alpha^2 v_\eta}{m_i} + \frac{v_\epsilon}{m_i^2} \sum_{j \in \mathcal{N}_{[i]} \cap \mathcal{U}} \sum_{k \in \mathcal{N}_{[i]} \cap \mathcal{U}} c(j,k) \\ &\quad - 2\frac{v_\epsilon}{m_i} \sum_{k \in \mathcal{N}_{[i]} \cap \mathcal{U}} c(i,k).\end{aligned} \tag{B.12}$$

□

B.2 Proof of Proposition 3.3

Proof. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph, where $\mathcal{V} = \{1, \ldots, N\}$ is a set of nodes and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ a set of edges. Let $\mathcal{X}$ be a set of N random variables, such that x_i represents the data value associated to node i in the graph.

Let us assume that

$$x_i = \left(\frac{r_s}{|\mathcal{N}_{[i]}^s|} \sum_{j \in \mathcal{N}_{[i]}^s} \epsilon_j + \frac{r_t}{|\mathcal{N}_{[i]}^t|} \sum_{k \in \mathcal{N}_{[i]}^t} \epsilon_k \right) + \alpha\eta_i, \tag{B.13}$$

where $\mathcal{N}^s_{[i]}$ and $\mathcal{N}^t_{[i]}$ are the closed sets of spatial and temporal neighbors, respectively, of node i; r_s is an arbitrary constant in $[0, 1]$, with $r_t = 1 - r_s$; ϵ_j and η_i are zero-mean independent random variables with variances v_{ϵ_j}, and v_{η_i}, respectively; and α is an arbitrary nonnegative real constant.

Consider the predictions given by

$$\hat{x}_i = \frac{w_s}{m^s_i} \sum_{j \in \mathcal{N}^s_i \cap \mathcal{U}} x_j + \frac{w_t}{m^t_i} \sum_{j \in \mathcal{N}^t_i \cap \mathcal{U}} x_j, \tag{B.14}$$

where $m^s_i = |\mathcal{N}^s_i \cap \mathcal{U}|$ and $m^t_i = |\mathcal{N}^t_i \cap \mathcal{U}|$.

Consider that $v_{\eta_i} = v_\eta$ and that $v_{\epsilon_i} = v_\epsilon$ for any $i \in \mathcal{V}$.

Estimate $\hat{x}_i$ is an unbiased estimate of x_i,

$$\mathbb{E}\{x_i\} = \frac{r_s}{|\mathcal{N}^s_{[i]}|} \sum_{j \in \mathcal{N}^s_{[i]}} \mathbb{E}\{\epsilon_j\} + \frac{r_t}{|\mathcal{N}^t_{[i]}|} \sum_{k \in \mathcal{N}^t_{[i]}} \mathbb{E}\{\epsilon_k\} + \alpha \mathbb{E}\{\eta_i\} = 0 \tag{B.15}$$

and

$$\mathbb{E}\{\hat{x}_i\} = \frac{w_s}{m^s_i} \sum_{j \in \mathcal{N}^s_i \cap \mathcal{U}} \mathbb{E}\{x_j\} + \frac{w_t}{m^t_i} \sum_{k \in \mathcal{N}^t_i \cap \mathcal{U}} \mathbb{E}\{x_k\} = 0. \tag{B.16}$$

The expected value of the squared error of node i can be written as:

$$\mathbb{E}\{(x_i - \hat{x}_i)^2\} = \mathbb{E}\{(x_i)^2\} + \mathbb{E}\{(\hat{x}_i)^2\} - 2\mathbb{E}\{x_i \hat{x}_i\} = \text{var}(x_i) + \text{var}(\hat{x}_i) - 2\mathbb{E}\{x_i \hat{x}_i\}. \tag{B.17}$$

where we have used that $\mathbb{E}\{x_i\} = \mathbb{E}\{\hat{x}_i\} = 0$.

Let us first calculate the variance of the model, var(x_i) :

$$
\begin{aligned}
\mathbb{E}\{(x_i)^2\} = \text{var}(x_i) &= \mathbb{E}\{(\frac{r_s}{|\mathcal{N}^s_{[i]}|}\sum_{j\in\mathcal{N}^s_{[i]}}\epsilon_j + \frac{r_t}{|\mathcal{N}^t_{[i]}|}\sum_{k\in\mathcal{N}^t_{[i]}}\epsilon_k + \alpha\eta_i)^2\} \\
&= \mathbb{E}\{\left(\frac{r_s}{|\mathcal{N}^s_{[i]}|}\sum_{j\in\mathcal{N}^s_{[i]}}\epsilon_j\right)^2 + \left(\frac{r_t}{|\mathcal{N}^t_{[i]}|}\sum_{k\in\mathcal{N}^t_{[i]}}\epsilon_k\right)^2 \\
&+ 2\frac{r_s}{|\mathcal{N}^s_{[i]}|}\frac{r_t}{|\mathcal{N}^t_{[i]}|}\sum_{j\in\mathcal{N}^s_{[i]}}\sum_{k\in\mathcal{N}^t_{[i]}}\epsilon_j\epsilon_k + \alpha^2\eta_i^2\} \\
&= \left(\frac{r_s}{|\mathcal{N}^s_{[i]}|}\right)^2\sum_{m\in\mathcal{N}^s_{[i]}}\sum_{n\in\mathcal{N}^s_{[i]}}\mathbb{E}\{\epsilon_m\epsilon_n\} + \left(\frac{r_t}{|\mathcal{N}^t_{[i]}|}\right)^2\sum_{m\in\mathcal{N}^t_{[i]}}\sum_{n\in\mathcal{N}^t_{[i]}}\mathbb{E}\{\epsilon_m\epsilon_n\} \\
&+ 2\frac{r_s}{|\mathcal{N}^s_{[i]}|}\frac{r_t}{|\mathcal{N}^t_{[i]}|}\sum_{m\in\mathcal{N}^s_{[i]}}\sum_{n\in\mathcal{N}^t_{[i]}}\mathbb{E}\{\epsilon_m\epsilon_n\} + \alpha^2\mathbb{E}\{\eta_i^2\} \\
&= \left(\frac{r_s}{|\mathcal{N}^s_{[i]}|}\right)^2 v_\epsilon\sum_{m\in\mathcal{N}^s_{[i]}}\sum_{n\in\mathcal{N}^s_{[i]}}\delta_{m,n} + \left(\frac{r_t}{|\mathcal{N}^t_{[i]}|}\right)^2 v_\epsilon\sum_{m\in\mathcal{N}^t_{[i]}}\sum_{n\in\mathcal{N}^t_{[i]}}\delta_{m,n} \\
&+ 2\frac{r_s}{|\mathcal{N}^s_{[i]}|}\frac{r_t}{|\mathcal{N}^t_{[i]}|}v_\epsilon\sum_{m\in\mathcal{N}^s_{[i]}}\sum_{n\in\mathcal{N}^t_{[i]}}\delta_{m,n} + \alpha^2\mathbb{E}\{\eta_i^2\} \\
&= \alpha^2 v_\eta + v_\epsilon\left(\frac{r_s^2}{|\mathcal{N}^s_{[i]}|} + \frac{r_t^2}{|\mathcal{N}^t_{[i]}|} + \frac{2r_sr_t}{|\mathcal{N}^s_{[i]}||\mathcal{N}^t_{[i]}|}\right),
\end{aligned}
\tag{B.18}
$$

where we have used that ϵ_j and η_i are zero-mean independent random variables, and that $v_\epsilon\sum_{m\in\mathcal{N}^s_{[i]}}\sum_{n\in\mathcal{N}^s_{[i]}}\delta_{m,n} = v_\epsilon|\mathcal{N}^s_{[i]}|$ (similarly for the temporal neighbors) and $v_\epsilon\sum_{m\in\mathcal{N}^s_{[i]}}\sum_{n\in\mathcal{N}^t_{[i]}}\delta_{m,n} = v_\epsilon$.

Next we obtain the variance of the estimator, var($\hat{x}_i$). Estimate $\hat{x}_i$ is unbiased, with variance:

$$\begin{aligned}
\text{var}(\hat{x}_i) &= \mathbb{E}\{(\frac{w_s}{m_i^s}\sum_{j\in\mathcal{N}_i^s\cap\mathcal{U}} x_j + \frac{w_t}{m_i^t}\sum_{k\in\mathcal{N}_i^t\cap\mathcal{U}} x_k)^2\} \\
&= \frac{w_s^2}{(m_i^s)^2}\sum_{m\in\mathcal{N}_i^s\cap\mathcal{U}}\sum_{n\in\mathcal{N}_i^s\cap\mathcal{U}}\mathbb{E}\{x_m x_n\} \\
&+ \frac{w_t^2}{(m_i^t)^2}\sum_{m\in\mathcal{N}_i^t\cap\mathcal{U}}\sum_{n\in\mathcal{N}_i^t\cap\mathcal{U}}\mathbb{E}\{x_m x_n\} \\
&+ \frac{2w_s w_t}{m_i^s m_i^t}\sum_{j\in\mathcal{N}_i^s\cap\mathcal{U}}\sum_{k\in\mathcal{N}_i^t\cap\mathcal{U}}\mathbb{E}\{x_j x_k\}. \qquad \text{(B.19)}
\end{aligned}$$

Note that

$$\begin{aligned}
\mathbb{E}\{x_j x_k\} &= \frac{r_s^2}{|\mathcal{N}_{[j]}^s||\mathcal{N}_{[k]}^s|}\sum_{m\in\mathcal{N}_{[j]}^s}\sum_{r\in\mathcal{N}_{[k]}^s}\mathbb{E}\{\epsilon_m\epsilon_r\} + \frac{r_t^2}{|\mathcal{N}_{[j]}^t||\mathcal{N}_{[k]}^t|}\sum_{n\in\mathcal{N}_{[j]}^t}\sum_{s\in\mathcal{N}_{[k]}^t}\mathbb{E}\{\epsilon_n\epsilon_s\} \\
&+ \frac{r_s r_t}{|\mathcal{N}_{[j]}^s||\mathcal{N}_{[k]}^t|}\sum_{m\in\mathcal{N}_{[j]}^s}\sum_{s\in\mathcal{N}_{[k]}^t}\mathbb{E}\{\epsilon_m\epsilon_s\} + \frac{r_t r_s}{|\mathcal{N}_{[j]}^t||\mathcal{N}_{[k]}^s|}\sum_{n\in\mathcal{N}_{[j]}^t}\sum_{r\in\mathcal{N}_{[k]}^s}\mathbb{E}\{\epsilon_n\epsilon_r\} \\
&+ \alpha^2\mathbb{E}\{\eta_j\eta_k\} \\
&= \frac{v_\epsilon r_s^2|\mathcal{N}_{[j]}^s\cap\mathcal{N}_{[k]}^s|}{|\mathcal{N}_{[j]}^s||\mathcal{N}_{[k]}^s|} + \frac{v_\epsilon r_t^2|\mathcal{N}_{[j]}^t\cap\mathcal{N}_{[k]}^t|}{|\mathcal{N}_{[j]}^t||\mathcal{N}_{[k]}^t|} + \frac{v_\epsilon r_s r_t|\mathcal{N}_{[j]}^s\cap\mathcal{N}_{[k]}^t|}{|\mathcal{N}_{[j]}^s||\mathcal{N}_{[k]}^t|} \\
&+ \frac{r_s r_t|\mathcal{N}_{[j]}^t\cap\mathcal{N}_{[k]}^s|}{|\mathcal{N}_{[j]}^t||\mathcal{N}_{[k]}^s|} + \alpha^2 v_\eta\delta_{j,k}, \qquad \text{(B.20)}
\end{aligned}$$

where we have used that ϵ and η are zero-mean independent random variables.

Define

$$D_{ab}^{cd}(i) = \sum_{j\in\mathcal{N}_i^a\cap\mathcal{U}}\sum_{k\in\mathcal{N}_i^b\cap\mathcal{U}}\frac{|\mathcal{N}_{[j]}^c\cap\mathcal{N}_{[k]}^d|}{|\mathcal{N}_{[j]}^c||\mathcal{N}_{[k]}^d|} \qquad \text{(B.21)}$$

for a, b, c, d equal to "s" or "t".

From (B.19), (B.20), and (B.21) we get:

$$\begin{aligned}\text{var}(\hat{x}_i) = &\frac{v_\epsilon w_s^2}{(m_i^s)^2}(r_s^2 D_{ss}^{ss}(i) + r_t^2 D_{ss}^{tt}(i) + 2r_s r_t(D_{ss}^{st}(i)) + \alpha^2 v_\eta m_i^s) \\ &+ \frac{v_\epsilon w_t^2}{(m_i^t)^2}(r_s^2 D_{tt}^{ss}(i) + r_t^2 D_{tt}^{tt}(i) + 2r_s r_t(D_{tt}^{st}(i)) + \alpha^2 v_\eta m_i^t) \\ &+ \frac{2v_\epsilon w_s w_t}{m_i^s m_i^t}(r_s^2 D_{st}^{ss}(i) + r_t^2 D_{st}^{tt}(i) + r_s r_t(D_{st}^{st}(i) + D_{st}^{ts}(i))).\end{aligned} \tag{B.22}$$

where we have used that $D_{ss}^{st}(i) = D_{ss}^{ts}(i)$ and $D_{tt}^{st}(i) = D_{tt}^{ts}(i)$.

Finally, we obtain the correlation between the model and the estimator $\mathbb{E}\{x_i\hat{x}_i\}$. Define

$$D_a^{cd}(i) = \sum_{j \in \mathcal{N}_i^a \cap \mathcal{U}} \frac{|\mathcal{N}_{[j]}^c \cap \mathcal{N}_{[i]}^d|}{|\mathcal{N}_{[j]}^c||\mathcal{N}_{[i]}^d|} \tag{B.23}$$

for a, c, d equal to "s" or "t".

We can write:

$$\begin{aligned}\mathbb{E}\{x_i\hat{x}_i\} &= \mathbb{E}\{x_i(\frac{w_s}{m_i^s}\sum_{j \in \mathcal{N}_i^s \cap \mathcal{U}} x_j + \frac{w_t}{m_i^t}\sum_{k \in \mathcal{N}_i^t \cap \mathcal{U}} x_k)\} \\ &= \frac{w_s}{m_i^s}\sum_{j \in \mathcal{N}_{is} \cap \mathcal{U}} \mathbb{E}\{x_j x_i\} + \frac{w_t}{m_i^t}\sum_{k \in \mathcal{N}_{it} \cap \mathcal{U}} \mathbb{E}\{x_k x_i\}.\end{aligned} \tag{B.24}$$

Using definition (B.23) in (B.24), we get:

$$\begin{aligned}\mathbb{E}\{x_i\hat{x}_i\} = &\frac{v_\epsilon w_s}{m_i^s}(r_s^2 D_s^{ss}(i) + r_t^2 D_s^{tt}(i) + r_s r_t(D_s^{st}(i) + D_s^{ts}(i))) \\ &+ \frac{v_\epsilon w_t}{m_i^t}(r_s^2 D_t^{ss}(i) + r_t^2 D_t^{tt}(i) + r_s r_t(D_t^{st}(i) + D_t^{ts}(i)).\end{aligned} \tag{B.25}$$

Fixing $r_s = w_s$ and $r_t = w_t$ and using (B.17), (B.19), (B.22) and (B.25), the expected value of the squared error of node i can be written as:

$$\begin{aligned}\mathbb{E}\{(x_i - \hat{x}_i)^2\} &= \alpha^2 v_\eta + v_\epsilon \left(\frac{w_s^2}{|\mathcal{N}_{[i]}^s|} + \frac{w_t^2}{|\mathcal{N}_{[i]}^t|} + \frac{2w_s w_t}{|\mathcal{N}_{[i]}^s||\mathcal{N}_{[i]}^t|} \right) \\ &+ \alpha^2 v_\eta \left(\frac{w_s^2}{m_i^s} + \frac{w_t^2}{m_i^t} \right) + v_\epsilon \left(\frac{w_s^2}{(m_i^s)^2} G + \frac{w_t^2}{(m_i^t)^2} H + \frac{2w_s w_t}{m_i^t m_i^s} I \right) \\ &- 2v_\epsilon \left(\frac{w_s}{m_i^s} J + \frac{w_t}{m_i^t} K \right),\end{aligned} \tag{B.26}$$

where

$$\begin{aligned} G &= w_s^2 D_{ss}^{ss} + w_t^2 D_{ss}^{tt} + 2w_s w_t D_{ss}^{st}, \\ H &= w_s^2 D_{tt}^{ss} + w_t^2 D_{tt}^{tt} + 2w_s w_t D_{tt}^{st}, \\ I &= w_s^2 D_{st}^{ss} + w_t^2 D_{st}^{tt} + w_s w_t (D_{st}^{st} + D_{st}^{ts}), \\ J &= w_s^2 D_s^{ss} + w_t^2 D_s^{tt} + w_s w_t (D_s^{st} + D_s^{ts}), \\ K &= w_s^2 D_t^{ss} + w_t^2 D_t^{tt} + w_s w_t (D_t^{st} + D_t^{ts}). \end{aligned} \tag{B.27}$$

□

Appendix C

Optimal Weighting for a Given Graph and $\mathcal{U}/\mathcal{P}$ Assignment

This appendix contains the formulation to obtain the optimal weights that minimize the detail coefficient energy using a given graph and $\mathcal{U}/\mathcal{P}$ assignment, and assuming one-hop filters defined below. First, in Section C.1, we consider a video representation example. Then, in section C.2, we extend this result to F kinds of edges with different correlations.

C.1 Optimal Weighting for a Video Representation Given an $\mathcal{U}/\mathcal{P}$ Assignment

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph, where $\mathcal{V} = \{1, \ldots, N\}$ is a set of nodes and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ a set of edges. Let $\mathcal{S}, \mathcal{T}$ be the set of spatial and temporal edges, respectively, with $\mathcal{S} \cup \mathcal{T} = \mathcal{E}$. Denote one-hop spatial neighborhood of i as $\mathcal{N}_i^{\mathcal{S}} = \{j \in \mathcal{V} : ij \in \mathcal{S}\}$. Let $m_i^{\mathcal{S}} = |\mathcal{N}_i^{\mathcal{S}} \cap \mathcal{U}|$ be the number of one-hop spatial update neighbors of node $i \in \mathcal{P}$. Thus, the mean value of the update spatial neighbors of a node $i \in \mathcal{P}$ is defined as

$$\bar{x}_i^s = \frac{1}{m_i^{\mathcal{S}}} \sum_{j \in \mathcal{N}_i^{\mathcal{S}} \cap \mathcal{U}}^{m_i^{\mathcal{S}}} x_j, \tag{C.1}$$

and is defined similarly for the temporal neighbors. Assuming that every node $i \in \mathcal{P}$ is linearly predicted from its spatial and temporal update neighbors as:

$$\hat{x}_i = w_s \bar{x}_i^s + w_t \bar{x}_i^t, \tag{C.2}$$

we want to find the weights w_s and w_t that minimize the quadratic prediction error over all the nodes $i \in \mathcal{P}$:

$$\min_{w_s,w_t} \sum_{i\in\mathcal{P}} \left(x_i - \hat{x}_i\right)^2 = \min_{w_s,w_t} \sum_{i\in\mathcal{P}} \left(d_i\right)^2 = \min_{w_s,w_t} \sum_{i\in\mathcal{P}} \left(x_i - w_s\bar{x}_i^s - w_t\bar{x}_i^t\right)^2. \tag{C.3}$$

Differentiating with respect to w_s and w_t we obtain the solution:

$$\mathbf{w}^* = (w_s^*, w_t^*) = \mathbf{R^{-1}r} \tag{C.4}$$

where

$$\mathbf{R} = \begin{bmatrix} \sum_{i\in\mathcal{P}} \bar{x}_i^s\bar{x}_i^s & \sum_{i\in\mathcal{P}} \bar{x}_i^s\bar{x}_i^t \\ \sum_{i\in\mathcal{P}} \bar{x}_i^t\bar{x}_i^s & \sum_{i\in\mathcal{P}} \bar{x}_i^t\bar{x}_i^t \end{bmatrix} \tag{C.5}$$

and

$$\mathbf{r} = \sum_{i\in\mathcal{P}} x_i \begin{bmatrix} \bar{x}_i^s \\ \bar{x}_i^t \end{bmatrix} \tag{C.6}$$

are the correlation matrices.

Next, we express the optimal weight vector $\mathbf{w}^*$ as a function of matrices derived from the spatial and temporal adjacency matrices of the graph.

Let $\mathbf{A_s} = \left[a_{s_{i,j}}\right]$ and $\mathbf{A_t} = \left[a_{t_{i,j}}\right]$ be the adjacency matrices of the subgraphs containing only the spatial and temporal edges, respectively.

Let $\mathbf{i}_\mathcal{U}$ (resp. $\mathbf{i}_\mathcal{P}$) be a $N \times 1$ indicator vector in which $\{i_h\} = 1$ if node $h \in \mathcal{U}$ (resp. $\in \mathcal{P}$), and zero otherwise. Let $\mathbf{I}_\mathcal{U}$ (resp. $\mathbf{I}_\mathcal{P}$) be a diagonal matrix in which the main diagonal is $\mathbf{i}_\mathcal{U}$ (resp. $\mathbf{i}_\mathcal{P}$). Denote $\mathbf{B_s} = \mathbf{I}_\mathcal{U}\mathbf{A_s}\mathbf{I}_\mathcal{P}$ and $\mathbf{B_t} = \mathbf{I}_\mathcal{U}\mathbf{A_t}\mathbf{I}_\mathcal{P}$, and let $\overline{\mathbf{B}}_\mathbf{s} = \left[\overline{b}_{s_{i,j}}\right]$ and $\overline{\mathbf{B}}_\mathbf{t} = \left[\overline{b}_{t_{i,j}}\right]$ be the $\mathbf{B_s}$ and $\mathbf{B_t}$ matrices where each column is normalized (i.e., defining $|\mathcal{N}^j|$ as the number of non-zero elements of column j, $\overline{b}_{s_{i,j}} = 1/|\mathcal{N}^j|$ if $b_{s_{i,j}} = 1$; $\overline{b}_{s_{i,j}} = 0$ if $b_{s_{i,j}} = 0$).

"Vectorize" the graph data into a $1 \times N$ row vector $\mathbf{x}$ (e.g., if the data is a video sequence, we obtain a $1 \times (L \times H \times K)$ row vector, where $L \times H$ is the frame size and K the number of frames considered).

The optimal weight vector $\mathbf{w}^*$ can be written as:

$$\mathbf{w}^* = \begin{bmatrix} \mathbf{x}\overline{\mathbf{B}}_\mathbf{s}\overline{\mathbf{B}}_\mathbf{s}^\mathbf{T}\mathbf{x}^\mathbf{T} & \mathbf{x}\overline{\mathbf{B}}_\mathbf{s}\overline{\mathbf{B}}_\mathbf{t}^\mathbf{T}\mathbf{x}^\mathbf{T} \\ \mathbf{x}\overline{\mathbf{B}}_\mathbf{t}\overline{\mathbf{B}}_\mathbf{s}^\mathbf{T}\mathbf{x}^\mathbf{T} & \mathbf{x}\overline{\mathbf{B}}_\mathbf{t}\overline{\mathbf{B}}_\mathbf{t}^\mathbf{T}\mathbf{x}^\mathbf{T} \end{bmatrix}^{-1} \cdot \begin{bmatrix} \mathbf{x}\overline{\mathbf{B}}_\mathbf{s}\mathbf{x}^\mathbf{T} \\ \mathbf{x}\overline{\mathbf{B}}_\mathbf{t}\mathbf{x}^\mathbf{T} \end{bmatrix}. \tag{C.7}$$

Proof. The matrix product $\mathbf{I}_\mathcal{U}\mathbf{A}_\mathbf{s}$ is the spatial adjacency matrix, but setting $p \in \mathcal{P}$ rows as zero vectors, and similarly, $\mathbf{A}_\mathbf{s}\mathbf{I}_\mathcal{P}$ is the spatial adjacency matrix in which $u \in \mathcal{U}$ columns are zero vectors. Therefore, $\mathbf{B}_\mathbf{s} = \mathbf{I}_\mathcal{U}\mathbf{A}_\mathbf{s}\mathbf{I}_\mathcal{P}$ can be interpreted as a matrix in which the non-zero column i represents the node $i \in \mathcal{P}$ and the indices of its non-zero elements are the $u \in \mathcal{U}$ spatial neighbors of i.

Normalizing by columns, $\overline{\mathbf{B}}_\mathbf{s} = \left[\overline{b}_{s_{i,j}}\right]$, where $\overline{b}_{s_{i,j}} = 1/\left|\mathcal{N}^j\right|$. Note that $\overline{b}_{s_{i,j}} \neq 0$ if $j \in \mathcal{P}, i \in \mathcal{U}$, and $ij \in \mathcal{S}$, and that $|\mathcal{N}^j|$ is the number of $u \in \mathcal{U}$ spatial neighbors that the node $\in \mathcal{P}$ of column j has. Multiplying the $1 \times N$ data vector $\mathbf{x}$ by $\overline{\mathbf{B}}_\mathbf{s}$, $\mathbf{a} = [a_k] = \mathbf{x}\overline{\mathbf{B}}_\mathbf{s}$, we obtain a $1 \times N$ row vector:

$$[a_k] = \begin{cases} \bar{x}_k^s, & \text{if } k \in \mathcal{P}, \\ 0, & \text{if } k \in \mathcal{U}. \end{cases} \tag{C.8}$$

The above reasoning is equivalent for the temporal adjacency matrix $\mathbf{A}_\mathbf{t}$.

Therefore, $\mathbf{a}\mathbf{a}^\mathbf{T} = \sum_{i\in\mathcal{P}} \bar{x}_i^s \bar{x}_i^s = \mathbf{x}\overline{\mathbf{B}}_\mathbf{s}\left(\mathbf{x}\overline{\mathbf{B}}_\mathbf{s}\right)^T = \mathbf{x}\overline{\mathbf{B}}_\mathbf{s}\overline{\mathbf{B}}_\mathbf{s}^\mathbf{T}\mathbf{x}^\mathbf{T}$.

Similarly, $\sum_{i\in\mathcal{P}} \bar{x}_i^s \bar{x}_i^t = \mathbf{x}\overline{\mathbf{B}}_\mathbf{s}\overline{\mathbf{B}}_\mathbf{t}^\mathbf{T}\mathbf{x}^\mathbf{T}$.

□

C.2 Optimal Weighting for F Different Kinds of Links Given an $\mathcal{U}/\mathcal{P}$ Assignment

We now generalize the result in (C.7) to the case of F different kinds of links with different correlations.

For every node $i \in \mathcal{P}$, let us define the mean value of its update neighbors linked by means of links of class f as:

$$\bar{x}_i^f = \frac{1}{m_i^f} \sum_{j \in \mathcal{N}_i^f \cap \mathcal{U}}^{m_i^f} x_j. \tag{C.9}$$

Assuming that every node in $\mathcal{P}$ is linearly predicted from its F types of neighbors, we would like to find the weights $w_1, w_2, \ldots, w_F$ that minimize the quadratic prediction error over all the nodes $\in \mathcal{P}$:

$$\min_{w_1,w_2,\ldots,w_F} \sum_{i \in \mathcal{P}}^{|\mathcal{P}|} (x_i - \hat{x}_i)^2 = \min_{w_1,w_2,\ldots,w_F} \sum_{i \in \mathcal{P}}^{|\mathcal{P}|} \left(x_i - w_1 \bar{x}_i^1 - w_2 \bar{x}_i^2 - \ldots - w_F \bar{x}_i^F\right)^2. \tag{C.10}$$

The optimal weights can be obtained as:

$$\mathbf{w}^* = \begin{bmatrix} \mathbf{x\overline{B}_1\overline{B}_1^T x^T} & \mathbf{x\overline{B}_1\overline{B}_2^T x^T} & \cdots & \mathbf{x\overline{B}_1\overline{B}_F^T x^T} \\ \mathbf{x\overline{B}_2\overline{B}_1^T x^T} & \mathbf{x\overline{B}_2\overline{B}_2^T x^T} & \cdots & \mathbf{x\overline{B}_2\overline{B}_F^T x^T} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x\overline{B}_F\overline{B}_1^T x^T} & \mathbf{x\overline{B}_F\overline{B}_2^T x^T} & \cdots & \mathbf{x\overline{B}_F\overline{B}_F^T x^T} \end{bmatrix}^{-1} \cdot \begin{bmatrix} \mathbf{x\overline{B}_1 x^T} \\ \mathbf{x\overline{B}_2 x^T} \\ \vdots \\ \mathbf{x\overline{B}_F x^T} \end{bmatrix}. \quad \text{(C.11)}$$

Bibliography

[1] M. Do and M. Vetterli, "The contourlet transform: an efficient directional multiresolution image representation," *Image Processing, IEEE Transactions on*, vol. 14, no. 12, pp. 2091 –2106, dec. 2005.

[2] W. Sweldens, "The lifting scheme: A construction of second generation wavelets," 1995, tech. report 1995:6, Industrial Math. Initiative, Dept. of Math., University of South Carolina, 1995.

[3] E. J. Candès and D. L. Donoho, "Curvelets – a surprisingly effective nonadaptive representation for objects with edges," 2000.

[4] V. Velisavljevic, B. Beferull-Lozano, M. Vetterli, and P. L. Dragotti, "Directionlets: anisotropic multidirectional representation with separable filtering," *Image Processing, IEEE Transactions on*, vol. 15, no. 7, pp. 1916 –1933, July 2006.

[5] G. Shen and A. Ortega, "Compact image representation using wavelet lifting along arbitrary trees," in *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, October 2008, pp. 2808 –2811.

[6] A. Secker and D. Taubman, "Lifting-based invertible motion adaptive transform (limat) framework for highly scalable video compression," *Image Processing, IEEE Transactions on*, vol. 12, no. 12, pp. 1530 – 1542, December 2003.

[7] G. Pau, C. Tillier, B. Pesquet-Popescu, and H. Heijmans, "Motion compensation and scalability in lifting-based video coding," *Signal Processing: Image Communication*, vol. 19, no. 7, pp. 577 – 600, 2004, special Issue on Subband/Wavelet Interframe Video Coding.

[8] M. Flierl and B. Girod, "Video coding with motion-compensated lifted wavelet transforms." *Sig. Proc.: Image Comm.*, vol. 19, no. 7, pp. 561–575, 2004.

[9] R. Wagner, H. Choi, and R. Baraniuk, "Distributed wavelet transform for irregular sensor network grids," in *in IEEE Statistical Signal Processing (SSP) Workshop*, 2005.

[10] S. K. Narang and A. Ortega, "Lifting based wavelet transforms on graphs," in *APSIPA ASC 2009: Asia-Pacific Signal and Information Processing Association, 2009 Annual Summit and Conference*, October 2009.

[11] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Applied and Computational Harmonic Analysis*, vol. 30, no. 2, pp. 129–150, Mar 2011.

[12] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, 2013.

[13] S. K. Narang and A. Ortega, "Perfect reconstruction two-channel wavelet filter banks for graph structured data," *IEEE Transactions on Signal Processing*, vol. 60, no. 6, pp. 2786–2799, 2012.

[14] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs," *CoRR*, vol. abs/1210.4752, 2012. [Online]. Available: http://dblp.uni-trier.de/db/journals/corr/corr1210.html#abs-1210-4752

[15] P. Milanfar, "A tour of modern image filtering: New insights and methods, both practical and theoretical." *IEEE Signal Process. Mag.*, vol. 30, no. 1, pp. 106–128, 2013. [Online]. Available: http://dblp.uni-trier.de/db/journals/spm/spm30.html#Milanfar13

[16] G. Shen and A. Ortega, "Tree-based wavelets for image coding: Orthogonalization and tree selection," in *Picture Coding Symposium, 2009. PCS 2009*, May 2009, pp. 1 –4.

[17] S. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 11, no. 7, pp. 674–693, 1989.

[18] W. Sweldens, "The lifting scheme: A new philosophy in biorthogonal wavelet constructions," in *in Wavelet Applications in Signal and Image Processing III*, 1995, pp. 68–79.

[19] W. Sweldens and P. Schröder, "Building your own wavelets at home," 1995, tech. report 1995:5, Industrial Math. Initiative, Dept. of Math., University of South Carolina, 1995.

[20] W. Sweldens, "The lifting scheme: A custom-design construction of biorthogonal wavelets," *Applied and Computational Harmonic Analysis*, vol. 3, no. 2, pp. 186 – 200, 1996.

[21] S. Mallat, *A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way*, 3rd ed. Academic Press, 2008.

[22] G. Shen, "Lifting transforms on graphs: theory and applications." Ph.D. dissertation, University of Southern California, 2010.

[23] A. Cohen, I. Daubechies, and J.-C. Feauveau, "Biorthogonal bases of compactly supported wavelets," *Communications on Pure and Applied Mathematics*, vol. 45, no. 5, pp. 485–560, 1992. [Online]. Available: http://dx.doi.org/10.1002/cpa.3160450502

[24] R. Fattal, "Edge-avoiding wavelets and their applications," in *SIGGRAPH '09: ACM SIGGRAPH 2009 papers*. New York, NY, USA: ACM, 2009, pp. 1–10.

[25] J. Solé and P. Salembier, "Generalized lifting prediction optimization applied to lossless image compression," *Signal Processing Letters, IEEE*, vol. 14, no. 10, pp. 695 –698, oct. 2007.

[26] D. Taubman, "Adaptive, non-separable lifting transforms for image compression," in *Image Processing, 1999. ICIP 99. Proceedings. 1999 International Conference on*, vol. 3, 1999, pp. 772–776 vol.3.

[27] B. Pesquet-Popescu and V. Bottreau, "Three-dimensional lifting schemes for motion compensated video compression," in *ICASSP '01: Proceedings of the Acoustics, Speech, and Signal Processing, 2001. on IEEE International Conference*, Washington, DC, USA, 2001, pp. 1793–1796.

[28] G. Shen and A. Ortega, "Optimized distributed 2d transforms for irregularly sampled sensor network grids using wavelet lifting," in *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, March 2008, pp. 2513 –2516.

[29] ——, "Transform-based distributed data gathering," *Signal Processing, IEEE Transactions on*, vol. 58, no. 7, pp. 3802–3815, 2010.

[30] R. S. Wagner, R. G. Baraniuk, S. Du, D. B. Johnson, and A. Cohen, "An architecture for distributed wavelet analysis and processing in sensor networks," in *Proceedings of the 5th international conference on Information processing in sensor networks*, ser. IPSN '06. New York, NY, USA: ACM, 2006, pp. 243–250. [Online]. Available: http://doi.acm.org/10.1145/1127777.1127816

[31] J. Asensio-Cubero, J. Gan, and R. Palaniappan, "Extracting common spatial patterns based on wavelet lifting for brain computer interface design," in *Computer Science and Electronic Engineering Conference (CEEC), 2012 4th*, 2012, pp. 160–163.

[32] ——, “Multiresolution analysis over simple graphs for brain computer interfaces,” *Journal of neural engineering*, vol. 10, no. 4, p. 046014, 2013.

[33] I. Dutta, R. Banerjee, T. Acharya, and S. DasBit, “An energy efficient audio compression scheme using wavelet with dynamic difference detection technique in wireless sensor network,” in *Proceedings of the International Conference on Advances in Computing, Communications and Informatics*, ser. ICACCI ’12. New York, NY, USA: ACM, 2012, pp. 360–366. [Online]. Available: http://doi.acm.org/10.1145/2345396.2345456

[34] E. Le Pennec and S. Mallat, “Sparse geometric image representations with bandelets,” *Image Processing, IEEE Transactions on*, vol. 14, no. 4, pp. 423 –438, April 2005.

[35] E. L. Pennec and S. Mallat, “Bandelet image approximation and compression,” *SIAM Journal of Multiscale Modeling and Simulation*, vol. 4, p. 2005, 2005.

[36] B. Zeng and J. Fu, “Directional discrete cosine transforms-a new framework for image coding,” *IEEE Trans. Cir. and Sys. for Video Technol.*, vol. 18, no. 3, pp. 305–313, Mar. 2008. [Online]. Available: http://dx.doi.org/10.1109/TCSVT.2008.918455

[37] R. Claypoole, G. Davis, W. Sweldens, and R. Baraniuk, “Nonlinear wavelet transforms for image coding,” in *IEEE Trans. Image Process*, 1997, pp. 1449–1459.

[38] R. L. Claypoole, G. M. Davis, W. Sweldens, and R. G. Baraniuk, “Nonlinear wavelet transforms for image coding via lifting,” *Trans. Img. Proc.*, vol. 12, no. 12, pp. 1449–1459, Dec. 2003. [Online]. Available: http://dx.doi.org/10.1109/TIP.2003.817237

[39] N. Adami, A. Signoroni, and R. Leonardi, “State-of-the-art and trends in scalable video compression with wavelet-based approaches,” *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 17, no. 9, pp. 1238 –1255, September 2007.

[40] A. Sánchez, G. Shen, and A. Ortega, “Edge-preserving depth-map coding using graph-based wavelets,” in *Signals, Systems and Computers, 2009 Conference Record of the Forty-Third Asilomar Conference on*, 2009, pp. 578–582.

[41] W.-S. Kim, S. Narang, and A. Ortega, “Graph based transforms for depth video coding,” in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, 2012, pp. 813–816.

[42] E. Martinez-Enriquez, F. Diaz-de Maria, J. Cid-Sueiro, and A. Ortega, "Filter optimization and complexity reduction for video coding using graph-based transforms," in *Image Processing (ICIP), 2013 20th IEEE International Conference on*, Sept 2013, pp. 1948–1952.

[43] E. Martinez-Enriquez and A. Ortega, "Lifting transforms on graphs for video coding," in *Data Compression Conference (DCC), 2011*, march 2011, pp. 73 –82.

[44] E. Martinez-Enriquez, F. Diaz-de Maria, and A. Ortega, "Video encoder based on lifting transforms on graphs," in *Image Processing (ICIP), 2011 18th IEEE International Conference on*, sept. 2011, pp. 3509 –3512.

[45] S. Narang, G. Shen, and A. Ortega, "Unidirectional graph-based wavelet transforms for efficient data gathering in sensor networks," in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, 2010, pp. 2902 –2905.

[46] S. Narang and A. Ortega, "Local two-channel critically sampled filter-banks on graphs," in *Image Processing (ICIP), 2010 17th IEEE International Conference on*, sept. 2010, pp. 333 –336.

[47] C.-P. Hsu, "Minimum-via topological routing," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 2, no. 4, pp. 235 – 246, 1983.

[48] J. A. Cadzow, D. M. Wilkes, R. A. Peters, II, R. Alan, P. Ii, and X. K. Li, "Image texture synthesis-by-analysis using moving-average models," *IEEE Trans on Aerospace and Electrical Systems*, vol. 29, 1993.

[49] H. Kaufman, J. Woods, S. Dravida, and A. Tekalp, "Estimation and identification of two-dimensional images," *Automatic Control, IEEE Transactions on*, vol. 28, no. 7, pp. 745–756, 1983.

[50] G. Demoment, "Image reconstruction and restoration: overview of common estimation structures and problems," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 37, no. 12, pp. 2024–2036, 1989.

[51] N. Boulgouris, D. Tzovaras, and M. Strintzis, "Lossless image compression based on optimal prediction, adaptive lifting, and conditional arithmetic coding," *Image Processing, IEEE Transactions on*, vol. 10, no. 1, pp. 1 –14, jan 2001.

[52] A. Deever and S. Hemami, "Lossless image compression with projection-based and adaptive reversible integer wavelet transforms," *Image Processing, IEEE Transactions on*, vol. 12, no. 5, pp. 489 – 499, may 2003.

[53] G. P. Christophe, C. Tillier, and B. Pesquet-popescu, “Optimization of the predict operator in lifting-based motion compensated temporal filtering,” in *in Proc. of Visual Communications and Image Processing*, 2004.

[54] G. Shen, S. K. Narang, and A. Ortega, “Adaptive distributed transforms for irregularly sampled wireless sensor networks,” *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, vol. 0, pp. 2225–2228, 2009.

[55] B. Girod and S. Han, “Optimum update for motion-compensated lifting,” *Signal Processing Letters, IEEE*, vol. 12, no. 2, pp. 150 – 153, feb. 2005.

[56] C. Tillier, B. Pesquet-Popescu, and M. van der Schaar, “Improved update operators for lifting-based motion-compensated temporal filtering,” *Signal Processing Letters, IEEE*, vol. 12, no. 2, pp. 146 – 149, feb. 2005.

[57] J. M. Shapiro, “An embedded wavelet hierarchical image coder,” in *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, vol. 4, Mar. 1992, pp. 657 –660 vol.4.

[58] B.-J. Kim and W. A. Pearlman, “An embedded wavelet video coder using three-dimensional set partitioning in hierarchical trees (SPIHT),” in *Data Compression Conference, 1997. DCC ’97. Proceedings*, Mar. 1997, pp. 251 –260.

[59] E. Martinez-Enriquez, M. de Frutos-Lopez, J. C. Pujol-Alcolado, and F. Diaz-de Maria, “A fast motion-cost based algorithm for H.264/AVC inter mode decision,” in *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, vol. 5, 2007, pp. V – 325–V – 328.

[60] E. Martinez-Enriquez, A. Jimenez-Moreno, and F. Diaz-de Maria, “An adaptive algorithm for fast inter mode decision in the H.264/AVC video coding standard,” *Consumer Electronics, IEEE Transactions on*, vol. 56, no. 2, pp. 826–834, 2010.

[61] E. Martinez-Enriquez, A. Jimenez-Moreno, M. Angel-Pellon, and F. Diaz-de Maria, “A two-level classification-based approach to inter mode decision in H.264/AVC,” *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 21, no. 11, pp. 1719–1732, 2011.

[62] E. Martinez-Enriquez, A. Jimenez-Moreno, and F. Diaz-de Maria, “A novel fast inter mode decision in H.264/AVC based on a regionalized hypothesis testing,” in *Picture Coding Symposium, 2009. PCS 2009*, 2009, pp. 1–4.

[63] E. Martinez-Enriquez and F. Diaz-de Maria, “A hierarchical classification-based approach to inter mode decision in H.264/AVC,” in *Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on*, 2009, pp. 221–224.

[64] A. Jimenez-Moreno, E. Martinez-Enriquez, and F. Diaz-de Maria, "Mode decision-based algorithm for complexity control in H.264/AVC," *Multimedia, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2013.

[65] H. Everett, "Generalized Lagrange Multiplier Method for Solving Problems of Optimum Allocation of Resources," *Operations Research*, vol. 11, no. 3, pp. 399–417, May - Jun., 1963.

[66] K. Ramchandran and M. Vetterli, "Best wavelet packet bases in a rate-distortion sense," *Image Processing, IEEE Transactions on*, vol. 2, no. 2, pp. 160–175, 1993.

[67] A. Ortega and K. Ramchandran, "Rate-distortion methods for image and video compression," *Signal Processing Magazine, IEEE*, vol. 15, no. 6, pp. 23 –50, Nov. 1998.

[68] H. Gish and J. Pierce, "Asymptotically efficient quantizing," *Information Theory, IEEE Transactions on*, vol. 14, no. 5, pp. 676–683, 1968.

[69] S. Mallat and F. Falzon, "Analysis of low bit rate image transform coding," *Signal Processing, IEEE Transactions on*, vol. 46, no. 4, pp. 1027–1042, 1998.

[70] G. J. Sullivan, T. Wiegand, and P. Corporation, "Rate-distortion optimization for video compression," *IEEE Signal Processing Magazine*, vol. 15, pp. 74–90, 1998.

[71] B. Usevitch, "Optimal bit allocation for biorthogonal wavelet coding," in *Data Compression Conference, 1996. DCC '96. Proceedings*, 1996, pp. 387–395.

[72] T. André, M. Cagnazzo, M. Antonini, and M. Barlaud, "A scalable video coder with scan-based lifted MCWT and model-based bitrate allocation," *I3S Internal Report*, no. I3S/RR-2004-42-FR, 2004.

[73] A. Gouze, C. Parisot, M. Antonini, and M. Barlaud, "Optimal weighted model-based bit allocation for quincunx sampled images," in *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, vol. 3, 2003, pp. III–221–4 vol.2.

[74] C. Zhang and D. A. F. Florêncio, "Analyzing the optimality of predictive transform coding using graph-based models." *IEEE Signal Process. Lett.*, vol. 20, no. 1, pp. 106–109, 2013. [Online]. Available: http://dblp.uni-trier.de/db/journals/spl/spl20.html#ZhangF13