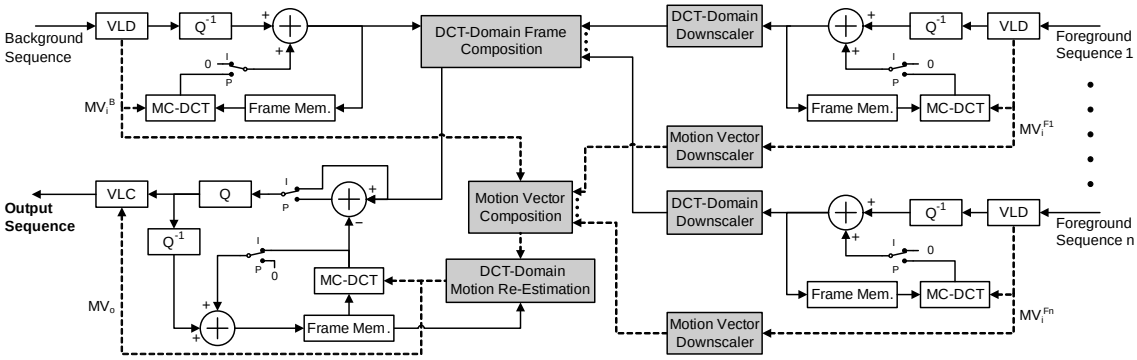
UNIVERSIDADE TÉCNICA DE LISBOA

INSTITUTO SUPERIOR TÉCNICO



Transform Domain Transcoding Systems for Static and Dynamic Video Composition

Nuno Filipe Valentim Roma

(Mestre)

Dissertação para obtenção do Grau de Doutor em

Engenharia Electrotécnica e de Computadores

Orientador: Doutor Leonel Augusto Pires Seabra de Sousa

Júri

Presidente: Reitor da Universidade Técnica de Lisboa

Vogais: Doutor Arlindo Manuel Limede de Oliveira
Doutor Luís António Pereira de Meneses Côrte-Real
Doutor Fernando Manuel Bernardo Pereira
Doutor Leonel Augusto Pires Seabra de Sousa
Doutor Vítor Manuel Mendes da Silva

Maio de 2008

This thesis is dedicated to my wife Cristina,
for her love, endless support
and encouragement.

Abstract

Video delivery systems and service providers often face the need to manipulate compressed video-streams. As a consequence, video transcoding has emerged as a new research area. It concerns a broad set of processing, manipulation and adaptation techniques to convert one video bit stream into another, with a more convenient set of parameters targeted to a given application. The prime focus of the conducted research was the development of efficient and flexible transcoding algorithms and architectures for video composition in the DCT-domain. Two distinct applications were targeted: static video composition, consisting on the insertion of stationary visible data over the received video sequence; and dynamic video composition, consisting on the composition of one or more foreground video sequences over the displaying area corresponding to the background video sequence. To support the implementation of the proposed transcoding structures, several common video processing operations had to be adapted and transposed into the DCT-domain, such as video space-scaling by an arbitrary integer scaling factor, and motion estimation using the blocks of DCT-coefficients obtained from the received video-streams. When compared with the corresponding pixel-domain counterparts, the proposed DCT-domain transcoders have shown to provide significant advantages, both in terms of the video quality, coding efficiency and computational cost.

Keywords

- Discrete Cosine Transform;
- Video Transcoding;
- Insertion of video objects;
- Video composition in the DCT-domain;
- Motion Estimation in the DCT-domain;
- Space-scaling in the DCT-domain.

Resumo

Na distribuição e fornecimento de serviços de vídeo é frequentemente necessário proceder à manipulação das tramas de vídeo comprimidas. Consequentemente, a transcodificação de vídeo tem vindo a emergir como uma nova área de investigação, abarcando um conjunto vasto de técnicas de processamento, manipulação e adaptação, para converter uma trama de vídeo numa outra, com uma gama de parâmetros ajustada a uma dada aplicação. O objectivo principal da investigação realizada foi o desenvolvimento de algoritmos e arquitecturas de transcodificação eficientes e flexíveis, para realizar a composição de vídeo no domínio da DCT. Focaram-se, principalmente, dois tipos de aplicações: composição de vídeo estática, em que informação fixa e visível é sobreposta à sequência de vídeo recebida; e composição de vídeo dinâmica, consistindo na composição de uma ou mais sequências de vídeo sobre a área correspondente à sequência de fundo. Para permitir a implementação das estruturas de transcodificação propostas, várias operações típicas de processamento de vídeo tiveram de ser adaptadas e transpostas para o domínio da DCT, tais como: o escalamento espacial de vídeo por um factor inteiro arbitrário, e a estimação de movimento usando os blocos com coeficientes DCT obtidos a partir da trama de vídeo. Quando comparados com implementações correspondentes no domínio do píxel, os transcodificadores propostos no domínio da DCT oferecem vantagens significativas em termos de qualidade de vídeo, da eficiência da codificação e do custo computacional.

Palavras Chave

- Transformada de Coseno Discreta;
- Transcodificação de Vídeo;
- Inserção de Objectos de Vídeo;
- Composição de Vídeo no Domínio da Transformada DCT;
- Estimação de Movimento no Domínio da Transformada DCT;
- Escalamento Espacial no Domínio da Transformada DCT.

Acknowledgments

Along all these years, a significant number of people have directly or indirectly contributed, in several different ways, to the successful completion of this research work. I would like to take this opportunity to thank each of them.

First and foremost, I would like to express my deepest gratitude to my adviser, Prof. Leonel Sousa, for giving me the opportunity and confidence to pursue this work. It wouldn't have been possible to finish this dissertation without his reviewing and thorough critical comments, targeting its permanent improvement and refinement. But he was certainly more than an adviser to me. I must thank him for his kind friendship, infinite patience, invaluable suggestions and advises, as well as for the constant motivation that he has provided me in order to drive this dissertation until this point. His guidance, generous support, constant assistance and encouragement have definitely helped me to overcome tough times. He has always shown a full respect for my individual choices and tried to guarantee the best working environment.

I would like also to extend my appreciation to my colleagues Tiago Dias and Pedro Tomás, for their invaluable help in the revision of this thesis, as well as for the constructive and enlightened discussions that provided me valuable feedback to improve this dissertation.

I would like to thank all my colleagues at the Signal Processing Systems group (SiPS) for their friendship, with a particular appreciation to Oliver Sinnen, Gonçalo Tavares, José Germano, Ricardo Chaves and Nuno Sebastião, with whom I have shared unforgettable moments and who truly contributed to provide a really nice and friendly working environment. I could not forget to express a special appreciation to Prof. Moisés Piedade, for his permanent friendship and encouragement.

I must also extend my gratitude to the following individuals and institutions, from which I have received important support and that significantly helped me to accomplish this research:

- To Instituto de Engenharia de Sistemas e Computadores: Investigação e Desenvolvimento em Lisboa (INESC-ID), for supporting me with the necessary working conditions along all these years, with a particular appreciation for

providing me the possibility to use the GRID computational platform that is available at its facilities and that significantly helped me in the experimental procedures that were conducted along this research;

- To all my colleagues at the Department of Computer Science and Engineering (DEI) of Instituto Superior Técnico (IST), with a special thank to those of the Architectures and Operating Systems group (ASO), for their support and permanent encouragement;

- To the Portuguese Foundation for Science and Technology, for the financial support provided by the Ph.D. grant that was given to me during the first years of this research.

I would like also to express a special thank to all my family and, in particular, to my parents and my brother, for their encouragement and support along all these years.

And last, but certainly not least, I would like to express a special gratitude to my wife, Cristina, not only for showing me that life is much more than work, but particularly for all the love, dedication, support and encouragement that she gave me during all these years and for allowing me to insistently exploit her generous patience. Finally, it is now time to start that long list of things that we have postponed until *"after the thesis is finished"*.

Nuno Roma
Lisbon, May 2008

Contents

List of Figures

List of Tables

List of Acronyms

1-D	one-dimensional
2-D	two-dimensional
ADVS	Activity Dominant Vector Selection
AWA	Area Weighted Average
CDRS	Constrained Dynamic Rate Shaping
CIF	Common Intermediate Format
CPAT	Cascaded Pixel Averaging Transcoder
DCT	Discrete Cosine Transform
DDT	DCT Decimation Transcoder
DFT	Discrete Fourier Transform
DST	Discrete Sine Transform
DTT	Discrete Trigonometric Transform
DWT	Discrete Wavelet Transform
FDVS	Forward Dominant Vector Selection
FFT	Fast Fourier Transform
FPS	Frames per Second
FSBM	Full-Search Block-Matching
FT	Fourier Transform
GDFT	Generalized Discrete Fourier Transform

GDRS	Unconstrained Dynamic Rate Shaping
GOP	Group of Pictures
GPU	Graphics Processing Unit
HA	Half-Sample Anti-symmetry
HDT	Hybrid Downscaling Transcoder
HS	Half-Sample Symmetry
HT	Haar Transform
IDCT	Inverse Discrete Cosine Transform
IEC	International Electrotechnical Commission
IntDCT	Integer Discrete Cosine Transform
INTER	Interframe
INTRA	Intraframe
ISO	International Organization for Standardization
ITU	International Telecommunications Union
JVT	Joint Video Team
KLT	Karhunen-Loève Transform
LOG	Logarithmic Search
LPOS	Left Point of Symmetry
LSE	Least Squares Estimation
LSME	Least Squares Motion Estimation
MB	Macroblock
MC-DCT	Transform-Domain Motion Compensation
MC	Motion Compensation
MDN	Median

ME-DCT	Transform-Domain Motion Estimation
ME	Motion Estimation
MITD	Modified Inverse Transformation and Decimation
MPEG	Moving Pictures Experts Group
MQBA	Maximum Quantization step-size times number of Bits-Area
MSE	Mean Squared Error
MVCS	Motion Vector Compositing and Scaling
MV	Motion Vector
NPR	Non-Peak-to-Peak Ratio
NRSO	Non-Regular Shaped Object
NZ	number of Non-Zero quantized DCT coefficients
PAP	Picture-And-Picture
PDA	Personal Digital Assistant
PD-FSBM	Pixel-Domain Full-Search Block-Matching
PDIT-MV	Pixel-Domain Insertion Transcoder with MV Re-Estimation
PDIT-nMV	Pixel-Domain Insertion Transcoder without MV Re-Estimation
PDVCT	Pixel-Domain Video Compositing Transcoder
PIPCT	Picture-In-Picture Cascaded Transcoder
PIP	Picture-In-Picture
POP	Picture-Over-Picture
POS	Point of Symmetry
PRET	Partial Re-Encoding Transcoder
PSNR	Peak Signal-to-Noise Ratio
QB	Quantization step-size times number of Bits

QCIF	Quarter Common Intermediate Format
RDRE	Rate-Distortion Re-Encoding
RPOS	Right Point of Symmetry
SAAWA	Spatial Activity-Area Weighted Average
SAD	Sum of Absolute Differences
SA	Simple Average
SMES	Simple Motion Estimation Scaling
SOB	System-On-Board
SOC	System-On-Chip
SPMD	Single-Program Multiple-Data
SPS	Symmetric-Periodic Sequence
ST	Slant Transform
SUB	Subsampled Search
TDIT-CL	Closed-Loop Transform-Domain Insertion Transcoder
TDIT-FCL	Fast Closed-Loop Transform-Domain Insertion Transcoder
TDIT-OL	Open-Loop Transform-Domain Insertion Transcoder
TD-LSME	Transform-Domain Least Squares Motion Estimation
TDVCT-MRE	Transform-Domain Video Compositing Transcoder with Motion Re-estimation
TDVCT	Transform-Domain Video Compositing Transcoder
TSS	Three Step Search
VLC	Variable Length Coder
VLD	Variable Length Decoder
WA	Whole-Sample Anti-symmetry
WHT	Walsh-Hadamard Transform
WS	Whole-Sample Symmetry

Nomenclature

Matrix related symbols:

$\mathbf{I}_{n\times n}$ - $(n \times n)$ identity matrix;

$\mathbf{0}_{n\times n}$ - $(n \times n)$ null matrix;

$\mathbf{A}(l, :)$ - l^{th} line of matrix $\mathbf{A}$;

$\mathbf{A}(:, c)$ - c^{th} column of matrix $\mathbf{A}$;

$\mathbf{A}(l_x : l_y, :)$ - submatrix of $\mathbf{A}$, composed by the range of lines between l_x and l_y;

$\mathbf{A}(:, c_x : c_y)$ - submatrix of $\mathbf{A}$, composed by the range of columns between c_x and c_y;

Matrix related operations:

$\mathbf{A}^{\text{T}}$ - matrix transpose;

$\cdot$ - matrix multiplication[†];

$\odot$ - element-by-element matrix multiplication;

$\circledast$ - symmetric convolution;

Ⓢ - skew-circular convolution;

Set related symbols:

$\emptyset$ - empty set;

$\mathbb{N}$ - set of all natural numbers;

$\mathbb{Z}$ - set of all integer numbers;

[†]This symbol is suppressed in many equations.

$\mathbb{Q}$ - set of all rational numbers;

$\mathbb{R}$ - set of all real numbers;

Set related operations:

$\#\{\mathcal{A}\}$ - number of elements of a given set $\mathcal{A}$;

$\max\{\mathcal{A}\}$ - maximum value of a given set $\mathcal{A}$;

$\min\{\mathcal{A}\}$ - minimum value of a given set $\mathcal{A}$;

Transform related symbols:

$\mathcal{C}_{ne}$ - kernel matrix of *even* type-n discrete cosine transform (orthogonal formulation);

$\mathcal{C}_{no}$ - kernel matrix of *odd* type-n discrete cosine transform (convolution formulation);

$\mathcal{S}_{ne}$ - kernel matrix of *even* type-n discrete sine transform (orthogonal formulation);

$\mathcal{S}_{no}$ - kernel matrix of *odd* type-n discrete sine transform (convolution formulation);

$\mathbf{T}$ - Even type-II DCT kernel matrix ($\mathcal{C}_{2e}$);

Miscellaneous functions:

$\lfloor x \rfloor$ - greatest integer less or equal to x;

$\lceil x \rceil$ - smallest integer greater or equal to x;

$\mathrm{mod}_N(x)$ - integer division remainder of x by N;

$\delta(n)$ - discrete impulse function;

$\mathcal{O}(f(x))$ - class of algorithms that grow no faster than $f(x)$ (worst case complexity);

$\mathcal{M}$ - number of arithmetic operations;

Video processing related symbols:

$\mathbf{x}$ - block of pixel values;

$\mathbf{X}$ - block of DCT coefficient values;

$\mathcal{S}_{\mathcal{F}}$ - scale factor;

v - motion vector.

1

Introduction

Contents

1.1 Motivation

In the last few years there has been a general proliferation of advanced video services and multimedia applications. However, despite the fast technological growth in the semiconductor industry, efficient compression algorithms and technologies had to be devised in order to bridge the gap between the huge amount of data that is required to represent video scenes and the strict limitations related to the transmission, the storage and the processing capabilities. As a consequence, several video standards have been developed along the past few decades, to store and broadcast video information in the digital form. Some of such video standards are the MPEG-1 Video [26], the MPEG-2 Video [27] and the MPEG-4 Visual [28], proposed by the Moving Pictures Experts Group (MPEG) of the International Organization for Standardization (ISO) and by the International Electrotechnical Commission (IEC); the H.261 [29] and the H.263 [30], proposed by the International Telecommunications Union (ITU); and the H.264/AVC [31], also known as MPEG-4 Part 10 or MPEG-4 AVC, recently developed by the Joint Video Team (JVT) of MPEG and ITU. A similar partnership between MPEG and ITU had been previously responsible for the definition of the MPEG-2 Video standard. In fig. 1.1 it is represented the evolution of these standards along the time.

Most of these standards make use of block-based video coding techniques that exploit the energy compaction properties of the Discrete Cosine Transform (DCT), allied with quantization schemes, to reduce the amount of irrelevant information that is present in each frame. Moreover, temporal prediction mechanisms based on motion estimation and compensation procedures are usually applied, to reduce the amount of redundant information in consecutive frames. By applying such compression techniques, it is possible to obtain a substantial decrease of the amount

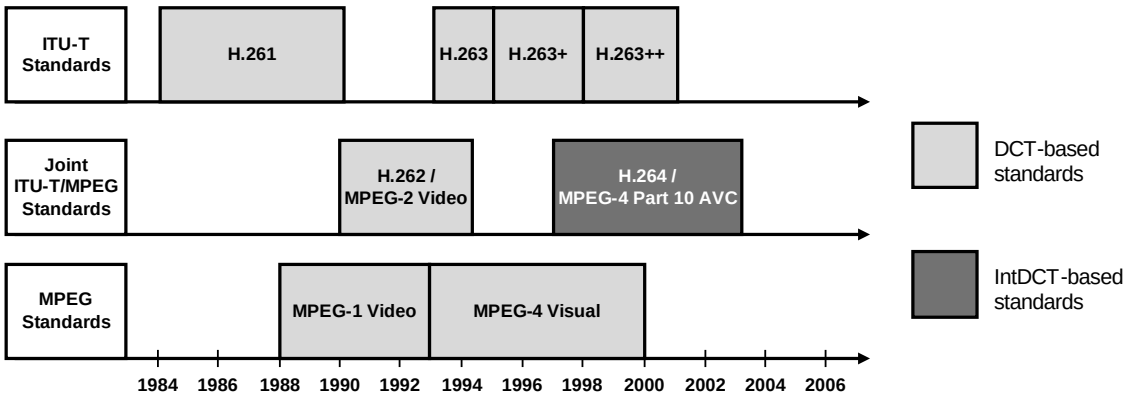


Figure 1.1: Evolution of the main video standards along the past few years.

of data required to store or transmit a given video sequence. As a consequence, many successful applications have arisen in the areas of digital television, digital storage, video streaming over the network, surveillance, mobile and portable video transmission, and many others.

However, once video signals have been compressed, delivery systems and service providers frequently face the need for further manipulations and processing of the compressed video-streams. Such data processing usually focuses on adapting the characteristics of the video sequences not only to the terminal devices but also to the available channel bandwidth or to the characteristics of the storage media. In this scope, a video transcoder is defined as a device that converts one or more video streams into other video streams that possess a more desirable set of characteristics [107].

As a consequence, video transcoding has emerged as a new research area concerning a broad set of processing, manipulation and adaptation techniques to convert one video bit stream into another bit stream with a more convenient set of parameters targeted to a given application. This parameter adaptation may include changes on: syntax, spatial and temporal resolutions, bit rate adjustment, added functionalities or even other adjustments imposed by hardware requirements. Moreover, with the advent of a broad range of different portable and battery supplied terminal equipments, including handheld computers, Personal Digital Assistants (PDAs), set-top boxes and smart cellular phones, new requirements and demands have recently arisen. With the significant set of different characteristics that are presented by these devices in what concerns the available computational resources, display capability and power consumption, it became increasingly important to ensure the feasibility of such video processing algorithms in a wide range of different embedded computational systems.

Meanwhile, the developments on video transcoding that have been published in the last few years have shown that significant advantages can be obtained by fully processing the video streams in the compressed DCT-domain [6]. Some of such advantages are enumerated as follows:

- *lower computational cost*, since it avoids the implementation of both the forward transform and its inverse;
- *smaller data volume*, since it takes advantage of the presence of a large number of null quantized DCT coefficients, which heavily reduce the data manipulation rate [57, 99];
- *increased image quality*, resulting from the absence of degradation effects that

arise during the computation of the direct and of the inverse DCT, which usually emerge from round-off errors that are introduced by the usage of finite precision arithmetic; this quality loss becomes even more serious in embedded systems with limitative hardware for fixed-point arithmetic and operating with a restricted number of bits.

As a consequence, there has been a growing interest in the research and industrial communities to develop processing techniques and to implement many of the most required video parameter adjustment and processing algorithms either directly in the compressed-domain or in an intermediate domain representation, where the processing is directly applied to the DCT coefficients. In the following, such intermediate processing domain will be simply denoted by DCT-domain [107].

Due to the complexity of the inherent procedures and to the significant number of coding mechanisms that must be adapted, one of the most challenging processing operations is related to the DCT-domain composition of multiple precoded video objects, in order to obtain one single video sequence [11]. Contrary to what happens with some video standards (such as MPEG-4 Visual [28]) where multiple video objects can be independently encoded and transmitted, the objective of these composition operations is to obtain a single video plane composited by the several input video objects. In this scope, it will be generically denoted by *video object* any image or video entity that can be characterized by a given shape and texture data structure, as well as by a precise size and target position within the composited plane.

As an example, with the widespread dissemination and usage of video data over the several communication means that have become increasingly available, intellectual property management and protection issues become one of the most common video processing operations that is carried out over broadcasted video sequences. Such operations usually comprehend a set of stationary or *quasi*-stationary manipulations on precoded video streams, in order to insert extra visual information in the video sequence that is delivered to the receiver. Among the several possible manipulations, the algorithms for the insertion of visible logos, subtitles, fixed images or graphical symbols in video sequences represent the most commonly used. Such objects usually occupy a small area of the encoded frame and tend to be static, from frame to frame. As a consequence, the insertion of visible objects in the compressed-domain has faced a growing interest by broadcasting television networks as well as by digital video producers and distributors, in order to provide the possibility of inserting their own logos and subtitles in pre-encoded video streams [61]. To take into account for the several DCT-based video standards currently in use, this mech-

anism should provide the insertion capability in several block-based video standards, such as the simplest MPEG-1 and MPEG-2 and other real-time low bit rate services based on H.261 and H.263 video standards [29, 30, 64, 74].

Another video processing operation that has become increasingly popular along the past few years is related to the composition of multiple video sequences into a single scene, according to any arbitrary composition layout. One of such layouts that is particularly adopted in surveillance and video-conferencing applications is the *picture-in-picture* setup, where one or more foreground video sequences are continuously displayed, at a reduced dimension, over a given region of the background displaying area. However, the implementation of these dynamic video compositing schemes in the compressed DCT-domain poses some additional highly complex challenges. Contrary to what happens with the static video compositing operations, it requires the simultaneous processing and combination of the several and possibly different encoding strategies that are adopted by the video sequences under processing.

An example of one of the most challenging operations that is often required is the adaptation of the temporal prediction schemes of the input video sequences, in order to provide an efficient encoding of the output composited video stream. In fact, many video transcoding operations require the refinement or even the computation of entirely new Motion Vectors (MVs), in order to reduce the magnitude of the prediction error signal and the drift that is introduced along the time by such techniques. Consequently, the implementation of motion estimation and compensation algorithms by directly using the DCT coefficients obtained from the received video sequences is one of the processing techniques that has been given more attention by the research community.

Another important challenge is related to the efficiency and reliability of the scaling procedures that are required to adapt the spatial resolutions of the involved foreground video sequences. Although such issue has been considered by many researchers in the last few years, the characteristics of the proposed algorithms do not always comply with the requisites of the adopted space-scaling transcoding system: either they are only directly applied to scaling operations using integer power of 2 scaling factors, or their performance significantly degrades when other arbitrary scaling factors are applied. Moreover, some of these algorithms do not allow the direct processing of pre-coded data using the fixed block structure that is adopted by most digital video standards.

1.2 Main objectives

In accordance with the identification of open research issues that was described in the previous section, the research work that is presented in this thesis targets the implementation of complex composition schemes of compressed video objects, by directly processing each encoded video sequence in the intermediate DCT-domain representation. Contrasting with other simpler pixel-domain approaches - composed of a cascading structure of one or more complete decoding stages, a dedicated pixel-domain composition module and a final video encoding structure - the adopted compressed-domain approach takes advantage of many of the encoding parameters and statistics that are available in the input compressed video streams. As it will be shown in the presentation, such significant amount of information can be used not only to simplify the overall computation, but also to improve the video quality and coding efficiency.

The main research efforts that were conducted in the scope of this thesis were focused on the design of efficient algorithms and processing architectures to implement video composition transcoding operations in current video processing structures. Along this study, a particular attention was devoted to the inherent trade-off between the following key factors that significantly affect the actual feasibility of these transcoding structures in current video processing systems: *i)* the involved computational cost, *ii)* the output video quality, and *iii)* the resulting bit rate. Consequently, many of the algorithms that are proposed provide a complexity scalability feature that confer the designer the possibility to adapt the characteristics of the obtained transcoder not only to the computational capabilities of the target system, but also to the coding quality and bit rate requirements of the intended application.

The following paragraphs outline the main objectives of the conducted research:

- **Development of new transcoding algorithms for video composition**
  One of the prime focuses of the research presented in this thesis is the development of efficient and flexible transcoding algorithms for video composition in the compressed DCT-domain. Two distinct video compositing schemes will be considered:

  - Static video composition: consisting on the insertion of stationary or *quasi*-stationary visible data (such as logos or other non-regular shaped objects, etc.) over the received "background" video sequence;
  - Dynamic video composition: consisting on the composition of one or more

"foreground" video sequences over the displaying area corresponding to the "background" video sequence.

The implementation of any of these transcoding schemes should support a wide flexibility and variability of the insertion parameters, mainly, in what concerns the adopted composition layout, the position of the several video objects, the spacial scaling of the inserted scenes, etc.

- **Development and improvement of DCT-domain video processing operations**
  To support the implementation of the proposed transcoding algorithms for video composition, several common video processing operations will have to be adapted and transposed into the DCT-domain. Despite the several algorithms that have already been proposed in the literature by other authors, some significant changes and improvements have to be carried out. These improvements are important to optimize not only the processing of the video objects, by directly using the DCT coefficients of the decoded video objects, but also to better adapt and suit their implementations to the data structures adopted by the considered video standards. Some examples of research topics that have to be tackled are:
  - DCT-domain video space-scaling: by considering an arbitrary integer scaling factor and a proper adaption to the block-based data structures of the video standards;
  - DCT-domain motion estimation: by directly using the decoded blocks of DCT-coefficients obtained from the received video sequences.
- **Development of transcoding architectures for video composition**
  The efficient concretization of the several transcoding algorithms that will be proposed along this thesis can only be achieved through a careful design and implementation procedure. In particular, a tight relation should exist between the transcoding algorithms development stage and their corresponding architectures design phase. Such relation allows to cope with the several restrictions that are imposed not only by the data structures associated to the considered video standards, but also with the computational and memory access requirements and characteristics of the target implementation platforms.
- **Optimization of the considered transcoding algorithms**
  One of the main premises that will serve as the basis for the development

of the algorithms and architectures that are proposed in this thesis is the possibility to properly adapt the computational cost required by the several involved operations with both the bit rate and the quality requirements of the intended transcoding system. In order to attain such objective, a flexible scaling of the overall complexity level that is involved in each operation, as well as an accurate assessment of the resulting influence on both the obtained video quality and bit rate, should be carefully analyzed. By adopting such approach, it will be possible to confer the transcoder designer the possibility to trade-off the involved computational cost with the resulting output video quality performance measures.

From the stated objectives, it should be clear that although some basic theoretical concepts and principles of digital video coding, as well as of the associated standards, will have to be extensively used, the main focus of this thesis will be on digital video transcoding techniques, algorithms and architectures, and on the more practical aspects related to their implementation in currently available computational systems.

1.3 Summary of original contributions

In this section, it is presented a summary of the most relevant and original contributions that have resulted from the research work performed in the scope of this thesis. Such achievements are mainly related to the proposal of innovative or improved transcoding algorithms and architectures to provide video manipulation schemes in the compressed DCT-domain.

Along the comprehensive study that was performed during the development stage of such algorithms and architectures, a constant attention was devoted to the validation of the proposed techniques within the scientific community. As a consequence, many of the most relevant contributions have already been published either in international scientific journals or in the proceedings of several international conferences.

The most relevant contributions are listed as follows:

- **Fast Transcoding Architectures to Insert Non-Regular Shaped Objects in the Compressed DCT-Domain**
  From the developed research work, it was proposed a set of fast transcoding architectures to insert non-regular shaped objects, such as visible logos or subtitles, in compressed video signals. The proposed transcoders incorporate a logo insertion module on their structure that manipulates the DCT coefficient

blocks directly obtained from the decoded video streams. To avoid the presence of undesired semi-transparent rectangular regions around the inserted objects, the proposed technique makes use of the multiplication-convolution property of the DCT. Such transcoding structures have proved to provide significant advantages, both in terms of the subjective video quality and computational cost. Moreover, the distinctive features presented by the proposed architectures provide the means to adapt the insertion scheme to the particular requirements of the target application.
Most of the related main contributions have already been published in:

[86] N. Roma and L. Sousa, "Insertion of irregular-shaped logos in the compressed DCT domain," in *Proceedings of the IEEE International Conference on Digital Signal Processing (DSP)*, vol. 1. Santorini, Greece: IEEE, Jul. 2002, pp. 125–128.

[87] N. Roma and L. Sousa, "Transcoding architectures for object insertion in compressed video," INESC-ID – Lisboa, Portugal, Tech. Rep. RT/006/2002, Oct. 2002.

[88] N. Roma and L. Sousa, "Fast transcoding architectures for insertion of non-regular shaped objects in the compressed DCT-domain," *Signal Processing: Image Communication*, vol. 18, no. 8, pp. 659–683, Sep. 2003.

- **Arbitrary Space Scaling Algorithm in the Compressed DCT-Domain**
  It was proposed an efficient video downscaling algorithm for any arbitrary integer scaling factor. This algorithm receives the encoded DCT coefficient blocks of the input video sequence and efficiently computes the DCT coefficients of the scaled video stream. The involved steps are properly tailored, so that all operations are performed using the video standard block structure, independently of the adopted scaling factor. As a result, the proposed algorithm offers a significant optimization of the computational cost without compromising the output video quality, by taking into account the scaling mechanism and by restricting the involved operations to avoid useless computations. In order to meet any system needs, an optional and possible combination of the presented algorithm with high order AC frequency DCT coefficients discarding techniques was also proposed. Such combination provides a flexible complexity scalability feature and gives rise to an adaptable trade-off between the involved scalable computational cost and the resulting video quality and bit rate.

Most of the related main contributions have already been published in:

[90] N. Roma and L. Sousa, "Efficient hybrid DCT-domain algorithm for any arbitrary integer re-size video downscaling," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, no. 57291, pp. 1–16, Sep. 2007.

- **Block-Based Motion Re-Estimation Algorithm in the Compressed DCT-Domain**

  The research work that was performed on DCT-domain spatial downscaling algorithms and on dynamic composition of multiple video sequences required the development and the proposal of a new compressed-domain motion estimation algorithm. This algorithm directly processes the DCT coefficients obtained from the decoded video streams and is based on an iterative scheme that computes the new MVs by applying a least squares estimation technique. To reduce the computational effort, the proposed algorithm may also only consider an arbitrary subset of non-null DCT coefficients. The resulting MVs provide the means to significantly enhance the quality of the temporal prediction mechanism on the processed video sequences, with a consequent reduction of the required bit rate.

  Most of the related main contributions have already been published in:

  [89] N. Roma and L. Sousa, "Least squares motion estimation algorithm in the compressed DCT domain for H.26x/MPEG-x video sequences," in *Proceedings of the IEEE International Conference on Advanced Video and Signal-Based Surveillance (AVSS)*. Como - Italy: IEEE, Sep. 2005, pp. 576–581.

- **Dynamic Video Composition Algorithm in the Compressed DCT-Domain**

  It was proposed an efficient architecture to perform the composition of encoded video sequences that fully operates in the compressed DCT-domain. By directly operating with the partially decoded DCT coefficients of the involved video sequences, the proposed approach may provide significant advantages in what concerns the output video quality performance. Such advantages are owed to the absence of both the Inverse Discrete Cosine Transform (IDCT) and DCT processing blocks that naturally make it less prone to round-off

and fixed-precision arithmetic errors. Moreover, the presented approach significantly improves the temporal prediction mechanism, by incorporating the developed DCT-domain motion re-estimation algorithm. Contrary to the refinement schemes proposed by other authors, this refinement procedure may consider any dimension for the search area, which significantly improves the output video quality and reduces the resulting bit rate. Furthermore, the presented DCT-domain approach does not impose any limitation on the composition setup, allowing each "foreground" video sequence to be placed over any location of the "background" video scene. Therefore, it also offers a flexible and easy implementation of most current video composition setups, such as Picture-In-Picture (PIP), Picture-And-Picture (PAP) and Picture-Over-Picture (POP).
Most of the related main contributions have already been published in:

[91] N. Roma and L. Sousa, "Fully compressed-domain transcoder for PIP/PAP video composition," in *Proceedings of the Picture Coding Symposium (PCS)*, Lisbon - Portugal, Nov. 2007, pp. CD–ROM.

1.4 Computational framework

To support the development of the several algorithms and architectures that were proposed in the scope of the research presented in this thesis, a complete and dedicated computational framework was implemented. This computational framework was also used to evaluate and assess all the presented transcoding algorithms and architectures. In the following, it will be presented a brief description of the main aspects related to such framework.

Video coding platform

Without any generalization or applicability loss of the several transcoding algorithms and architectures that are presented in this thesis, the implemented framework was based on a generic block-based video coding structure, incorporating most of the processing modules that integrate many current DCT-based video encoders. In accordance, such coding structure makes use of the DCT and of a motion-compensation prediction mechanism to reduce the amount of irrelevant and redundant information within the processed video data. The frames of the processed video sequences are divided into Macroblocks (MBs), where each MB consists of four luminance blocks (Y), along with the corresponding sub-sampled chrominance

blocks (C_b) and (C_r) [7]. By taking these general assumptions into account, any of the proposed transcoding structures can be easily implemented and integrated within the transcoding architecture of any other compatible video standard. Nevertheless, to support the evaluation and the assessment procedures of the presented transcoding schemes, the above referred transcoding structure made use of the same syntax and semantic rules of the H.263 [30] video standard. The selection of this specific standard arises from the simple encoding mechanism that characterizes it, thus greatly simplifying the development and integration of new processing modules. In addition, such approach does not compromise the general applicability of the proposed techniques to other DCT-based video standards, such as the H.261 [29], the MPEG-1 Video [26], the MPEG-2 Video [27] and the MPEG-4 Visual [28].

Software implementation

The conception of this computational framework comprised the full implementation of the used video encoding and decoding systems, as well as of all the processing modules that are required to implement the proposed video transcoding architectures. All these computational blocks were developed using the GNU Octave [69] high-level language, primarily intended for numerical and matrix operations, running over a general purpose distribution of the Linux operating system. Such computational platform provides a convenient batch-oriented interface to perform numerical computations, using a language that is mostly compatible with Matlab from Mathworks [60]. The development of complex processing systems, such as the video transcoding schemes that are proposed in this thesis, is somewhat facilitated by its extensible and customizable facility via user-defined functions written in Octave's own language.

Furthermore, contrasting with other general purpose programming languages (such as ANSI C [37] or C++), that could potentially provide better computational performances, the programming language adopted in the implementation of this framework comprises a comprehensive library of mathematical functions that has proved to be special adequate to allow a fast and easy prototyping platform, capable of implementing all the processing structures required by the proposed transcoders.

Computational resources

To accelerate the execution of the proposed transcoding structures, the GRID [18] computational platform that is available at INESC-ID facilities was extensively used to run several parallel instances of the developed software transcoder

implementations. The GRID infrastructure, depicted in fig. 1.2, is composed of 22 computational nodes, each one equipped with a 3.2 MHz Intel Pentium 4 processor with 1 GB of memory RAM. The several nodes that compose this computational pool are managed by the Globus software toolkit [19, 21] using the Condor scheduler [15]. The combination of these two middleware toolkits provides a complete and efficient platform to control the job queuing mechanism, the adopted scheduling policy and priority schemes, and a complete set of resources to monitor and manage the several computational nodes.

The parallelization of the implemented transcoding structures was carried out by using a simple Single-Program Multiple-Data (SPMD) parallel computational model, by executing the same transcoding software implementation in all nodes of the GRID infrastructure. The input streams, corresponding to the encoded video sequences that have to be processed by the transcoding system, were sliced into multiple pieces of data. Each of these slices was then assigned to a distinct job that was submitted to the queuing mechanism.

To comply the required slicing procedure of the input video sequences with the data dependencies that are inherent to the video coding syntax and semantic rules, each of these slices comprehends an integer multiple of closed Group of

Figure 1.2: Grid infrastructure available at INESC-ID.

Pictures (GOP) units. By adopting such approach, it was removed any INTER frame dependency between the data sets that were submitted to the several different jobs. The processed output video sequence was then obtained by implementing an ordered concatenation mechanism of the output video streams produced by the several processed jobs.

By using such a parallel computational platform, it was possible to obtain a faster and reliable prototyping system to implement all the proposed transcoding architectures. In fact, without such platform, some of the conducted evaluation and assessment experimental procedures would have taken several days to be obtained.

1.5 Organization of the thesis

This thesis is organized in seven chapters and two appendices.

In the present chapter it is presented the motivation and an introductory approach to the recent advent of transcoding techniques in current video processing applications. The main challenges and target objectives of the presented research are also outlined, as well as a brief overview of the main resulting contributions.

In chapter 2 it is presented a set of theoretical foundations concerning the DCT that will be extensively used in the presentation of the several algorithms proposed in this thesis. A formal definition of this transform is provided, as well as the presentation of its most important properties and relations with other discrete trigonometric transforms.

Chapter 3 presents a brief overview of the main state-of-the-art techniques and tools that have been proposed over the last few years to implement many of the most required operations for video transcoding of video bit streams in the DCT-domain.

In chapter 4 it is addressed the problem concerning the insertion of non-regular shaped objects, such as visible logos, in compressed video signals. A different approach from the usual pixel-domain compositing operation is adopted, in order to avoid the presence of undesired semi-transparent rectangular regions around the inserted objects. The transposition of this technique to the compressed DCT-domain is presented, as well as the integration of the logo insertion module in several architectures of compressed-domain video transcoders.

Chapter 5 presents a set of efficient transcoding operations to process and manipulate one or more precoded video sequences in the compressed DCT-domain. In particular, section 5.2 addresses the implementation of an innovative and efficient transcoding algorithm for video downscaling by any arbitrary integer scaling factor in the transform-domain. Such algorithm offers a considerable advantage in what

concerns the computational cost and the obtained video quality, when arbitrary scaling factors are applied. In section 5.3 it is proposed a new compressed-domain motion estimation algorithm. This algorithm is based on an iterative scheme to estimate the new MVs, by using the DCT coefficients directly obtained from the input video stream. These two important algorithms were then applied in the implementation of a flexible and efficient compressed-domain video compositing architecture, presented in section 5.4.

In chapter 6 it is described the several experimental procedures that were conducted in the scope of the presented research, in order to assess the performance and the efficiency of the proposed algorithms and architectures. A thorough discussion of the obtained results is also presented. Such discussion provides a detailed evaluation of the objective and subjective video quality levels, as well as of the involved computational cost and the resulting bit rate, obtained with the proposed static and dynamic video processing algorithms.

Finally, chapter 7 concludes this thesis with a presentation of the main accomplished objectives and the drawing of some possible future research directions.

Besides these seven chapters, this thesis also includes two appendices where it is provided some complementary information concerning the presentation of a motion estimation algorithm mentioned in the description of the current state-of-the-art, and some additional and supplementary results obtained with the proposed architectures.

References

[6] P. Assunção and M. Ghanbari, "A frequency-domain video transcoder for dynamic bit-rate reduction of MPEG-2 bitstreams," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 8, pp. 953–967, Dec. 1998.

[7] V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards: Algorithms and Architectures*, 2nd ed. Kluwer Academic Publishers, Jun. 1997.

[11] S.-F. Chang and D. G. Messerschmitt, "Manipulation and compositing of MC-DCT compressed video," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 1, pp. 1–11, Jan. 1995.

[15] "Condor webpage," http://www.cs.wisc.edu/condor, 2008.

[18] I. Foster, *The Grid: Blueprint for a New Computing Infrastructure*, 2nd ed. Morgan Kaufmann, 2004.

[19] I. Foster, "Globus toolkit version 4: Software for service-oriented systems," in *Proceedings of the IFIP International Conference on Network and Parallel Computing (NPC)*, vol. LNCS 3779. Springer-Verlag, 2006, pp. 2–13.

[21] "Globus webpage," http://www.globus.org, 2008.

[26] *MPEG-1: ISO/IEC JTC1 CD 11172 - "Coding of moving pictures and associated audio for digital storage media up to 1.5 Mbit/s – Part 2: Video"*, ISO, 1992.

[27] *MPEG-2: ISO/IEC JTC1 CD 13818 - "Generic coding of moving pictures and associated audio – Part 2: Video"*, ISO, 1994.

[28] *MPEG-4: ISO/IEC 14496-2:2004. Information technology – Coding of audio-visual objects – Part 2: Visual*, ISO, 2004.

[29] *ITU-T Recommendation H.261 - "Video Codec for Audiovisual Services at p×64 Kbit/s"*, ITU-T, Mar. 1993.

[30] *ITU-T Recommendation H.263 - "Video Coding for Low Bitrate Communication"*, ITU-T, Feb. 1998.

[31] *ITU-T Recommendation H.264, "Advanced Video Coding for Generic Audiovisual Services"*, ITU-T, May 2003.

[37] B. Kernighan and D. Ritchie, *The C Programming Language*, 2nd ed. Prentice Hall Software, 1988.

[57] S. Liu and A. C. Bovik, "Local bandwidth constrained fast inverse motion compensation for DCT-domain video transcoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 5, pp. 309–319, May 2002.

[60] "Matlab webpage," http://www.mathworks.com/products/matlab, 2008.

[61] J. Meng and S.-F. Chang, "Embedding visible video watermarks in the compressed domain," *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, vol. 1, pp. 474–477, 1998.

[64] J. L. Mitchell, W. B. Pennebaker, C. E. Fogg, and D. J. Legall, *MPEG video compression standard*. Chapman & Hall, 1996.

[69] "Octave webpage," http://www.octave.org, 2008.

[74] F. Pereira and T. Ebrahimi, Eds., *The MPEG-4 Book*. Prentice Hall PTR, 2002.

[86] N. Roma and L. Sousa, "Insertion of irregular-shaped logos in the compressed DCT domain," in *Proceedings of the IEEE International Conference on Digital Signal Processing (DSP)*, vol. 1. Santorini, Greece: IEEE, Jul. 2002, pp. 125–128.

[87] N. Roma and L. Sousa, "Transcoding architectures for object insertion in compressed video," INESC-ID – Lisboa, Portugal, Tech. Rep. RT/006/2002, Oct. 2002.

[88] N. Roma and L. Sousa, "Fast transcoding architectures for insertion of non-regular shaped objects in the compressed DCT-domain," *Signal Processing: Image Communication*, vol. 18, no. 8, pp. 659–683, Sep. 2003.

[89] N. Roma and L. Sousa, "Least squares motion estimation algorithm in the compressed DCT domain for H.26x/MPEG-x video sequences," in *Proceedings of the IEEE International Conference on Advanced Video and Signal-Based Surveillance (AVSS)*. Como - Italy: IEEE, Sep. 2005, pp. 576–581.

[90] N. Roma and L. Sousa, "Efficient hybrid DCT-domain algorithm for any arbitrary integer re-size video downscaling," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, no. 57291, pp. 1–16, Sep. 2007.

[91] N. Roma and L. Sousa, "Fully compressed-domain transcoder for PIP/PAP video composition," in *Proceedings of the Picture Coding Symposium (PCS)*, Lisbon - Portugal, Nov. 2007, pp. CD–ROM.

[99] B. Shen and I. K. Sethi, "Block-based manipulations of transformed-compressed images and videos," *ACM Multimedia System Journal*, vol. 6, no. 2, pp. 113–124, Mar. 1998.

[107] H. Sun, X. Chen, and T. Chiang, *Digital Video Transcoding for Transmission and Storage*. CRC Press, 2004.

2

The Discrete Cosine Transform

Contents

2.1 Introduction

Transform-based coding has been extensively used in image and video coding. The adoption of transform functions to encode pixel data relies on the general premise that adjacent pixels exhibit a significant level of spatial correlation. Such correlation can be highly exploited to predict the value of a given pixel from its corresponding neighbors. As a consequence, many digital image and video coding techniques that have been proposed take advantage of these transform functions to map the spatial correlated data into a set of less correlated transform-domain coefficients.

In fig. 2.1 it is depicted the general structure of a typical digital video transmission system. The main objective of the *source encoder* is to exploit the redundancies and the irrelevancies of the pixel data, in order to obtain the highest possible compression level. Such objective is attained by reducing the contents entropy, thus decreasing the average number of bits required to represent each frame. On the other hand, to enhance the reliability of the transmission, the *channel encoder* adds a certain redundancy level to the output of the source encoder.

To maximize the compression, each component of the source encoder exploits a different redundancy level of the pixel data:

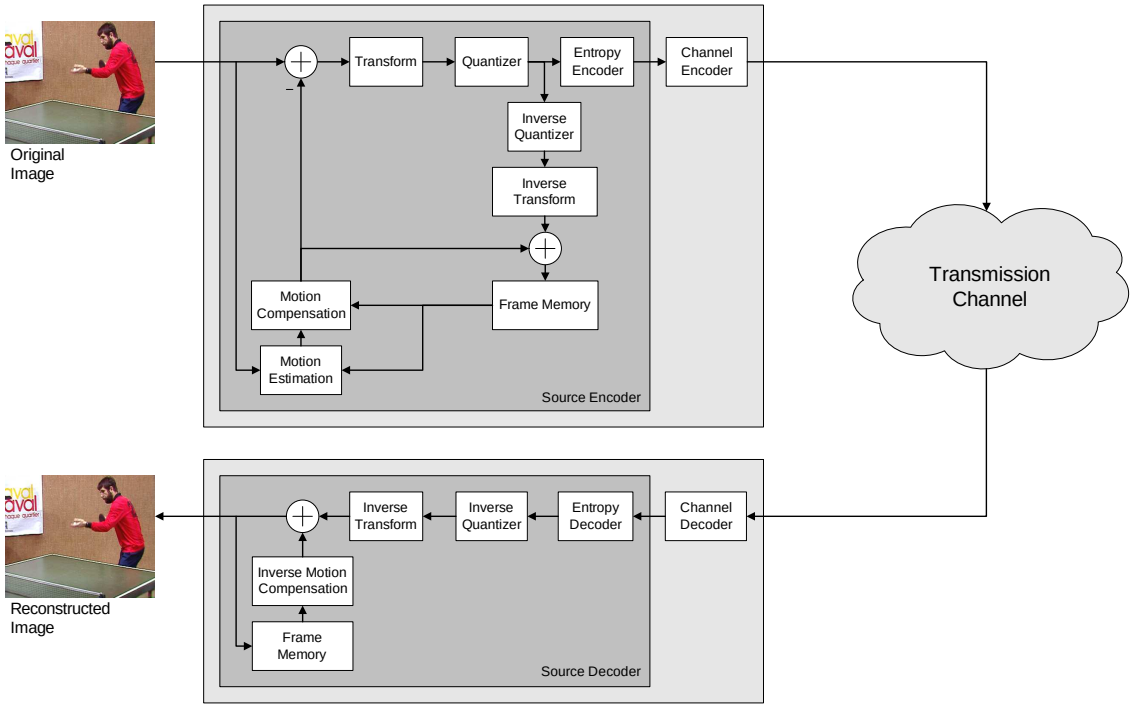


Figure 2.1: Block diagram of a general video transmission system.

- The *motion compensation* module decorrelates the pixel data in order to minimize the temporal redundancy between consecutive frames. Its main objective is to compensate for the displacement of moving objects, from one frame to another, in order to minimize the magnitude of the resulting difference signal. The implementation of this module in the encoder side is tightly coupled with a *motion estimation* module, to compute the MVs that describe the relative motion of a given block of pixels.

- The *transform* module decorrelates the pixel data in order to minimize the spatial redundancy between adjacent pixels. Several different transforms can be adopted in the implementation of this module; a thorough and detailed discussion about the transform functions that are usually used by such module will be provided in the following sections.

- The *quantizer* module takes advantage of the inability of the human eye to perceive small differences between pixel values. As a consequence, such small differences are often considered irrelevant and can actually be discarded without introducing serious visual artifacts. Such irrelevancy is often denoted by psychovisual redundancy. This idea is often extended and exploited in low bit rate transmission systems, characterized by strict bandwidth restrictions, where visual quality has to be sacrificed in order to reduce the bit rate to the available bandwidth. Hence, even though other sources of degradation may also have to be considered (as it will be described in section 3.1.2), this module is usually the main source of irreversible loss in these encoding schemes.

- The *entropy encoder* implements a final lossless compression stage in the encoding scheme, by using, for example, a Huffman run-length encoder [26, 27, 29, 30]. The main purpose of this module is to represent each symbol of the quantizer output with the minimum amount of bits as possible. In the following, this encoding block will be generally denoted by Variable Length Coder (VLC), while the corresponding decoder block will be denoted by Variable Length Decoder (VLD).

In video coding terminology, a compressed frame that only uses spatial redundancy reduction techniques is usually referred to as an Intraframe (INTRA), and is often simply denoted by I-frame. On the other hand, a compressed frame that uses both spatial and temporal redundancy reduction techniques is referred to as an Interframe (INTER) and is denoted by P-frame.

Most transform-based coding techniques process the pixel data by grouping the pixels in block structures. These blocks are then transformed and mapped into another domain, such as the frequency-domain. By defining two $(N \times N)$ transformation matrices:

$$\boldsymbol{\mathcal{T}}_c = \{\mathcal{T}_c(m,i)\} \ , \ m,i = 0,1,\ldots,N-1, \tag{2.1}$$

$$\boldsymbol{\mathcal{T}}_r = \{\mathcal{T}_r(n,j)\} \ , \ n,j = 0,1,\ldots,N-1, \tag{2.2}$$

and by considering a two-dimensional (2-D) image block in the spatial-domain $\mathbf{x}$, the corresponding transform-domain representation $\mathbf{X}$ is obtained by applying the $(N \times N)$ linear transformation process:

$$\mathbf{X} = \boldsymbol{\mathcal{T}}_c \cdot \mathbf{x} \cdot \boldsymbol{\mathcal{T}}_r{}^{\mathrm{T}}. \tag{2.3}$$

The $\boldsymbol{\mathcal{T}}_c$ and $\boldsymbol{\mathcal{T}}_r$ matrices are usually referred to as transformation kernels or basis functions.

As it was previously referred, the main motivation for applying this exchange of the signal domain and compute the transform representation $\mathbf{X}$ of the pixel-domain signal $\mathbf{x}$ is to obtain a more compact representation of the pixel-data. Nevertheless, such transformation process has to be reversible, so that $\mathbf{x}$ can be reconstructed from $\mathbf{X}$ in the decoder side of the video transmission system.

Some of the most commonly used transforms that have been considered for image and video coding are the Karhunen-Loève Transform (KLT), the Discrete Fourier Transform (DFT), the Discrete Cosine Transform (DCT), the Discrete Sine Transform (DST), and the Discrete Wavelet Transform (DWT) [34, 53, 78, 82].

The KLT is the most efficient, in terms of compaction efficiency. The basis functions of this transform are obtained from the statistical properties of the pixel data. As a consequence, it gives rise to the optimum performance in terms of energy compaction, thus placing as much energy as possible in as few coefficients as possible. However, since the transform kernel of this transform depends on the image data under processing, it cannot be computed using a fast matrix-multiplication form, based on a separable pre-computed matrix kernel. It also requires a continuous transmission of the transform kernel coefficients to the decoder end, thus implying a subsequent increase of the required bandwidth. Moreover, for block-based coding, the derivation of the basis kernel corresponding to each image block introduces an extra computational effort, that is often not compatible with most current image and video coding applications.

As a consequence, other less efficient but image independent transforms have been preferred. Among them, the DFT is characterized by a linear, separable and

symmetric definition. Contrary to the KLT, it is represented by fixed basis functions and exhibits good decorrelation and energy compaction characteristics. However, the output of the DFT is defined in the complex-domain and therefore gives rise to both magnitude and phase components to be encoded. Furthermore, the implicit periodicity of DFT introduces boundary discontinuities that result in a significant high-frequency content.

As a result, discrete transforms characterized by smoother basis functions have been preferred. In particular, the transform output provided by the DCT usually leads to compaction efficiency levels quite close to the optimum performance provided by the KLT. Consequently, the DCT has been widely adopted in many digital image and video standards, such as the JPEG [32], H.261 [29], H.263 [30], MPEG-1 Video [26], MPEG-2 Video [27] and MPEG-4 Visual [28] (see fig. 1.1 on page 2).

Independently of the adopted standard, the computation of the transform coefficients usually implies the usage of floating point precision accuracy. Nevertheless, for practical implementations, the floating-point DCT and IDCT are usually implemented with finite precision. This often leads to an accuracy mismatch in the computation of these transforms at both the encoder and decoder ends of the video transmission system. These errors in the IDCT mismatch will accumulate and result in a non-negligible distortion component with visible artifacts, that can only be removed by periodically inserting INTRA-coded blocks, to stop this accumulation.

To circumvent this mismatch problem, recent video standards have adopted alternative transforms. These transforms can be accurately implemented with reduced precision, at the cost of a slight decrease of the provided decorrelation performance. Some examples of these transforms are the Walsh-Hadamard Transform (WHT) [79], the Slant Transform (ST) [80] and the Haar Transform (HT) [22]. Another example of such transforms is the Integer Discrete Cosine Transform (IntDCT) [8], recently adopted by the H.264/AVC video standard [31] (see fig. 1.1 on page 2). It is defined as:

$$\mathbf{X}_{\mathrm{I}} = \mathbf{T}_{\mathrm{I}} \cdot \mathbf{x} \cdot {\mathbf{T}_{\mathrm{I}}}^{\mathrm{T}}, \tag{2.4}$$

where the transform kernel matrix ($\mathbf{T}_{\mathrm{I}}$) is:

$$\mathbf{T}_{\mathrm{I}} = \begin{pmatrix} 1/2 \\ 1/\sqrt{10} \\ 1/2 \\ 1/\sqrt{10} \end{pmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}. \tag{2.5}$$

Since the scaling factors associated with this kernel can be absorbed in the quantization process, all arithmetic operations of this transform can be accurately computed

with 16-bit integers and using solely additions and shift operations; there is no need to perform any multiplication.

However, despite the orthogonal nature and the computational simplicity that is offered by these transforms, many important mathematical relations already presented for the DCT have not yet been extended to such transforms. One such relation is the *multiplication-convolution property*, whose definition, in the DCT-domain, will be presented in section 2.5.2. Such limitations will restrict the application of some transcoding operations that will be presented in the following chapters to DCT-based video encoding systems, namely, those based on the H.261 [29], H.263 [30], MPEG-1 Video [26], MPEG-2 Video [27] and MPEG-4 Visual [28] standards. As a consequence, the general designation of *DCT-H.26x/MPEG-x* that henceforward will be extensively adopted, will only accommodate this broad family of DCT-based video standards and excludes those based on integer transforms, such as the H.26L [33] and H.264/AVC [31].

In the following sections it is presented a brief description of the discrete cosine transform family, as well as its definition and main properties. From now on, the subscripts i and j will denote coordinates in the spatial (pixel) domain, whereas the subscripts m and n will denote coordinates in the transform-domain. Likewise, lowercase symbols will denote pixel-domain signal values, whereas uppercase symbols will denote transform-domain values. E.g.: $X(m,n) = \mathrm{DCT}(x(i,j))$.

2.2 Definition

Similarly to what happens with other Fourier-related transforms, the so called Discrete Trigonometric Transforms (DTTs), such as the Discrete Cosine Transform (DCT) and the Discrete Sine Transform (DST), represent a function or a signal as a sum of trigonometric terms (cosine or sine), with different frequencies and amplitudes. Just like the DFT, the DCT and the DST also operate with a finite number of discrete data samples of a given function. Nevertheless, while the DCT only makes use of cosine functions, the DFT uses both cosines and sines (in the form of complex exponentials) to represent each signal. However, this difference is just a direct consequence of a more important characteristic of these transforms. As it will be seen in the following subsections, the DCT and the DST imply different boundary conditions on sample data than the DFT or other related transforms.

To simplify this description, the presentation that follows will be focused on one-dimensional (1-D) data sequences. Nevertheless, the same definitions can equally be extended to 2-D signals, without any loss of generalization.

2.2.1 Extension properties of sampled data beyond original boundaries

Just like any other Fourier-related transform that operates on a given function $f(n)$ over a finite discrete domain, the DFT, the DCT or the DST can be thought of as implicitly defining an extension of that function outside the original domain. Such implicit extension, defined as a sum of trigonometric functions, will then allow the evaluation of that function at any arbitrary point n, even for points where the original function $f(n)$ was not defined. Nevertheless, while the DFT and the Fourier series imply a periodic extension of the original function, the extension properties that are implicit in the DCT and DST imply quite distinct characteristics that provide useful applications in image and video processing.

Since the DCT and the DST operate on finite and discrete sequences, two issues arise concerning the symmetry properties of the extensions that are obtained from the input samples, which do not arise for the continuous cosine transform. Firstly, each boundary of the input data set can be extended *symmetrically* (also known as *even* extension) or *anti-symmetrically* (also known as *odd* extension). Secondly, the symmetry or anti-symmetry point of such extension must be specified. As an example, by considering a symmetric (even) extension of the left boundary of a simple data sequence composed of four equally spaced data points *abcd*, two distinct possible solutions arise in terms of the symmetry point: either the data is symmetrically extended about the sample a, in which case the even extension is *dcbabcd*; or the data is even about a hypothetical point, halfway between a and the previous point, in which case the symmetric extension is *dcbaabcd* (a is repeated).

By adopting these different extension setups, infinite sequences can easily be obtained by simple extending the input data samples of a given finite signal. Such infinite extensions are classified according to the types of symmetry adopted at each boundary of the original signal. The four possible extension setups are illustrated in fig. 2.2 and can be enumerated as follows [58]:

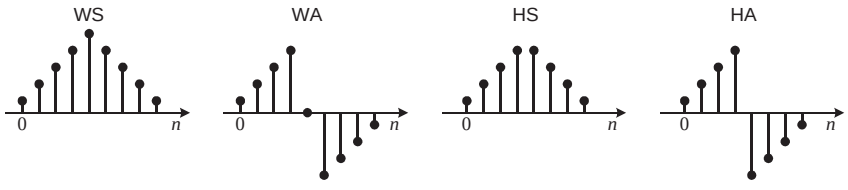


Figure 2.2: Symmetric-periodic extensions of a finite sequence.

- Whole-Sample Symmetry (WS),
- Whole-Sample Anti-symmetry (WA),
- Half-Sample Symmetry (HS),
- Half-Sample Anti-symmetry (HA),

where the designations *whole-sample* and *half-sample* refer to the position of the point of symmetry: either coincident with one of the original samples or at a theoretical half-way between two samples. Hence, a given finite sequence $f(n)$ can be easily converted into an infinite sequence by symmetrically extending each Point of Symmetry (POS) using any of the above four possible setups and by continuing that extension indefinitely, to obtain a Symmetric-Periodic Sequence (SPS), according to the following rules [58]:

$$\begin{aligned}
\mathrm{HSHS}(x_1,\ldots,x_n) &= \mathcal{P}(x_1,\ldots,x_n,x_n,\ldots,x_2) && (2.6)\\
\mathrm{HAHA}(x_1,\ldots,x_n) &= \mathcal{P}(x_1,\ldots,x_n,-x_n,\ldots,-x_2) && (2.7)\\
\mathrm{WSWS}(x_1,\ldots,x_n) &= \mathcal{P}(x_1,\ldots,x_{n-1},x_n,x_{n-1},\ldots,x_2) && (2.8)\\
\mathrm{WAWA}(x_1,\ldots,x_n) &= \mathcal{P}(0,x_2,\ldots,x_{n-1},0,-x_{n-1},\ldots,-x_2) && (2.9)
\end{aligned}$$

where $\mathcal{P}(\varphi)$ denotes the periodic replication of the sequence φ. The POS are either all of the same type or of two different types. If the adopted types are different, then they alternate along the length of the SPS. The obtained extensions are then usually denominated by concatenating the mnemonics of the symmetry types used at each of its ends (e.g.: WSWS, HAHA, WAHS, etc.).

Two POS are associated with the base period: a Left Point of Symmetry (LPOS) and a Right Point of Symmetry (RPOS); between them lie the representative samples. At each POS, it is implemented one of the four defined types of symmetry: WS, WA, HS or HA.

Four possible extension types at each of the two endpoints leads to a total of 16 distinct SPSs. All these different extensions characterize a broad set of standard variants of discrete cosine and sine transforms. For each of these 2 transforms, each of the 2 data set boundaries can be either symmetrically or anti-symmetrically extended (2 possibilities per boundary) and can be extended about a data point or a point halfway between two sample points (2 choices per boundary), thus giving rise to a total of $2 \times 2 \times 2 \times 2 = 16$ different possibilities. Half of these setups, corresponding to those where the left boundary is symmetrically extended, correspond to the 8 different types of DCTs, while the remaining half comprises the 8 types of

DSTs. In table 2.1 it is presented a comprehensive description of the several properties that are implicit to both the input and output extensions of the considered discrete cosine ($\mathcal{C}$) and sine ($\mathcal{S}$) transforms [58].

At this point, it is worth to recall that any discontinuity of the considered function potentially reduces the rate of convergence of its Fourier series, so that more sine or cosine terms are needed to represent it with a given accuracy. As a consequence, these different boundary conditions lead to different but useful properties for the several DCT and DST variants. In fact, such characteristic significantly influences the actual usefulness of the DFT and of other transforms for signal compression: the smoother a function is, the fewer terms in its DFT, DCT or DST are required to represent it accurately, and the more it can be compressed. However, the implicit periodicity of the input sequence that characterizes the DFT often implies considerable discontinuities at the signal boundaries, since any random segment of a given signal is unlikely to have the same pattern at both the left and right boundaries. In contrast, when both boundaries of a given signal are symmetrically extended, it naturally yields a continuous and smooth extension at the boundaries. This is why some types of DCT that have two symmetrically extended boundaries (in particular, DCTs of types I, II, V, and VI, as will be defined in the following subsections) generally perform better for signal compression than the other DTTs.

2.2.2 Discrete cosine transforms

From a pure and rather simplistic mathematical point of view, each DCT can be defined as a linear and invertible trigonometric function $\mathcal{F} : \mathbb{R}^N \rightarrow \mathbb{R}^N$. As it was referred in the previous section, there are several different variants of the DCT, presenting distinct formal definitions and characteristics. Nevertheless, all of them share a common important property: they all operate and output data sequences characterized by symmetric extensions at their left boundary. Hence, 8 different variants of the DCT are available, corresponding to all possible extension combinations in both boundaries of the output sequence that comply with this specific characteristic. A comprehensive list of extension alternatives and their corresponding variants of the DCT is presented in table 2.1.

Among these transforms, it is often denoted by *even* DCTs those transforms that also present the same characteristic in terms of the symmetry point in both boundaries (half-sample or whole-sample symmetry). Such transforms are usually denominated by DCT I, II, III and IV. On the other hand, those transforms with distinct characteristics in terms of the adopted point of symmetry in the two boundaries are usually denoted by *odd* DCTs and are denominated by DCT V, VI, VII

Table 2.1: Properties of the implicit input and output extensions of the considered discrete sine and cosine transforms.

Transform	Input extension properties					Output extension properties					
	SPS	**Length**	**Index Range**	**LPOS**	**RPOS**	**SPS**	**Length**	**Index Range**	**LPOS**	**RPOS**	
$\mathcal{C}_{1e}$	WSWS	N	$0 \to N-1$	0	$N-1$	WSWS	N	$0 \to N-1$	0	$N-1$	${\mathcal{C}_{1e}}^{-1}$
$\mathcal{C}_{2e}$	HSHS	N	$0 \to N-1$	$-\frac{1}{2}$	$N-\frac{1}{2}$	WSWA	N	$0 \to N-1$	0	N^*	${\mathcal{C}_{3e}}^{-1}$
$\mathcal{C}_{3e}$	WSWA	N	$0 \to N-1$	0	N^*	HSHS	N	$0 \to N-1$	$-\frac{1}{2}$	$N-\frac{1}{2}$	$\mathcal{C}_{2e}$−1
$\mathcal{C}_{4e}$	HSHA	N	$0 \to N-1$	$-\frac{1}{2}$	$N-\frac{1}{2}$	HSHA	N	$0 \to N-1$	$-\frac{1}{2}$	$N-\frac{1}{2}$	${\mathcal{C}_{4e}}^{-1}$
$\mathcal{C}_{1o}$	WSHS	N	$0 \to N-1$	0	$N-\frac{1}{2}$	WSHS	N	$0 \to N-1$	0	$N-\frac{1}{2}$	${\mathcal{C}_{1o}}^{-1}$
$\mathcal{C}_{2o}$	HSWS	N	$0 \to N-1$	$-\frac{1}{2}$	$N-1$	WSHA	N	$0 \to N-1$	0	$N-\frac{1}{2}$	${\mathcal{C}_{3o}}^{-1}$
$\mathcal{C}_{3o}$	WSHA	N	$0 \to N-1$	0	$N-\frac{1}{2}$	HSWS	N	$0 \to N-1$	$-\frac{1}{2}$	$N-1$	${\mathcal{C}_{2o}}^{-1}$
$\mathcal{C}_{4o}$	HSWA	N	$0 \to N-1$	$-\frac{1}{2}$	N^*	HSWA	N	$0 \to N-1$	$-\frac{1}{2}$	N^*	${\mathcal{C}_{4o}}^{-1}$
$\mathcal{S}_{1e}$	WAWA	N	$0 \to N-1$	-1^*	N^*	WAWA	N	$0 \to N-1$	-1^*	N^*	${\mathcal{S}_{1e}}^{-1}$
$\mathcal{S}_{2e}$	HAHA	N	$0 \to N-1$	$-\frac{1}{2}$	$N-\frac{1}{2}$	WAWS	N	$0 \to N-1$	-1^*	$N-1$	${\mathcal{S}_{3e}}^{-1}$
$\mathcal{S}_{3e}$	WAWS	N	$0 \to N-1$	-1^*	$N-1$	HAHA	N	$0 \to N-1$	$-\frac{1}{2}$	$N-\frac{1}{2}$	${\mathcal{S}_{2e}}^{-1}$
$\mathcal{S}_{4e}$	HAHS	N	$0 \to N-1$	$-\frac{1}{2}$	$N-\frac{1}{2}$	HAHS	N	$0 \to N-1$	$-\frac{1}{2}$	$N-\frac{1}{2}$	${\mathcal{S}_{4e}}^{-1}$
$\mathcal{S}_{1o}$	WAHA	N	$0 \to N-1$	-1^*	$N-\frac{1}{2}$	WAHA	N	$0 \to N-1$	-1^*	$N-\frac{1}{2}$	${\mathcal{S}_{1o}}^{-1}$
$\mathcal{S}_{2o}$	HAWA	N	$0 \to N-1$	$-\frac{1}{2}$	N^*	WAHS	N	$0 \to N-1$	-1^*	$N-\frac{1}{2}$	${\mathcal{S}_{3o}}^{-1}$
$\mathcal{S}_{3o}$	WAHS	N	$0 \to N-1$	-1^*	$N-\frac{1}{2}$	HAWA	N	$0 \to N-1$	$-\frac{1}{2}$	N^*	${\mathcal{S}_{2o}}^{-1}$
$\mathcal{S}_{4o}$	HAWS	N	$0 \to N-1$	$-\frac{1}{2}$	$N-1$	HAWS	N	$0 \to N-1$	$-\frac{1}{2}$	$N-1$	${\mathcal{S}_{4o}}^{-1}$
	SPS	**Length**	**Index Range**	**LPOS**	**RPOS**	**SPS**	**Length**	**Index Range**	**LPOS**	**RPOS**	**Transform**
	Output extension properties					**Input extension properties**					

Legend: SPS - Symmetric-Periodic Sequence; LPOS - Left Point of Symmetry; RPOS - Right Point of Symmetry;

* - Extra data sample of the implicitly extended sequence, corresponding to an anti-symmetric point of symmetry.

and VIII. Hence, while one of the boundaries presents a symmetry/anti-symmetry characteristic around an original data point, the other is extended around an implicit halfway point, between two data samples. These odd transforms, however, have been rarely used in practical image and video coding applications.

Independently of the specific type of DCT, the mathematical definition of each of these transforms is represented as a sum of product terms that multiply a cosine function $\mathcal{C}(m,i)$ with the input sequence $x(i)$:

$$X(m) = \sum_i \mathcal{C}(m,i).x(i) \tag{2.10}$$

According to the previously defined nomenclature, the subscript i denotes the coordinate in the spatial (pixel) domain, whereas the subscript m represents the coordinate in the transform-domain.

Equivalent matrix definitions are often adopted in the literature, which represent the DCT computation as a simple $(N \times N)$ matrix multiplication of a kernel matrix $\mathcal{C}$ by an input data vector $\mathbf{x}$, in the form:

$$\mathbf{X} = \mathcal{C} \cdot \mathbf{x} \tag{2.11}$$

In such matrix formulations, an e or o subscript is often appended to the kernel matrix definition ($\mathcal{C}$), to denote *even* or *odd* transforms (e.g.: $\mathcal{C}_\mathbf{e}$, $\mathcal{C}_\mathbf{o}$).

Independently of the adopted formalization, the term $\mathcal{C}(m,i)$ can be defined by the product:

$$\mathcal{C}(m,i) = A.w_1(m).w_2(i).t(m,i), \tag{2.12}$$

where $t(m,i) = \cos\left(f(m,i)\right)$ is the transform kernel; the term $w_1(m)$ is a weighting function that is used in some transforms to make the column vectors orthogonal to one another; the weighting function $w_2(i)$ is required by some transforms to make the row vectors orthogonal to one another; and the scalar A is a final multiplier that normalizes the rows and columns in order to produce a normal matrix. Hence, the mutual contribution of all these weighting functions leads to an orthonormal definition of each considered transform. These properties will be further described in section 2.2.5.

In table 2.2 it is presented a comprehensive list of the several orthogonal $(N \times N)$-point DCT kernel matrix definitions. The $w_1(m)$ and $w_2(i)$ terms are implemented by the orthogonalization functions $\xi\left(p\right)$ and $\zeta\left(p\right)$, defined in eqs. 2.13 and 2.14, respectively.

$$\xi\left(p\right) = \begin{cases} \sqrt{\frac{1}{2}} & , \text{ for } p = 0 \text{ or } p = N \\ 1 & , \text{ for } p = 1, 2, \ldots, N-1. \end{cases} \tag{2.13}$$

Table 2.2: Definition of the orthogonal DCT and DST kernel matrices, as defined by Rao and Yip [82], Strang [105] and Püschel and Moura [81, table 5.1 and table A.1].

DTT	Definition	Length	Index Range
DCT-I DCT-1e	$[\mathcal{C}_{1e}]_{\mathrm{m,i}} = \sqrt{\frac{2}{N-1}}\, \xi(m)\, \zeta(m)\, \xi(i)\, \zeta(i) \cos\left(\frac{mi\pi}{N-1}\right)$	N	$m = 0, 1, \ldots, (N-1)$ $i = 0, 1, \ldots, (N-1)$
DCT-II DCT-2e	$[\mathcal{C}_{2e}]_{\mathrm{m,i}} = \sqrt{\frac{2}{N}}\, \xi(m) \cos\left(\frac{m\left(i+\frac{1}{2}\right)\pi}{N}\right)$		
DCT-III DCT-3e	$[\mathcal{C}_{3e}]_{\mathrm{m,i}} = \sqrt{\frac{2}{N}}\, \xi(i) \cos\left(\frac{\left(m+\frac{1}{2}\right)\, i\pi}{N}\right)$		
DCT-IV DCT-4e	$[\mathcal{C}_{4e}]_{\mathrm{m,i}} = \sqrt{\frac{2}{N}}\, \cos\left(\frac{\left(m+\frac{1}{2}\right)\left(i+\frac{1}{2}\right)\pi}{N}\right)$		
DCT-V DCT-1o	$[\mathcal{C}_{1o}]_{\mathrm{m,i}} = \sqrt{\frac{2}{N-\frac{1}{2}}}\, \xi(m)\, \xi(i) \cos\left(\frac{mi\pi}{N-\frac{1}{2}}\right)$		
DCT-VI DCT-2o	$[\mathcal{C}_{2o}]_{\mathrm{m,i}} = \sqrt{\frac{2}{N-\frac{1}{2}}}\, \xi(m)\, \zeta(i) \cos\left(\frac{m\left(i+\frac{1}{2}\right)\pi}{N-\frac{1}{2}}\right)$		
DCT-VII DCT-3o	$[\mathcal{C}_{3o}]_{\mathrm{m,i}} = \sqrt{\frac{2}{N-\frac{1}{2}}}\, \zeta(m)\, \xi(i) \cos\left(\frac{\left(m+\frac{1}{2}\right) i\pi}{N-\frac{1}{2}}\right)$		
DCT-VIII DCT-4o	$[\mathcal{C}_{4o}]_{\mathrm{m,i}} = \sqrt{\frac{2}{N+\frac{1}{2}}}\, \cos\left(\frac{\left(m+\frac{1}{2}\right)\left(i+\frac{1}{2}\right)\pi}{N+\frac{1}{2}}\right)$		
DST-I DST-1e	$[\mathcal{S}_{1e}]_{\mathrm{m,i}} = \sqrt{\frac{2}{N+1}}\, \sin\left(\frac{(m+1)(i+1)\pi}{N+1}\right)$		
DST-II DST-2e	$[\mathcal{S}_{2e}]_{\mathrm{m,i}} = \sqrt{\frac{2}{N}}\, \xi(m) \sin\left(\frac{(m+1)\left(i+\frac{1}{2}\right)\pi}{N}\right)$		
DST-III DST-3e	$[\mathcal{S}_{3e}]_{\mathrm{m,i}} = \sqrt{\frac{2}{N}}\, \xi(i) \sin\left(\frac{\left(m+\frac{1}{2}\right)(i+1)\pi}{N}\right)$		
DST-IV DST-4e	$[\mathcal{S}_{4e}]_{\mathrm{m,i}} = \sqrt{\frac{2}{N}}\, \sin\left(\frac{\left(m+\frac{1}{2}\right)\left(i+\frac{1}{2}\right)\pi}{N}\right)$		
DST-V DST-1o	$[\mathcal{S}_{1o}]_{\mathrm{m,i}} = \sqrt{\frac{2}{N+\frac{1}{2}}}\, \sin\left(\frac{(m+1)(i+1)\pi}{N+\frac{1}{2}}\right)$		
DST-VI DST-2o	$[\mathcal{S}_{2o}]_{\mathrm{m,i}} = \sqrt{\frac{2}{N+\frac{1}{2}}}\, \sin\left(\frac{(m+1)\left(i+\frac{1}{2}\right)\pi}{N+\frac{1}{2}}\right)$		
DST-VII DST-3o	$[\mathcal{S}_{3o}]_{\mathrm{m,i}} = \sqrt{\frac{2}{N+\frac{1}{2}}}\, \sin\left(\frac{\left(m+\frac{1}{2}\right)(i+1)\pi}{N+\frac{1}{2}}\right)$		
DST-VIII DST-4o	$[\mathcal{S}_{4o}]_{\mathrm{m,i}} = \sqrt{\frac{2}{N-\frac{1}{2}}}\, \zeta(m)\, \zeta(i) \sin\left(\frac{\left(m+\frac{1}{2}\right)\left(i+\frac{1}{2}\right)\pi}{N-\frac{1}{2}}\right)$		

$$\zeta(p) = \begin{cases} 1 & , \text{ for } p = 0, 1, \ldots, N-2 \\ \sqrt{\frac{1}{2}} & , \text{ for } p = N-1. \end{cases} \tag{2.14}$$

As it can be observed, the definitions of the odd DCTs are quite similar to the corresponding even definitions, where the denominators of the cosine arguments are replaced by the value $N \pm \frac{1}{2}$.

2.2.3 Discrete sine transforms

Similarly to the previously defined DCTs, each DST can be defined, from a pure and rather simplistic mathematical point of view, as a linear and invertible trigonometric function $\mathcal{F}: \mathbb{R}^N \rightarrow \mathbb{R}^N$. The several variants of the DST also share a common important property, since they all operate and output data sequences characterized by anti-symmetric extensions at their left boundary. Consequently, 8 different variants of the DST are also available, corresponding to all possible extension combinations in both boundaries of the output sequence that comply with this specific characteristic. A comprehensive list of extension alternatives and the corresponding variants of the DST is also presented in table 2.1.

As it was mentioned for the cosine transform, DSTs are also divided in *even* and *odd* transforms, whenever they present the same characteristic in terms of the symmetry point in both boundaries (half-sample or whole-sample symmetry), or when they have distinct characteristics in terms of the adopted point of symmetry in the two boundaries, respectively. Hence, *even* DSTs are also denoted by DST I, II, III and IV, while *odd* DSTs are usually referred to as DST V, VI, VII and VIII.

Similarly to the DCTs, the mathematical definition of each DST is represented as a sum of product terms that multiply a sine function $\mathcal{S}(m,i)$ with the input sequence $x(i)$:

$$X(m) = \sum_i \mathcal{S}(m,i).x(i) \tag{2.15}$$

where the term $\mathcal{S}(m,i)$ is defined by the product:

$$\mathcal{S}(m,i) = A.w_1(m).w_2(i).t(m,i), \tag{2.16}$$

and $t(m,i) = \sin(f(m,i))$ is the corresponding transform kernel.

The equivalent matrix definitions represent the computation of each DST as an $(N \times N)$ matrix multiplication of a kernel matrix $\mathcal{S}$ by an input data vector $\mathbf{x}$, in the form:

$$\mathbf{X} = \mathcal{S} \cdot \mathbf{x} \tag{2.17}$$

In table 2.2 it is also presented a comprehensive list of the several orthogonal $(N \times N)$-point DST kernel matrix definitions. Similarly to the DCT definitions,

the transform kernel of each odd DSTs are quite similar to the corresponding even definition, where the denominators of the sine arguments are replaced by the value $N \pm \frac{1}{2}$.

2.2.4 Inverse transforms

The following expressions represent the relations between each inverse kernel matrix and the respective forward kernel matrix:

$$\mathcal{C}_1{}^{-1} = \mathcal{C}_1 \tag{2.18}$$

$$\mathcal{C}_2{}^{-1} = \mathcal{C}_3 \tag{2.19}$$

$$\mathcal{C}_3{}^{-1} = \mathcal{C}_2 \tag{2.20}$$

$$\mathcal{C}_4{}^{-1} = \mathcal{C}_4 \tag{2.21}$$

$$\mathcal{S}_1{}^{-1} = \mathcal{S}_1 \tag{2.22}$$

$$\mathcal{S}_2{}^{-1} = \mathcal{S}_3 \tag{2.23}$$

$$\mathcal{S}_3{}^{-1} = \mathcal{S}_2 \tag{2.24}$$

$$\mathcal{S}_4{}^{-1} = \mathcal{S}_4 \tag{2.25}$$

The designations for *even* or *odd* transforms have been omitted from these equations, since the same relation holds for both the *even* and *odd* cases.

2.2.5 Main properties

Several useful properties can be derived from the previously defined sine and cosine transforms. This section presents a brief overview of the most important and useful properties of these transforms for image and video coding [23, 34, 53]. In such presentation, a generic discrete cosine transform will be used for illustration. Nevertheless, the presented properties are equally valid for the whole family of trigonometric transforms defined in the previous subsections.

A - Linearity

According to the several definitions presented in table 2.2, any trigonometric transform can be regarded as a linear combination of linear functions (sine ($\mathcal{S}$) or cosine ($\mathcal{C}$) functions), which are added together using the input signal samples as weighting factors:

$$X(m) = \sum_i x(i)\, \mathcal{C}(m, i) \tag{2.26}$$

As a consequence, by denoting by $X(m)$ and $Y(m)$ the cosine transforms of the input samples $x(i)$ and $y(i)$, the following statement defines the linearity property of this transform for any scalar α and $\beta \in \mathbb{R}$:

$$\mathrm{DCT}\Big[\alpha x(i) + \beta y(i)\Big] = \alpha X(m) + \beta Y(m) \tag{2.27}$$

B - Orthogonality

The row and column vectors that compose each discrete sine and cosine transform kernel matrix define a set of orthogonal basis functions. Let $\mathcal{C}$ denote the $(n \times m)$ kernel matrix of a given cosine transform:

$$\mathcal{C} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{bmatrix} \tag{2.28}$$

This matrix is said to be orthogonal because all column vectors $\mathbf{c}_i = \begin{bmatrix} a_{1,i} & a_{2,i} & \dots & a_{n,i} \end{bmatrix}^{\mathrm{T}}$ fulfill the following relation, denoted by $\mathbf{c}_i \perp \mathbf{c}_j$:

$$\mathbf{c}_1{}^{\mathrm{T}} \cdot \mathbf{c}_2 = \mathbf{c}_1{}^{\mathrm{T}} \cdot \mathbf{c}_3 = \ldots = \mathbf{c}_i{}^{\mathrm{T}} \cdot \mathbf{c}_j = 0, \ \forall i \neq j \tag{2.29}$$

with $1 \leq i, j \leq m$. Hence, these relations imply that:

$$\mathcal{C}^{\mathrm{T}} = \mathcal{C}^{-1} \tag{2.30}$$

Entirely similar relations also apply for the row vectors $\mathbf{r}_i$.

C - Normalization

Each column vector $\mathbf{c}_i$ of any discrete sine and cosine transform kernel matrix also fulfills the following property:

$$\|\mathbf{c}_i\| = 1, \ \forall i: \ 1 \leq i \leq m. \tag{2.31}$$

As a consequence, the kernel matrices defined in table 2.2 are said to be normalized.

D - Orthonormality

Considering that each column or row vector of any discrete sine or cosine transform kernel matrix are both orthogonal and normalized, these matrices are also said to be orthonormal, thus presenting the following important properties:

$$\mathcal{C}^{\mathrm{T}} = \mathcal{C}^{-1}, \ \mathcal{C}^{\mathrm{T}}\mathcal{C} = \mathbf{I}, \ \mathcal{C}\,\mathcal{C}^{\mathrm{T}} = \mathbf{I} \tag{2.32}$$

These relations lead to a quite important consequence, since the matrix inversion operation is reduced to a simple matrix transpose, resulting in a significant computational cost reduction.

E - Energy conservation (Parseval's theorem)

Another important property of these discrete sine and cosine transform kernel matrices is related to the conservation of the signal energy after the computation of its transform. This property is also denoted by Parseval's theorem and can be formulated by the following expression:

$$\sum_{m=0}^{N-1} |X(m)|^2 = \|\mathbf{X}\|^2 = \mathbf{X}^{\mathrm{T}}\mathbf{X} = \mathbf{x}^{\mathrm{T}}\mathcal{C}^{\mathrm{T}}\mathcal{C}\,\mathbf{x} = \mathbf{x}^{\mathrm{T}}\mathbf{x} = \|\mathbf{x}\|^2 = \sum_{i=0}^{N-1} |x(i)|^2 \quad (2.33)$$

2.3 Multidimensional transforms

Multidimensional extensions of the several previously described sine and cosine transforms can be straightforwardly defined by considering separable decompositions along each dimension.

In fact, it can be easily shown that a two-dimensional transform can be regarded as the application of the same one-dimensional transform, performed firstly along the rows and then along the columns (see fig. 2.3). As an example, by considering the generic one-dimensional definition of the cosine transform, the extension to two dimensions can be defined as:

$$X(m,n) = \sum_i \sum_j \mathcal{C}(m,i)\,\mathcal{C}(n,j)\,x(i,j) \quad (2.34)$$

By re-arranging this equation, one can obtain:

$$X(m,n) = \sum_i \mathcal{C}(m,i) \sum_j \mathcal{C}(n,j)\,x(i,j) \quad (2.35)$$

$$= \sum_i \mathcal{C}(m,i)\,\tilde{X}^{1D}(i,n) \quad (2.36)$$

where $\tilde{X}^{1D}(i,n) = \left[\mathcal{C}\cdot\mathbf{x}^{\mathrm{T}}\right]^{\mathrm{T}} = \mathbf{x}\cdot\mathcal{C}^{\mathrm{T}}$ is a matrix whose lines are the 1-D discrete cosine transform of the lines of $x(i,j)$.

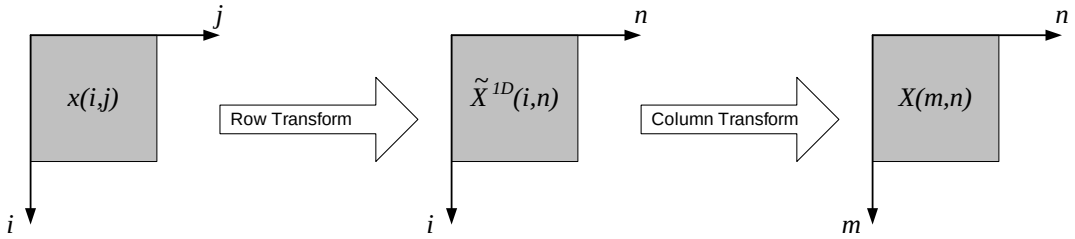


Figure 2.3: Row-Column decomposition of a 2-D transform.

Hence, the 2-D discrete cosine transform of the matrix $\mathbf{x} = [\mathbf{x}]_{i,j}$ can be represented in a matrix-product form as follows:

$$\mathbf{X} = [\mathbf{X}]_{m,n} = \mathcal{C} \cdot \mathbf{x} \cdot \mathcal{C}^{\mathrm{T}} \tag{2.37}$$

The same formulation could equally be applied to any other type of sine or cosine transform, as well as to their 2-D inverse transforms.

The corresponding inverse transform can then be derived as:

$$\mathbf{x} = [\mathbf{x}]_{i,j} = (\mathcal{C})^{-1} \cdot \mathbf{X} \cdot \left(\mathcal{C}^{\mathrm{T}}\right)^{-1} \tag{2.38}$$

By taking into account the orthogonality property of the kernel matrices, defined in section 2.2.5 ($\mathcal{C}^{-\mathbf{1}} = \mathcal{C}^{\mathrm{T}}$), the following expression can be easily derived for the inverse transform:

$$\mathbf{x} = [\mathbf{x}]_{i,j} = \mathcal{C}^{\mathrm{T}} \cdot \mathbf{X} \cdot \mathcal{C} \tag{2.39}$$

This row-column decomposition defines an additional property that is often referred to as **separability property**. It provides a quite important computational advantage: $X(m,n)$ can be computed in two steps by using successive 1-D operations on the rows and columns of the image data. From an implementation point of view, this row-column approach may significantly simplify the hardware requirements, at the expense of a slight increase in the overall operation-count.

2.4 Application of the DCT to image and video coding

Contrasting to the other previously described DTTs, the even type-II discrete cosine transform (DCT-II) has been widely adopted in image and video processing applications and is currently the basis of many image and video standards (e.g.: JPEG [32], H.263 [30], MPEG-2 Video [27], etc.). As it was previously referred, such fact is mainly owed to its particular well suited characteristics to exploit the spatial irrelevancies of a given pixels area, by concentrating most of the pixels energy in a restricted set of DCT coefficients [7]. Hence, most DCT-H.26x/MPEG-x standards transform each $(N \times N)$ pixels block from the spatial-domain into a $(N \times N)$ matrix of DCT-domain coefficients, by considering $N = 8$. The selection of this particular block size is historically related to several reasons. In what concerns the hardware and software implementation point of view, an (8×8) block size does not impose significant memory requirements and its DCT computation is easily manageable on

most computing platforms. On the other hand, in what concerns the compaction efficiency point of view, it has been observed that block sizes larger than (8×8) pixels do not offer any significantly better compression levels [7].

As a consequence, unless otherwise stated, from now on it will be adopted the even type-II discrete cosine transform, whenever the DCT operation is mentioned. Furthermore, the corresponding (8×8) kernel matrix will be simply referred by $\mathbf{T}$:

$$[\mathbf{T}(m,i)] \triangleq [\mathcal{C}_{2e}(m,i)] = \sqrt{\frac{2}{N}}\, \xi(m)\ \cos\left(\frac{m\left(i+\frac{1}{2}\right)\pi}{N}\right), \tag{2.40}$$

with $m, i = 0, 1, \ldots, (N-1)$, $N = 8$ and $\xi(m)$ defined in eq. 2.13.

2.4.1 One-dimensional discrete cosine transform

According to the above definitions, the 1-D DCT can be formulated as follows [2, 34, 53, 76, 82]:

$$X(m) = \sqrt{\frac{2}{N}}\, \xi(m) \sum_{i=0}^{N-1} \cos\left(\frac{m\left(i+\frac{1}{2}\right)\pi}{N}\right) x(i) \quad \longleftrightarrow \quad \mathbf{X} = \mathbf{T}\cdot\mathbf{x} \tag{2.41}$$

$$x(i) = \sqrt{\frac{2}{N}} \sum_{m=0}^{N-1} \xi(m)\ \cos\left(\frac{m\left(i+\frac{1}{2}\right)\pi}{N}\right) X(m) \quad \longleftrightarrow \quad \mathbf{x} = \mathbf{T}^{\mathrm{T}}\cdot\mathbf{X} \tag{2.42}$$

with $m, i = 0, 1, \ldots, (N-1)$ and $\xi(m)$ as defined in eq. 2.13.

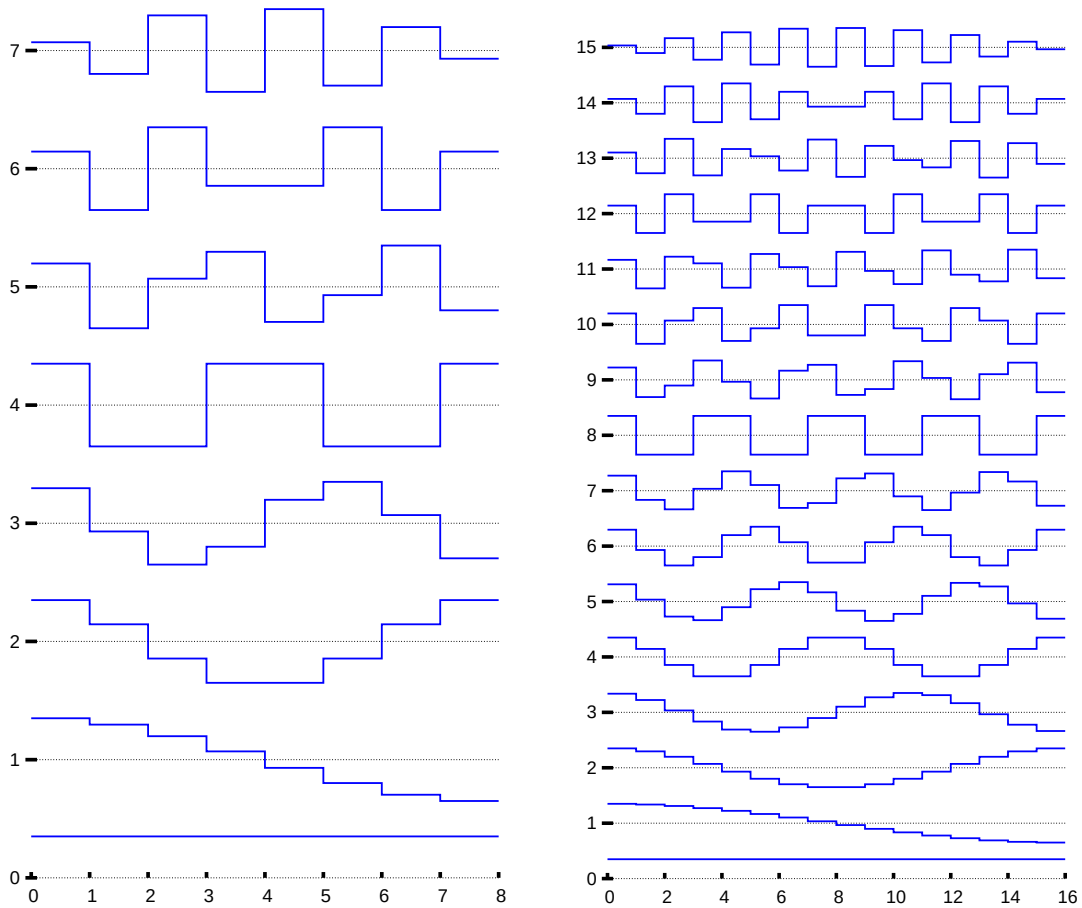
From a careful observation of equations 2.41 and 2.42, it can be seen that the computations of the forward and inverse DCT are nearly the same. Thus, from a hardware implementation point of view, the same computational unit can be used for both the forward and the inverse DCTs.

From eq. 2.41, it can also be observed that the first transform coefficient $(X(0))$ represents the average value of the sample sequence. As a consequence, this value is often referred to as the *DC Coefficient*, in analogy to what happens with the circuit analysis theory in electrical engineering. In accordance, all other transform coefficients are denoted by *AC Coefficients*.

The DCT decomposes each signal into a series of waveforms or basis functions, each one with a particular spatial frequency. In fig. 2.4(a) it is depicted the set of 8 basis functions corresponding to the discrete cosine transform, with $N = 8$. These waveforms actually correspond to the set of functions defined by the sum $\sum_{i=0}^{N-1} \cos\left(\frac{m\left(i+\frac{1}{2}\right)\pi}{N}\right)$, with $N = 8$ and m varying from $m = 0$ to $m = N-1$. In accordance with the previous paragraph, the bottom waveform $(m = 0)$ renders a constant (DC) value, whereas all other waveforms $(m = 1, 2, \ldots, 7)$ represent

different basis for progressively increasing frequencies. All these basis functions are orthogonal. As a consequence, the multiplication between any pair of these waveforms followed by a summation over all sample points yields a zero (scalar) value, whereas the multiplication of any of these waveforms with itself followed by a summation operation yields a constant (scalar) value. Hence, according to the orthogonal definition, these waveforms are said to be independent: none of the basis functions can be represented as a combination of the other basis functions. As a consequence, the DCT can be regarded as the process of finding the weight $X(m)$, corresponding to each waveform shown in fig. 2.4(a), so that the sum of the 8 waveforms, scaled by the corresponding weights $X(m)$, yields the reconstructed version of the original 8-point vector $\mathbf{x}$.

For illustrations purposes, it is also presented in fig. 2.4(b) the set of basis functions corresponding to a $N = 16$-point DCT.



(a) $N = 8$. **(b)** $N = 16$.

Figure 2.4: 1-D DCT basis functions.

2.4.2 Two-dimensional discrete cosine transform

The extension of the above defined transform to a bi-dimensional space is straightforward [2, 34, 53, 76, 82]:

$$X(m,n) = \frac{2}{N}\,\xi(m)\,\xi(n)\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} x(i,j)\cos\left(\frac{m\left(i+\frac{1}{2}\right)\pi}{N}\right)\cos\left(\frac{n\left(j+\frac{1}{2}\right)\pi}{N}\right) \tag{2.43}$$

$$\Leftrightarrow\ \mathbf{X} = \mathbf{T}\cdot\mathbf{x}\cdot\mathbf{T}^{\mathrm{T}} \tag{2.44}$$

$$x(i,j) = \frac{2}{N}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}\xi(m)\ \xi(n)\ X(m,i)\cos\left(\frac{m\left(i+\frac{1}{2}\right)\pi}{N}\right)\cos\left(\frac{n\left(j+\frac{1}{2}\right)\pi}{N}\right) \tag{2.45}$$

$$\Leftrightarrow\ \mathbf{x} = \mathbf{T}^{\mathrm{T}}\cdot\mathbf{X}\cdot\mathbf{T} \tag{2.46}$$

with $m,n = 0,1,\ldots,(N-1)$; $i,j = 0,1,\ldots,(N-1)$; and $\xi(m)$ and $\xi(n)$ defined in eq. 2.13.

Just like the definition of the 1-D DCT, the transform coefficient $X(0,0)$ represents the average value of the sample sequence and is denoted by *DC Coefficient*, while all other transform coefficients are denoted by *AC Coefficients*.

In fig. 2.5 it is depicted the set of 64 basis functions corresponding to the 2-D discrete cosine transform, with $N = 8$. These 2-D basis functions can be generated by multiplying the horizontally oriented 1-D basis functions (shown in fig. 2.4(a)) with a vertically oriented set of the same functions. As it was observed for the 1-D case, it can be noted that the basis functions exhibit a progressive increase with the frequency, both in the vertical and horizontal directions. A particular case occurs with the top-left basis function, which results from the multiplication of two constant vectors, corresponding to the DC component in fig. 2.4(a). Just like the 1-D case, such function assumes a constant value (*DC Coefficient*). Hence, each 2-D DCT can be regarded as the process of finding the weight $X(m,n)$, corresponding to each waveform shown in fig. 2.5, so that the sum of the 64 waveforms, scaled by the corresponding weights $X(m,n)$, yields the reconstructed version of the original (8×8) pixel matrix $\mathbf{x}$.

2.5 Multiplication-convolution property

As it will be seen in the following chapters, some processing functions that directly operate with the DCT coefficients of an encoded image or video stream require the application of other more complex properties, besides those presented in section 2.2.5. One of such properties will be used by the static video composition operations, proposed in chapter 4, and concerns the relation between the convolution

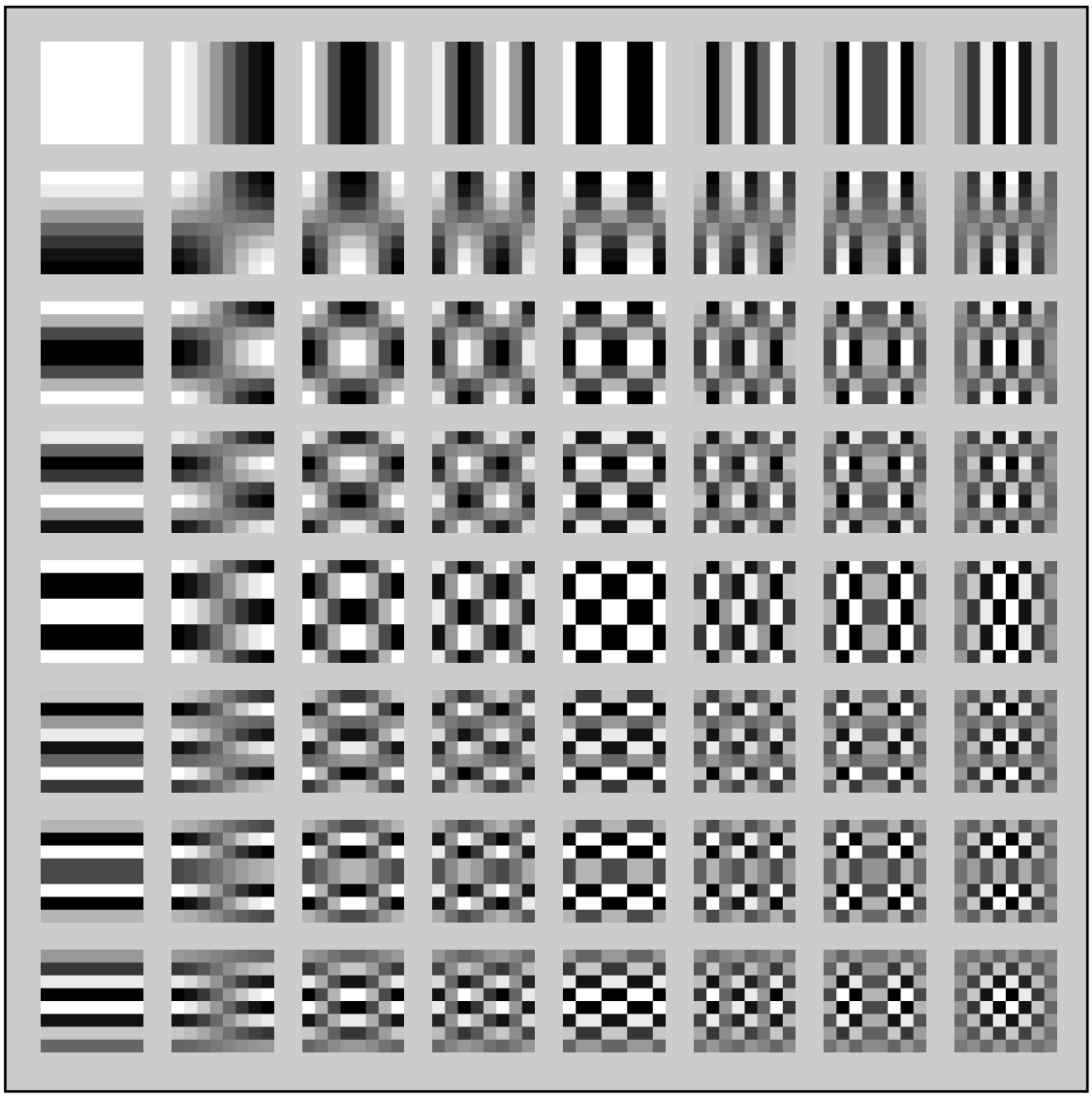


Figure 2.5: 2-D DCT basis functions.

operation, implemented in the transform-domain, and the corresponding pixel-wise multiplication of two blocks of pixels.

The formulation of such relations is quite common in other transform domains. As an example, the Fast Fourier Transform (FFT) domain component-wise multiplication is widely used to accomplish spatial-domain convolution, required by many feature extraction and filtering operations. Nevertheless, a similar relation, but implemented in the DCT-domain, will be required by the set of applications that will be considered in this thesis. However, although the DCT is closely related to the DFT, the multiplication-convolution theorem for the DCT was established much after the corresponding relationship for the DFT. In fact, despite the several

attempts to establish this relation [104], a complete and more consistent formalization was only presented relatively recently [11, 58]. In particular, Martucci [58] presented a formalized and detailed description of the convolution operation for the entire family of discrete sine and cosine transforms. In his presentation, the DCT and the DST are regarded as special cases of the so called Generalized Discrete Fourier Transform (GDFT), which operates on infinite sequences that are strictly periodic or antiperiodic, with period N. Martucci denoted such definition as *symmetric convolution*, since it proved to be specially suited to convolve symmetrically extended sequences. In practice, such type of convolution can be regarded as a conventional convolution sum, that has been suitably modified to incorporate the implied symmetric extensions of both operands.

In this section it will be presented the formulation of the multiplication-convolution property, in the DCT-domain. Such property will provide the means to replace the spatial-domain pixel-wise multiplication by a DCT-domain symmetric convolution operation. A fast computational method to compute such convolution will be presented in subsection 2.5.3.

2.5.1 Generic discrete trigonometric transform

The definition of the discrete cosine transform that was presented in the previous section corresponds to the first formulation of the DCT, reported by Ahmed, Natarajan, and Rao, in 1974 [2]. It is also the most commonly used in image and video standards. As it was referred in section 2.2, several other DTTs have also been proposed, such as the 8 types of DCTs and the 8 types of DSTs defined in tables 2.1 and 2.2 [58, 82]. Since the kernel matrices of all these transforms are orthogonal and invertible, the kernel matrices of their inverses can be easily obtained by their transposes.

Meanwhile, Martucci [58] proposed a new formulation for these DTTs, denoted by *convolution form*, where the orthogonality of the kernel matrices was lost for most DTT types. This convolution form was shown to be more appropriate for applying the *convolution-multiplication property* than the above *orthogonal form*, derived by Wang [110], since it avoids the need for adding any scaling factors or weighting functions to the convolution-multiplication formula. The definition and formulation of each of these kernel matrices for the DCT and the DST is presented in tables 2.3 and 2.4 [44, 58]. Since the kernel functions of some of these transforms evaluate to zero for some values of the indices m and i, Martucci rearranged some of their index ranges in order to avoid null values. As a consequence, contrasting with the definitions presented in tables 2.1 and 2.2, the index ranges for both m

Table 2.3: Properties of the implicit input and output extensions of the convolution formulations of the considered discrete sine and cosine transforms.

Transform	Input extension properties					Output extension properties					
	SPS	**Length**	**Index Range**	**LPOS**	**RPOS**	**SPS**	**Length**	**Index Range**	**LPOS**	**RPOS**	
$\mathcal{C}_{1e}$	WSWS	$N+1$	$0 \to N$	0	N	WSWS	$N+1$	$0 \to N$	0	N	$\mathcal{C}_{1e}{}^{-1}$
$\mathcal{C}_{2e}$	HSHS	N	$0 \to N-1$	$-\frac{1}{2}$	$N-\frac{1}{2}$	WSWA	N	$0 \to N-1$	0	N^*	$\mathcal{C}_{3e}{}^{-1}$
$\mathcal{C}_{3e}$	WSWA	N	$0 \to N-1$	0	N^*	HSHS	N	$0 \to N-1$	$-\frac{1}{2}$	$N-\frac{1}{2}$	$\mathcal{C}_{2e}{-1}$
$\mathcal{C}_{4e}$	HSHA	N	$0 \to N-1$	$-\frac{1}{2}$	$N-\frac{1}{2}$	HSHA	N	$0 \to N-1$	$-\frac{1}{2}$	$N-\frac{1}{2}$	$\mathcal{C}_{4e}{}^{-1}$
$\mathcal{C}_{1o}$	WSHS	N	$0 \to N-1$	0	$N-\frac{1}{2}$	WSHS	N	$0 \to N-1$	0	$N-\frac{1}{2}$	$\mathcal{C}_{1o}{}^{-1}$
$\mathcal{C}_{2o}$	HSWS	N	$0 \to N-1$	$-\frac{1}{2}$	$N-1$	WSHA	N	$0 \to N-1$	0	$N-\frac{1}{2}$	$\mathcal{C}_{3o}{}^{-1}$
$\mathcal{C}_{3o}$	WSHA	N	$0 \to N-1$	0	$N-\frac{1}{2}$	HSWS	N	$0 \to N-1$	$-\frac{1}{2}$	$N-1$	$\mathcal{C}_{2o}{}^{-1}$
$\mathcal{C}_{4o}$	HSWA	$N-1$	$0 \to N-2$	$-\frac{1}{2}$	$N-1^*$	HSWA	$N-1$	$0 \to N-2$	$-\frac{1}{2}$	$N-1^*$	$\mathcal{C}_{4o}{}^{-1}$
$\mathcal{S}_{1e}$	WAWA	$N-1$	$1 \to N-1$	0^*	N^*	WAWA	$N-1$	$1 \to N-1$	0^*	N^*	$\mathcal{S}_{1e}{}^{-1}$
$\mathcal{S}_{2e}$	HAHA	N	$0 \to N-1$	$-\frac{1}{2}$	$N-\frac{1}{2}$	WAWS	N	$1 \to N$	0^*	N	$\mathcal{S}_{3e}{}^{-1}$
$\mathcal{S}_{3e}$	WAWS	N	$1 \to N$	0^*	N	HAHA	N	$0 \to N-1$	$-\frac{1}{2}$	$N-\frac{1}{2}$	$\mathcal{S}_{2e}{}^{-1}$
$\mathcal{S}_{4e}$	HAHS	N	$0 \to N-1$	$-\frac{1}{2}$	$N-\frac{1}{2}$	HAHS	N	$0 \to N-1$	$-\frac{1}{2}$	$N-\frac{1}{2}$	$\mathcal{S}_{4e}{}^{-1}$
$\mathcal{S}_{1o}$	WAHA	$N-1$	$1 \to N-1$	0^*	$N-\frac{1}{2}$	WAHA	$N-1$	$1 \to N-1$	0^*	$N-\frac{1}{2}$	$\mathcal{S}_{1o}{}^{-1}$
$\mathcal{S}_{2o}$	HAWA	$N-1$	$0 \to N-2$	$-\frac{1}{2}$	$N-1^*$	WAHS	$N-1$	$1 \to N-1$	0^*	$N-\frac{1}{2}$	$\mathcal{S}_{3o}{}^{-1}$
$\mathcal{S}_{3o}$	WAHS	$N-1$	$1 \to N-1$	0^*	$N-\frac{1}{2}$	HAWA	$N-1$	$0 \to N-2$	$-\frac{1}{2}$	$N-1^*$	$\mathcal{S}_{2o}{}^{-1}$
$\mathcal{S}_{4o}$	HAWS	N	$0 \to N-1$	$-\frac{1}{2}$	$N-1$	HAWS	N	$0 \to N-1$	$-\frac{1}{2}$	$N-1$	$\mathcal{S}_{4o}{}^{-1}$
	SPS	**Length**	**Index Range**	**LPOS**	**RPOS**	**SPS**	**Length**	**Index Range**	**LPOS**	**RPOS**	**Transform**
	Output extension properties					**Input extension properties**					

Legend: SPS - Symmetric-Periodic Sequence; LPOS - Left Point of Symmetry; RPOS - Right Point of Symmetry;

* - Extra data sample of the implicitly extended sequence, corresponding to an anti-symmetric point of symmetry.

Table 2.4: Definition of the convolution formulation of the DCT and DST kernel matrices [58].

DTT	Definition	Length	Index Range
cDCT-I cDCT-1e	$[\mathcal{C}_{1e}]_{\mathrm{m,i}} = 2\ \xi^2(i) \cos\left(\frac{mi\pi}{N}\right)$	$(N+1)$	$m, i = 0, 1, \ldots, N$
cDCT-II cDCT-2e	$[\mathcal{C}_{2e}]_{\mathrm{m,i}} = 2\ \cos\left(\frac{m\left(i+\frac{1}{2}\right)\pi}{N}\right)$	N	$m, i = 0, 1, \ldots, (N-1)$
cDCT-III cDCT-3e	$[\mathcal{C}_{3e}]_{\mathrm{m,i}} = 2\ \xi^2(i) \cos\left(\frac{\left(m+\frac{1}{2}\right) i\,\pi}{N}\right)$	N	$m, i = 0, 1, \ldots, (N-1)$
cDCT-IV cDCT-4e	$[\mathcal{C}_{4e}]_{\mathrm{m,i}} = 2\ \cos\left(\frac{\left(m+\frac{1}{2}\right)\left(i+\frac{1}{2}\right)\pi}{N}\right)$	N	$m, i = 0, 1, \ldots, (N-1)$
cDCT-V cDCT-1o	$[\mathcal{C}_{1o}]_{\mathrm{m,i}} = 2\ \xi^2(i) \cos\left(\frac{mi\pi}{N-\frac{1}{2}}\right)$	N	$m, i = 0, 1, \ldots, (N-1)$
cDCT-VI cDCT-2o	$[\mathcal{C}_{2o}]_{\mathrm{m,i}} = 2\ \zeta^2(i) \cos\left(\frac{m\left(i+\frac{1}{2}\right)\pi}{N-\frac{1}{2}}\right)$	N	$m, i = 0, 1, \ldots, (N-1)$
cDCT-VII cDCT-3o	$[\mathcal{C}_{3o}]_{\mathrm{m,i}} = 2\ \xi^2(i) \cos\left(\frac{\left(m+\frac{1}{2}\right) i\pi}{N-\frac{1}{2}}\right)$	N	$m, i = 0, 1, \ldots, (N-1)$
cDCT-VIII cDCT-4o	$[\mathcal{C}_{4o}]_{\mathrm{m,i}} = 2\ \cos\left(\frac{\left(m+\frac{1}{2}\right)\left(i+\frac{1}{2}\right)\pi}{N-\frac{1}{2}}\right)$	$(N-1)$	$m, i = 0, 1, \ldots, (N-2)$
cDST-I cDST-1e	$[\mathcal{S}_{1e}]_{\mathrm{m,i}} = 2\ \sin\left(\frac{mi\pi}{N}\right)$	$(N-1)$	$m, i = 1, 2, \ldots, (N-1)$
cDST-II cDST-2e	$[\mathcal{S}_{2e}]_{\mathrm{m,i}} = 2\ \sin\left(\frac{m\left(i+\frac{1}{2}\right)\pi}{N}\right)$	N	$m = 1, 2, \ldots, N$ $i = 0, 1, \ldots, (N-1)$
cDST-III cDST-3e	$[\mathcal{S}_{3e}]_{\mathrm{m,i}} = 2\ \xi^2(i) \sin\left(\frac{\left(m+\frac{1}{2}\right) i\pi}{N}\right)$	N	$m = 0, 1, \ldots, (N-1)$ $i = 1, 2, \ldots, N$
cDST-IV cDST-4e	$[\mathcal{S}_{4e}]_{\mathrm{m,i}} = 2\ \sin\left(\frac{\left(m+\frac{1}{2}\right)\left(i+\frac{1}{2}\right)\pi}{N}\right)$	N	$m, i = 0, 1, \ldots, (N-1)$
cDST-V cDST-1o	$[\mathcal{S}_{1o}]_{\mathrm{m,i}} = 2\ \sin\left(\frac{mi\pi}{N-\frac{1}{2}}\right)$	$(N-1)$	$m, i = 1, 2, \ldots, (N-1)$
cDST-VI cDST-2o	$[\mathcal{S}_{2o}]_{\mathrm{m,i}} = 2\ \sin\left(\frac{m\left(i+\frac{1}{2}\right)\pi}{N-\frac{1}{2}}\right)$	$(N-1)$	$m = 1, 2, \ldots, (N-1)$ $i = 0, 1, \ldots, (N-2)$
cDST-VII cDST-3o	$[\mathcal{S}_{3o}]_{\mathrm{m,i}} = 2\ \sin\left(\frac{\left(m+\frac{1}{2}\right) i\pi}{N-\frac{1}{2}}\right)$	$(N-1)$	$m = 0, 1, \ldots, (N-2)$ $i = 1, 2, \ldots, (N-1)$
cDST-VIII cDST-4o	$[\mathcal{S}_{4o}]_{\mathrm{m,i}} = 2\ \zeta^2(i) \sin\left(\frac{\left(m+\frac{1}{2}\right)\left(i+\frac{1}{2}\right)\pi}{N-\frac{1}{2}}\right)$	N	$m, i = 0, 1, \ldots, (N-1)$

and i for some of these transform definitions are no longer the same. With this new formulation, there is a direct link between all DTTs and the GDFT.

Similarly to the orthogonal DCT and DST definitions, the following expressions represent the relations between each inverse kernel matrix and its own, or another, forward kernel matrix:

$$\mathcal{C}_1{}^{-1} = \frac{1}{M}\,\mathcal{C}_1 \tag{2.47}$$

$$\mathcal{C}_2{}^{-1} = \frac{1}{M}\,\mathcal{C}_3 \tag{2.48}$$

$$\mathcal{C}_3{}^{-1} = \frac{1}{M}\,\mathcal{C}_2 \tag{2.49}$$

$$\mathcal{C}_4{}^{-1} = \frac{1}{M}\,\mathcal{C}_4 \tag{2.50}$$

$$\mathcal{S}_1{}^{-1} = \frac{1}{M}\,\mathcal{S}_1 \tag{2.51}$$

$$\mathcal{S}_2{}^{-1} = \frac{1}{M}\,\mathcal{S}_3 \tag{2.52}$$

$$\mathcal{S}_3{}^{-1} = \frac{1}{M}\,\mathcal{S}_2 \tag{2.53}$$

$$\mathcal{S}_4{}^{-1} = \frac{1}{M}\,\mathcal{S}_4 \tag{2.54}$$

where $M = 2N$ for even transforms and $M = 2N-1$ for odd transforms. As before, since the same relation holds for both the *even* and *odd* cases, the designations for *even* or *odd* transforms have been omitted.

2.5.2 Definition

The DCT-domain convolution operation and its relation with the element-by-element multiplication, in the space-domain, can be defined as follows. Let $f(i)$ and $g(i)$ represent two vectors, each one with N elements. The length-N vector $h(i)$, obtained from the element-by-element multiplication, between $f(i)$ and $g(i)$ is denoted as:

$$\mathbf{h} = \mathbf{f} \odot \mathbf{g} \tag{2.55}$$

The multiplication-convolution property that is now defined relates the DCT coefficients vectors $F(m)$ and $G(m)$, corresponding to the input vectors $f(i)$ and $g(i)$, with the resulting DCT coefficients output vector $H(m)$.

Martucci [58] presented the formulation of this property for all considered DTTs using the following notation:

$$\mathbf{\Omega}_n = \varepsilon_a\{\mathbf{\Phi}_n\} \circledast \varepsilon_b\{\mathbf{\Psi}_n\} = \tau_c^{-1}\Big\{\tau_a\{\mathbf{\Phi}_n\} \odot \tau_b\{\mathbf{\Psi}_n\}\Big\} \tag{2.56}$$

Table 2.5: Multiplication-convolution properties of WSWA DTT extensions.

ϵ_a	ϵ_b	Output Extension	$\circledast$	Input Index Ranges		Output Index Range	τ_a	τ_b	${\tau_c}^{-1}$
				Φ_n	Ψ_n				
WSWA	WSWA	WSWA	Ⓢ	$0 \rightarrow N-1$	$0 \rightarrow N-1$	$0 \rightarrow N-1$	$\mathcal{C}_{3e}$	$\mathcal{C}_{3e}$	${\mathcal{C}_{3e}}^{-1}$
WSWA	WAWS	WAWS	Ⓢ	$0 \rightarrow N-1$	$1 \rightarrow N$	$1 \rightarrow N$	$\mathcal{C}_{3e}$	$\mathcal{S}_{3e}$	${\mathcal{S}_{3e}}^{-1}$
WAWS	WAWS	WSWA	Ⓢ	$1 \rightarrow N$	$1 \rightarrow N$	$0 \rightarrow N-1$	$\mathcal{S}_{3e}$	$\mathcal{S}_{3e}$	$-{\mathcal{C}_{3e}}^{-1}$

where $\mathbf{\Phi}_n$ and $\mathbf{\Psi}_n$ are two input sequences of finite length and $\mathbf{\Omega}_n$ is the output convolved sequence. In this expression, ε_a and ε_b are two generic symmetric or anti-symmetric extension operators, as defined in section 2.2.1, $\circledast$ denotes the symmetric convolution operation, defined in terms of a conventional convolution sum that has been suitably modified to incorporate the implied symmetric extensions of both operands, and $\odot$ denotes the element-by-element multiplication. The operators τ_a, τ_b and τ_c^{-1} define three invertible convolution-form DTTs, as defined in table 2.4, that transform from one domain to another.

Martucci grouped such relations in families of three or four DTTs, whose input and output data sequences share the same type of symmetry. Since most DCT-H.26x/MPEG-x video standards make use of the even type-II DCT (DCT-IIe), characterized by a WSWA symmetry extension (see table 2.1), the family of such relations that deal with the WS and the WA extensions shows to be specially relevant:

$$\mathbf{\Omega}^{\mathrm{WSWA}} = f(\mathbf{\Phi}^{\mathrm{WSWA}}, \mathbf{\Psi}^{\mathrm{WSWA}}) \tag{2.57}$$

In table 2.5 it is presented the several properties of such group, where Ⓢ denotes the skew-circular convolution operation. Such particular form of the symmetric convolution of two length-M sequences $\alpha(n)$ and $\beta(n)$, with $n = 0, 1, \ldots, M-1$ is defined as:

$$\alpha(n) \text{ Ⓢ } \beta(n) = \sum_{k=0}^{n} \alpha(n)\beta(n-k) - \sum_{k=n+1}^{M-1} \alpha(n)\beta(n-k+M) \tag{2.58}$$

Hence, the skew-circular convolution operation of two length-M sequences defines an output sequence that is equivalent to the corresponding period of a periodic convolution of anti-symmetric sequences, with period M.

Among the properties presented in table 2.5, the first formulation is considered specially adequate, since it relates two input WSWA symmetric extensions and outputs one WSWA extension. By applying such property to the formulation stated in eq. 2.56, the following relation is obtained:

$$\mathbf{\Omega}^{\mathrm{WSWA}} = \mathbf{\Phi}^{\mathrm{WSWA}} \text{ Ⓢ } \mathbf{\Psi}^{\mathrm{WSWA}} = {\mathcal{C}_{\mathbf{3e}}}^{-1}\Big\{\mathcal{C}_{\mathbf{3e}}\{\mathbf{\Phi}\} \odot \mathcal{C}_{\mathbf{3e}}\{\mathbf{\Psi}\}\Big\}. \tag{2.59}$$

By recalling the relation between the forward and inverse definition of the $\mathcal{C}_{2e}$ and the $\mathcal{C}_{3e}$ transforms, stated in eqs. 2.48 and 2.49, the above equation comes as follows:

$$\mathbf{\Omega}^{\text{WSWA}} = \mathbf{\Phi}^{\text{WSWA}} \circledS \mathbf{\Psi}^{\text{WSWA}} = \mathcal{C}_{\mathbf{2e}}\Big\{ {\mathcal{C}_{\mathbf{2e}}}^{-1}\{\mathbf{\Phi}\} \odot {\mathcal{C}_{\mathbf{2e}}}^{-1}\{\mathbf{\Psi}\} \Big\}. \tag{2.60}$$

Under this assumption, if we denote the generic WSWA data sequences $\mathbf{\Phi}$, $\mathbf{\Psi}$ and $\mathbf{\Omega}$ by the corresponding convolution form of the even type-II DCT coefficients $F(m)$, $G(m)$ and $H(m)$, it comes:

$$H(m)^{\text{WSWA}} = F^{\text{WSWA}}(m) \circledS G^{\text{WSWA}}(m) = \mathcal{C}_{\mathbf{2e}}\Big\{ f(i) \odot g(i) \Big\}. \tag{2.61}$$

According to eq. 2.61, the DCT coefficients $H(m)$, corresponding to the pixels vector $h(i)$, that is obtained by the element-by-element multiplication of the pixels sequences $f(i)$ and $g(i)$, can be obtained by computing a skew-circular convolution operation with the DCT coefficients $F(m)$ and $G(m)$, corresponding to the pixel sequences $f(i)$ and $g(i)$, respectively. Such skew-circular convolution is computed by a circular convolution between WSWA symmetrically-extended versions of the input operands.

Chang and Messerschmitt [11] presented an equivalent formulation of this definition that directly operates with the orthogonal form of the even type-II DCT (DCT-IIe) of the input signals. According with such formulation, the DCT of $h(i)$ can be computed by applying the symmetric convolution operation to the DCT coefficients of the two length-N WSWA sequences $\mathbf{F} = \text{DCT-IIe}(\mathbf{f})$ and $\mathbf{G} = \text{DCT-IIe}(\mathbf{g})$, as defined as follows:

$$H(m) = F(m) \circledast G(m) = W_N(m) \left(\tilde{F}(m) \circledS \tilde{G}(m) \right). \tag{2.62}$$

The vectors $\tilde{F}(m)$ and $\tilde{G}(m)$ correspond to symmetric length-$2N$ WSWA extended sequences of $F(m)$ and $G(m)$, defined as:

$$\tilde{X}(m) = \begin{cases} 0 & , \; m = 0 \\ \hat{X}(N-m) & , \; m = 1 \ldots (N-1) \\ \hat{X}(m-N) & , \; m = N \ldots (2N-1) \end{cases} \tag{2.63}$$

where $\hat{X}(m) = \frac{X(m)}{\xi(m)}$, with $X(m) = \text{DCT-IIe}\,[x(i)]$ and $\xi(m)$ as defined in eq. 2.13. The skew-circular convolution $\circledS$, defined in eq. 2.58, is computed as follows:

$$\tilde{F}(m) \circledS \tilde{G}(m) = \frac{1}{\sqrt{2N}} \cdot \xi(m) \cdot \left[\sum_{n=0}^{m} \tilde{F}(n)\tilde{G}(m-n) - \sum_{n=m+1}^{2N-1} \tilde{F}(n)\tilde{G}(m-n+2N) \right] \tag{2.64}$$

$$= \frac{1}{\sqrt{2N}} \cdot \xi(m) \cdot \left[\sum_{n=0}^{2N-1} \tilde{F}(n)\tilde{G}\left[\text{mod}_{2N}\,(m-n)\right] \cdot S(m-n) \right] \tag{2.65}$$

where:

$$S(m-n) = \begin{cases} 1 & , \ m-n \in [0,\ (2N-1)] \\ -1 & , \ \text{otherwise} \end{cases} \tag{2.66}$$

and $W_N(m)$ is a length-N rectangular window, which is used to extract the representative samples out of the base period of the result of the convolution.

The extension of the above definition to the 2-D domain can be easily formulated as shown in eq. 2.67, where $\xi(m)$ and $S(m)$ were defined in eq. 2.13 and 2.66 and $\tilde{X}(m_1, m_2)$ is a $(2N \times 2N)$ symmetric WSWA extended sequence defined as shown in eq. 2.68.

$$F(m_1,m_2) \circledS G(m_1,m_2) = \frac{1}{2N}\, \xi(m_1)\, \xi(m_2) \left[\sum_{n_1=0}^{2N-1} \sum_{n_2=0}^{2N-1} \tilde{F}(n_1,n_2) \tilde{G}\left[\mathrm{mod}_{2N}(m_1-n_1), \mathrm{mod}_{2N}(m_2-n_2)\right] \cdot S(m_1-n_1) \cdot S(m_2-n_2) \right] \tag{2.67}$$

$$\tilde{X}(m_1,m_2) = \begin{cases} 0 & , \ m_1 = 0 \text{ or } m_2 = 0 \\ \hat{X}(N-m_1, N-m_2) & , \ m_1 = 1 \ldots (N-1),\ m_2 = 1 \ldots (N-1) \\ \hat{X}(m_1-N, N-m_2) & , \ m_1 = N \ldots (2N-1),\ m_2 = 1 \ldots (N-1) \\ \hat{X}(N-m_1, m_2-N) & , \ m_1 = 1 \ldots (N-1),\ m_2 = N \ldots (2N-1) \\ \hat{X}(m_1-N, m_2-N) & , \ m_1,\ m_2 = N \ldots (2N-1) \end{cases} \tag{2.68}$$

2.5.3 Fast computation of the convolution operation in the DCT-domain

By analyzing the definition of the convolution operation, expressed in equations 2.62 through 2.66, it can be observed that the total number of multiplications required to perform the convolution between two length-N sequences is proportional to $(2N)^2 = 4N^2$. By extending this analysis to the 2-D domain, one can conclude that $\left[(2N)^2\right]^2 = 16N^4$ multiplications are required to evaluate the convolution between two $(N \times N)$ bidimensional sequences (see eq. 2.67).

Recently, Shen et al. [100] proposed a different approach to compute the DCT-domain convolution, by exploiting the symmetry and the orthogonality properties of the DCT to reduce the overall computational cost of the convolution operation. They started their formulation from the 1-D length-N DCT-II transform definition, as follows:

$$X(m) = \sum_{i=0}^{N-1} \mathcal{C}_{2e}(m,i)\, x(i), \tag{2.69}$$

where $\mathcal{C}_{2e}(m,i) = \sqrt{\frac{2}{N}}\, \xi(m) \cos\left(\frac{m\left(i+\frac{1}{2}\right)\pi}{N}\right)$, defined in table 2.2, with $\xi(m)$ defined in eq. 2.13.

Each element of the pixel-domain sequence $x(i)$ can be reconstructed using the IDCT as follows:

$$x(i) = \sum_{m=0}^{N-1} \mathcal{C}_{2e}(m,i)X(m). \tag{2.70}$$

From the definition of the pixel-wise multiplication (eq. 2.55) and by using eq. 2.70, each $h(i)$ element can be expressed as:

$$h(i) = f(i) \odot g(i) = \sum_{m_1=0}^{N-1} \sum_{m_2=0}^{N-1} \mathcal{C}_{2e}(m_1,i)\ \mathcal{C}_{2e}(m_2,i)\ F(m_1)\ G(m_2), \tag{2.71}$$

where the vectors $\mathbf{F}$ and $\mathbf{G}$ are the discrete cosine transforms of $\mathbf{f}$ and $\mathbf{g}$, respectively. Since only these transform data sequences are actually available from a video compressed bitstream, there is a considerable interest in computing $\mathbf{H}$ directly from $\mathbf{F}$ and $\mathbf{G}$. The discrete cosine transform of $h(i)$, expressed as $H(m)$, is stated from eq. 2.69 as follows:

$$H(m) = \sum_{i=0}^{N-1} \mathcal{C}_{2e}(m,i)h(i) \tag{2.72}$$

$$= \sum_{i=0}^{N-1} \mathcal{C}_{2e}(m,i) \left(\sum_{m_1=0}^{N-1} \sum_{m_2=0}^{N-1} \mathcal{C}_{2e}(m_1,i)\ \mathcal{C}_{2e}(m_2,i)\ F(m_1)\ G(m_2) \right) \tag{2.73}$$

By performing some simple manipulations, eq. 2.73 can be expressed as:

$$H(m) = \sum_{m_1=0}^{N-1} \sum_{m_2=0}^{N-1} W(m,m_1,m_2)\ F(m_1)\ G(m_2), \tag{2.74}$$

where:

$$W(m,m_1,m_2) = \sum_{i=0}^{N-1} \mathcal{C}_{2e}(m,i)\mathcal{C}_{2e}(m_1,i)\mathcal{C}_{2e}(m_2,i). \tag{2.75}$$

Hence, eq. 2.74 expresses the DCT-domain convolution operation corresponding to the spatial-domain pixel-wise multiplication of eq. 2.55. Nevertheless, by comparing these two approaches in what concerns the computational cost, the pixel-domain approach seems to require significantly less operations ($\mathcal{O}(N)$) than the DCT-domain counterpart, which implies a computational cost proportional to $\mathcal{O}(N^3)$ (disregarding the cost of performing two IDCTs and one direct DCT in the pixel-domain approach). However, by considering that many of the DCT coefficients of typical compressed video streams are zero, the DCT-domain convolution between $\mathbf{F}$ and $\mathbf{G}$ requires, in practice, only $N_F \times N_G \times N$ multiplications, where N_F and N_G represent the number of nonzero coefficients in $\mathbf{F}$ and $\mathbf{G}$, respectively. Hence,

since only the nonzero DCT coefficients need to be used in the convolution process, the convolution operation in the DCT-domain may have, in practice, a lower computational cost than the spatial-domain approach.

Besides these observations, Shen et al. [100] have shown that the computation cost inherent to this operation can still be significantly reduced, by taking advantage of the sparse characteristics of the $W(m, m_1, m_2)$ matrix. They proved that each cross product between any two arbitrary elements of $\mathbf{F}$ and $\mathbf{G}$ (e.g.: $F(m_1)$ and $G(m_2)$), contributes to no more than two elements $H(a)$ and $H(b)$ of the resulting DCT output vector $\mathbf{H}$, where these indexes a and b are given by:

$$a = \begin{cases} m_1 + m_2 & , if\ m_1 + m_2 < N \\ 2N - (m_1 + m_2) & , if\ m_1 + m_2 > N \\ \emptyset\ \text{(empty)} & , if\ m_1 + m_2 = N \end{cases} \tag{2.76}$$

and

$$b = |m_1 - m_2|. \tag{2.77}$$

The justification for this fact comes from the orthogonality property of the DCT, stated as:

$$\sum_{m=0}^{N-1} \mathcal{C}_{2e}(m, i) \,.\, \mathcal{C}_{2e}(m, j) = \begin{cases} 1 & , \ i = j \\ 0 & , \ i \neq j \end{cases} . \tag{2.78}$$

In fact, from eq. 2.75,

$$W(m, m_1, m_2) = \sum_{i=0}^{N-1} \mathcal{C}_{2e}(m, i) \left[\mathcal{C}_{2e}(m_1, i)\ \mathcal{C}_{2e}(m_2, i)\right]. \tag{2.79}$$

By using simple trigonometric relations, the term $\mathcal{C}_{2e}(m_1, i)\ \mathcal{C}_{2e}(m_2, i)$ of the equation above can be formulated as follows:

$$\begin{aligned} \mathcal{C}_{2e}(m_1, i)\ \mathcal{C}_{2e}(m_2, i) &= \sqrt{\frac{2}{N}}\ \xi\left(m_1\right) \cos\left(\frac{m_1\left(i + \frac{1}{2}\right)\pi}{N}\right) \times \\ &\quad \sqrt{\frac{2}{N}}\ \xi\left(m_2\right) \cos\left(\frac{m_2\left(i + \frac{1}{2}\right)\pi}{N}\right) \qquad (2.80) \\ &= \frac{\xi\left(m_1\right)\xi\left(m_2\right)}{N} \cos\left(\frac{\left(m_1 + m_2\right)\left(i + \frac{1}{2}\right)\pi}{N}\right) + \\ &\quad \frac{\xi\left(m_1\right)\xi\left(m_2\right)}{N} \cos\left(\frac{\left(m_1 - m_2\right)\left(i + \frac{1}{2}\right)\pi}{N}\right) \qquad (2.81) \\ &= \frac{\xi\left(m_1\right)\xi\left(m_2\right)}{N} \left[\frac{\mathcal{C}_{2e}(m_1 + m_2, i)}{\sqrt{\frac{2}{N}}\ \xi\left(m_1 + m_2\right)} + \frac{\mathcal{C}_{2e}(m_1 - m_2, i)}{\sqrt{\frac{2}{N}}\ \xi\left(m_1 - m_2\right)}\right] \qquad (2.82) \end{aligned}$$

$$= \sqrt{\frac{1}{2N}}\,\frac{\xi(m_1)\,\xi(m_2)}{\xi(m_1+m_2)}\,\mathcal{C}_{2e}(m_1+m_2,i)+ \sqrt{\frac{1}{2N}}\,\frac{\xi(m_1)\,\xi(m_2)}{\xi(m_1-m_2)}\,\mathcal{C}_{2e}(m_1-m_2,i) \tag{2.83}$$

Hence, eq. 2.79 can be written as:

$$W(m,m_1,m_2) = \sqrt{\frac{1}{2N}}\,\frac{\xi(m_1)\,\xi(m_2)}{\xi(m_1+m_2)}\,\sum_{i=0}^{N-1}\mathcal{C}_{2e}(m,i)\mathcal{C}_{2e}(m_1+m_2,i) + \sqrt{\frac{1}{2N}}\,\frac{\xi(m_1)\,\xi(m_2)}{\xi(m_1-m_2)}\,\sum_{i=0}^{N-1}\mathcal{C}_{2e}(m,i)\mathcal{C}_{2e}(m_1-m_2,i) \tag{2.84}$$

By taking eq. 2.78 into account, one concludes that the first term on the right side of eq. 2.84 is non-null only when $m = m_1 + m_2$. Similarly, the second term will be nonzero when $m = m_1 - m_2$. However, since $\cos(m_1 - m_2) = \cos(m_2 - m_1)$, one realizes that this term will be nonzero whenever $m = |m_1 - m_2|$. A detailed proof of this formulation can be found in [100].

Shen et al. also proposed an efficient algorithm for computing $H(m)$, by observing that the entries in $W(m, m_1, m_2)$ take only three possible nonzero values: $\sqrt{\frac{1}{N}}$ and $\pm\sqrt{\frac{1}{2N}}$.

1) if $\underline{m_1 = 0 \text{ and } m_2 = 0}$ then $m = (m_1 + m_2) = |m_1 - m_2| = 0$. From eq. 2.78 and eq. 2.84, it comes that:

$$W(m,m_1,m_2) = \frac{1}{\sqrt{2N}}\,\frac{\sqrt{1/2}.\sqrt{1/2}}{\sqrt{1/2}}.1 + \frac{1}{\sqrt{2N}}\,\frac{\sqrt{1/2}.\sqrt{1/2}}{\sqrt{1/2}}.1 \tag{2.85}$$

$$= 2\,\frac{1}{\sqrt{2N}}\,\sqrt{1/2} = \sqrt{\frac{1}{N}} \tag{2.86}$$

2) if $\underline{m_1 = 0 \text{ or } m_2 = 0}$, it comes that $m = (m_1 + m_2) = |m_1 - m_2| \neq 0$ and

$$W(m,m_1,m_2) = \frac{1}{\sqrt{2N}}\,\frac{1.\sqrt{1/2}}{1}.1 + \frac{1}{\sqrt{2N}}\,\frac{1.\sqrt{1/2}}{1}.1 = \sqrt{\frac{1}{N}} \tag{2.87}$$

3) if $\underline{m_1 \neq 0 \text{ and } m_2 \neq 0}$, it comes that $(m_1 + m_2) \neq |m_1 - m_2|$. Hence, two different cases should be taken into account: $m = a$ or $m = b$ (see eqs. 2.76 and 2.77):

$\underline{m = a}$: • if $m = (m_1 + m_2) < N$, then:

$$W(m,m_1,m_2) = \frac{1}{\sqrt{2N}}\,\frac{1.1}{1}.1 + 0 = \sqrt{\frac{1}{2N}} \tag{2.88}$$

- if $(m_1+m_2) > N$, then $m = 2N-(m_1+m_2)$; considering that $\mathcal{C}_{2e}(m_1+m_2, i) = -\mathcal{C}_{2e}(2N-(m_1+m_2), i)$, it comes:

$$W(m, m_1, m_2) = \frac{1}{\sqrt{2N}} \frac{1.1}{1}.(-1)+0 = -\sqrt{\frac{1}{2N}} \tag{2.89}$$

$\underline{m = b}$: • if $m = |m_1 - m_2| \neq 0$, then:

$$W(m, m_1, m_2) = 0 + \frac{1}{\sqrt{2N}} \frac{1.1}{1}.1 = \sqrt{\frac{1}{2N}} \tag{2.90}$$

- if $m = |m_1 - m_2| = 0$, then:

$$W(m, m_1, m_2) = 0 + \frac{1}{\sqrt{2N}} \frac{1.1}{\sqrt{1/2}}.1 = \sqrt{\frac{1}{2N}} \tag{2.91}$$

The weighting factors $W(a, m_1, m_2)$ and $W(b, m_1, m_2)$ for each cross product $F(m_1)G(m_2)$, applied in the computation of the two elements $H(a)$ and $H(b)$ of the resulting DCT output vector $H(m)$ (see eqs. 2.76 and 2.77) are shown in table 2.6, with $\alpha = \sqrt{1/N}$ and $\beta = \sqrt{1/(2N)}$. As it was shown, from the previous description, when either m_1 or m_2 are zero, the product $F(m_1)G(m_2)$ contributes to exactly one output component. Otherwise, at most two components are required in the evaluation.

By adopting this technique, the computational cost is now reduced to $N_F \times N_G \times N_W$ multiplications, with N_W representing the number of nonzero $W(m, m_1, m_2)$ elements for each fixed m (1 or 2). Actually, this computational cost can be further reduced by exploiting the symmetry property of the DCT (in particular, of the $W(m, m_1, m_2)$ matrix) in the mapping of m_1, m_2 to m, to reduce the number of multiplications required to compute eq. 2.74. In fact, eq. 2.74 can be computed as follows [100]:

$$\begin{aligned} H(m) \;=\; & \sum_{m_1=0}^{N-2} \sum_{m_2=m_1+1}^{N-1} W(m, m_1, m_2)\,[F(m_1)\; G(m_2) + F(m_2)\; G(m_1)] \\ & + \sum_{m_1=0}^{N-1} W(m, m_1, m_1)\; F(m_1)\; G(m_1) \end{aligned} \tag{2.92}$$

Hence, if $F(m_1)G(m_2)$ and $F(m_2)G(m_1)$ are both nonzero, their multiplications with the weighting factor $W(m, m_1, m_2)$ can still be merged, thus reducing the amount of required multiplications.

The procedure described above to fasten the computation of the convolution operation in the DCT-domain can be extended to the 2-D case using an entirely

Table 2.6: Weighting factors $W(a, m_1, m_2)$ and $W(b, m_1, m_2)$, with $N = 8$ and a and b given by eqs. 2.76 and 2.77.

m_1	m_2							
	0	1	2	3	4	5	5	7
0	α, α	α, α	α, α	α, α	α, α	α, α	α, α	α, α
1	α, α	β, α	β, β	β, β	β, β	β, β	β, β	$-$, β
2	α, α	β, β	β, α	β, β	β, β	β, β	$-$, β	-β, α
3	α, α	β, β	β, β	β, α	β, β	$-$, β	-β, α	-β, α
4	α, α	β, β	β, β	β, β	$-$, α	-β, α	-β, α	-β, α
5	α, α	β, β	β, β	$-$, β	-β, α	-β, α	-β, α	-β, α
6	α, α	β, β	$-$, β	-β, α	-β, α	-β, α	-β, α	-β, α
7	α, α	$-$, β	-β, α	-β, α	-β, α	-β, α	-β, α	-β, α

similar procedure. It can be shown [100] that, in such a case, each element of the **W** matrix takes on one of five possible nonzero values: $\frac{1}{N}$, $\pm\frac{1}{2N}$ and $\pm\frac{\sqrt{2}}{2N}$. Each cross product contributes to no more than four output values. As in the 1-D case, this fast algorithm provides the means to significantly reduce the computational load: instead of $16N^4$ multiplications, only about $4N^4$ operations are now required.

2.6 Conclusion

In this chapter it was presented a general overview of the main discrete trigonometric transforms, with a special emphasis to the presentation of the main properties of the even type-II discrete cosine transform. This transform has been particularly adopted in image and video standards due to its high suitability to exploit inter-pixel redundancies, rendering excellent decorrelation for most image data. In fact, since this DCT packs the energy content in a reduced number of low frequency coefficients, it provides the capability to discard some high frequency coefficients without significantly degrading the output image quality. This implicit coarse quantization scheme provides a significant reduction of the resulting signal entropy and a consequent reduction of the average number of required bits to encode each pixel.

The formulation of many of the presented properties was based on the boundary characteristics of the data sequences under processing and on the implicit symmetric extensions of those sequences outside the original domain.

Among the several properties that have been presented, a special attention was devoted to the formalization of the multiplication-convolution property. This prop-

erty will be applied by the static video composition operations, that will be presented in chapter 4, and concerns the relation between the convolution operation, implemented in the transform-domain, and the corresponding pixel-wise multiplication of two blocks of pixels.

References

[2] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Transactions on Computers*, vol. C-23, no. 1, pp. 90–93, 1974.

[7] V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards: Algorithms and Architectures*, 2nd ed. Kluwer Academic Publishers, Jun. 1997.

[8] W.-K. Cham, "Family of order-4 four-level orthogonal transforms," *IEE Electronic Letters*, vol. 19, no. 21, pp. 869–871, Oct. 1983.

[11] S.-F. Chang and D. G. Messerschmitt, "Manipulation and compositing of MC-DCT compressed video," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 1, pp. 1–11, Jan. 1995.

[22] A. Haar, "Zur theorie der orthogonalen funktionen-systeme [on the theory of orthogonal function systems]," *Mathematische Annalen*, no. 69, pp. 331–371, 1910.

[23] H. Hedberg and P. Nilsson, "A survey of various discrete transforms used in digital image compression algorithms," in *Proceedings of the Swedish System-On-Chip Conference*, Bastad - Sweden, Apr. 2004.

[26] *MPEG-1: ISO/IEC JTC1 CD 11172 - "Coding of moving pictures and associated audio for digital storage media up to 1.5 Mbit/s – Part 2: Video"*, ISO, 1992.

[27] *MPEG-2: ISO/IEC JTC1 CD 13818 - "Generic coding of moving pictures and associated audio – Part 2: Video"*, ISO, 1994.

[28] *MPEG-4: ISO/IEC 14496-2:2004. Information technology – Coding of audio-visual objects – Part 2: Visual*, ISO, 2004.

[29] *ITU-T Recommendation H.261 - "Video Codec for Audiovisual Services at* $p\times64$ *Kbit/s"*, ITU-T, Mar. 1993.

[30] *ITU-T Recommendation H.263 - "Video Coding for Low Bitrate Communication"*, ITU-T, Feb. 1998.

[31] *ITU-T Recommendation H.264, "Advanced Video Coding for Generic Audiovisual Services"*, ITU-T, May 2003.

[32] *JPEG: ITU-T Recommendation T.81 - "Digital compression and coding of continuous-tone still images"*, ITU-T, 1993.

[33] *ITU-T/SG16/VCEG (Q.6), H.26L Test Model Long-Term Number 8 (TML-8)*, ITU-T, Video Coding Experts Group (VCEG), Sep. 2001.

[34] A. K. Jain, *Fundamentals of Digital Image Processing.* Prentice Hall, 1989.

[44] R. Kresch and N. Merhav, "Fast DCT domain filtering using the DCT and the DST," *IEEE Transactions on Image Processing*, vol. 8, no. 6, pp. 821–833, Jun. 1999.

[53] J. S. Lim, *Two-Dimensional Signal and Image Processing.* Prentice-Hall, 1990.

[58] S. A. Martucci, "Symmetric convolution and discrete sine and cosine transforms," *IEEE Transactions on Signal Processing*, vol. SP-42, no. 5, pp. 1038–1051, May 1994.

[76] B. Porat, *A Course in Digital Signal Processing.* John Wiley & Sons, Inc., 1997.

[78] W. K. Pratt, *Digital Image Processing.* John Wiley & Sons, Inc., 1978.

[79] W. K. Pratt, J. Kane, and H. C. Andrews, "Hadamard transform image coding," *Proceedings of the IEEE*, vol. 57, no. 1, pp. 58–68, Jan. 1969.

[80] W. K. Pratt, W.-H. Chen, and L. R. Welch, "Slant transform image coding," *IEEE Transactions on Communications*, vol. 22, no. 8, pp. 1075–1093, Aug. 1974.

[81] M. Püschel and J. M. F. Moura, "The algebraic approach to the discrete cosine and sine transforms and their fast algorithms," *Society for Industrial and Applied Mathematics Journal on Computing*, vol. 32, no. 5, pp. 1280–1316, 2003.

[82] K. R. Rao and P. Yip, *Discrete Cosine Transform: algorithms, advantages and applications.* Academic Press, Inc., 1990.

[100] B. Shen, I. K. Sethi, and V. Bhaskaran, "DCT convolution and its application in compressed domain," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 8, pp. 947–952, Dec. 1998.

[104] B. C. Smith and L. A. Rowe, "Algorithms for manipulating compressed images," *IEEE Computer Graphics and Applications*, pp. 34–42, Sep. 1993.

[105] G. Strang, "The discrete cosine transform," *Society for Industrial and Applied Mathematics Review*, vol. 41, no. 1, pp. 135–147, 1999.

[110] Z. Wang, "Fast algorithms for the discrete W transform and for the discrete Fourier transform," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 4, pp. 803–816, Aug. 1984.

3

Video Transcoding in the DCT-Domain

Contents

3.1 Introduction

As it was referred in the previous chapters, the main reasons for transform-domain video transcoding techniques have become increasingly popular in the last few years are related to the inherent advantages that they provide, both in terms of the computational efficiency and of the obtained video quality.

3.1.1 Computational efficiency

The increased computational efficiency levels provided by transform-domain processing algorithms result from the smaller number of arithmetic operations that are required to perform the same video processing task, when compared with the traditional pixel-domain transcoding approaches.

However, this computational advantage is not always evident, since many transform-domain algorithms are clearly distinct from their pixel-domain counterparts. Nevertheless, they all share a common advantage: they do not require the implementation of both the direct and inverse DCTs. By considering that two matrix multiplications are required for each of these operations, and that $N^2(N-1)$ additions and N^3 multiplications are required to perform each $(N \times N)$ matrix multiplication, these direct and inverse transforms impose a fixed overhead of about $4(N-1)$ additions and $4N$ multiplications in the processing of each pixel, when pixel-domain processing algorithms are used.

3.1.2 Reduced influence of degradation effects

The main advantages of the transform-domain processing algorithms in terms of the obtained video quality result as a consequence of the absence of degradation effects that are directly introduced during the computation of the direct and inverse DCTs. Such degradation usually emerges from round-off errors that are introduced by the usage of finite precision arithmetic. Hence, the influence of this degradation effect will be enforced by every DCT/IDCT processing modules at the several transcoding nodes that process the video encoded signal, since the primary encoding device until the final decoder-end of the whole video transcoding system. Consequently, its influence will be greatly aggravated when the number of transcoding nodes, between the primary encoder and the final decoder, increases.

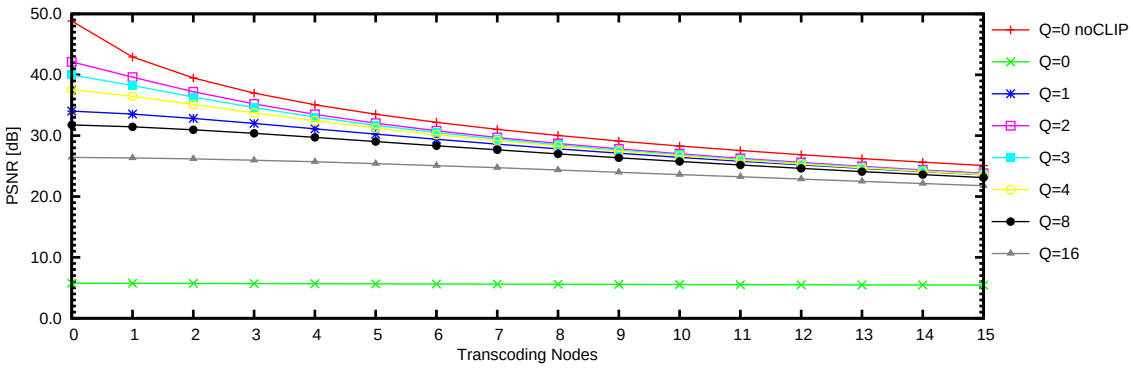
This degradation effect will also accumulate with the result of other degradation sources that are inherent to the video encoding process, such as the quantization and the clipping of the DCT coefficients. In particular, the clipping effect is usually imposed by certain video standards that confine the reconstruction levels of all

coefficients, other than the INTRA DC coefficients, to an interval between -2048 and 2047, represented with 12 bits (e.g.: H.263 [30]). All together, these sources of degradation will contribute to the introduction of a gradual distortion along the processing chain.

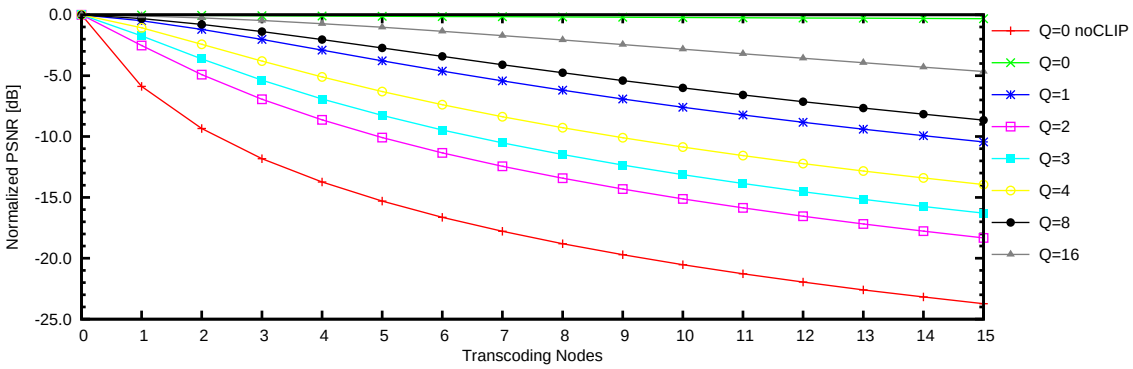
To illustrate this phenomenon, fig. 3.1 presents the Peak Signal-to-Noise Ratio (PSNR) degradation effect introduced in an INTRA type encoded frame, from the *Mobile & Calendar* video sequence. The processed frames were obtained using several different quantization setups of a H.263 [30] video encoder, by passing it through successive transcoding nodes, since its primary encoder until the final decoder. The selection of this particular video sequence was due to its characteristics in terms of the amount of spatial detail, since this feature is specially affected by this type of degradation.

Two different setups were considered for the particular case of a null quantizer ($Q = 0$):

- Full-precision, *without* the clipping effect of the transmitted DCT coefficients imposed by the H.263 video standard [30] (Legend Key '`Q=0 noCLIP`');



(a) Obtained PSNR measures.



(b) Normalized PSNR measures.

Figure 3.1: Degradation effect in pixel-domain transcoding nodes.

- Full-precision, *with* the clipping effect in the transmitted DCT coefficients (Legend Key '`Q=0`').

To eliminate any remaining influence of the quantizer modules over the obtained video quality, this null quantizer was also applied to the DC coefficients of all INTRA blocks. This violation of this video standard rules was only considered for these two quantization setups ($Q = 0$).

The first of these two setups (`Q=0 noCLIP`), characterized by the absence of any quantization or clipping effect, clearly demonstrates the degradation effect introduced by the usage of fixed-point arithmetic for computing the direct and inverse DCTs. Such effect is even more noticeable in encoded frames with high quality levels and may lead to significant losses in the first nodes of the cascaded transcoding system, reaching magnitudes of about 5 dB per node.

The second setup (`Q=0`) illustrates the effect of clipping in the obtained video quality in a situation without any quantization in the video encoding setup. The extremely low video quality level (about 5.5 dB) that was obtained with this configuration arises from the fact that most DCT coefficients exceed the 12-bit dynamic range supported by the considered video standard [30]. This fact clearly illustrates the strict restrictions imposed by the considered standard to encode lossless video data.

The remaining plots, presented in the charts of fig. 3.1, illustrate the degradation effect introduced by the several cascaded transcoding nodes in situations with non-null quantizers. Although this degradation cannot be directly attributed to the lack of precision in the arithmetic operations that are performed in the computation of the direct and inverse DCT, the following aspects can be observed:

- The clipping effect is still noticeable in transcoding setups using small non-null quantization steps, such as $Q = 1$; this gives rise to higher degradation effects than those that are obtained with schemes using greater quantization steps (e.g. $Q = 2, 3, 4$); this was already observed in the setup with $Q = 0$.

- From the chart presented in fig. 3.1(b), which plots the normalized PSNR measures in terms of their maximum value, obtained at the output of the first encoder, it can be observed that the relative influence of the degradation effect is more significant in video sequences encoded with a greater PSNR level; this was also observed for the setup with '`Q=0 noCLIP`' and illustrates the influence of the quantization procedure not only of the INTRA DC coefficients (with a fixed step size of 8 ($Q = 4$)) but also of the remaining AC coefficients, on

the degradation level that is introduced by transmitting the encoded frame through several cascaded transcoding nodes.

In the following sections, it will be presented a brief overview of the main video transcoding architectures and algorithms that have been proposed in the last few years. By entirely performing the processing of the incoming video sequences in the compressed DCT-domain, most of these transcoding systems demonstrate great advantages both in terms of the computational efficiency and of the resulting video quality.

3.2 Video transcoding architectures

As it was referred before, video transcoding can be regarded as a process of changing or converting a pre-compressed bit stream into another bit stream, with different coding characteristics.

As an example, a simple drift-free transcoding process to reduce the bit rate of a precoded video sequence can be simply carried out by applying the operations depicted in fig. 3.2: decoding of the original bit stream into its reconstructed pixels and re-encoding with a coarser quantization step. However, apart from being computationally expensive, such cascaded architectures make the transcoding process rather slow and difficult to be implemented in real-time. Although the effect of such drawback may be negligible in most video storage applications, it is a crucial issue in networking and real-time applications. As a consequence, other more efficient structures have been proposed in the last few years to implement transcoding tasks in the pixel-domain, in the transform-domain [6] or even in a hybrid pixel/transform-domain [90].

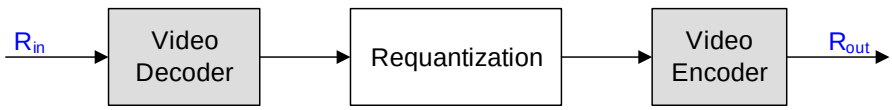


Figure 3.2: Cascaded transcoder.

3.2.1 Pixel-domain transcoding architectures

The most straightforward drift-free transcoding architectures are implemented in the pixel-domain, by cascading a video decoder and a video encoder, as it is illustrated in fig. 3.3. As it was previously seen, such architectures are frequently adopted to reduce the bit rate of a given precoded bit stream, by considering coarser

quantization step sizes in the encoding part of the transcoder. These architectures may also be used to change the video standard that is adopted by the output sequence, by considering distinct syntax codes in the VLD and the VLC modules. The motion estimation block of the encoder side is frequently absent in such architectures, since the motion vectors of the incoming bit stream are usually re-used, instead of computing new ones. These motion vectors are then applied to implement the motion compensation prediction mechanism considering the previous and future frames, stored in the corresponding frame memories (denoted as "Ref. Mem. P" and "Ref. Mem. F", in fig. 3.3) of both the decoding and encoding sides of the transcoder architecture. Such simplification is often a very significant step towards a simple and fast architecture, since motion estimation is undoubtedly the most complex processing step of the video coding algorithm. For the same reason, the GOP structure of the input video stream is also kept unchanged for the sake of simplicity. Otherwise, it would be necessary to reorder the picture sequence, which would introduce a delay of several pictures and would make the implementation and application of those architectures unsuited for real-time and low latency transcoding applications.

In the following sections, it will be analyzed the transcoding process of three different types of coded frames: Intra (I), Predicted (P) and Bidirectionally Interpolated (B). For the sake of illustration, it will be considered the above referred transcoder architecture to reduce the bit rate of a given precoded bit stream, by adopting a coarser quantization step size $Q_2 > Q_1$ in the encoding part of the transcoder. As it will be described in section 3.3.2, several different approaches may be devised to select the optimum coarser quantization step size Q_2 to be adopted in the encoding part of the transcoder. Nevertheless, the following description will only focus on the aspects concerning to the transcoder architecture point of view.

It is worth noting that the block diagram of the cascaded structures illustrated in fig. 3.3, composed by a pixel-domain decoder and encoder, may be wholly or

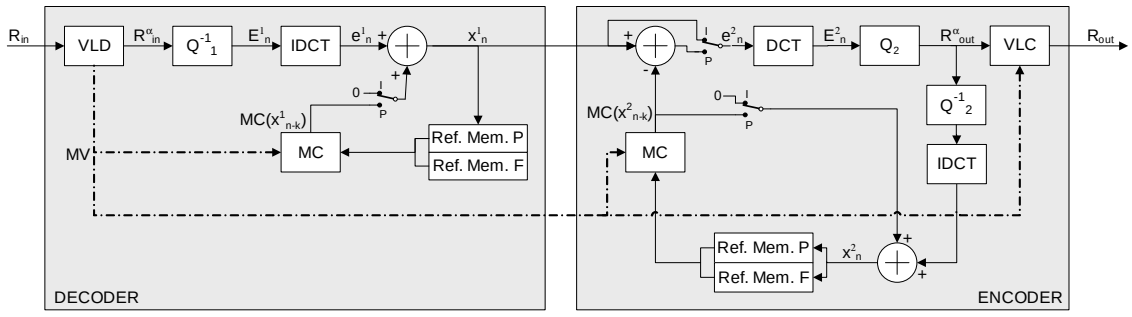


Figure 3.3: Typical pixel-domain cascaded transcoder architecture.

partially used to process these three types of frames. While B type frames employ all modules of the codec, P type frames make use of only one of the two reference frame buffers in the feedback loop, and I type frames do not use the feedback loop at all. In the following, $\mathbf{R}_{in}$ and $\mathbf{R}_{out}$ will denote the input and output streams, respectively. To represent the input and output streams for each type of frame, the representations $\mathbf{R}_{in}^{\alpha}$ and $\mathbf{R}_{out}^{\alpha}$ will be adopted, with $\alpha = I,\ P,\ B$. The inverse quantization operations will be denoted as Q^{-1}.

A - Intra (I) type frames

The transcoding process of I type frames is conducted by coarsely encoding the decoded pictures $\mathbf{x}_n^1$ (see fig. 3.3) with $Q_2 > Q_1$. As a consequence, the output sequence for this type of pictures $\mathbf{R}_{out}^{I}$ is given by:

$$\mathbf{R}_{out}^{I} = Q_2 \left[\mathrm{DCT} \left(\mathbf{x}_n^1 \right) \right], \tag{3.1}$$

where $\mathbf{x}_n^1$ is given by:

$$\mathbf{x}_n^1 = \mathrm{IDCT} \left[Q_1^{-1} \left(\mathbf{R}_{in}^{I} \right) \right]. \tag{3.2}$$

By substituting eq. 3.2 in eq. 3.1, the transcoding equation for I type frames can be formulated:

$$\mathbf{R}_{out}^{I} = Q_2 \left[Q_1^{-1} \left(\mathbf{R}_{in}^{I} \right) \right]. \tag{3.3}$$

It should be noted that since many video standards apply a fixed quantization step of 8 to encode the DC coefficient of each INTRA coded block, these equations only apply to the AC DCT coefficients of each encoded block.

B - Predicted (P) type frames

According to the block diagram of the cascaded pixel-domain transcoder, illustrated in fig. 3.3, the P type frames are accumulated in both Motion Compensation (MC) loops, as they are used to decode and encode consecutive frames. In fact, if the n^{th} incoming picture is of type P, then the $(n - M_B)^{th}$ previously decoded anchor frame is required to reconstruct the current frame, where M_B is the distance between anchor pictures. Such anchor is usually the last decoded P or I type frame. Hence, the input and output prediction errors $\mathbf{e}_n^1$ and $\mathbf{e}_n^2$, respectively, are related to the input and output bit streams $\mathbf{R}_{in}^{P}$ and $\mathbf{R}_{out}^{P}$ as follows:

$$\mathbf{e}_n^1 = \mathrm{IDCT} \left[Q_1^{-1} \left(\mathbf{R}_{in}^{P} \right) \right] \tag{3.4}$$

$$\mathbf{R}_{out}^{P} = Q_2 \left[\mathrm{DCT} \left(\mathbf{e}_n^2 \right) \right]. \tag{3.5}$$

The decoded frame $\mathbf{x}_n^1$ and the corresponding prediction error $\mathbf{e}_n^2$ are given by:

$$\mathbf{x}_n^1 = \mathbf{e}_n^1 + \mathrm{MC}\left(\mathbf{x}_{n-M_B}^1\right) \tag{3.6}$$

$$\mathbf{e}_n^2 = \mathbf{x}_n^1 - \mathrm{MC}\left(\mathbf{x}_{n-M_B}^2\right), \tag{3.7}$$

where $\mathbf{x}_{n-M_B}^2$ is the previous anchor frame, decoded after coarser quantization, and MC is the motion compensation operation for P type frames. By substituting eq. 3.6 into 3.7, the prediction error $\mathbf{e}_n^2$ is computed as follows:

$$\mathbf{e}_n^2 = \mathbf{e}_n^1 + \mathrm{MC}\left(\mathbf{x}_{n-M_B}^1\right) - \mathrm{MC}\left(\mathbf{x}_{n-M_B}^2\right). \tag{3.8}$$

According to eq. 3.8, the prediction error $\mathbf{e}_n^2$ can be obtained by adding the input prediction error e_n^1 to the difference between the input and output motion-compensated anchor frames. This difference can be regarded as the *transcoding error* that is introduced in each anchor picture. Since the motion vectors that are used by both motion compensated loops are considered to be the same, this difference can be calculated prior to the accumulation:

$$\mathbf{e}_n^2 = \mathbf{e}_n^1 + \mathrm{MC}\left(\mathbf{x}_{n-M_B}^1 - \mathbf{x}_{n-M_B}^2\right). \tag{3.9}$$

It should be noted, however, that even when the considered motion vectors are the same, the previous simplification may introduce some distortion in the overall transcoding process. In fact, if one takes into account that the motion compensation function is not a linear operation, as a consequence of the round-off errors inherent to integer truncation, it is easily observed that:

$$\mathrm{MC}\left(\mathbf{a}_n\right) - \mathrm{MC}\left(\mathbf{b}_n\right) \neq \mathrm{MC}\left(\mathbf{a}_n - \mathbf{b}_n\right). \tag{3.10}$$

Even so, Assunção and Ghanbari [5] have shown that no significant drift error is introduced when such simplification is considered and only one frame buffer is used to accumulate the transcoding error. Hence, by substituting eq. 3.9 into eq. 3.5 and by taking into account the linear property of the DCT, one can obtain:

$$\mathbf{R}_{out}^P = Q_2\left[\mathbf{E}_n^1 + \mathrm{DCT}\left(\mathrm{MC}\left(\mathbf{x}_{n-M_B}^1 - \mathbf{x}_{n-M_B}^2\right)\right)\right], \tag{3.11}$$

where $\mathbf{E}_n^1 = Q_1^{-1}\left(\mathbf{R}_{in}^P\right)$. According to this equation, it can be observed that for transcoding a P type frame, the accumulated transcoding error has to be added to the incoming DCT coefficients and then coarsely quantized.

C - Bidirectionally interpolated (B) type frames

The transcoding procedure for B type frames is entirely similar to the processing of P type pictures. In fact, the only difference is that the motion compensated prediction makes use of two reference anchor frames: one in the past and other in the future. As a consequence, eq. 3.11 should be modified as:

$$\mathbf{R}_{out}^{B} = Q_2\left[\mathbf{E}_n^1 + \mathrm{DCT}\left(\mathrm{MC}\left(\mathbf{x}_p^1 - \mathbf{x}_p^2, \mathbf{x}_f^1 - \mathbf{x}_f^2\right)\right)\right], \tag{3.12}$$

where the indices p and f denote the past and future anchor frames, respectively. As a consequence, two frame memories are needed, one for each anchor frame. Moreover, the accumulated transcoding errors for both frames have to be taken into account, in order to keep track of the drift in B type frames. It should be also noted, however, that since B type frames are not used as references for further prediction in most DCT-H.26x/MPEG-x video standards, the resulting output picture $\mathbf{x}_b^2$ does not need to be stored in such buffers. Hence, only the corresponding prediction difference $\mathbf{R}_{out}^{b}$ needs to be supplied to the output of the transcoding system.

D - Reduced computational cost of pixel-domain architectures

The transcoding equations corresponding to the processing of I, P and B type frames (see eqs. 3.3, 3.11 and 3.12) can be efficiently implemented, provided that the transcoder architecture presented in fig. 3.3 is re-designed in order to only implement the computational blocks that are really needed to fulfill the processing - see fig. 3.4. In this architecture, the transcoding error is directly computed in the DCT-domain. As it was referred before, such simplification can be performed by taking into account the linearity property of the DCT, and allows the use of only one pair of DCT/IDCT operations. Moreover, by re-using the decoded motion vectors in just one reconstruction loop, this architecture is significantly more computational efficient than the one presented in fig. 3.3.

It should be also noted that the DCT and IDCT processing blocks included in the transcoder architecture illustrated in fig 3.4 are only necessary because the motion compensation function is defined as a pixel-domain operation. As a consequence, this transcoder architecture is defined as a *pixel-domain* transcoder. In fact, to further improve the efficiency and to reduce the computational cost of this architecture, fully frequency-domain transcoders that operate entirely in the DCT-domain have also been derived.

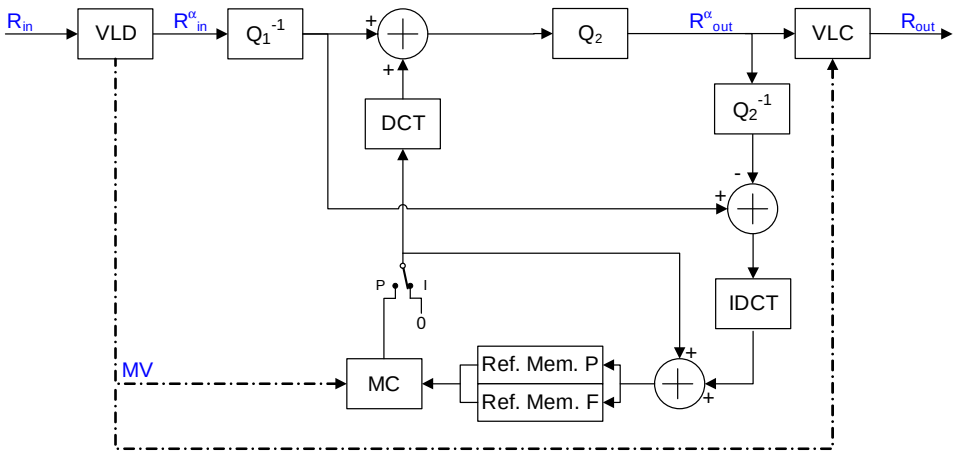


Figure 3.4: Reduced computational cost pixel-domain transcoder architecture.

3.2.2 DCT-domain transcoding architectures

As it was previously referred, if the motion compensation operation could be performed directly in the frequency-domain, there would be no need for the DCT and IDCT processing blocks of the transcoder architecture illustrated in fig. 3.4. In fact, as it will be detailed in subsection B of section 3.3.1 (page 69), such operation can be efficiently performed in the transform-domain, which provides a significant simplification of the architecture described above, as illustrated in fig. 3.5.

In the *transform-domain*, the transcoding error, given by the difference between the inverse quantized input and the inverse quantized output DCT coefficients, is added to the DCT of the current picture after Transform-Domain Motion Compensation (MC-DCT) and accumulated directly in the transform-domain.

As it will be shown in the following sections, computational and distortion advan-

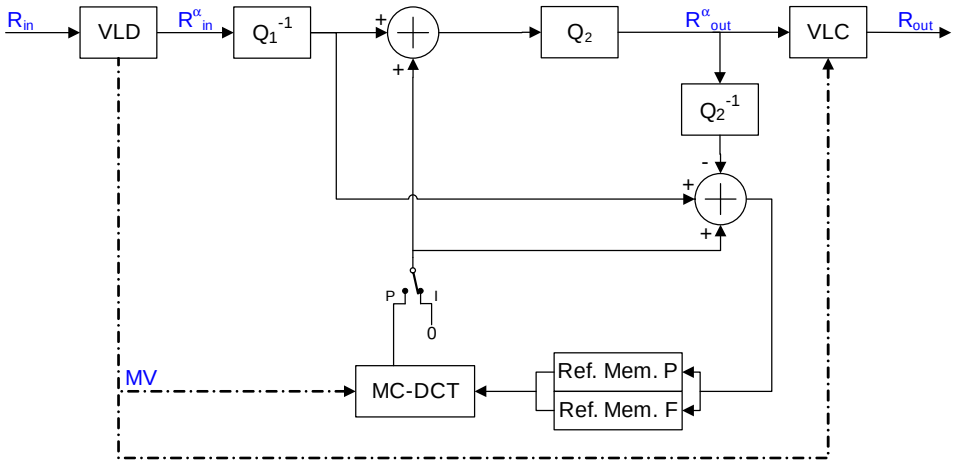


Figure 3.5: Transform-domain transcoder architecture.

tages can be obtained, by implementing the processing functions, on the incoming video sequences, directly in the transform-domain.

3.3 Video processing algorithms in the DCT-domain

To sustain a good video quality at a specified compression ratio, existing video standards compress video streams by customizing their characteristics, in order to meet a set of constraints of their target application scenarios or terminal devices. Therefore, video streams that have been compressed for one specific application are often not applicable or optimized for other scenarios, with different constraints. In such situations, delivery systems and service providers frequently face the need for further manipulations and processing of such compressed video streams. The main aim of such operations is to adapt the characteristics of the video stream not only to the adopted transmission channel but also to the characteristics of the terminal devices. Some common sources of mismatches include lower transmission bandwidths, smaller display screens, reduced computational capabilites, etc. Such mismatches often impose serious hardnesses and hinder the efficient sharing and broadcast of compressed videos among today's heterogeneous network and terminal devices [108].

To alleviate this problem, a number of scalable (or layered) video coding techniques have been proposed, in order to obtain compressed video streams that can be easily tailored to meet different application or network constraints. In layered video coding, a given stream is usually compressed into one base layer and one or more hierarchically dependent enhancement layers. While the base layer can be independently decoded to provide a pre-defined basic video quality, the several enhancement layers can be used to progressively improve the video quality whenever sufficient channel transmission capacity, as well as decoder display and computational processing resources, are available. However, most of these layered video coding techniques only provide a part of the solution for the above mentioned problems. Often, the available channel capacity is still scarce to transmit the base layer bit stream, making real-time video transmission not possible. In addition, the target devices, or even the adopted video standards, may not be capable of processing multi-layered video streams. In such situations, transcoding techniques are usually necessary and preferred.

In contrast to layered video coding techniques, transcoding methods are primarily regarded as a set of manipulation and adaptation techniques to process video

sequences. Many of these techniques allow their application directly on compressed precoded streams, thus offering significant advantages in what concerns the computational requirements and distortion level. This processing may include:

- *syntax conversion*, so that an existing precoded video can be processed by a decoder of a different video standard, such as the conversion between MPEG-2 Video and H.264/AVC or H.263 and MPEG-4 Visual standards [96];
- *spatial resolution reduction*, to allow a given video stream to be received in devices equipped with display screens with lower resolution;
- *temporal resolution reduction*, to allow devices with reduced computational capabilities to decode the video stream in real-time;
- *bit rate adjustment*, to allow real-time transmission using variable bandwidth communication channels;
- *functionality and hardware compliancy requirements*, to adjust the video stream to the functional characteristics of the target terminal devices, such as the amount of memory available at the decoder buffer, the processing resources, etc. In fact, the computational resources that are available in many terminal decoding systems, such as portable, mobile and battery supplied devices, poses severe restrictions to the computational cost of the adopted transcoding algorithms [6, 89].

To accomplish the set of different manipulations discussed above, a large number of transcoding techniques have been proposed in the literature. In particular, for DCT compressed video streams, such as DCT-H.26x/MPEG-x videos, three transcoding approaches are generally adopted:

- *bit rate and quality adaptation* [106, 107];
- *spatial resolution reduction* [16, 63, 66, 90];
- *temporal resolution reduction* [13, 115].

Depending on the specific application requirements, these methods present their own inherent advantages and disadvantages, and can even be individually or jointly applied. As an example, while frame rate reduction can better preserve video spatial details, it may introduce motion irregularities, due to frame skipping. On the other hand, frame size reduction techniques tend to preserve smooth motion at the expense of distorted video spatial details.

In the following subsections, it will be presented a brief overview of each of these techniques. Further details and some additional improvements or alternative algorithms for these techniques that are proposed in the scope of this thesis will be presented in subsequent chapters.

3.3.1 Motion compensated temporal prediction

Most video compression algorithms use temporal prediction techniques based on motion compensation to predict each $N \times N$ pixels block ($\mathbf{x}$) of the current image, by searching the best matching block ($\hat{\mathbf{x}}$) in a reference frame that maximizes a given similarity criteria. The residual signal that is obtained by subtracting the prediction block from the current image block, $\mathbf{e} = \mathbf{x} - \hat{\mathbf{x}}$, is subsequently encoded using the Discrete Cosine Transform (DCT) to obtain the prediction signal, $\mathbf{E} = \text{DCT}(\mathbf{e})$. At the decoder end, the DCT coefficients of the current block ($\mathbf{X}$) are recovered by summing up the coefficients of the selected prediction block ($\hat{\mathbf{X}}$) with the DCT coefficients of the prediction error signal ($\mathbf{E}$), that was received from the encoder:

$$\mathbf{X} = \hat{\mathbf{X}} + \mathbf{E}. \tag{3.13}$$

A - Motion-compensation in the pixel-domain

To implement the motion compensation operation in the pixel-domain, it is necessary to extract the pixel information of the prediction block ($\hat{\mathbf{x}}$) from previous or future images. This is achieved by shifting the current horizontal and vertical positions a number of pixels dictated by the previously computed motion vector (v). However, since usually neither the vertical nor the horizontal components of the motion vector are an integer multiple of the block size, there is no perfect alignment of the prediction block ($\hat{\mathbf{x}}$) with the grid defined by the several blocks ($\hat{\mathbf{x}}_i$) of the reference image (see fig. 3.6). Therefore, the area corresponding to this block is frequently composited by pixels that belong to the four neighboring blocks $\hat{\mathbf{x}}_1$, $\hat{\mathbf{x}}_2$, $\hat{\mathbf{x}}_3$ and $\hat{\mathbf{x}}_4$. As a consequence, the prediction block ($\hat{\mathbf{x}}$) is usually comprised by four separate sub-regions, corresponding to the areas that are intersected with each block $\hat{\mathbf{x}}_i$. Assuming that the region of block $\hat{\mathbf{x}}$ that is superimposed with block $\hat{\mathbf{x}}_1$ is composited by h lines and w columns, where $0 \leq h, w \leq N$, the intersections of the block $\hat{\mathbf{x}}$ with blocks $\hat{\mathbf{x}}_2$, $\hat{\mathbf{x}}_3$ and $\hat{\mathbf{x}}_4$ are composited by $h \times (N - w)$, $(N - h) \times w$ and $(N - h) \times (N - w)$ pixels, respectively.

All these sub-regions can be extracted from the corresponding blocks ($\hat{\mathbf{x}}_i$) by multiplying each of these blocks by the appropriate matrices ($\mathbf{h}_{i1}$ and $\mathbf{h}_{i2}$), that perform the separation and the displacement of the required pixel areas [11, 62].

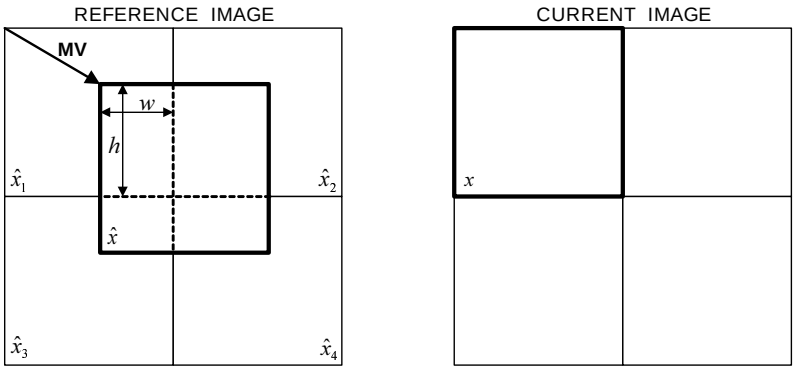


Figure 3.6: Motion compensation procedure.

The matrices pair ($\mathbf{h}_{i1}$, $\mathbf{h}_{i2}$) is defined by the number of lines and columns of the area intersected by each block $\hat{\mathbf{x}}_i$ with the block $\hat{\mathbf{x}}$. Hence, the motion compensation operation can be described as follows:

$$\hat{\mathbf{x}} = \sum_{i=1}^{4} \mathbf{h}_{i1}\ \hat{\mathbf{x}}_i\ \mathbf{h}_{i2}, \tag{3.14}$$

where matrices $\mathbf{h}_{ij}$, for $i = 1, \ldots, 4$ and $j = 1, 2$, are $N \times N$ sparse matrices whose structure is similar to the upper and lower triangular matrices $\mathbf{u}_h$ and $\boldsymbol{\ell}_w$ (see eqs. 3.15 and 3.16), where $\mathbf{I}_h$ and $\mathbf{I}_w$ are $h \times h$ and $w \times w$ identity matrixes, respectively.

$$\mathbf{h}_{11} = \mathbf{h}_{21} = \mathbf{u}_h \triangleq \begin{bmatrix} 0 & \mathbf{I}_h \\ 0 & 0 \end{bmatrix} \tag{3.15}$$

$$\mathbf{h}_{12} = \mathbf{h}_{32} = \boldsymbol{\ell}_w \triangleq \begin{bmatrix} 0 & 0 \\ \mathbf{I}_w & 0 \end{bmatrix} \tag{3.16}$$

Similarly,

$$\mathbf{h}_{31} = \mathbf{h}_{41} = \boldsymbol{\ell}_{N-h} \tag{3.17}$$

and

$$\mathbf{h}_{22} = \mathbf{h}_{42} = \mathbf{u}_{N-w}. \tag{3.18}$$

Considering that $\mathbf{u}_z = \boldsymbol{\ell}_z^{\mathrm{T}}$, $\forall z$, the previous definitions can be generalized by using only upper or lower triangular matrices:

$$\begin{aligned} \mathbf{h}_{11} &= \boldsymbol{\ell}_h^{\mathrm{T}} = \mathbf{u}_h & \mathbf{h}_{12} &= \boldsymbol{\ell}_w = \mathbf{u}_w^{\mathrm{T}} \\ \mathbf{h}_{21} &= \boldsymbol{\ell}_h^{\mathrm{T}} = \mathbf{u}_h & \mathbf{h}_{22} &= \boldsymbol{\ell}_{N-w}^{\mathrm{T}} = \mathbf{u}_{N-w} \\ \mathbf{h}_{31} &= \boldsymbol{\ell}_{N-h} = \mathbf{u}_{N-h}^{\mathrm{T}} & \mathbf{h}_{32} &= \boldsymbol{\ell}_w = \mathbf{u}_w^{\mathrm{T}} \\ \mathbf{h}_{41} &= \boldsymbol{\ell}_{N-h} = \mathbf{u}_{N-h}^{\mathrm{T}} & \mathbf{h}_{42} &= \boldsymbol{\ell}_{N-w}^{\mathrm{T}} = \mathbf{u}_{N-w} \end{aligned} \tag{3.19}$$

Hence, eq. 3.14 is given by:

$$\hat{\mathbf{x}} = \sum_{i=1}^{4} \mathbf{h}_{i1}\ \hat{\mathbf{x}}_i\ \mathbf{h}_{i2} \tag{3.20}$$

$$= \boldsymbol{\ell}_h^{\mathrm{T}}\ \hat{\mathbf{x}}_1\ \boldsymbol{\ell}_w + \boldsymbol{\ell}_h^{\mathrm{T}}\ \hat{\mathbf{x}}_2\ \boldsymbol{\ell}_{N-w}^{\mathrm{T}} + \boldsymbol{\ell}_{N-h}\ \hat{\mathbf{x}}_3\ \boldsymbol{\ell}_w + \boldsymbol{\ell}_{N-h}\ \hat{\mathbf{x}}_4\ \boldsymbol{\ell}_{N-w}^{\mathrm{T}} \tag{3.21}$$

$$= \mathbf{u}_h\ \hat{\mathbf{x}}_1\ \mathbf{u}_w^{\mathrm{T}} + \mathbf{u}_h\ \hat{\mathbf{x}}_2\ \mathbf{u}_{N-w} + \mathbf{u}_{N-h}^{\mathrm{T}}\ \hat{\mathbf{x}}_3\ \mathbf{u}_w^{\mathrm{T}} + \mathbf{u}_{N-h}^{\mathrm{T}}\ \hat{\mathbf{x}}_4\ \mathbf{u}_{N-w} \tag{3.22}$$

B - Motion-compensation in the DCT-domain

The described motion compensation operation can be directly implemented in the DCT-domain [11, 62] by taking into account the linearity property of the DCT and by applying the distributive property of matrix multiplication with respect to the DCT. By following this approach, the set of matrices $\mathbf{H}_{ij} \equiv \mathrm{DCT}(\mathbf{h}_{ij})$ can be directly used to compute the DCT of the prediction block $\hat{\mathbf{X}} \equiv \mathrm{DCT}(\hat{\mathbf{x}})$ from the DCT of the previous image blocks $\hat{\mathbf{X}}_i \equiv \mathrm{DCT}(\hat{\mathbf{x}}_i)$. Hence, a set of expressions entirely equivalent to eqs. 3.20–3.22, but defined in the DCT-domain, are stated as:

$$\hat{\mathbf{X}} = \sum_{i=1}^{4} \mathbf{H}_{i1}\ \hat{\mathbf{X}}_i\ \mathbf{H}_{i2} \tag{3.23}$$

$$= \mathbf{L}_h^{\mathrm{T}}\ \hat{\mathbf{X}}_1\ \mathbf{L}_w + \mathbf{L}_h^{\mathrm{T}}\ \hat{\mathbf{X}}_2\ \mathbf{L}_{N-w}^{\mathrm{T}} + \mathbf{L}_{N-h}\ \hat{\mathbf{X}}_3\ \mathbf{L}_w + \mathbf{L}_{N-h}\ \hat{\mathbf{X}}_4\ \mathbf{L}_{N-w}^{\mathrm{T}} \tag{3.24}$$

$$= \mathbf{U}_h\ \hat{\mathbf{X}}_1\ \mathbf{U}_w^{\mathrm{T}} + \mathbf{U}_h\ \hat{\mathbf{X}}_2\ \mathbf{U}_{N-w} + \mathbf{U}_{N-h}^{\mathrm{T}}\ \hat{\mathbf{X}}_3\ \mathbf{U}_w^{\mathrm{T}} + \mathbf{U}_{N-h}^{\mathrm{T}}\ \hat{\mathbf{X}}_4\ \mathbf{U}_{N-w} \tag{3.25}$$

By applying the distributive property of the matrix-product regarding to the matrix-addition operation, one can easily reduce the number of matrix-products of the above equations to only six:

$$\hat{\mathbf{X}} = \mathbf{L}_h^{\mathrm{T}} \left(\hat{\mathbf{X}}_1\ \mathbf{L}_w + \hat{\mathbf{X}}_2\ \mathbf{L}_{N-w}^{\mathrm{T}} \right) + \mathbf{L}_{N-h} \left(\hat{\mathbf{X}}_3\ \mathbf{L}_w + \hat{\mathbf{X}}_4\ \mathbf{L}_{N-w}^{\mathrm{T}} \right) \tag{3.26}$$

$$= \mathbf{U}_h \left(\hat{\mathbf{X}}_1\ \mathbf{U}_w^{\mathrm{T}} + \hat{\mathbf{X}}_2\ \mathbf{U}_{N-w} \right) + \mathbf{U}_{N-h}^{\mathrm{T}} \left(\hat{\mathbf{X}}_3\ \mathbf{U}_w^{\mathrm{T}} + \hat{\mathbf{X}}_4\ \mathbf{U}_{N-w} \right) \tag{3.27}$$

Nevertheless, considering that each matrix multiplication requires N^3 products and $N^2(N-1)$ additions, the above operation still implies a considerable complexity level of about $\mathcal{O}(N^3)$, independently of the adopted computational scheme. Moreover, similarly to the pixel-domain approach, the matrices $\mathbf{H}_{ij}$ are constant and can be pre-computed and stored in memory. Hence, if one takes into account that if $\mathbf{u}_z = \boldsymbol{\ell}_z^{\mathrm{T}},\ \forall z$, their transforms $\mathbf{U}_z$ and $\mathbf{L}_z$ are also related as $\mathbf{U}_z = \mathbf{L}_z^{\mathrm{T}}$, then only seven different matrices need to be stored. However, contrary to what happened in the pixel-domain approach, the $\mathbf{H}_{ij}$ matrices, obtained from the computation of the DCT of the $\mathbf{h}_{ij}$ matrices, are no longer sparse.

C - Motion compensation in the DCT-domain with sub-pixel resolution

It can be shown that the previous described motion compensation algorithm in the compressed DCT-domain can be easily generalized to the sub-pixel resolution. In such case, most video standards state that the required sub-pixel values should be obtained by interpolation, namely, by using the bilinear function [26, 27, 30]. One example of such situation is illustrated in fig. 3.7. If $P(x, y)$ denotes the intensity of the pixel located at position (x, y), the value $P(x_\alpha, y_\beta)$ can be obtained in two stages:

1. Compute the auxiliary values:

$$P(x_\alpha, y_0) = (1-\alpha)P(x_0, y_0) + \alpha P(x_1, y_0) \tag{3.28}$$

$$P(x_\alpha, y_1) = (1-\alpha)P(x_0, y_1) + \alpha P(x_1, y_1) \tag{3.29}$$

2. Compute the desired value:

$$\begin{aligned} P(x_\alpha, y_\beta) &= (1-\beta)P(x_\alpha, y_0) + \beta P(x_\alpha, y_1) && (3.30)\\ &= (1-\alpha)(1-\beta)P(x_0, y_0) + (1-\alpha)\beta P(x_0, y_1) + \\ &\quad \alpha(1-\beta)P(x_1, y_0) + \alpha\beta P(x_1, y_1). && (3.31) \end{aligned}$$

In practice, the above computation can be regarded as a weighted average of the four neighboring points of the desired pixel. The corresponding weights are given by $(1 - d_x)$ and $(1 - d_y)$, where $d = (d_x, d_y)$ is the vector that connects each neighbor point to the desired pixel.

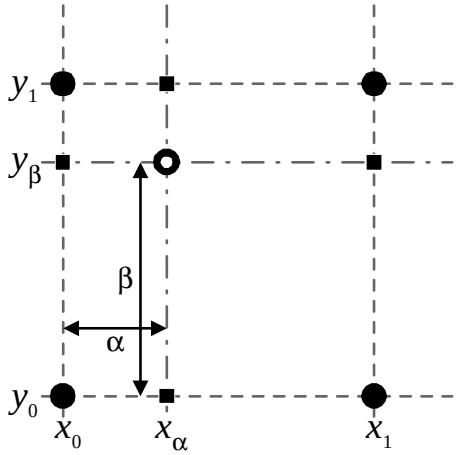


Figure 3.7: Sub-pixel resolution using bilinear interpolation.

Table 3.1: Intersected regions and the corresponding bilinear interpolation weights.

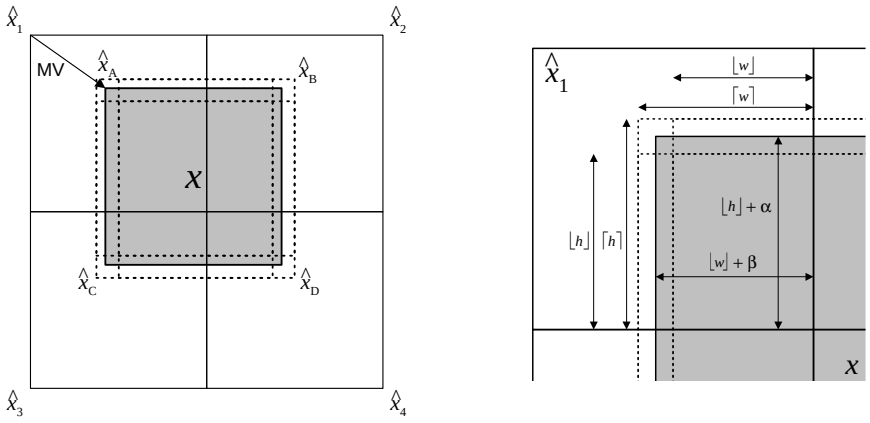
Prediction Block	Height	Width	Weight(h)	Weight(w)
$\hat{\mathbf{x}}_A$	$\lceil h \rceil$	$\lceil w \rceil$	α	β
$\hat{\mathbf{x}}_B$	$\lceil h \rceil$	$\lfloor w \rfloor$	α	$1-\beta$
$\hat{\mathbf{x}}_C$	$\lfloor h \rfloor$	$\lceil w \rceil$	$1-\alpha$	β
$\hat{\mathbf{x}}_D$	$\lfloor h \rfloor$	$\lfloor w \rfloor$	$1-\alpha$	$1-\beta$

In a block-based coding environment, it is highly convenient to implement the above interpolation in a block basis. Let us consider the non-integer motion vector v, which points to the intersected region between $\hat{\mathbf{x}}_1$ and the prediction block $\mathbf{x}$, composited by $h \times w$ pixels, where h and w can be any non-integer values (see fig. 3.8):

$$(h, w) = (N, N) - v, \tag{3.32}$$

with $h = \lfloor h \rfloor + \alpha$ and $w = \lfloor w \rfloor + \beta$.

In this case, for each prediction pixel, the four neighboring pixels can be obtained from the four adjacent prediction blocks $\hat{\mathbf{x}}_A$, $\hat{\mathbf{x}}_B$, $\hat{\mathbf{x}}_C$ and $\hat{\mathbf{x}}_D$, which, in turn, are obtained from the four original blocks $\hat{\mathbf{x}}_1$, $\hat{\mathbf{x}}_2$, $\hat{\mathbf{x}}_3$ and $\hat{\mathbf{x}}_4$ using an integer resolution whose intersected regions with $\hat{\mathbf{x}}_1$ have the dimensions given in table 3.1.



(a) Desired prediction block and the four adjacent blocks $\hat{\mathbf{x}}_A$, $\hat{\mathbf{x}}_B$, $\hat{\mathbf{x}}_C$ and $\hat{\mathbf{x}}_D$.

(b) Detailed view.

Figure 3.8: Bilinear interpolation using the four adjacent blocks $\hat{\mathbf{x}}_A$, $\hat{\mathbf{x}}_B$, $\hat{\mathbf{x}}_C$ and $\hat{\mathbf{x}}_D$.

As before, the desired sub-pixel prediction block is obtained by bilinear interpolating each four neighboring pixels using the weights presented in table 3.1:

$$\hat{\mathbf{x}} = \alpha\ \beta\ \hat{\mathbf{x}}_A + \alpha\ (1-\beta)\ \hat{\mathbf{x}}_B + (1-\alpha)\ \beta\ \hat{\mathbf{x}}_C + (1-\alpha)\ (1-\beta)\ \hat{\mathbf{x}}_D \tag{3.33}$$

$$\begin{aligned} = \quad & \alpha\ \beta\ \left[\boldsymbol{\ell}^{\mathrm{T}}_{\lceil h\rceil}\hat{\mathbf{x}}_1\boldsymbol{\ell}_{\lceil w\rceil} + \boldsymbol{\ell}^{\mathrm{T}}_{\lceil h\rceil}\hat{\mathbf{x}}_2\boldsymbol{\ell}^{\mathrm{T}}_{N-\lceil w\rceil} + \boldsymbol{\ell}_{N-\lceil h\rceil}\hat{\mathbf{x}}_3\boldsymbol{\ell}_{\lceil w\rceil} + \boldsymbol{\ell}_{N-\lceil h\rceil}\hat{\mathbf{x}}_4\boldsymbol{\ell}^{\mathrm{T}}_{N-\lceil w\rceil}\right] + \\ & \alpha\ (1-\beta)\ \left[\boldsymbol{\ell}^{\mathrm{T}}_{\lceil h\rceil}\hat{\mathbf{x}}_1\boldsymbol{\ell}_{\lfloor w\rfloor} + \boldsymbol{\ell}^{\mathrm{T}}_{\lceil h\rceil}\hat{\mathbf{x}}_2\boldsymbol{\ell}^{\mathrm{T}}_{N-\lfloor w\rfloor} + \boldsymbol{\ell}_{N-\lceil h\rceil}\hat{\mathbf{x}}_3\boldsymbol{\ell}_{\lfloor w\rfloor} + \boldsymbol{\ell}_{N-\lceil h\rceil}\hat{\mathbf{x}}_4\boldsymbol{\ell}^{\mathrm{T}}_{N-\lfloor w\rfloor}\right] + \\ & (1-\alpha)\ \beta\ \left[\boldsymbol{\ell}^{\mathrm{T}}_{\lfloor h\rfloor}\hat{\mathbf{x}}_1\boldsymbol{\ell}_{\lceil w\rceil} + \boldsymbol{\ell}^{\mathrm{T}}_{\lfloor h\rfloor}\hat{\mathbf{x}}_2\boldsymbol{\ell}^{\mathrm{T}}_{N-\lceil w\rceil} + \boldsymbol{\ell}_{N-\lfloor h\rfloor}\hat{\mathbf{x}}_3\boldsymbol{\ell}_{\lceil w\rceil} + \boldsymbol{\ell}_{N-\lfloor h\rfloor}\hat{\mathbf{x}}_4\boldsymbol{\ell}^{\mathrm{T}}_{N-\lceil w\rceil}\right] + \\ & (1-\alpha)\ (1-\beta)\ \left[\boldsymbol{\ell}^{\mathrm{T}}_{\lfloor h\rfloor}\hat{\mathbf{x}}_1\boldsymbol{\ell}_{\lfloor w\rfloor} + \boldsymbol{\ell}^{\mathrm{T}}_{\lfloor h\rfloor}\hat{\mathbf{x}}_2\boldsymbol{\ell}^{\mathrm{T}}_{N-\lfloor w\rfloor} + \boldsymbol{\ell}_{N-\lfloor h\rfloor}\hat{\mathbf{x}}_3\boldsymbol{\ell}_{\lfloor w\rfloor} + \boldsymbol{\ell}_{N-\lfloor h\rfloor}\hat{\mathbf{x}}_4\boldsymbol{\ell}^{\mathrm{T}}_{N-\lfloor w\rfloor}\right] \end{aligned} \tag{3.34}$$

The above equation can be expressed with its dependencies in $\hat{\mathbf{x}}_1$, $\hat{\mathbf{x}}_2$, $\hat{\mathbf{x}}_3$ and $\hat{\mathbf{x}}_4$, as follows:

$$\hat{\mathbf{x}} = f(\hat{\mathbf{x}}_1) + f(\hat{\mathbf{x}}_2) + f(\hat{\mathbf{x}}_3) + f(\hat{\mathbf{x}}_4), \tag{3.35}$$

where

$$\begin{aligned} f(\hat{\mathbf{x}}_1) = {} & \alpha\ \boldsymbol{\ell}^{\mathrm{T}}_{\lceil h\rceil}\hat{\mathbf{x}}_1\boldsymbol{\ell}_{\lceil w\rceil}\ \beta\ +\ \alpha\ \boldsymbol{\ell}^{\mathrm{T}}_{\lceil h\rceil}\hat{\mathbf{x}}_1\boldsymbol{\ell}_{\lfloor w\rfloor}\ (1-\beta)\ + \\ & (1-\alpha)\ \boldsymbol{\ell}^{\mathrm{T}}_{\lfloor h\rfloor}\hat{\mathbf{x}}_1\boldsymbol{\ell}_{\lceil w\rceil}\ \beta\ +\ (1-\alpha)\ \boldsymbol{\ell}^{\mathrm{T}}_{\lfloor h\rfloor}\hat{\mathbf{x}}_1\boldsymbol{\ell}_{\lfloor w\rfloor}\ (1-\beta) \end{aligned} \tag{3.36}$$

$$= \alpha\ \boldsymbol{\ell}^{\mathrm{T}}_{\lceil h\rceil}\hat{\mathbf{x}}_1\Big[\boldsymbol{\ell}_{\lceil w\rceil}\ \beta + \boldsymbol{\ell}_{\lfloor w\rfloor}\ (1-\beta)\Big] + (1-\alpha)\ \boldsymbol{\ell}^{\mathrm{T}}_{\lfloor h\rfloor}\hat{\mathbf{x}}_1\Big[\boldsymbol{\ell}_{\lceil w\rceil}\ \beta + \boldsymbol{\ell}_{\lfloor w\rfloor}\ (1-\beta)\Big] \tag{3.37}$$

$$= \Big[\alpha\ \boldsymbol{\ell}^{\mathrm{T}}_{\lceil h\rceil}\ + (1-\alpha)\ \boldsymbol{\ell}^{\mathrm{T}}_{\lfloor h\rfloor}\Big]\hat{\mathbf{x}}_1\Big[\beta\ \boldsymbol{\ell}_{\lceil w\rceil}\ + (1-\beta)\ \boldsymbol{\ell}_{\lfloor w\rfloor}\Big] \tag{3.38}$$

$$= \boldsymbol{\ell}^{\mathrm{T}}(h)\ \hat{\mathbf{x}}_1\ \boldsymbol{\ell}(w), \tag{3.39}$$

with

$$\boldsymbol{\ell}(m) = \theta\ \boldsymbol{\ell}_{\lceil m\rceil} + (1-\theta)\ \boldsymbol{\ell}_{\lfloor m\rfloor} \tag{3.40}$$

and

$$\theta = m - \lfloor m\rfloor. \tag{3.41}$$

In a similar way, one could derive:

$$f(\hat{\mathbf{x}}_2) = \boldsymbol{\ell}^{\mathrm{T}}(h)\ \hat{\mathbf{x}}_2\ \boldsymbol{\ell}^{\mathrm{T}}(N-w), \tag{3.42}$$

$$f(\hat{\mathbf{x}}_3) = \boldsymbol{\ell}(N-h)\ \hat{\mathbf{x}}_3\ \boldsymbol{\ell}(w), \tag{3.43}$$

$$f(\hat{\mathbf{x}}_4) = \boldsymbol{\ell}(N-h)\ \hat{\mathbf{x}}_4\ \boldsymbol{\ell}^{\mathrm{T}}(N-w). \tag{3.44}$$

Hence, for the DCT-domain MC, with sub-pixel resolution, one can obtain an expression entirely similar to eq. 3.21:

$$\begin{aligned}\hat{\mathbf{x}} &= \boldsymbol{\ell}^{\mathrm{T}}(h)\ \hat{\mathbf{x}}_1\ \boldsymbol{\ell}(w) + \boldsymbol{\ell}^{\mathrm{T}}(h)\ \hat{\mathbf{x}}_2\ \boldsymbol{\ell}^{\mathrm{T}}(N-w) + \\ &\quad \boldsymbol{\ell}(N-h)\ \hat{\mathbf{x}}_3\ \boldsymbol{\ell}(w) + \boldsymbol{\ell}(N-h)\ \hat{\mathbf{x}}_4\ \boldsymbol{\ell}^{\mathrm{T}}(N-w),\end{aligned} \tag{3.45}$$

with $h,\ w \in \mathbb{Q}$. It is worth noting that, in the particular case when $m \in \mathbb{N}$, $\lceil m \rceil = \lfloor m \rfloor = m$ and $\boldsymbol{\ell}(m) = \boldsymbol{\ell}_m$, making eq. 3.45 entirely similar to eq. 3.21.

The transposition of eq. 3.45 to the compressed DCT-domain is performed in an entirely similar manner as it was done before, leading to:

$$\begin{aligned}\hat{\mathbf{X}} &= \mathbf{L}^{\mathrm{T}}(h)\ \hat{\mathbf{X}}_1\ \mathbf{L}(w) + \mathbf{L}^{\mathrm{T}}(h)\ \hat{\mathbf{X}}_2\ \mathbf{L}^{\mathrm{T}}(N-w) + \\ &\quad \mathbf{L}(N-h)\ \hat{\mathbf{X}}_3\ \mathbf{L}(w) + \mathbf{L}(N-h)\ \hat{\mathbf{X}}_4\ \mathbf{L}^{\mathrm{T}}(N-w) \qquad (3.46)\\ &= \mathbf{L}^{\mathrm{T}}(h)\left[\hat{\mathbf{X}}_1\ \mathbf{L}(w) + \hat{\mathbf{X}}_2\ \mathbf{L}^{\mathrm{T}}(N-w)\right] + \mathbf{L}(N-h)\left[\hat{\mathbf{X}}_3\ \mathbf{L}(w) + \hat{\mathbf{X}}_4\ \mathbf{L}^{\mathrm{T}}(N-w)\right] \qquad (3.47)\end{aligned}$$

where

$$\mathbf{L}(m) = \mathrm{DCT}\left(\boldsymbol{\ell}\left(m\right)\right). \tag{3.48}$$

However, contrasting to what was previously described, although the set of matrices $\mathbf{L}(m)$ are constant, there is no longer any feasibility to pre-compute and store all possible values in memory. In fact, while only seven different $\mathbf{L}_m$ matrices were required in the previously described integer resolution approach, the extension to the sub-pixel resolution gives rise to the need of considering a significant number of possible values $h,\ w \in \mathbb{Q}$. Even so, it is still possible to compute their values using a restricted set of pre-computed and stored matrices, by taking into consideration eq. 3.40:

$$\mathbf{L}(m) = \theta\ \mathbf{L}_{\lceil m \rceil} + (1-\theta)\ \mathbf{L}_{\lfloor m \rfloor}, \tag{3.49}$$

where $\theta = m - \lfloor m \rfloor$ are scalars. Hence, the previous setup, composed by seven pre-computed and stored matrices, can still be kept unchanged, with an additional computational cost of $2N^2$ scalar multiplications and N^2 additions to compute each $\mathbf{L}(m)$ matrix. If one takes into consideration that eq. 3.47 requires 6 matrix multiplications and 3 matrix sums, representing $6N^3$ products and $6N^3 - 3N^2$ additions, these extra operations will represent an increase of $8N^2$ scalar multiplications and $4N^2$ additions, leading to a total of $6N^3 + 8N^2$ products and $6N^3 + N^2$ additions. Considering the usual 8×8 block dimension ($N = 8$), it represents an increase of 16.7% and 8.9% of the number of multiplications and additions, respectively. This

contrasts with the significant increase of the number of operations that would arise if a brute-force approach was followed by separately computing each of the four neighboring blocks.

As a concluding remark, the above described algorithm provides the ability to extend the computation of the motion compensation algorithm in the compressed DCT-domain to any sub-pixel resolution, keeping the original complexity level unchanged ($\mathcal{O}(N^3)$) but requiring more memory.

D - Computationaly reduced algorithms for motion-compensation in the DCT-domain

Several proposals have been presented in order to reduce the computational load of the motion compensation algorithm presented for the DCT-domain. In the remaining of this section, two of the most used methods will be presented: *Bandwidth constrained motion-compensation* and *Decomposition of the DCT-kernel matrix*.

Bandwidth constrained motion-compensation

The previously described motion compensation algorithm requires a considerable amount of operations. According to eq. 3.26 or eq. 3.27, six $(N \times N)$ matrix multiplications and three $(N \times N)$ matrix additions are required, leading to the computation of $6N^3 - 6N^2$ elementary products and $6N^3 + 3N^2$ sums.

Recently, Li and Shi [51] and Liu and Bovik [56, 57] have proposed two rather equivalent strategies to reduce the computational load of the motion compensation algorithm. Their algorithms exploit the sparseness property of the $\hat{\mathbf{X}}_i$ blocks, as well as the spatial continuity and the high correlation between neighboring blocks. The proposed motion compensation algorithms are based on two elementary operations: *cropping* and *shifting*. The *cropping* operation keeps unchanged all the data inside the window (effective area) and zeroes all the pixels outside this area. The *shifting* operation displaces the interest region, according to the motion vector of the macroblock under processing.

This procedure, however, usually gives rise to an undesired and significant increase of the bandwidth required to represent each subblock. Such increase is the result of the introduction of the steep change at the edge of the interest window, between the effective area and the blank area. As a consequence, this abrupt boundary usually introduces, at the resulting block, some DCT coefficients with higher frequencies than those of the original block. The effect of this phenomenom can be easily observed in the example illustrated in fig. 3.9, where it is considered a macroblock composited by four luminance blocks $\hat{\mathbf{x}}_1$, $\hat{\mathbf{x}}_2$, $\hat{\mathbf{x}}_3$ and $\hat{\mathbf{x}}_4$. The motion

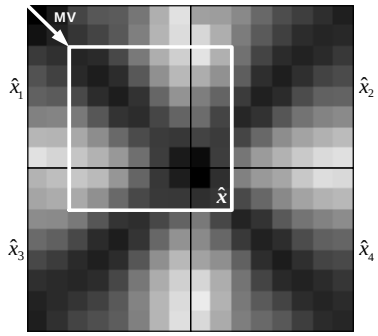


Figure 3.9: Macroblock composited by four luminance blocks $\hat{x}_1$, $\hat{x}_2$, $\hat{x}_3$ and $\hat{x}_4$.

compensation algorithm is applied to obtain the prediction block $\hat{\mathbf{x}}$ given by the motion vector $v = (2, 2)$. The DCT coefficients of each block $\hat{\mathbf{x}}_i$ are presented in eq. 3.50. These coefficients were obtained at the output of the inverse quantizer block of the video decoder, using a given quantization step size ($Q = 15$).

$$\hat{\mathbf{X}}_1 = \begin{bmatrix} 816 & -90 & 120 & -60 & 0 & 0 & 0 & 0 \\ -90 & -390 & 60 & -30 & 0 & 0 & 0 & 0 \\ 120 & 60 & -120 & 0 & 0 & 0 & 0 & 0 \\ -60 & 0 & 0 & -60 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \hat{\mathbf{X}}_2 = \begin{bmatrix} 816 & 60 & 120 & 30 & 0 & 0 & 0 & 0 \\ -90 & 360 & 60 & 0 & 0 & 0 & 0 & 0 \\ 120 & -90 & -120 & -30 & 0 & 0 & 0 & 0 \\ -60 & -30 & 0 & 30 & 0 & 0 & 0 & 0 \\ 0 & -30 & -30 & -30 & -30 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\hat{\mathbf{X}}_3 = \begin{bmatrix} 816 & -90 & 120 & -60 & 0 & 0 & 0 & 0 \\ 60 & 360 & -90 & 0 & 0 & 0 & 0 & 0 \\ 120 & 60 & -120 & 0 & 0 & 0 & 0 & 0 \\ 30 & -30 & -30 & 30 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \hat{\mathbf{X}}_4 = \begin{bmatrix} 816 & 60 & 120 & 30 & 0 & 0 & 0 & 0 \\ 60 & -390 & -90 & -30 & -30 & 0 & 0 & 0 \\ 120 & -90 & -120 & -30 & 0 & 0 & 0 & 0 \\ 30 & 0 & -30 & -60 & -30 & 0 & 0 & 0 \\ 0 & -30 & -30 & -30 & -30 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.50}$$

By observing these DCT coefficients, one can easily realize that each $\hat{\mathbf{X}}_i$ is a sparse matrix with most of its high frequency coefficients equal to zero. In fact, they can be represented as:

$$\hat{\mathbf{X}}_i = \begin{bmatrix} [\mathbf{A}]_{k_i \times k_i} & 0 \\ 0 & 0 \end{bmatrix} \tag{3.51}$$

where $[\mathbf{A}]_{k_i \times k_i}$ denotes a $(k_i \times k_i)$ matrix, with at least a non-null element in its k_i line or k_i column. Hence, assuming that $d(x_i, y_i)$ is the non-zero highest effective frequency element in $\hat{\mathbf{X}}_i$ and $k_i = \max\{x_i, y_i\}$, then $d(m, n) = 0$ for all $m, n > k_i$. For convenience of representation, k_i will be used to denote the bandwidth of the

block $\hat{\mathbf{x}}_i$. Hence, the bandwidths of the four blocks that composite the macroblock represented in fig. 3.9 are $k_1 = 4$, $k_2 = 5$, $k_3 = 4$ and $k_4 = 5$.

The DCT coefficients of the prediction block ($\hat{\mathbf{X}}$) are obtained by summing up the four components $\tilde{\mathbf{X}}_1$, $\tilde{\mathbf{X}}_2$, $\tilde{\mathbf{X}}_3$ and $\tilde{\mathbf{X}}_4$, where $\tilde{\mathbf{X}}_i = \mathbf{H}_{i1}\hat{\mathbf{X}}_i\mathbf{H}_{i2}$ and $\hat{\mathbf{X}} = \sum_{i=1}^{4} \tilde{\mathbf{X}}_i$. Each $\tilde{\mathbf{X}}_i$ component represents the contribution of each block $\hat{\mathbf{x}}_i$ to the prediction block. In figs. 3.10 through 3.13 it is represented each of these components $\tilde{\mathbf{x}}_i$. The corresponding DCT representations $\tilde{\mathbf{X}}_i$ are presented in eqs. 3.52 through 3.55.

Figure 3.10: Block $\tilde{\mathbf{x}}_1$.

$$\tilde{\mathbf{X}}_1 = \begin{bmatrix} 427.125 & 155.458 & -118.705 & 135.373 & 5.875 & -55.038 & 75.011 & -47.359 \\ 157.688 & -65.885 & -151.049 & 118.185 & 3.346 & -48.600 & 63.690 & -40.789 \\ -111.631 & -153.590 & -89.645 & -5.422 & 1.238 & -4.620 & 6.123 & -4.215 \\ 140.859 & 112.125 & -18.066 & -35.841 & 2.406 & 4.686 & -3.618 & 2.330 \\ 3.875 & 5.073 & 5.317 & 2.526 & -3.375 & -2.832 & 0.097 & -1.021 \\ -56.806 & -46.281 & 1.225 & 6.454 & -3.213 & -2.266 & -0.135 & -0.388 \\ 76.410 & 61.855 & 1.123 & -5.476 & 0.704 & 0.233 & 1.395 & -0.727 \\ -47.917 & -39.022 & -0.412 & 3.614 & -1.619 & -0.364 & -0.415 & -0.507 \end{bmatrix} \tag{3.52}$$

Figure 3.11: Block $\tilde{\mathbf{x}}_2$.

$$\tilde{\mathbf{X}}_2 = \begin{bmatrix} 162.500 & -206.943 & 145.442 & -64.230 & -12.250 & 62.167 & -73.503 & 47.923 \\ 131.470 & -166.270 & 113.476 & -44.012 & -20.868 & 62.199 & -69.591 & 44.588 \\ 17.916 & -22.508 & 14.919 & -4.964 & -4.266 & 10.022 & -10.798 & 6.831 \\ 16.629 & -22.260 & 18.808 & -14.018 & 9.003 & -4.781 & 1.954 & -0.540 \\ 10.750 & -12.508 & 5.339 & 3.873 & -12.000 & 16.268 & -15.200 & 9.109 \\ -3.188 & 4.870 & -5.788 & 6.826 & -7.428 & 7.110 & -5.642 & 3.130 \\ 3.402 & -5.204 & 6.201 & -7.331 & 7.991 & -7.658 & 6.080 & -3.374 \\ -3.274 & 4.696 & -4.836 & 4.909 & -4.733 & 4.158 & -3.120 & 1.678 \end{bmatrix} \tag{3.53}$$

Figure 3.12: Block $\tilde{\mathbf{x}}_3$.

$$\tilde{\mathbf{X}}_3 = \begin{bmatrix} 162.750 & 128.663 & 12.484 & 7.375 & -2.500 & -6.644 & 7.276 & -5.125 \\ -207.079 & -162.895 & -15.748 & -10.706 & 3.124 & 8.777 & -9.685 & 6.754 \\ 145.003 & 111.687 & 10.626 & 11.363 & -2.022 & -7.092 & 8.030 & -5.411 \\ -63.073 & -44.274 & -3.895 & -11.948 & 0.580 & 4.800 & -5.755 & 3.589 \\ -14.000 & -18.767 & -2.372 & 11.887 & 0.750 & -2.494 & 3.418 & -1.767 \\ 64.143 & 59.073 & 6.331 & -10.696 & -1.566 & 0.712 & -1.525 & 0.378 \\ -75.216 & -66.576 & -6.969 & 8.160 & 1.649 & 0.238 & 0.373 & 0.329 \\ 48.915 & 42.757 & 4.441 & -4.429 & -1.035 & -0.369 & 0.040 & -0.368 \end{bmatrix} \tag{3.54}$$

Figure 3.13: Block $\tilde{\mathbf{x}}_4$.

$$\tilde{\mathbf{X}}_4 = \begin{bmatrix} 18.250 & -23.914 & 18.774 & -11.841 & 5.000 & 0.055 & -2.364 & 1.997 \\ -23.940 & 31.322 & -24.449 & 15.194 & -6.090 & -0.581 & 3.534 & -2.869 \\ 18.869 & -24.547 & 18.761 & -11.011 & 3.471 & 1.902 & -4.014 & 2.967 \\ -12.023 & 15.414 & -11.135 & 5.462 & -0.072 & -3.536 & 4.534 & -3.026 \\ 5.250 & -6.410 & 3.702 & -0.184 & -3.000 & 4.837 & -4.780 & 2.930 \\ -0.216 & -0.225 & 1.622 & -3.359 & 4.762 & -5.238 & 4.482 & -2.583 \\ -2.133 & 3.228 & -3.764 & 4.360 & -4.685 & 4.448 & -3.511 & 1.943 \\ 1.865 & -2.692 & 2.820 & -2.920 & 2.866 & -2.553 & 1.934 & -1.046 \end{bmatrix} \tag{3.55}$$

The result of the addition of all these components $\tilde{\mathbf{X}}_i$ is the DCT representation

of the prediction block $\hat{\mathbf{X}}$ and is given by:

$$\hat{\mathbf{X}} = \sum_{i=1}^{4} \tilde{\mathbf{X}}_i = \begin{bmatrix} 767.244 & 62.464 & 54.077 & 64.433 & 3.784 & -10.028 & 16.311 & -8.584 \\ 67.579 & -374.041 & -72.821 & 81.106 & -29.710 & 34.538 & -23.856 & 15.038 \\ 65.571 & -83.412 & -47.463 & -11.040 & 3.066 & -5.834 & 4.645 & -3.367 \\ 80.382 & 63.241 & -15.658 & -57.225 & 14.345 & -1.836 & -0.009 & 0.753 \\ 13.832 & -41.617 & 16.198 & 20.019 & -24.742 & 26.608 & -25.741 & 14.926 \\ -6.501 & 29.916 & -2.202 & -4.132 & 3.556 & -14.661 & 10.979 & -7.781 \\ 12.095 & -18.418 & 2.279 & 2.585 & -4.088 & 11.427 & -9.107 & 6.164 \\ -6.860 & 12.742 & -1.459 & -0.622 & 1.863 & -8.162 & 6.269 & -4.406 \end{bmatrix} \tag{3.56}$$

By applying the IDCT operation, one can finally obtain the prediction block ($\hat{\mathbf{x}}$), illustrated in fig. 3.14 and presented in eq. 3.57.

Figure 3.14: Prediction block $\hat{\mathbf{x}}$

$$\hat{\mathbf{x}} = \text{IDCT}(\hat{\mathbf{X}}) = \begin{bmatrix} 38 & 44 & 73 & 121 & 172 & 205 & 224 & 173 \\ 43 & 30 & 44 & 87 & 142 & 180 & 170 & 144 \\ 74 & 44 & 28 & 49 & 100 & 145 & 123 & 99 \\ 121 & 86 & 49 & 42 & 70 & 100 & 105 & 67 \\ 167 & 139 & 103 & 75 & 61 & 58 & 66 & 58 \\ 194 & 175 & 150 & 111 & 65 & 32 & 11 & 60 \\ 195 & 183 & 147 & 98 & 54 & 30 & 10 & 42 \\ 166 & 142 & 104 & 71 & 57 & 55 & 43 & 51 \end{bmatrix} \tag{3.57}$$

By observing the DCT coefficients of the obtained motion-compensated block $\hat{\mathbf{X}}$, depicted in eq. 3.56, the previously referred undesired increase of the required bandwidth to represent this block becomes evident. In fact, while the four original blocks $\hat{\mathbf{X}}_i$ are characterized by a maximum bandwidth of $k = 5$, the DCT coefficients matrix of the obtained block is no longer sparse and is characterized by a bandwidth of $k = 8$.

Algorithm

If one takes into account that the several effective areas of the four components $\tilde{\mathbf{x}}_i$ are spatially adjacent, it may be expected that there should be no subblock boundaries in $\hat{\mathbf{x}}$. Consequently, the number of non-zero coefficients in $\hat{\mathbf{X}}$ should be much smaller than those presented in $\hat{\mathbf{X}}_i$. This implies that the bandwidth of $\hat{\mathbf{x}}$ should be also much smaller that the bandwidth of each component $\tilde{\mathbf{x}}_i$.

Hence, one can decompose each component $\tilde{\mathbf{X}}_i$ into two coefficient matrixes $\mathbf{F}_i$ and $\mathbf{E}_i$. The matrix $\mathbf{F}_i$ represents the actual contribution of $\tilde{\mathbf{X}}_i$ to $\hat{\mathbf{X}}$ and it usually contains low frequency components. The matrix $\mathbf{E}_i$ represents the subblock boundary effect and it is characterized by a higher bandwidth. Thus:

$$\hat{\mathbf{X}} = \sum_{i=1}^{4} \tilde{\mathbf{X}}_i = \sum_{i=1}^{4} (\mathbf{F}_i + \mathbf{E}_i) = \sum_{i=1}^{4} \mathbf{F}_i + \sum_{i=1}^{4} \mathbf{E}_i \tag{3.58}$$

Although many elements of the $\mathbf{E}_i$ matrices are non-zero, the sum of the four matrices should be an all-zero matrix: $\sum_{i=1}^{4} \mathbf{E}_i = [0]_{8\times 8}$, since no subblock boundary

effect should be present in $\hat{\mathbf{x}}$. This implies that there is no need to calculate $\mathbf{E}_i$, providing the ability to restrict the calculation to the computation of the fraction $\sum_{i=1}^{4} \mathbf{F}_i$. Consequently, if one estimates the frequency bandwidth (k_f) of $\hat{\mathbf{X}}$ prior to actually computing $\hat{\mathbf{X}}$, it is only necessary to compute those frequency components below k_f in the calculation of $\tilde{\mathbf{X}}_i$. This is the basic idea of this bandwidth constrained algorithm, that can be implemented simply by applying low-pass filtering in the computation process.

Hence, by observing that $\hat{\mathbf{x}}$ is obtained from $\hat{\mathbf{x}}_i$ and by assuming that the maximum bandwidth of $\hat{\mathbf{x}}$ is k_f, from the continuity properties and from the high correlation among the pixels inside the macroblock area, one can naturally estimate k_f as:

$$k_f = \max\{k_1, k_2, k_3, k_4\}. \tag{3.59}$$

Consequently, one can apply a low-pass filter $\mathfrak{L}_{k_f}$ in the computation process, where:

$$\mathfrak{L}_{k_f} = \begin{bmatrix} [\mathbf{I}]_{k_f \times k_f} & 0 \\ 0 & 0 \end{bmatrix} \tag{3.60}$$

and $[\mathbf{I}]_{k_f \times k_f}$ is a $(k_f \times k_f)$ identity matrix. The coefficients corresponding to the effective area in $\hat{\mathbf{X}}$ are extracted as:

$$\hat{\mathbf{X}} = \sum_{i=1}^{4} \mathbf{F}_i = \mathfrak{L}_{k_f} \left(\sum_{i=1}^{4} \mathbf{H}_{i1} \hat{\mathbf{X}}_i \mathbf{H}_{i2} \right) \mathfrak{L}_{k_f}^{\mathrm{T}} \tag{3.61}$$

$$= \sum_{i=1}^{4} \left(\mathfrak{L}_{k_f} \mathbf{H}_{i1} \, \mathfrak{L}_{k_f}^{\mathrm{T}} \right) \left(\mathfrak{L}_{k_f} \hat{\mathbf{X}}_i \, \mathfrak{L}_{k_f}^{\mathrm{T}} \right) \left(\mathfrak{L}_{k_f} \mathbf{H}_{i2} \, \mathfrak{L}_{k_f}^{\mathrm{T}} \right) \tag{3.62}$$

where $\mathbf{H}_{i1}$ and $\mathbf{H}_{i2}$ are the discrete cosine transform of matrices $\mathbf{h}_{i1}$ and $\mathbf{h}_{i2}$, defined in eqs. 3.19. Since the bandwidth (k_i) of each component $\hat{\mathbf{X}}_i$ is lower that k_f (see eq. 3.59), it comes:

$$k_i \leq k_f \quad \Rightarrow \quad \mathfrak{L}_{k_f} \hat{\mathbf{X}}_i \, \mathfrak{L}_{k_f}^{\mathrm{T}} = \hat{\mathbf{X}}_i. \tag{3.63}$$

and

$$\hat{\mathbf{X}} = \sum_{i=1}^{4} \mathbf{F}_i = \sum_{i=1}^{4} \left(\mathfrak{L}_{k_f} \mathbf{H}_{i1} \, \mathfrak{L}_{k_f}^{\mathrm{T}} \right) \hat{\mathbf{X}}_i \left(\mathfrak{L}_{k_f} \mathbf{H}_{i2} \, \mathfrak{L}_{k_f}^{\mathrm{T}} \right) \tag{3.64}$$

$$= \sum_{i=1}^{4} \mathbf{H}_{i1}^{k_f} \, \hat{\mathbf{X}}_i \, \mathbf{H}_{i2}^{k_f} \tag{3.65}$$

where $\mathbf{H}_{i1}^{k_f}$ and $\mathbf{H}_{i2}^{k_f}$ are obtained by applying the low-pass filter $\mathfrak{L}_{k_f}$ to the original

matrices $\mathbf{H}_{i1}$ and $\mathbf{H}_{i2}$:

$$\mathbf{H}_{ij}^{k_f}(m,n) = \begin{cases} \mathbf{H}_{ij}(m,n) & , \ m,n \leq k_f \\ 0 & , \ \text{otherwise.} \end{cases} \tag{3.66}$$

The result of the application of the described procedure to the macroblock used in the previous example is presented in fig. 3.15. The corresponding DCT representation $\hat{\mathbf{X}}$ is shown in eq. 3.67.

Figure 3.15: Prediction block $\hat{\mathbf{x}}$

$$\hat{\mathbf{X}} = \begin{bmatrix} 767.244 & 62.464 & 54.077 & 64.433 & 3.784 & 0 & 0 & 0 \\ 67.579 & -374.041 & -72.821 & 81.106 & -29.710 & 0 & 0 & 0 \\ 65.571 & -83.412 & -47.463 & -11.040 & 3.066 & 0 & 0 & 0 \\ 80.382 & 63.241 & -15.658 & -57.225 & 14.345 & 0 & 0 & 0 \\ 13.832 & -41.617 & 16.198 & 20.019 & -24.742 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.67}$$

By applying the IDCT operation, the prediction block $\hat{\mathbf{x}}$ can be obtained:

$$\hat{\mathbf{x}} = \text{IDCT}(\hat{\mathbf{X}}) = \begin{bmatrix} 33 & 47 & 72 & 115 & 174 & 216 & 211 & 183 \\ 46 & 38 & 46 & 87 & 144 & 176 & 165 & 140 \\ 73 & 45 & 26 & 48 & 102 & 138 & 127 & 96 \\ 116 & 86 & 47 & 38 & 72 & 109 & 104 & 75 \\ 168 & 143 & 105 & 75 & 62 & 61 & 62 & 61 \\ 200 & 176 & 149 & 117 & 62 & 13 & 13 & 43 \\ 194 & 167 & 139 & 109 & 59 & 13 & 10 & 36 \\ 172 & 144 & 105 & 74 & 55 & 44 & 41 & 41 \end{bmatrix} \tag{3.68}$$

By comparing the obtained results with the pixel values presented in eq. 3.57 it can be observed that the application of this algorithm leads to quite similar results. Nevertheless, it also presents some slightly differences in certain pixel values of the processed macroblock. Such differences arise as a result of restricting the bandwidth in the computation of the involved signals to k_f.

Computational Requirements

By comparing eqs. 3.23 and 3.65 one realizes that the formalism of the motion compensation algorithm was kept entirely unchanged. However, the same result can now be obtained by performing the operations on $(k_f \times k_f)$ matrices, instead of using (8×8) matrices as in eq. 3.23.

Thus, the main advantage of using this bandwidth constrained scheme comes from the significant reduction of the involved computational load. In fact, by using a computational scheme entirely similar to the one illustrated in eqs. 3.26 and 3.27, requiring six matrix multiplications and three matrix additions, one concludes that $6k_f^3$ elementary products and $6k_f^3 - 3k_f^2$ sums are now required. Hence, depending

on the block bandwidth (k_f), this can provide a significant computational saving in the overall motion compensation procedure.

Distortion

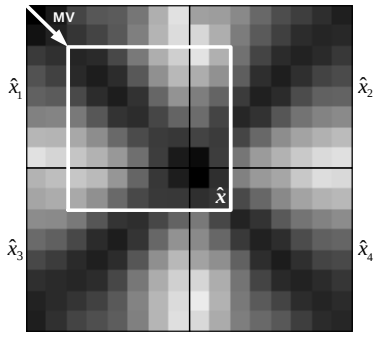
In eq. 3.58 it was stated that the sum of the contributions of the four macroblocks to the subblock boundary effect is canceled, i.e. $\sum_{i=1}^{4} \mathbf{E}_i = 0$, so that only the effective area fractions will contribute to the final result:

$$\hat{\mathbf{X}} = \sum_{i=1}^{4} \mathbf{F}_i + \sum_{i=1}^{4} \mathbf{E}_i = \sum_{i=1}^{4} \mathbf{F}_i \tag{3.69}$$

However, by considering that the maximum bandwidth k_i of the four blocks used in the above example is 5, and by analyzing the DCT coefficients of the macroblock obtained using the full-precision method $\hat{\mathbf{X}}$ (see eq. 3.56), the presence of coefficients representing frequencies with order higher than k_f can be easily observed, with $k_f = \max\{k_1, k_2, k_3, k_4\} = 5$.

This fact can be justified by the presence of inherent discontinuities along the original boundaries of the four blocks $\hat{\mathbf{x}}_i$. These edges, whose representations were not present in the original DCT coefficients of each block $\hat{\mathbf{x}}_i$, are now located in the middle of the area occupied by $\hat{\mathbf{x}}$ and therefore must now be represented, leading to increased order DCT coefficients. Since the described bandwidth constrained algorithm does not take into account for this fact, it may give rise to the introduction of a minor distortion in the image. On the other hand, the low-pass filtering effect that characterizes this method often contributes to a noticeable decrease of the blocking effect that is present in the image, therefore slightly improving the image quality [57].

In fig. 3.16 it is presented the original macroblock, as well as the resulting prediction block using the full-precision method (fig. 3.16(b)) and using the bandwidth constrained algorithm (fig. 3.16(c)). The PSNR of these blocks is 24.89 dB and 24.34 dB, respectively, corresponding to an almost non-noticeable degradation of about 0.55 dB. However, while the full-precision algorithm required 4096 multiplications and 3776 sums, the bandwidth constrained method needed only 1000 multiplications and 875 sums, representing an approximate speedup of about 4. This significant speedup can easily justify the slight degradation that is introduced by this method. Its magnitude is directly related to the block bandwidth, which is greatly dependent on the quantization process that is used in the video coding procedure. Thus, greater speedups can easily be achieved when higher quantization steps are used.



(a) Original macroblock.

(b) Prediction block, obtained using the full-precision method.

(c) Prediction block, obtained using the bandwidth constrained method.

Figure 3.16: DCT-domain motion compensation using the full-precision and the bandwidth constrained methods.

In conclusion, this bandwidth constrained algorithm may offer significant speedup capabilities by avoiding unnecessary computations at a minor cost of introducing slight and non-noticeable distortions in the decoded images. These distortions can often be compensated and minimized by an inherent reduction of the blocky artifacts (previously introduced in the quantization process) as a consequence of the low-pass filtering that is performed by this algorithm.

Decomposition of the DCT-kernel matrix

To fasten the computation of the operation presented in eq. 3.23, some decomposition methods have also been proposed by several authors. Merhav and Bhaskaran [62, 63] introduced the usage of one of the fastest algorithms for computing the 8-point DCT, proposed by Arai et al. [3]. Considering the computation of the DCT of a given block of pixels ($\mathbf{X}$) using the traditional matrix product: $\mathbf{X} = \mathbf{T}\ \mathbf{x}\ \mathbf{T}^{\mathrm{T}}$, Arai et al. proposed a factorization for matrix $\mathbf{T}$ so that:

$$\mathbf{T} = \mathbf{DPB}_1\mathbf{B}_2\mathbf{MA}_1\mathbf{A}_2\mathbf{A}_3, \tag{3.70}$$

where $\mathbf{D}$ is a diagonal matrix, $\mathbf{P}$ is a sparse permutation matrix (all non-null elements are equal to 1), $\mathbf{M}$ is a sparse matrix, and $\mathbf{B}_1$, $\mathbf{B}_2$, $\mathbf{A}_1$, $\mathbf{A}_2$ and $\mathbf{A}_3$ are all sparse matrices with all non-null elements equal to 1 or -1 (see [63]). Hence by pre-computing the fixed upper and lower matrices:

$$\mathbf{J}_i \triangleq \mathbf{u}_i(\mathbf{MA}_1\mathbf{A}_2\mathbf{A}_3)^{\mathrm{T}}, \quad i = 1, 2, \ldots, 8 \tag{3.71}$$

and

$$\mathbf{K}_i \triangleq \boldsymbol{\ell}_i(\mathbf{MA}_1\mathbf{A}_2\mathbf{A}_3)^{\mathrm{T}}, \quad i = 1, 2, \ldots, 8, \tag{3.72}$$

Merhav and Bhaskaran computed the DCT of the prediction block $\hat{X}$ using one of the following expressions:

$$\begin{aligned}\hat{\mathbf{X}} &= \mathbf{T}\left[\mathbf{J}_h\mathbf{B}_2^{\mathrm{T}}\mathbf{B}_1^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\mathbf{D}\left(\mathbf{X}_1\mathbf{D}\mathbf{P}\mathbf{B}_1\mathbf{B}_2\mathbf{J}_w^{\mathrm{T}} + \mathbf{X}_2\mathbf{D}\mathbf{P}\mathbf{B}_1\mathbf{B}_2\mathbf{K}_{8-w}^{\mathrm{T}}\right) + \right. \\ &\quad \left. \mathbf{K}_{8-h}\mathbf{B}_2^{\mathrm{T}}\mathbf{B}_1^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\mathbf{D}\left(\mathbf{X}_3\mathbf{D}\mathbf{P}\mathbf{B}_1\mathbf{B}_2\mathbf{J}_w^{\mathrm{T}} + \mathbf{X}_4\mathbf{D}\mathbf{P}\mathbf{B}_1\mathbf{B}_2\mathbf{K}_{8-w}^{\mathrm{T}}\right)\right]\mathbf{T}^{\mathrm{T}} \quad (3.73)\end{aligned}$$

$$\begin{aligned}\hat{\mathbf{X}} &= \mathbf{T}\left[\left(\mathbf{J}_h\mathbf{B}_2^{\mathrm{T}}\mathbf{B}_1^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\mathbf{D}\mathbf{X}_1 + \mathbf{K}_{8-h}\mathbf{B}_2^{\mathrm{T}}\mathbf{B}_1^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\mathbf{D}\mathbf{X}_3\right)\mathbf{D}\mathbf{P}\mathbf{B}_1\mathbf{B}_2\mathbf{J}_w^{\mathrm{T}} + \right. \\ &\quad \left. \left(\mathbf{J}_h\mathbf{B}_2^{\mathrm{T}}\mathbf{B}_1^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\mathbf{D}\mathbf{X}_2 + \mathbf{K}_{8-h}\mathbf{B}_2^{\mathrm{T}}\mathbf{B}_1^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\mathbf{D}\mathbf{X}_4\right)\mathbf{D}\mathbf{P}\mathbf{B}_1\mathbf{B}_2\mathbf{K}_{8-w}^{\mathrm{T}}\right]\mathbf{T}^{\mathrm{T}} \quad (3.74)\end{aligned}$$

by selecting the expression which requires less computations for the given w and h. According to the authors, in comparison to the brute-force method (DCT decoding, followed by pixel-domain MC, followed by DCT encoding), the usage of this scheme provides a reduction of the computational load of about 32% for the worst case and 46.8% on the average case.

Another proposal was presented by Assunção and Ghanbari, who achieved a reduction of the computational cost by approximating the elements of matrices $\mathbf{H}_{ij}$ to finite sums of powers of 2, with a maximum distortion of 1/32 [6]. By doing so, only basic integer operations, such as *shift-right* and *additions*, are required to compute eq. 3.23. According to these authors, when compared with the pixel-domain approach, the proposed fast DCT-domain MC method using approximate matrices provides a reduction of about 81% in the overall computational cost.

3.3.2 Bit rate and quality adaptation

The main objective of a bit rate or quality adaptation transcoder is to convert a high rate bit stream into a lower rate sequence of bits. This reduction makes it possible to store the encoded video in a restricted storage medium or to transmit it over constrained or heterogeneous networks, where the transmission bandwidth is not always sufficient to meet the requirements imposed by the original encoded video sequence. Such system accepts a precoded bit stream at its input and produces a scaled bit stream, with new constraints that are not usually known *a priori*, i.e., at the time of creation of the original bit stream [106, 107].

Two basic principles are usually adopted by these transcoders:

1. the information of the original bit stream should be exploited as much as possible;

2. the resulting image quality of the lower bit rate video sequence should be as high as possible, or as close as possible to a hypothetical bit stream that

would have been created if the original source video was encoded at the target reduced rate.

Consequently, several different requirements may have to be met by the output signal of this transcoder, such as bit rates, minimum visual quality, constrained delay, limited buffer size and periodic random access points.

As it was briefly described in section 3.2, a straightforward approach to implement this transcoder is to fully decode the input sequence and then re-encode it with a new set of coding parameters (see fig. 3.2). Such implementation, usually referred to as the *cascaded* approach, implies the maximum computational cost. As a consequence, several other alternative approaches are usually adopted in order to avoid the significant number of operations required by the pixel-domain cascaded transcoder.

Most of these approaches are based on a partial decoding of the input bit stream up to the dequantized DCT coefficients, followed by a new requantization of the coefficients with a coarser quantization step. However, while for INTRA type frames of a given video stream the overall computational cost is reduced, with no significant penality in the coding efficiency, the same is generally not true in INTER type frames. As it was shown in section 3.2.1, the requantization change in INTRA type images, which are used as reference for predictive INTER type frames, will imply that the displaced frame difference of INTER type frames will have to be recomputed in order to close the prediction loop and avoid the introduction of drift and inherent degradation of the video quality. As a consequence, such a process usually involves the whole recomputation of the prediction residuals, the re-estimation of the motion vectors and the selection of new coding modes, which often require similar computational costs as those of a full encoder.

In general, four different types of architectures are usually adopted by these bit rate adaptation transcoders [106, 107]:

Architecture I – transcoder with truncation of the high frequency DCT coefficients;
Architecture II – transcoder with requantization of the DCT coefficients;
Architecture III – transcoder with re-encoding of the reconstructed pictures, using the motion vectors and the coding modes of the original bit stream;
Architecture IV – transcoder with re-encoding of the reconstructed pictures, using the motion vectors extracted from the original high-quality video stream but applying a new coding mode decision.

While the first two architectures are often referred to as *open loop* transcoders, since

the encoder loop is not closed in the re-encoding process, the other two architectures are denominated as *closed loop* transcoders. The former significantly reduce the computational cost of the structure, at the expense of introducing loss in picture quality, since they do not recompute the prediction difference signal in the re-encoding loop. On the other hand, the last two architectures close the encoding loop and do not introduce any drift error, thus presenting significantly better quality performances. However, such advantage is usually achieved at the expense of an increased computational cost, since more processing functions will have to be recomputed.

Architecture I: Truncation of high frequency DCT coefficients

The block diagram of this architecture is presented in fig. 3.17. As it was referred before, the bit rate adaptation is achieved by discarding part of the high frequency DCT coefficients of the input bit stream. To accomplish this objective, the *VLD parser* of the top branch of the block diagram does not perform any decoding task, but simply determines the codeword lengths that correspond to the AC coefficients of the frame under processing. Meanwhile, the *bit allocation analyzer* computes an AC bit usage profile ($\mathrm{PV_N}$), consisting of a running sum of the number of bits required to encode the AC frequency DCT coefficients until macroblock N, as shown in fig. 3.18:

$$\mathrm{PV_N} = \sum \mathrm{AC\_bits} \tag{3.75}$$

In addition, the analyzer also counts the sum of all bits that are required to encode the frame under processing: TB (total bits). After all macroblocks of the considered frame have been analyzed, a target value ($\mathrm{TV_{AC}}$) of the bits used to encode the AC frequency DCT coefficients of each frame is calculated as [106, 107]:

$$\mathrm{TV_{AC}} = \mathrm{PV_{LS}} - \alpha.\mathrm{TB} - \mathrm{B_{EX}} \tag{3.76}$$

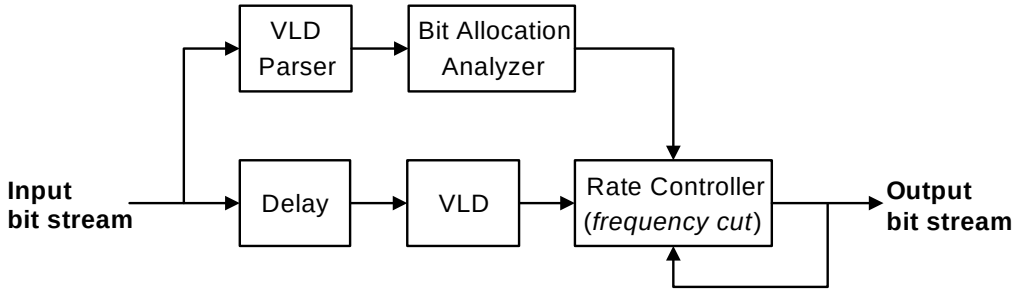


Figure 3.17: Architecture I: Truncation of high frequency DCT coefficients [106, 107].

where PV_{LS} is the same profile value used in the last macroblock, α is the percentage by which the precoded bit stream is to be reduced and B_{EX} is the amount of bits by which the previous frame missed its desired target.

With this newly scaled profile, the *rate controller* simply discards all the DCT coefficients that exceed the computed scale profile. All the codewords other than the AC coefficients should be kept, in order to maintain a compliant bit stream.

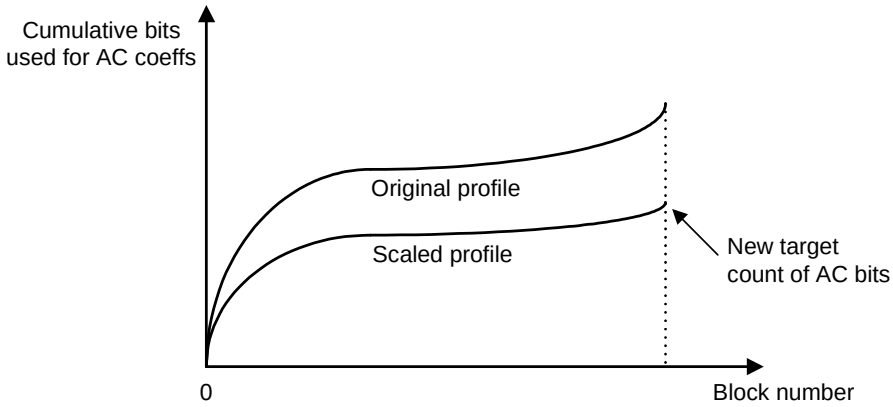


Figure 3.18: AC bit usage profile [106, 107].

Hence, the main advantage of this approach is related to its simplicity: it does not need to perform variable length decoding and inverse quantization. Its computational cost is even lower than the one corresponding to the decoder. However, it usually introduces a serious amount of drift error, since the decoding loop is not closed along the transcoding process.

Architecture II: Requantization of the DCT coefficients

The block diagram of architecture II is presented in fig. 3.19. This transcoder slightly increases the computational cost by including a *VLD*, a modified *rate controller* (requantizer), implemented by cascading an inverse and a direct quantization modules, and a *VLC*. Just like architecture I, it also performs a preliminary variable length decoding pass on the input bit stream and computes a similar scaled target profile of cumulative codeword bits versus the macroblock count, to be used for rate control (see fig. 3.18).

However, contrary to architecture I, the quantized DCT coefficients are inverse quantized as soon as this profile is computed and requantized with a coarser quantizer scale. This new quantizer scale is adaptively determined by making slight adjustments after the processing of every macroblock, so that the scaled target profile

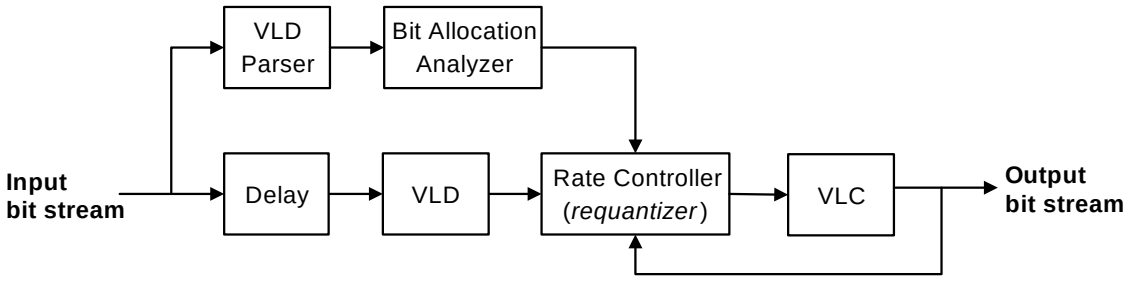


Figure 3.19: Architecture II: Requantization of the DCT coefficients [106, 107].

is respected as the several macroblocks within the frame are processed [106, 107]:

$$Q_{\mathrm{N}} = Q_{\mathrm{NOM}} + G \times \left(\sum_{N-1} \mathrm{BU} - \mathrm{PV}_{\mathrm{N-1}} \right), \tag{3.77}$$

where Q_{N} is the quantization factor for macroblock N, Q_{NOM} is an estimate of the new nominal quantization factor for the frame under processing, $\sum_{N-1} \mathrm{BU}$ is the cumulative amount of coded bits up to macroblock $(N-1)$, and G is a gain factor that controls how tightly the profile curve is tracked through the frame. Q_{NOM} is usually initialized to an average guess value before the processing of the very first frame, and updated for the next frame by setting it to Q_{LS} (the quantization factor for the last macroblock) from the frame just completed.

The coarsely requantized block of DCT coefficients is then variable-length encoded to generate the scaled bit stream.

Architecture III: Recoding with old motion vectors and old coding modes

The block diagram of architecture III is shown in fig 3.20. In this architecture, the reconstructed frames are obtained from the normal decoding procedure, by extracting the motion vectors and the macroblocks coding modes from the original bit stream. The scaled bit stream is then obtained by re-encoding the reconstructed frames at the encoder side of the transcoder, by re-using the old motion vectors and macroblock coding modes, but using a different requantization step, thus minimizing the introduction of drift error.

Consequently, the advantage of this architecture, when compared to fully decoding and re-encoding, is that no motion estimation and coding mode decisions will have to be carried out.

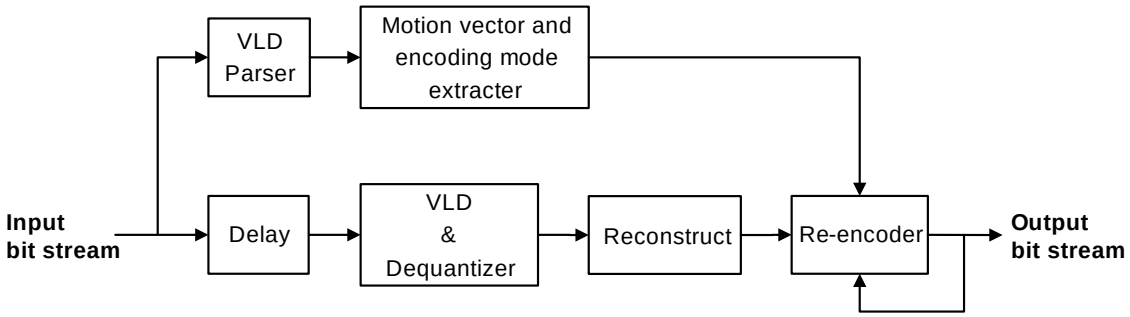


Figure 3.20: Architecture III: Recoding with old motion vectors and old coding modes [106, 107].

Architecture IV: Recoding with old motion vectors and re-avaliation of the coding modes

Architecture IV can be regarded as an improved and modified version of architecture III, since it computes new macroblock coding modes during the re-encoding stage, at the expense of additional computational cost. When the encoder side of the transcoder compresses the video signal, it selects the new coding mode based on the rate-distortion compromise that was selected for the considered operating point. At higher bit rates, the encoder sends more motion vectors and reduces the residual difference signal energy. On the other hand, the motion vectors are too expensive to be encoded at lower rates. Consequently, while at higher rates optimal coding mode decision for a given macroblock is more likely to favor bidirectional motion compensation over forward motion compensation, at lower rates the opposite is usually preferred. As a consequence, bidirectional motion compensation is often changed to forward or backward motion compensation in these transcoders.

As it can be easily realized, the computational cost involved in this architecture is quite close to the one corresponding to the cascaded transcoder, except for the computational demanding motion estimation part, that may be absent. To further enhance the video quality, the transcoder may also perform the whole or partial re-estimation of the motion vectors. Alternatively, the motion estimation process can re-use the old motion vectors, extracted from the input bit sequence. These motion vectors can then be used as starting points to a subsequent refinement operation, thus providing a considerable quality improvement at the expense of a slight increase in the system computational cost. Some of these motion vector refinement techniques, both in the pixel and in the transform DCT-domain, will be described in section 3.3.5.

Architectures comparison

The performance of the described bit rate and quality adaptation transcoders was assessed and compared by Sun, Kwok, and Zdepski [106] and by Sun, Chen, and Chiang [107]. In table 3.2 it is presented a summary of the computational blocks required by the four considered transcoder architectures, as well as the cascaded approach, denoted by "C". While the first architecture presents minimum computational cost, at the expense of the inherent drift problem, the requantization approach, introduced in architecture II, requires the introduction of the *requantizer* and of the *variable decoder/encoding* modules. However, it still keeps the processing entirely in the DCT-domain, with moderate drift degradation. Even so, the usage of coarser quantization steps often produces annoying coding artifacts, such as blocking and ringing effects, or even severely blurs the video content. To remove this drift distortion, the decoding and re-encoding loops have to be implemented in the transcoder. This is achieved at the expense of the integration of frame memory devices and of a motion compensation module in architectures III and IV.

To assess the quality performance (in terms of the PSNR) presented by the several considered architectures, Sun et al. [106, 107] evaluated seven coding setups for each test sequence:

Ref. A – Compression at 15 Mbps using the original sequence;
Ref. B – Compression at 4 Mbps using the original sequence;
Ref. C – Cascaded pixel-domain transcoder from 15 Mbps to 4 Mbps;
Arch. I – Cascaded transcoder from 15 Mbps to 4 Mbps with Architecture I;
Arch. II – Cascaded transcoder from 15 Mbps to 4 Mbps with Architecture II;
Arch. III – Cascaded transcoder from 15 Mbps to 4 Mbps with Architecture III;
Arch. IV – Cascaded transcoder from 15 Mbps to 4 Mbps with Architecture IV.

The first three experiments are used as references, to compare the scaling perfor-

Table 3.2: Processing modules required by the several considered bit rate and quality adaptation transcoders [106, 107].

Processing Module	Architectures				
	I	II	III	IV	C
VLD bit stream parser and rate controller	•	•	•	•	•
Quantizer/dequantizer/VLD/VLC		•	•	•	•
Decoding/encoding loops/frame memory/motion compensation			•	•	•
Mode decision				•	•
Motion estimation					•

Table 3.3: PSNR measure when transcoding from 15 Mbps to 4 Mbps [106, 107].

Output Sequence	PSNR [dB]		
	Flower Garden	Bicycle	Bus
Ref. A	37.02	35.12	37.74
Ref. B	29.60	27.26	30.44
Ref. C	29.34	27.14	30.22
Arch. I	20.89	20.18	21.87
Arch. II	27.41	25.04	28.44
Arch. III	29.19	27.02	30.14
Arch. IV	29.32	27.11	30.20

mances. The obtained results, presented in table 3.3, were obtained with an MPEG-2 encoder for the test video sequences illustrated in fig. 3.21, each one composed of 150 frames.

As it can be observed, the first significant quality improvement was obtained by applying the requantization procedure in architecture II, as opposed to simply truncation of the DCT coefficients, performed in architecture I. The next level of improvement is the result of closing the encoding loop, in architecture III. As it can also be observed, the advantage of using architecture IV is not very significant: the main difference lies in the procedure related to the decision of the macroblock coding mode. As it was referred before, the performance gap is the result of adjustments on the overhead bits of the motion vectors at various bit rate operation points. Moreover, one can realize that architecture IV is already very close to the theoretical optimal case, corresponding to a compression at 4 Mbps from the original sequence (Ref. B), thus leaving little room for further improvements on this architecture, in what concerns the output video quality.

Although the results corresponding to the simplest structure (architecture I)

(a) *Flower Garden.*

(b) *Bicycle.*

(c) *Bus.*

Figure 3.21: Test video sequences adopted in the evaluation of the bit rate and quality adaptation architectures.

that are presented in table 3.3 evidence a significant loss in the obtained video quality, recent advances proposed by Eleftheriadis and Anastassiou [17] have proven a significant improvement of its usefulness. These authors presented a dynamic rate-shaping approach to select the DCT coefficients that should be kept in the transcoding process [17]. Two different methods to select this set were proposed: the Constrained Dynamic Rate Shaping (CDRS) and the Unconstrained Dynamic Rate Shaping (GDRS). While in the first approach a set of coefficients at the end of each block is removed from the video bit stream, with the breakpoint of each block selected by the transcoder, in the second technique there is a 64-element binary vector indicating which coefficients will be kept. In both approaches, the optimization process consists of an iterative minimization of a Lagrangian cost function, computed as $L = D + \lambda R$. This incremental cost reduction takes into account the number of bits (R) resulting from the inclusion of the run length codes of the considered DCT coefficients, as well as the resulting distortion (D) associated to the coefficients truncation [17].

Transform-domain architectures

The previously described architectures for bit rate and quality adaptation can be further simplified if the processing modules required by each specific structure are entirely operated in the DCT-domain, as it was presented in section 3.2.2. In fact, considering that most of the described architectures perform the bit rate adaptation by directly manipulating the received DCT coefficients and by recalling the linearity properties of both the DCT and the motion compensation operations, it becomes clear that all the involved manipulations can be easily performed without the need to perform either the direct or inverse DCT. Moreover, such approach may even avoid the introduction of additional distortion, as a consequence of precision and round off errors that are always introduced during the computation of the DCT and IDCT operations (see section 3.1.2).

One example of such architecture was already presented in fig. 3.5 [6]. It is represented again, in fig. 3.22, but with the inclusion of the previously described *Bit Allocation Analyzer*. In this architecture, the motion compensation operation is entirely performed in the DCT-domain, using an algorithm quite similar to the one described in subsection B of section 3.3.1 (page 69). As it was previously observed, this module merges both the decoding and the encoding loops in one single loop, which, in turn, avoids the introduction of any additional drift. Such mechanism is implemented by means of a frame update memory, to store the residual signal that is obtained as a result of the transcoding operation that affects both the INTRA

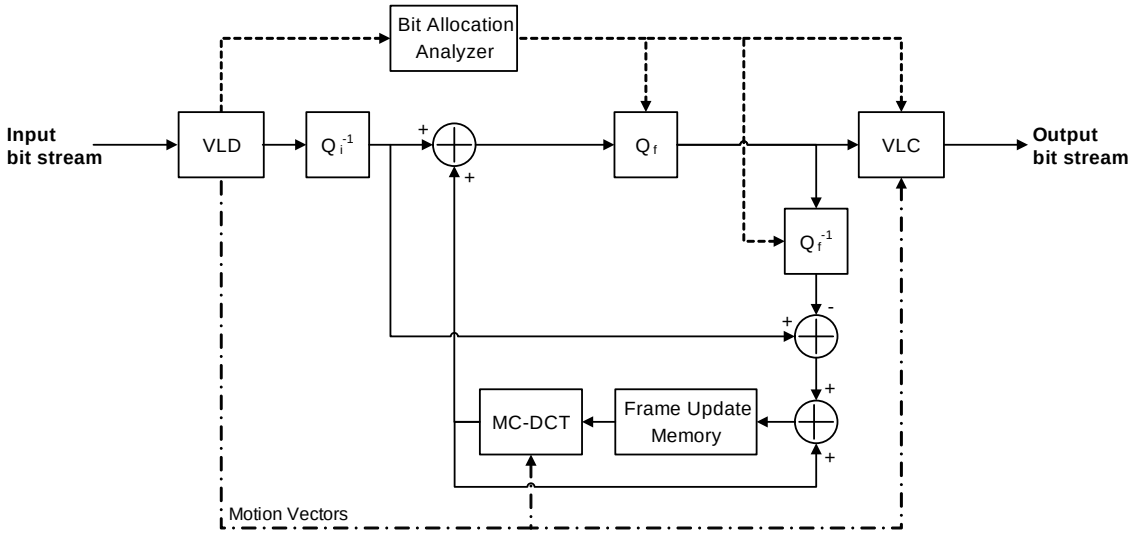


Figure 3.22: Frequency domain bit rate and quality adaptation architecture [6].

and the INTER type frames.

By adopting these DCT-domain processing structures, it is possible to obtain significant gains, both in terms of the computational cost of the transcoder and in terms of output video quality.

Requantization distortion error

Many of the described quality and bit rate adaptation techniques rely on a new requantization of the received DCT coefficients. In particular, such algorithms usually apply a coarser quantization step $Q_f > Q_i$ to the AC frequency DCT coefficients of the received blocks, in order to convert or adapt an incoming high rate bit stream into a lower sequence of bits.

As it was previously described (see, for example, section 3.2.1), the involved computational cost of such procedure greatly depends on the type of picture under processing, i.e. I, P or B type frame. Nevertheless, independently of the considered image type, the requantization process is usually implemented as a cascade sequence of an inverse quantizer Q_i^{-1} followed by a coarser quantizer Q_f:

$$\mathbf{X}_{out} = Q_f \left[Q_i^{-1} \left(\mathbf{X}_{in} \right) \right]. \tag{3.78}$$

However, this coarse requantization of the DCT coefficients of the incoming video sequence is often responsible for the introduction of an inherent degradation effect, consisting of three distinct components [4]:

1. some nonzero coefficients of the input frame that become zero after coarse requantization;

2. quantization error;

3. requantization error.

While the first two are well-known causes of distortion, the later is specific to this requantization procedure. In fact, under certain specific conditions, the requantization can introduce an additional error that would not be introduced if the original DCT coefficients had been directly quantized with the target coarser quantization step size. Such situation is illustrated in fig. 3.23.

In this figure, the reconstruction levels of the DCT coefficients A and B, quantized with a first quantizer step size Q_1, are Q_1^A and Q_1^B, respectively. On the other hand, if the coarser quantizer step size Q_2 had been used, both coefficients would be reconstructed to the same level $Q_2^A = Q_2^B$. However, if A and B are first quantized with Q_1 and then coarsely requantized with Q_2, the reconstruction level of A will be the same as that of direct coarser quantization with Q_2, i.e., $Q_{12}^A = Q_2^A$. On the other hand, in the case of B, the reconstruction value after requantization will be different from that of direct coarser quantization i.e., $Q_{12}^B \neq Q_2^B$. Hence, while the requantization error of A is zero, the error introduced for coefficient B is not.

Therefore, whenever the coarse quantization interval entirely contains the finer one, the direct coarse quantization and requantization distortions will be equal. On the other hand, if the finer interval overlaps two different coarser intervals, the requantization distortion will be larger whenever the reconstruction value of the first quantization and the original coefficient fall into different coarser quantization intervals [4]. Since the first quantization is performed independently of subsequent requantization procedures, the requantization error cannot be avoided. The distortion resulting from this error may lead to a drop in picture quality of about 1.5 dB [4].

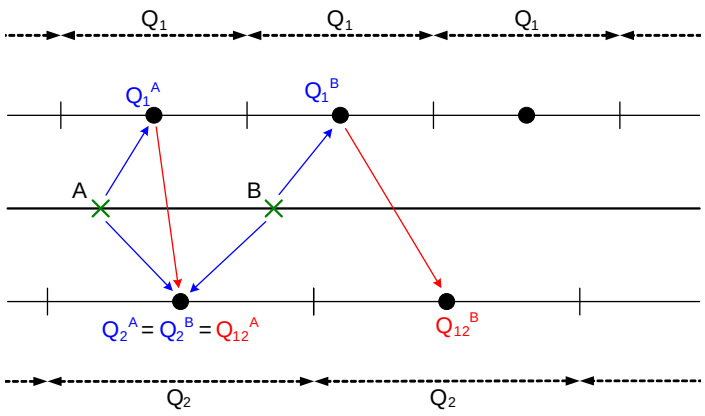


Figure 3.23: Requantization error.

3.3.3 Space scaling

Spatial frame scale is often required to reduce the image resolution by a given scale factor ($\mathcal{S}_{\mathcal{F}}$) before transmission or storage, thus reducing the output bit rate [1, 95]. From a straightforward point of view, image resizing of a compressed video sequence can be performed by cascading: *i)* a video decoder block; *ii)* a pixel-domain resizing module, to process the decompressed sequence; and *iii)* an encoding module, to compress the resized video. Such structure is illustrated in fig. 3.24. To implement

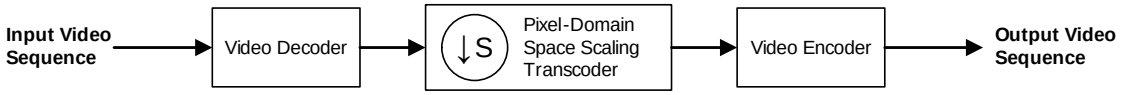


Figure 3.24: Cascaded pixel-domain space-scaling transcoder.

this downscaling process directly in the DCT-domain, alternative approaches have also been proposed [1, 95, 113]. These techniques can be classified in three different categories:

- **filtering and downsampling**: by using an arbitrary low-pass filter, the image is downsampled by dropping alternate pixels in both directions [59, 66, 114];
- **averaging and downsampling**: by representing every set of $\left(\mathcal{S}_{\mathcal{F}_x} \times \mathcal{S}_{\mathcal{F}_y}\right)$ pixels by a single output pixel, corresponding to its average value [11, 24, 63, 99, 116];
- **DCT decimation**: by discarding a subset of high order AC frequency DCT coefficients, a group of blocks composited by $(K \times K)$ DCT coefficients will form a downsampled $(N \times N)$ DCT coefficients block, with $K \leq N$ [16, 46–48, 73, 85].

From a strict digital signal processing point of view, the first two techniques may be regarded as somewhat equivalent approaches. However, they are applied in different contexts, as it will be seen in the following.

A - Space scaling using filtering and downsampling

Most of the techniques within this category adopt a traditional cascaded *filtering and downsampling* process. As an example, Natarajan and Vasudev proposed a scaling scheme, with $\mathcal{S}_{\mathcal{F}} = 2$, that replaces every set of four neighboring blocks $\mathbf{B}_{i,j}$,

with $i, j = 0 \ldots 1$, by a single $(N \times N)$ transform block $\mathbf{B}$ [66]:

$$\hat{\mathbf{B}} = \begin{pmatrix} \mathbf{F}_1 & \mathbf{F}_2 \end{pmatrix} \begin{pmatrix} \mathbf{B}_{0,0} & \mathbf{B}_{0,1} \\ \mathbf{B}_{1,0} & \mathbf{B}_{1,1} \end{pmatrix} \begin{pmatrix} {\mathbf{F}_1}^{\mathrm{T}} \\ {\mathbf{F}_2}^{\mathrm{T}} \end{pmatrix} \tag{3.79}$$

where $\mathbf{F}_1$ and $\mathbf{F}_2$ are the proposed filter matrices that were specially derived to downscale the original DCT coefficients blocks $\mathbf{B}_{i,j}$ [66]. $\mathbf{F}_1$ and $\mathbf{F}_2$ only differ in the signs of their entries:

$$\mathbf{F}_1 = \begin{bmatrix} 0.500 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.453 & 0.208 & -0.037 & 0.011 & 0.000 & -0.011 & 0.037 & -0.208 \\ 0.000 & 0.500 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & -0.500 \\ -0.159 & 0.396 & 0.257 & -0.049 & 0.000 & 0.049 & -0.257 & -0.396 \\ 0.000 & 0.000 & 0.500 & 0.000 & 0.000 & 0.000 & -0.500 & 0.000 \\ 0.106 & -0.176 & 0.384 & 0.245 & 0.000 & -0.245 & -0.384 & 0.176 \\ 0.000 & 0.000 & 0.000 & 0.500 & 0.000 & -0.500 & 0.000 & 0.000 \\ -0.090 & 0.139 & -0.188 & 0.433 & 0.000 & -0.433 & 0.188 & -0.139 \end{bmatrix} \tag{3.80}$$

$$\mathbf{F}_2 = \begin{bmatrix} 0.500 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ -0.453 & 0.208 & 0.037 & 0.011 & 0.000 & -0.011 & -0.037 & -0.208 \\ 0.000 & -0.500 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.500 \\ 0.159 & 0.396 & -0.257 & -0.049 & 0.000 & 0.049 & 0.257 & -0.396 \\ 0.000 & 0.000 & 0.500 & 0.000 & 0.000 & 0.000 & -0.500 & 0.000 \\ -0.106 & -0.176 & -0.384 & 0.245 & 0.000 & -0.245 & 0.384 & 0.176 \\ 0.000 & 0.000 & 0.000 & -0.500 & 0.000 & 0.500 & 0.000 & 0.000 \\ 0.090 & 0.139 & 0.188 & 0.433 & 0.000 & -0.433 & -0.188 & -0.139 \end{bmatrix} \tag{3.81}$$

To reduce the involved computational cost, Natarajan and Vasudev also proposed an approximation to these filter matrices, by rounding off their entries to a restricted set of sub-powers of 2: 0, $\pm\frac{1}{8}$, $\pm\frac{1}{4}$ and $\pm\frac{1}{2}$.

Another filtering approach was proposed by Martucci [59]. By applying the symmetric convolution property of the DCT (see section 2.5), it was shown that the discrete cosine transform of a scaled block can be obtained by pointwise multiplying the original block by the transform coefficients of a 2-D anti-aliasing filter. The obtained block $\hat{\mathbf{B}}$ is then obtained after a downsampling stage with $\mathcal{S}_{\mathcal{F}} = 2$, by following a procedure entirely equivalent to a pixel-domain downsampling process, i.e. by manipulating the DCT coefficients of the filtered block $\tilde{\mathbf{B}}$, as follows [59]:

$$\hat{\mathbf{B}} = \frac{\tilde{\mathbf{B}}(m) - \tilde{\mathbf{B}}(N-m)}{\sqrt{2}}, \quad \text{with} \;\; m, n = 0, 1, \ldots, \left(\frac{N}{2} - 1\right). \tag{3.82}$$

By folding over the coefficients and pointwise subtracting in both dimensions, the procedure gives rise to a $\left(\frac{N}{2} \times \frac{N}{2}\right)$ DCT coefficients block, corresponding to the scaled target block ($\mathcal{S}_{\mathcal{F}} = 2$). This is, in fact, one of the main drawbacks of this

method: since most video standards adopt a fixed $(N \times N)$ pixels block structure, it is usually required that both the input and output signals of these transcoders are structured in blocks with $(N \times N)$ elements.

More recently, Yin et al. [114] proposed an alternative approach based on the concept of frequency synthesis, to transform an input macroblock, consisting of four blocks, each one with (8×8) DCT coefficients, into a single (8×8) DCT coefficients block. The corresponding computations are actually performed on the rows and columns of the input macroblock using separable 1-D filters $(f_1,\ f_2)$. Let $\mathbf{B}_1$ and $\mathbf{B}_2$ denote the input vectors of size N. The output block $\hat{\mathbf{B}}$, also of size $(N \times N)$, is computed as:

$$\hat{\mathbf{B}} = \mathbf{f}_1 \cdot \mathbf{B}_1 + \mathbf{f}_2 \cdot \mathbf{B}_2 \tag{3.83}$$

where:

$$f_1(k,p) = \sum_{i=0}^{N-1} \psi_p^N(i) \cdot \psi_k^{2N}(i) \tag{3.84}$$

$$f_2(k,p) = \sum_{i=0}^{N-1} \psi_p^N(i) \cdot \psi_k^{2N}(i+N) \tag{3.85}$$

and

$$\psi_m^N(i) = \sqrt{\frac{2}{N}}\ \xi(m) \cos\left(\frac{2i+1}{2N}\ m\pi\right) \tag{3.86}$$

with $\xi\,(m)$ defined in eq. 2.13.

B - Space scaling using averaging and downsampling

Somewhat similar approaches were proposed by other authors, who presented some improvements in the algorithm efficiency by adopting simpler average low-pass filters. According to this technique, every $\left(\mathcal{S}_{\mathcal{F}x} \times \mathcal{S}_{\mathcal{F}y}\right)$ pixels block is represented by a single pixel with their average value, thus leading to an *averaging and downsampling* scheme [11, 24, 63, 99, 116].

By denoting $\mathbf{b}_{0,0}$, $\mathbf{b}_{0,1}$, $\mathbf{b}_{1,0}$ and $\mathbf{b}_{1,1}$ the four adjacent (8×8) pixels blocks, as shown in fig. 3.25, and considering a downsampling process characterized by $\mathcal{S}_{\mathcal{F}x} = \mathcal{S}_{\mathcal{F}y} = 2$, each (2×2) pixels sub-block will be replaced by their average value, in order to obtain the downsampled (8×8) pixels block $\hat{\mathbf{b}}$, as shown in fig. 3.25. This downsampling operation can be represented by the following expression:

$$\hat{\mathbf{b}} = \sum_{i=0}^{1} \sum_{j=0}^{1} \mathbf{h}_{i,j}.\mathbf{b}_{i,j}.\mathbf{w}_{i,j} \tag{3.87}$$

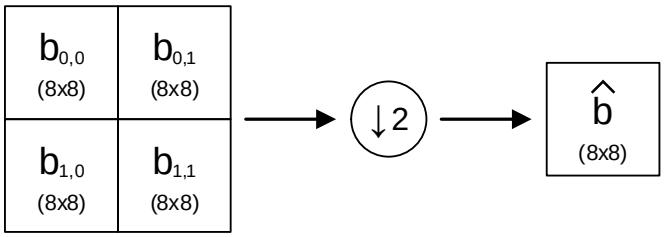


Figure 3.25: Downsampling four adjacent blocks to obtain a single (8×8) block.

In this equation, $\mathbf{h}_{i,j}$ and $\mathbf{w}_{i,j}$ are the downsampling filter matrices:

$$\mathbf{h}_{0,0} = \mathbf{h}_{0,1} = {\mathbf{w}_{0,0}}^{\mathrm{T}} = {\mathbf{w}_{1,0}}^{\mathrm{T}} = \frac{1}{2}\begin{bmatrix} \mathbf{u}_{4\times 8} \\ \mathbf{0}_{4\times 8} \end{bmatrix} \tag{3.88}$$

$$\mathbf{h}_{1,0} = \mathbf{h}_{1,1} = {\mathbf{w}_{0,1}}^{\mathrm{T}} = {\mathbf{w}_{1,1}}^{\mathrm{T}} = \frac{1}{2}\begin{bmatrix} \mathbf{0}_{4\times 8} \\ \mathbf{u}_{4\times 8} \end{bmatrix} \tag{3.89}$$

where $\mathbf{0}_{4\times 8}$ is a (4×8) zero matrix and $\mathbf{u}_{4\times 8}$ is defined as:

$$\mathbf{u}_{4\times 8} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}. \tag{3.90}$$

These scaling schemes can be directly implemented in the DCT-domain, by applying the DCT operator to both sides of eq. 3.87, as follows:

$$\mathrm{DCT}\left(\hat{\mathbf{b}}\right) = \mathrm{DCT}\left(\sum_{i=0}^{1}\sum_{j=0}^{1} \mathbf{h}_{i,j}.\mathbf{b}_{i,j}.\mathbf{w}_{i,j}\right). \tag{3.91}$$

By taking into account that the DCT is a linear and orthonormal transform, it is distributive over matrix multiplication. Hence, eq. 3.91 can be rewritten as:

$$\hat{\mathbf{B}} = \sum_{i=0}^{1}\sum_{j=0}^{1} \mathbf{H}_{i,j}.\mathbf{B}_{i,j}.\mathbf{W}_{i,j}, \tag{3.92}$$

where $\mathbf{X} = \mathrm{DCT}(\mathbf{x})$.

One advantage of this approach is the fact that all the filter matrices $\mathbf{H}_{i,j}$ and $\mathbf{W}_{i,j}$ can be pre-computed and stored in memory. Hence, given the blocks $\mathbf{B}_{i,j} = \mathrm{DCT}(\mathbf{b}_{i,j})$ from the input video sequence, for $i, j = 0 \ldots 1$, the downsampled DCT coefficients matrix $\hat{\mathbf{B}} = \mathrm{DCT}(\hat{\mathbf{b}})$ can be obtained by matrix multiplication.

It should be noted, however, that although the filter matrices $\mathbf{h}_{i,j}$ and $\mathbf{w}_{i,j}$ are highly sparse, $\mathbf{H}_{i,j} = \mathrm{DCT}(\mathbf{h}_{i,j})$ and $\mathbf{W}_{i,j} = \mathrm{DCT}(\mathbf{w}_{i,j})$ are not sparse at all. Consequently, this filtering and averaging approach may have a significant computational cost, often comparable to downsampling in the spatial-domain using the straightforward cascaded approach. This fact usually makes this technique impractical to be implemented in real-life transcoding devices. To overcome this drawback, some authors have proposed optimized factorizations of the filter matrices to minimize the involved computational cost [63]. Even so, the involved cost is usually still too high, since such factorizations will imply highly non-regular matrix manipulation algorithms.

Despite all these observations, it should be also noted that these techniques do not take advantage of the spatial redundancies that are generally present in the images under processing. As a consequence, the same number of computations will be always required, independently of the number of non-null AC frequency DCT coefficients that are present in each block.

C - Space scaling using DCT decimation

Another popular approach, usually referred to as *DCT decimation*, takes advantage of the fact that most of the pixels energy is concentrated in the lower frequency band of each block. Consequently, several video transcoding manipulations that have been proposed make use of this characteristic, by discarding some high order AC frequency DCT coefficients and retaining only a sub-set of the low order terms. As a consequence, this approach has also been denoted as Modified Inverse Transformation and Decimation (MITD) [98] and has been particularly adopted in DCT-domain inverse motion compensation [51, 54, 57] and spatial resolution downscaling [16, 46–48, 73, 85] schemes.

By following this principle, Dugad and Ahuja [16] proposed an efficient DCT decimation scheme for spatial downscaling in the DCT-domain. Their algorithm starts by extracting the (4×4) low frequency DCT coefficients from the four original adjacent blocks: $\mathbf{b}_{0,0}$, $\mathbf{b}_{0,1}$, $\mathbf{b}_{1,0}$ and $\mathbf{b}_{1,1}$ (see fig. 3.25). Each of these sub-blocks is then inverse DCT transformed, in order to obtain a sub-set of the original $(N \times N)$ pixels area that will represent the scaled version of the original block. The four (4×4) pixels sub-blocks are then merged and combined together, in order to obtain an (8×8) pixels block. The corresponding (8×8) DCT coefficients block is then obtained by applying the discrete cosine transform to this (8×8) pixels block.

This algorithm was shown to provide significant performance improvements over the previously described filtering schemes and can be formalized as follows. Let

$\mathbf{B}_{0,0}$, $\mathbf{B}_{0,1}$, $\mathbf{B}_{1,0}$ and $\mathbf{B}_{1,1}$ represent the four original (8×8) DCT coefficients blocks; $\mathbf{B}'_{0,0}$, $\mathbf{B}'_{0,1}$, $\mathbf{B}'_{1,0}$ and $\mathbf{B}'_{1,1}$ represent the four (4×4) low-frequency sub-blocks of $\mathbf{B}_{0,0}$, $\mathbf{B}_{0,1}$, $\mathbf{B}_{1,0}$ and $\mathbf{B}_{1,1}$, respectively:

$$\mathbf{B}'_{i,j} = \begin{bmatrix} \mathbf{I}_{4\times4} & \mathbf{0}_{4\times4} \end{bmatrix} \cdot \mathbf{B}_{i,j} \cdot \begin{bmatrix} \mathbf{I}_{4\times4} \\ \mathbf{0}_{4\times4} \end{bmatrix} \tag{3.93}$$

where $\mathbf{I}_{n\times n}$ is an $(n \times n)$ identity matrix and $\mathbf{0}_{n\times n}$ is an $(n \times n)$ null matrix. Let $\mathbf{b}'_{i,j} = \text{IDCT}\left(\mathbf{B}'_{i,j}\right)$, for $i, j = 0, \ldots, 1$.Then,

$$\hat{\mathbf{b}} = \begin{bmatrix} \left[\mathbf{b}'_{0,0}\right]_{4\times4} & \left[\mathbf{b}'_{0,1}\right]_{4\times4} \\ \left[\mathbf{b}'_{1,0}\right]_{4\times4} & \left[\mathbf{b}'_{1,1}\right]_{4\times4} \end{bmatrix}_{8\times8} \tag{3.94}$$

is the downscaled version of

$$\mathbf{b} = \begin{bmatrix} \left[\mathbf{b}_{0,0}\right]_{8\times8} & \left[\mathbf{b}_{0,1}\right]_{8\times8} \\ \left[\mathbf{b}_{1,0}\right]_{8\times8} & \left[\mathbf{b}_{1,1}\right]_{8\times8} \end{bmatrix}_{16\times16} . \tag{3.95}$$

To compute $\hat{\mathbf{B}} = \text{DCT}\left(\hat{\mathbf{b}}\right)$ directly from $\mathbf{B}_{0,0}$, $\mathbf{B}_{0,1}$, $\mathbf{B}_{1,0}$ and $\mathbf{B}_{1,1}$ (i.e., from $\mathbf{B}'_{0,0}$, $\mathbf{B}'_{0,1}$, $\mathbf{B}'_{1,0}$ and $\mathbf{B}'_{1,1}$), the following expression can be applied:

$$\hat{\mathbf{B}} = \mathbf{T}.\hat{\mathbf{b}}.\mathbf{T}^{\mathrm{T}} \tag{3.96}$$

where $\mathbf{T}$ is the (8×8) DCT operator matrix. It can be decomposed into two equal-sized (8×4) sub-matrices $\mathbf{T}_L$ and $\mathbf{T}_R$, where

$$\mathbf{T}_L = \mathbf{T}. \begin{bmatrix} \mathbf{I}_{4\times4} \\ \mathbf{0}_{4\times4} \end{bmatrix} \quad ; \quad \mathbf{T}_R = \mathbf{T}. \begin{bmatrix} \mathbf{0}_{4\times4} \\ \mathbf{I}_{4\times4} \end{bmatrix} \tag{3.97}$$

denote the four left and the four right columns, respectively, of the 8-point DCT operator $\mathbf{T}$. By denoting by $\mathbf{T}_4$ the 4-point DCT operator matrix, equation 3.96 can be evaluated as:

$$\hat{\mathbf{B}} = \begin{bmatrix} \mathbf{T}_L & \mathbf{T}_R \end{bmatrix} \begin{bmatrix} \hat{\mathbf{b}}_1 & \hat{\mathbf{b}}_2 \\ \hat{\mathbf{b}}_3 & \hat{\mathbf{b}}_4 \end{bmatrix} \begin{bmatrix} \mathbf{T}_L^{\mathrm{T}} \\ \mathbf{T}_R^{\mathrm{T}} \end{bmatrix} \tag{3.98}$$

$$= \begin{bmatrix} \mathbf{T}_L & \mathbf{T}_R \end{bmatrix} \begin{bmatrix} \mathbf{T}_4^{\mathrm{T}}.\mathbf{B}'_{0,0}.\mathbf{T}_4 & \mathbf{T}_4^{\mathrm{T}}.\mathbf{B}'_{0,1}.\mathbf{T}_4 \\ \mathbf{T}_4^{\mathrm{T}}.\mathbf{B}'_{1,0}.\mathbf{T}_4 & \mathbf{T}_4^{\mathrm{T}}.\mathbf{B}'_{1,1}.\mathbf{T}_4 \end{bmatrix} \begin{bmatrix} \mathbf{T}_L^{\mathrm{T}} \\ \mathbf{T}_R^{\mathrm{T}} \end{bmatrix} \tag{3.99}$$

$$\begin{aligned} = {} & \left(\mathbf{T}_L\mathbf{T}_4^{\mathrm{T}}\right) \mathbf{B}'_{0,0} \left(\mathbf{T}_L\mathbf{T}_4^{\mathrm{T}}\right)^{\mathrm{T}} + \left(\mathbf{T}_L\mathbf{T}_4^{\mathrm{T}}\right) \mathbf{B}'_{0,1} \left(\mathbf{T}_R\mathbf{T}_4^{\mathrm{T}}\right)^{\mathrm{T}} + \\ & \left(\mathbf{T}_R\mathbf{T}_4^{\mathrm{T}}\right) \mathbf{B}'_{1,0} \left(\mathbf{T}_L\mathbf{T}_4^{\mathrm{T}}\right)^{\mathrm{T}} + \left(\mathbf{T}_R\mathbf{T}_4^{\mathrm{T}}\right) \mathbf{B}'_{1,1} \left(\mathbf{T}_R\mathbf{T}_4^{\mathrm{T}}\right)^{\mathrm{T}} \end{aligned} \tag{3.100}$$

Dugad and Ahuja [16] have also proposed a matrix decomposition to convert the previous expression into an alternative form, so that the matrices involved in the previous matrix multiplication become more sparse and the computational cost is reduced. This scheme is formulated by defining the pair of matrices $\mathbf{C}$ and $\mathbf{D}$, characterized by a significant number of null elements, so that $\mathbf{T}_L{\mathbf{T}_4}^{\mathrm{T}} = \mathbf{C}+\mathbf{D}$ and $\mathbf{T}_R{\mathbf{T}_4}^{\mathrm{T}} = \mathbf{C}-\mathbf{D}$:

$$\mathbf{C} = \begin{bmatrix} 0.7071 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.2960 & 0.0000 & 0.0162 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.5594 & 0.0000 & -0.0690 \\ 0.0000 & 0.0000 & 0.7071 & 0.0000 \\ 0.0000 & -0.2492 & 0.0000 & 0.3468 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.1964 & 0.0000 & 0.6122 \end{bmatrix} \qquad \mathbf{D} = \begin{bmatrix} 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.6407 & 0.0000 & -0.0528 & 0.0000 \\ 0.0000 & 0.7071 & 0.0000 & 0.0000 \\ -0.2250 & 0.0000 & 0.3629 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.1503 & 0.0000 & 0.5432 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.7071 \\ -0.1274 & 0.0000 & -0.2654 & 0.0000 \end{bmatrix} \tag{3.101}$$

Hence, eq. 3.100 can be formulated as:

$$\begin{aligned} \hat{\mathbf{B}} &= (\mathbf{C}+\mathbf{D})\,\mathbf{B}'_{0,0}(\mathbf{C}+\mathbf{D})^{\mathrm{T}} + (\mathbf{C}+\mathbf{D})\,\mathbf{B}'_{0,1}(\mathbf{C}-\mathbf{D})^{\mathrm{T}} + \\ &\quad (\mathbf{C}-\mathbf{D})\,\mathbf{B}'_{1,0}(\mathbf{C}+\mathbf{D})^{\mathrm{T}} + (\mathbf{C}-\mathbf{D})\,\mathbf{B}'_{1,1}(\mathbf{C}-\mathbf{D})^{\mathrm{T}} && (3.102) \\ &= \left[\mathbf{C}\left(\mathbf{B}'_{0,0}+\mathbf{B}'_{1,0}\right)+\mathbf{D}\left(\mathbf{B}'_{0,0}-\mathbf{B}'_{1,0}\right)\right](\mathbf{C}+\mathbf{D})^{\mathrm{T}} + \\ &\quad \left[\mathbf{C}\left(\mathbf{B}'_{0,1}+\mathbf{B}'_{1,1}\right)+\mathbf{D}\left(\mathbf{B}'_{0,1}-\mathbf{B}'_{1,1}\right)\right](\mathbf{C}-\mathbf{D})^{\mathrm{T}} && (3.103) \\ &= \mathbf{X}(\mathbf{C}+\mathbf{D})^{\mathrm{T}} + \mathbf{Y}(\mathbf{C}-\mathbf{D})^{\mathrm{T}} && (3.104) \\ &= (\mathbf{X}+\mathbf{Y})\,\mathbf{C}^{\mathrm{T}} + (\mathbf{X}-\mathbf{Y})\,\mathbf{D}^{\mathrm{T}} && (3.105) \end{aligned}$$

where:

$$\mathbf{X} = \mathbf{C}\left(\mathbf{B}'_{0,0}+\mathbf{B}'_{1,0}\right)+\mathbf{D}\left(\mathbf{B}'_{0,0}-\mathbf{B}'_{1,0}\right) \tag{3.106}$$

and

$$\mathbf{Y} = \mathbf{C}\left(\mathbf{B}'_{0,1}+\mathbf{B}'_{1,1}\right)+\mathbf{D}\left(\mathbf{B}'_{0,1}-\mathbf{B}'_{1,1}\right) \tag{3.107}$$

By applying such decomposition, Dugad and Ahuja have shown that this down-sampling scheme requires only about 1.25 multiplications and 1.25 additions per pixel of the original non-scaled image.

Ridge [85] has also proposed an extension to this formulation by using a decomposition rather similar to the one presented by Merhav and Bhaskaran [63]. It provides a further reduction of the involved computational cost and still yields visual improvements, by reducing the aliasing effect. Even so, the main disadvantage of this approach is concerned with the inherent introduction of blocky artifacts, mainly in image areas with complex texture regions, due to the discard of high order AC frequency coefficients.

D - Discussion

Despite the several different strategies that have been presented, most of these proposals are only directly applied to scaling operations using a scale factor given by an integer power of two ($\mathcal{S}_{\mathcal{F}} = 2,\ 4,\ 8,\ 16,$ etc). Nevertheless, downscaling operations using any other arbitrary integer scaling factor are often required. As a consequence, in the last few years some techniques have been proposed in order to implement these algorithms for any integer scale factor [48, 72, 92, 102, 103]. However, although many of these proposals provide good video quality for scaling factors equal to integer powers of two, their performance significantly degrades when other scaling factors are applied. Furthermore, they often suffer from computational inefficiency in their processing: either by storing a large amount of data matrices [102] or by operating with large amount of data [72].

Another important issue is concerned with the block structure adopted by these algorithms. The $(N \times N)$ pixels block structure adopted by some digital image (e.g. JPEG [32]) and video standards (such as H.261 [29], H.263 [30], MPEG-1 Video [26] and MPEG-2 Video [27]), requires that both the input original frame and the output downscaled frame, together with all the data structures associated to the processing algorithm, are organized in $(N \times N)$ pixels blocks. As it was referred before, some of the described algorithms cannot comply with this important requisite.

Hence, contrary to the most recent proposals [48, 72, 92, 102, 103], in section 5.2 it will be proposed an innovative algorithm that offers a reliable and very efficient video downscaling method for any arbitrary integer scaling factor, with particularly notable performances for scaling factors other than integer powers of 2. The algorithm adopts an averaging and downsampling approach that is performed by directly processing the received DCT coefficients, in order to minimize the introduction of any inherent distortion. Moreover, the proposed algorithm also offers an alternative method for minimizing the computational cost: this method is implemented by restricting the involved operations in order to avoid spurious and useless computations and by only performing those that are really needed to obtain the output values. Furthermore, independently of the adopted scaling factor $\mathcal{S}_{\mathcal{F}}$, all the involved steps are properly tailored so that all operations are performed using $(N \times N)$ coefficient blocks, in order to comply the operations with most image and video standards. An optional and possible combination of the presented algorithm with high order AC frequency DCT coefficients discarding techniques is also proposed. These techniques, usually adopted by *DCT decimation* approaches, provide a flexible and often required complexity scalability feature.

Nevertheless, independently of the adopted methodology, the implementation

of a space scaling transcoder to process pre-coded video sequences requires a corresponding adaptation of the decoded motion vectors. The resulting motion-compensation prediction scheme of the processed video sequence has to comply with the new space resolution of the output sequence. To minimize the involved computational cost, such procedure is usually performed by re-using the decoded motion vectors and by applying several possible composition techniques, as it will be described in the following section.

3.3.4 Motion vector composition

Frequently, when an arbitrarily time or space manipulation function is performed over a precoded video sequence, each macroblock in the transcoded video is obtained as a composition of several macroblocks from the input sequence. Since the motion pattern of the composited macroblocks may be different from the original pattern, a set of new Motion Vectors (MVs) is generally required to implement the motion compensated prediction mechanism. However, it is often desired that these new MVs correlate well with the MVs of the compositing macroblocks. In fact, although the usual motion estimation algorithms may be used to estimate the new MV, this approach is often avoided for real-time applications, since motion estimation is computationally very intensive.

Consequently, fast methods have been devised to estimate the new MVs directly from the set of MVs of the precoded video with small computational cost and with satisfactory performance. In all these methods, it is necessary to take into account that the borders of the composited macroblock may not align with the original macroblock grid of the precoded frame. In fact, a given set of precoded macroblocks may be partially or wholly used to composite the transcoded macroblocks. On the other hand, the transcoded macroblocks may be composited of a different number of macroblocks from the precoded video.

Several fast MV compositing methods have been proposed in the literature [108]. In general, those methods can be classified into two different categories, based on the adopted process to derive the new motion vector from the precoded MVs. On the one hand, each new motion vector may be obtained by averaging the corresponding MVs from the precoded video. Such composition scheme may be obtained either as a simple average or as a more complex mean operation, that takes into account other characteristics of the compositing macroblocks, such as the area, the coding computational cost, etc. On the other hand, the new MV may be directly chosen from the set of original MVs, corresponding to the compositing macroblocks. Independently of the adopted procedure to obtain the new MV, and at the cost of

additional computations, this MV can be further refined by performing an additional motion estimation tuning operation over a restricted search area.

In the following, it will be presented a brief description of some of the composition techniques that have been proposed in the past few years. In all the presented methods, the symbol $\mathcal{P}$ denotes the set of compositing macroblocks that are involved in the transcoding operation, decoded from the original video stream.

A - Simple Average (SA)

According to this method, each new MV of the transcoded stream is obtained by computing the simple arithmetic mean of all the corresponding MVs from the precoded video:

$$\widehat{v}_{\mathrm{SA}} = \frac{1}{\#\mathcal{P}} \sum_{p \in \mathcal{P}} v(p) \tag{3.108}$$

where $\#\mathcal{P}$ denotes the cardinal of the set $\mathcal{P}$. According to the work presented by Shen et al. [101], this method usually provides good estimates of the new motion vector when all the corresponding MVs are similar and well aligned. However, it tends to perform poorly if the compositing macroblocks undergo different MVs [112].

B - Area Weighted Average (AWA)

The area weighted average method interpolates the new MV by weighting each precoded MV with the corresponding area of the compositing macroblock. According to this method, the larger the compositing area, the more the influence a precoded macroblock will have over the motion-compensated residue of the transcoded macroblock. This method can be stated as [52, 112]:

$$\widehat{v}_{\mathrm{AWA}} = \frac{\sum_{p \in \mathcal{P}} v(p) \times A(p)}{\sum_{p \in \mathcal{P}} A(p)} \tag{3.109}$$

where $A(p)$ denotes the portion (in terms of number of pixels) of the precoded macroblock $\mathrm{MB}(p)$ involved in the composition of the transcoded macroblock.

C - Spatial Activity-Area Weighted Average (SAAWA)

When the overlapping areas of the several compositing macroblocks are very similar, the MVs obtained with the Area Weighted Average (AWA) technique may be not meaningful. Consequently, several authors [13, 52, 101] have proposed different approaches to estimate the new MVs by also taking into account the spatial activity of each compositing macroblock. Their justification for this method lies in the

fact that, when their motion vectors deviate from the optimal values, the motion-compensated residues of macroblocks with high spatial activity increase faster than those corresponding to macroblocks with low spatial activity. Therefore, more emphasis should be given to the MVs of compositing macroblocks with higher spatial activity.

Hence, not only does this method rely on the usage of each macroblock relative compositing area, but it also considers the spatial activity to weight the influence of each compositing MV. Such spatial activity can be assessed by several means, such as [101]:

- by reconstructing the spatial-domain samples and measuring the energy or the gradients within the corresponding blocks;
- by directly using the received DCT coefficients and counting the number of non-null AC coefficients, thus avoiding the inherent cost of computing the inverse DCT;
- by summing the absolute values of the received AC coefficients.

As an example, Liang et al. [52] proposed a composition algorithm that uses the number of non-null AC coefficients of each DCT encoded block to estimate the spatial activity of the original macroblocks in the computation of the composited motion vector $\widehat{v}_{\text{SAWA}}$:

$$\widehat{v}_{\text{SAAWA}} = \frac{\sum_{p\in\mathcal{P}} v(p) \times \alpha(p) \times A(p)}{\sum_{p\in\mathcal{P}} \alpha(p) \times A(p)} \tag{3.110}$$

where $\alpha(p)$ denotes the number of non-null AC coefficients of macroblock $\text{MB}(p)$.

Tan et al. also proposed the usage of another activity measure, corresponding to the product of the quantization step-size and the number of bits used to encode the macroblock, denominated by QB-measure [108]:

$$\widehat{v}_{\text{SAAWA}} = \frac{\sum_{p\in\mathcal{P}} v(p) \times A(p) \times \text{QB}(p)}{\sum_{p\in\mathcal{P}} A(p) \times \text{QB}(p)} \tag{3.111}$$

where $\text{QB}(p)$ denotes the product of the quantization step-size $Q(p)$ with the total number of bits $B(p)$ used to encode the macroblock: $\text{QB}(p) = Q(p) \times B(p)$.

They claimed that not only does this measure provide similar transcoding performances when compared with the usage of the techniques proposed by Shen et al., but it also presents the advantage of being significantly easier to be obtained from the precoded video stream, since it does not require further computations.

D - Maximum Area (MA) / QB-Area (MQBA)

Despite the significant number of different possible approaches to composite the new MVs by averaging the set of corresponding precoded vectors, it is frequently observed that such vectors lead to poorer results than those obtained by using the original vectors [115]. Alternative techniques have therefore been proposed that obtain the new MV by selecting one from the set of precoded vectors. As an example, one of such techniques selects the motion vector that belongs to the macroblock with the largest compositing area $A(p)$, as follows:

$$\widehat{v}_{\mathrm{MA}} = v(p^*), \text{ where } p^* = \underbrace{\arg\max}_{p \in \mathcal{P}} \{A(p)\} \tag{3.112}$$

Another technique selects the MV that belongs to the macroblock with the largest product of compositing area $A(p)$ and $QB(p)$ measure [13], as follows:

$$\widehat{v}_{\mathrm{MQBA}} = v(p^*), \text{ where } p^* = \underbrace{\arg\max}_{p \in \mathcal{P}} \{\mathrm{QB}(p) \times A(p)\} \tag{3.113}$$

E - Median (MDN)

Another proposed approach suggests choosing the precoded MV that is most similar, on average, with the rest of the MVs corresponding to the set of compositing macroblocks $\mathcal{P}$. The implementation of this method can be stated as:

$$\widehat{v}_{\mathrm{MDN}} = v(p^*), \text{ where } p^* = \underbrace{\arg\min}_{p \in \mathcal{P}} \left\{ \sum_{p' \in \mathcal{P}} \| v(p) - v(p') \| \right\} \tag{3.114}$$

Since the implementation of this method resembles the computation of the median value of the compositing MVs, this technique has been denoted in the literature as the *median* method. Nevertheless, it should be noted that its computation does not entirely conform with the usual definition of the median estimate.

F - Comparative analysis

Despite all the presented compositing schemes, it should be noted that other possible alternatives could be equally adopted; each considering different cost functions to obtain the composited MV leading to the best prediction of the current transcoded macroblock. Some examples of such alternative schemes include weighted averaging strategies based on coding complexity, maximum area methods, maximum coding complexity schemes or weighted median methods.

Tan et al. [108] presented a performance evaluation of some of these methods when applied to MV composition, for the particular case of a space-scaling transcoding system. Such evaluation was performed by computing the PSNR of both the transcoded video stream and of the original uncompressed video, downsampled with the same scale factor. A set of standard benchmark video sequences was used on this evaluation. According to the obtained experimental results, the techniques that are based on the computation of a weighted average (SA, AWA and SAAWA) are generally outperformed by the MQBA and MDN methods. In fact, the median (MDN) method itself tends to perform consistently better than the MQBA, in terms of the PSNR measure.

Furthermore, it should be noted that whenever MV re-estimation is affordable, MVs of better precision can still be obtained at the cost of more computations. Once again, several possible schemes can be devised to estimate the best composited MV: either using the original data, decoded from the received video stream, or from one of the compositing schemes previously described. As an example, Wong and Au [112] proposed a fast motion re-estimation block-matching method that is performed over a restricted set of MV candidates. Such set includes both the n original compositing MVs and an additional MV candidate. This extra candidate is estimated as a weighted average of the precoded vectors using the reciprocal of their corresponding Sum of Absolute Differences (SAD) measures as weights:

$$v_{n+1} = \frac{\sum_{i=1}^{n} v_i \cdot \frac{1}{\mathrm{SAD}_i}}{\sum_{i=1}^{n} \frac{1}{\mathrm{SAD}_i}} \tag{3.115}$$

with

$$\mathrm{SAD}(v_i) = \sum_{m=0}^{(N-1)} \sum_{n=0}^{(N-1)} \left| F_{curr}(x+m, y+n) - F_{prev}(x+v_i^x+m, y+v_i^y+n) \right|, \tag{3.116}$$

where F_{curr} and F_{prev} denote the current and previously coded frames, respectively. If this extra MV, v_{n+1}, is not equal to one of the first n candidates, the value of the corresponding SAD (SAD_{n+1}) is computed, using the calculated value v_{n+1} as the MV. Finally, the MV within this set with the lowest SAD value is selected as the final motion vector v:

$$v = \underbrace{\arg\min}_{i \leq n+1} \{\mathrm{SAD}_i\} \tag{3.117}$$

It is worth noting that such re-estimation method could be equally applied using any of the compositing schemes previously described.

The presented MV refinement schemes, based on re-estimation techniques, involve, however, a significant amount of computations. As a consequence, its usage

often raises a difficult design compromise: it implies a trade-off between the overall system computational cost and the resulting encoding quality and efficiency. As it will be described in the following section, several different approaches have been proposed to accomplish this objective, by processing the received data either in the pixel or in the transform-domains.

3.3.5 Motion estimation

Among the several transcoding operations that have been addressed over the last few years to process already compressed video sequences directly in the DCT-domain, the Motion Estimation (ME) function has deserved a special attention by the scientific community. In fact, many video processing systems frequently require the computation of new MVs to encode the new video stream. Most ME techniques that have been applied up until now in the majority of the transcoding systems adopt the composition schemes previously described in section 3.3.4 to reuse the MVs decoded from the received video stream. However, as it was previously observed, most of these schemes frequently lead to non-optimal motion compensated prediction results, due to the mismatch that often exists between the prediction and residual components.

To overcome this quality loss without performing a full motion estimation operation, motion vector refinement schemes have been proposed [13, 41, 115]. The search windows typically adopted by these MV refinements are relatively small (e.g. $[-2, +2]$), compared to original search windows. Nevertheless, not only does this keep the additional computational cost at low and tolerable levels, but it still provides a significant amount of the achievable gains. As a consequence, such schemes have been frequently adopted by most bit rate reduction transcoding systems to improve the output video quality, as well as by spatial or temporal resolution reduction architectures.

However, few algorithms have been presented to estimate entirely new MVs in the absence of any precoded MV (e.g. INTRA to INTER frame conversion) or when the original MVs do not have any correlation with the motion activity in the new encoded frame (e.g. video composition, logo insertion [88], etc.). In such cases, the simple but computational intensive cascaded architecture is often applied, by performing the estimation of the new motion vectors in the pixel-domain.

Recently, some proposals have been presented to perform this motion estimation directly in the DCT-domain. Despite the different target application area, consisting of photogrammetric techniques for image registration of JPEG compressed aerial photos, Reeves and Kubik proposed an algorithm entirely different from the

traditional approaches, by trying to apply matching techniques using correlation and convolution techniques in the DCT-domain [83, 84].

Koc and Liu have also recently presented a new approach, denominated by DCT pseudo-phases technique, and applied it to DCT-domain motion estimation [12, 38, 39, 41]. However, unlike other fast block-matching motion estimation methods (such as logarithmic, three-step, cross, etc. [7]), which simply pick up a set of displacement candidates out of all possible displacement values in terms of the minimum SAD values, these techniques employ the sinusoidal orthogonal principles to extract the shift information from the pseudo-phases hidden in the DCT coefficients of the processed frames. Due to the novelty and peculiarity of this technique, the corresponding algorithm will be described in more detail.

In the following, it will be presented a brief overview of the main proposed approaches to implement the motion estimation procedure using the data received from the incoming video stream and to process it directly in the compressed DCT-domain.

A - Cascaded transcoders

Similarly to what happens in other video transcoding processing techniques, the most straightforward approaches to perform the (re-)estimation of the motion vectors of a given video stream are the cascaded architectures. Usually, to reduce the computational complexity, these structures re-use the motion vectors extracted from the incoming bit stream (v_i) and perform a prediction of the new motion vector (v_p) (see section 3.3.4). However, as it was referred before, these simple motion vector reuse schemes usually introduce a considerable quality degradation. Moreover, although the optimal motion vector can still be obtained by applying a new exhaustive motion estimation procedure, this alternative is often not desirable due to its high computational cost. As a consequence, a motion vector refinement stage is usually adopted, in order to improve the accuracy of the motion vectors that will be used by the output bit stream (v_o).

Pixel-domain cascaded transcoders

The block diagram of a pixel-domain cascaded architecture applying MV prediction techniques is presented in fig. 3.26. The advantage of these schemes is their ability to provide a video quality level that is comparable to the one that would be obtained by performing a new full-search motion estimation, but at the cost of considerably less computations. Even so, the incoming bit stream is first fully decoded and the block-matching motion (re-)estimation algorithm is performed in

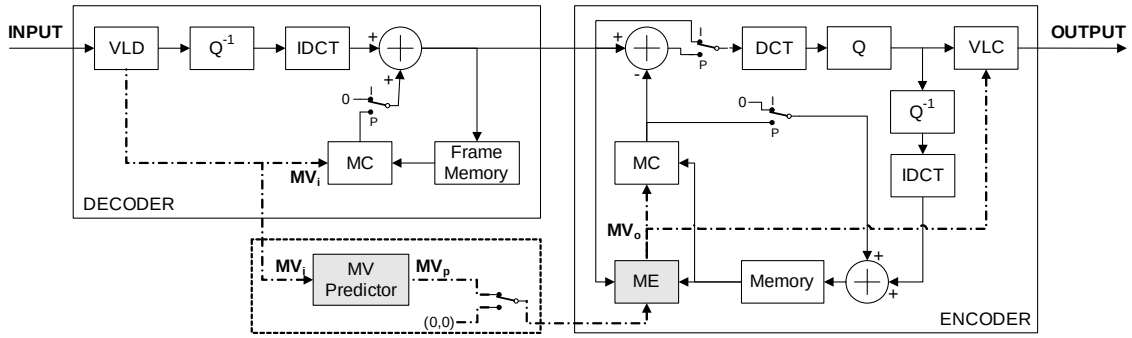


Figure 3.26: Cascaded pixel-domain motion estimation transcoder.

the pixel-domain, similarly to what happens in the front-end encoder of the video communication system.

An example of the application of this type of structures was presented by Youn, Sun, and Lin [115], who have proposed a cascaded pixel-domain architecture to be applied in a frame skipping transcoding structure. The proposed architecture refines the base motion vector obtained from the motion vectors decoded from the incoming bit stream, by using a maximum area composition technique ($\widehat{v}_{\mathrm{MA}}$) (see eq. 3.112). It then applies a set of fast search motion estimation algorithms within a small search window.

Chen, Chu, and Pan [13] have proposed a similar architecture that obtains the composited motion vector prediction (v_p) from a maximum spatial activity ($\widehat{v}_{\mathrm{MQBA}}$) composition algorithm (see eq. 3.113). It the applies a fast search algorithm to refine this prediction and to obtain a more accurate motion vector (v_o), in order to minimize the prediction error of the output video sequence.

Transform-domain cascaded transcoders

The described cascaded structure with a motion vector refinement module can equally be implemented in the transform-domain, by transposing all the involved operations to the DCT-domain. Such structure, illustrated in fig. 3.27, can be easily obtained from the block diagram of fig. 3.26, just by: *i)* removing all DCT and IDCT processing modules; and *ii)* replacing the MC module by its equivalent MC-DCT processing block, operating in the DCT-domain, as described in subsection B of section 3.3.1 (page 69). Once the incoming video frames are decoded to the DCT-domain, the predicted motion vectors (v_p) can be obtained in an entirely similar way, as it was done in the pixel-domain architectures. Such predictions can then be refined by applying them to a DCT-domain motion re-estimation algorithm.

As an example, Seo and Kim proposed a motion vector refinement algorithm

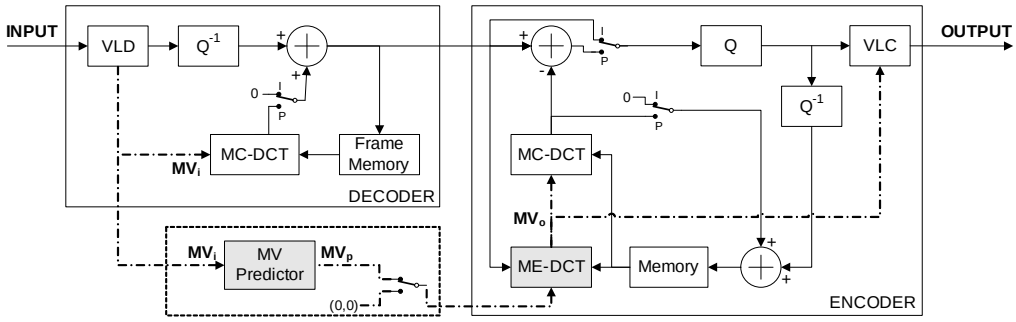


Figure 3.27: Cascaded transform-domain motion estimation transcoder.

based on a fast block-matching mechanism, entirely implemented in the DCT-domain [93, 94]. Such algorithm makes use of a predicted motion vector (v_p), obtained from the decoded motion vectors received from the incoming bit stream and from a mean/median compositing scheme, to determine a restricted set of candidate macroblocks to be checked by the block-matching algorithm. The adopted estimation algorithm is based on the minimization of the Mean Squared Error (MSE) criterion and on a computationally efficient method to extract the motion-compensated DCT candidate blocks. By extensively re-using the overlapped information corresponding to the set of 15×16 pixels that are shared with the previously considered adjacent candidate macroblock, they have shown that it is computationally feasible to implement a block-matching algorithm in the transform-domain, provided that the number of candidate macroblocks is reduced and that all these candidates are adjacent to each other.

B - Convolution based transcoder

As it was referred before, Reeves and Kubik proposed a convolution based technique to perform image matching, which was initially developed for image registration of JPEG compressed aerial photos [83, 84]. By taking into account that image registration is a problem entirely similar to estimating the motion vectors in predictive video coding, such technique can be equally applied to motion estimation in video transcoding systems.

The proposed technique is based on the convolution-multiplication relationship for the DCT and DST transforms, previously formulated by Martucci [58] (see section 2.5), which states the equivalence between the multiplication operation in the DCT-domain and the computational costly symmetric convolution operation in the pixel-domain. Hence, by providing a feasible alternative to compute the convolution of two shifted images, this technique makes it possible to easily determine the position of the peak within the convolution result, which indicates the magnitude

of the disparity of the two considered images.

However, the proposed symmetric convolution approach presents one important limitation: it only provides useful results for integer displacements of one position [83, 84]. Even so, some interest should still be devoted to this approach, mainly for motion re-estimation applications, where the search area is usually rather limited around a previously predicted candidate motion vector.

As it was described in section 2.5, this convolution-multiplication relationship between the pixel and the DCT-domains implies a symmetric extension of a N-point base sequence at each of its ends, in order to obtain a periodic sequence with length $2N$. As it was also observed, this symmetric convolution can be defined in a number of different ways, depending on how each end of the sequence to be convolved is extended. According to what was described in section 2.5, each point of symmetry can coincide with a sample point (odd symmetry) or can fall between two samples (even symmetry). Furthermore, the sequence can be extended in a symmetrically or anti-symmetrically way.

In the following, the description of the algorithm will be performed using 1-D signals. Such simplification is adopted in order to ease the explanation and does not introduce any limitation on the algorithm. Furthermore, instead of using the orthogonal form of the even type-II DCT kernel (see table 2.2):

$$X(m) = \sum_i \mathcal{C}_{2e}(m,i).x(i) \tag{3.118}$$

$$= \sqrt{\frac{2}{N}}\,\xi(m)\sum_{i=0}^{N-1} x(i)\cos\left(\frac{m\left(i+\frac{1}{2}\right)\pi}{N}\right), \tag{3.119}$$

with $m \in \{0,\ldots,N-1\}$ and $\xi(p)$ as defined in eq. 2.13, it will be adopted the convolution form of the second kind even DCT kernel matrix (see table 2.4), as defined by Martucci [58]:

$$X_{\mathcal{C}_{2e}}(m) = \sum_i \mathcal{C}_{2e}(m,i).x(i) \tag{3.120}$$

$$= 2\sum_{i=0}^{N-1} x(i)\cos\left(\frac{m\left(i+\frac{1}{2}\right)\pi}{N}\right). \tag{3.121}$$

In fact, it can be shown that these two DCT definitions are equivalent and a direct correspondence can be obtained between each other, with $X_{\mathcal{C}_{2e}}(0) = 2\sqrt{N}.X(0)$ and $X_{\mathcal{C}_{2e}}(m) = \sqrt{2N}.X(m)$.

In the considered case, the symmetric convolution operation is defined so that each of the pixel-domain convolved sequences $x(i)$ and $y(i)$ is extended at both ends

with half-sample symmetry, with:

$$c(i) = x(i) \circledS y(i) \tag{3.122}$$

$$= \mathcal{C}_{1e}{}^{-1}\{X_{\mathcal{C}_{2e}}(m)\, Y_{\mathcal{C}_{2e}}(m)\}, \tag{3.123}$$

where Ⓢ denotes the skew-circular convolution, defined in eq. 2.58, $X_{\mathcal{C}_{2e}}(m)$ and $Y_{\mathcal{C}_{2e}}(m)$ are the second kind even DCT transforms of $x(i)$ and $y(i)$, defined in eq. 3.120, and $\mathcal{C}_{1e}{}^{-1}$ denotes the inverse type-I even cosine transform, defined as:

$$Z_{\mathcal{C}_{1e}}(m) = 2\sum_{i=0}^{N} \xi^2(i)\, z(i) \cos\left(\frac{mi\pi}{N}\right) \tag{3.124}$$

$$z(i) = \mathcal{C}_{1e}{}^{-1}\{Z_{\mathcal{C}_{1e}}(m)\} = \frac{1}{N}\sum_{m=0}^{N} \xi^2(m)\, Z_{\mathcal{C}_{1e}}(m) \cos\left(\frac{im\pi}{N}\right). \tag{3.125}$$

As it was reported by Reeves and Kubik [83, 84], this symmetric convolution only provides useful results for integer displacement values of zero or one. Nevertheless, these authors have proposed a methodology to obtain fractional disparities within this region.

In order to compute the disparity value with greater accuracy, it is necessary to estimate the exact position (p_M) corresponding to the maximum value of the correlation function between two points: p_0 and p_1. However, since this technique only provides correlation values for these two samples, the exact position of p_M must be inferred from the output values at these two points. In fact, from a rather simplistic point of view, one can assume that sample p_0 will have a higher correlation value when p_M is closer to p_0, whereas sample p_1 will have a higher value when p_M is closer to p_1. Reeves and Kubik modeled the variation of this correlation function with two linear functions: the first passing through sample p_0, with positive slope α, and the second passing through sample p_1, with slope $-\alpha$.

Hence, for 2-D signals, the above algorithm is implemented by defining the convolution function:

$$c(i,j) = x(i,j) \circledS y(i,j) \tag{3.126}$$

obtained by extending eq. 3.122 for the 2-D case, where:

$$y(i,j) = x(i + d_i, j + d_j) \tag{3.127}$$

and d_i and d_j are the horizontal and vertical disparities, respectively. Similarly, the auto-convolution function can be defined as:

$$a(i,j) = x(i,j) \circledS x(i,j). \tag{3.128}$$

To simplify the following description, it will be presented the computation of the horizontal disparity (d_i) by assuming a null vertical disparity (d_j). Hence, by computing the convolution values $c(p_0)$ and $c(p_1)$ from the decoded DCT coefficients, corresponding to images $x(i,j)$ and $y(i,j)$, as well as the auto-convolution values $a(p_0)$ and $a(p_1)$ of the undisplaced image $x(i,j)$, an expression for the horizontal disparity d_n can be derived as follows:

$$d_n = \frac{a(p_0) - a(p_1) + c(p_1) - c(p_0)}{2\,[a(p_0) - a(p_1)]}. \tag{3.129}$$

A similar expression could equally be derived for the vertical disparity d_j.

Unfortunately, from the experimental assessment that was carried out by these authors, it was concluded that the above formulation consistently underestimates the true value of the disparity, since the slope of this relation varies from block to block. As a consequence, they have proposed the usage of a correcting factor that was empirically estimated from the results obtained using several test video sequences. However, as they have recognized, the accuracy of the obtained results is not enough to consider the usage of this technique in real life video processing systems. Furthermore, this technique also suffers from an important limitation: it cannot differentiate between a negative and a positive disparity, since it can only reveal the magnitude of the disparity, not its sign. As a consequence, such limitation must be overcome by other external means.

Consequently, despite the high interest that this technique could initially raise to reduce the computational cost in motion estimation applications performed directly in the DCT-domain, the poor accuracy of the results that it provides makes it unsuitable for most current video transcoding applications.

C - Least squares based transcoder

Reeves [83] has also proposed an alternative iterative scheme that performs image matching by applying a Least Squares Estimation (LSE) technique. Similarly to the convolution based technique, described in the previous subsection, this methodology was developed to be applied in digital image processing applications, namely, in the computation of the disparity of two shifted signals, for image registration of JPEG compressed aerial photos.

Based on this approach, in section 5.3.1 it will be presented a new motion estimation algorithm, directly operating in the DCT-domain, to be applied in video processing applications. Such algorithm computes the new MVs by applying a LSE technique and by following an approach somewhat similar to the one proposed by

Reeves. By adopting such methodology, it is possible to compute new MVs using the DCT coefficients directly obtained from the DCT-H.26x/MPEG-x video streams.

Furthermore, the proposed algorithm also provides the possibility to efficiently take advantage of the energy compaction property provided by the DCT. In fact, since most of the pixels energy is concentrated on the lower frequency band of the encoded blocks, most high frequency coefficients are set to zero after the quantization step. Hence, to reduce its computational effort, the proposed ME algorithm may consider only an arbitrary subset of non-null DCT coefficients, thus providing an adaptive capability to trade computational cost with the resulting accuracy of the MVs.

D - DCT pseudo-phases based transcoder

This technique was proposed by Koc and Liu and is based on the shift property of the Fourier Transform (FT) of a given signal $x(t)$ and its delayed version $x(t-\tau)$ [12, 38, 39, 41]:

$$\mathcal{F}\left\{x\left(t-\tau\right)\right\} = e^{-j\omega\tau}\mathcal{F}\left\{x\left(t\right)\right\}, \tag{3.130}$$

where $\mathcal{F}\left\{\cdot\right\}$ denotes the Fourier transform. According to elementary signal processing theory, the phase component of the Fourier transform of a shifted signal contains the information corresponding to the amount of the shift (τ). When t represents time, the shift τ denotes the amount of delay of the shifted signal. In general, such delay can be easily extracted from the phase component in the FT-domain. In contrast, contrary to what happens with the DFT, the discrete cosine/sine transforms of a given signal do not have any phase components. Even so, these authors have shown that some shift information can still be extracted from the DCT and DST coefficients of a shifted signal.

In the following, a detailed description of these techniques will be presented, firstly, in the simpler scope of 1-D discrete signals. The application of this methodology to 2-D motion estimation will be presented in appendix A.

Shift estimation in 1-D signals

Within the scope of the following presentation, it is assumed that a given signal $\{x_1(n):\ n \in \{0,\ldots,N-1\}\}$ is right shifted by an amount m (with $m > 0$), to obtain another signal $\{x_2(n):\ n \in \{0,\ldots,N-1\}\}$. It is also assumed that the sample values of $x_1(n)$ are all zeros outside the support region $\Psi\left(x_1\right)$. Therefore,

$$x_2(n) = \begin{cases} x_1(n-m), & \text{for } n-m \in \Psi\left(x_1\right), \\ 0, & \text{elsewhere.} \end{cases} \tag{3.131}$$

Koc and Liu have shown that, for $k = 1, \ldots, N-1$:

$$\begin{bmatrix} X_2^c(k) \\ X_2^s(k) \end{bmatrix} = \begin{bmatrix} +Z_1^c(k) & -Z_1^s(k) \\ +Z_1^s(k) & +Z_1^c(k) \end{bmatrix} \begin{bmatrix} g_m^c(k) \\ g_m^s(k) \end{bmatrix}, \tag{3.132}$$

where $g_m^s(k) \triangleq \sin\left[\frac{k\pi}{N}\left(m+\frac{1}{2}\right)\right]$ and $g_m^c(k) \triangleq \cos\left[\frac{k\pi}{N}\left(m+\frac{1}{2}\right)\right]$ are called *pseudo-phases*, analogous to the phases of the FT of shifted signals; $X_2^c(k)$ and $X_2^s(k)$ are orthogonal even type-II cosine (DCT-IIe) and sine (DST-IIe) trigonometric transforms of $x_2(n)$; $Z_1^c(k)$ and $Z_1^s(k)$ are orthogonal even type-I cosine (DCT-Ie) and sine (DST-Ie) trigonometric transforms of $x_1(n)$, respectively, defined as follows:

$$X_2^c(k) = \frac{2}{N}\,\xi(k) \sum_{n=0}^{N-1} x_2(n) \cos\left(\frac{k\left(n+\frac{1}{2}\right)\pi}{N}\right), \text{ with } k \in \{0, \ldots, N-1\} \tag{3.133}$$

$$X_2^s(k) = \frac{2}{N}\,\xi(k) \sum_{n=0}^{N-1} x_2(n) \sin\left(\frac{k\left(n+\frac{1}{2}\right)\pi}{N}\right), \text{ with } k \in \{1, \ldots, N\} \tag{3.134}$$

$$Z_1^c(k) = \frac{2}{N}\,\xi(k) \sum_{n=0}^{N-1} x_1(n) \cos\left(\frac{kn\pi}{N}\right), \text{ with } k \in \{0, \ldots, N\} \tag{3.135}$$

$$Z_1^s(k) = \frac{2}{N}\,\xi(k) \sum_{n=0}^{N-1} x_1(n) \sin\left(\frac{kn\pi}{N}\right), \text{ with } k \in \{1, \ldots, N-1\} \tag{3.136}$$

with $\xi(p)$ defined in eq. 2.13. It should be noted that the above definitions present slight differences when compared with those previously defined in table 2.2. Such differences arise not only from the fact that these alternative definitions do not comply with the normalization property, defined in section 2.2.5, but also because not all of these transforms are defined within the same domain.

By denoting the pseudo-phases matrix as $\theta(k) \triangleq \left[\, g_m^c(k)\ g_m^s(k) \,\right]^{\mathrm{T}}$, by defining the matrix $X(k) \triangleq \left[\, X_2^c(k)\ X_2^s(k) \,\right]^{\mathrm{T}}$ and by taking into account that the matrix:

$$Z_1(k) \triangleq \begin{bmatrix} Z_1^c(k) & -Z_1^s(k) \\ Z_1^s(k) & +Z_1^c(k) \end{bmatrix} \tag{3.137}$$

is orthogonal, i.e.,

$$\frac{1}{\left[Z_1^c(k)\right]^2 + \left[Z_1^s(k)\right]^2}\ {Z_1}^{\mathrm{T}}(k) Z_1(k) = \mathbf{I}_2, \tag{3.138}$$

equation 3.132 can be represented as:

$$\theta(k) = \frac{1}{\left[Z_1^c(k)\right]^2 + \left[Z_1^s(k)\right]^2}\ {Z_1}^{\mathrm{T}}(k) X(k). \tag{3.139}$$

By taking into account the sinusoidal orthogonal principles,

$$\frac{2}{N}\sum_{k=1}^{N}\xi^2(k)\sin\left(\frac{k\left(m+\frac{1}{2}\right)\pi}{N}\right)\sin\left(\frac{k\left(n+\frac{1}{2}\right)\pi}{N}\right)=\delta(m-n)-\delta(m+n+1), \tag{3.140}$$

$$\frac{2}{N}\sum_{k=0}^{N-1}\xi^2(k)\cos\left(\frac{k\left(m+\frac{1}{2}\right)\pi}{N}\right)\cos\left(\frac{k\left(n+\frac{1}{2}\right)\pi}{N}\right)=\delta(m-n)+\delta(m+n+1), \tag{3.141}$$

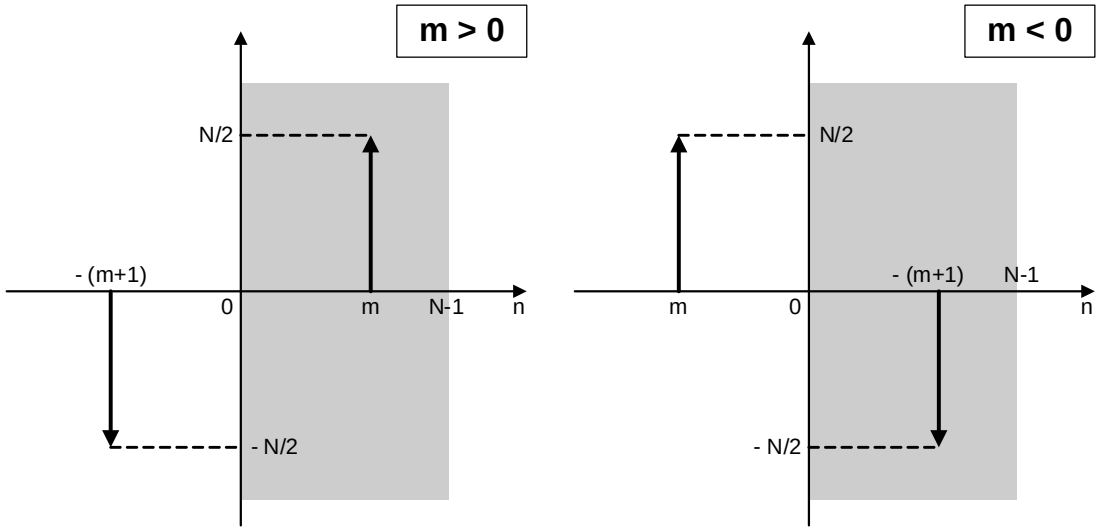
where $\delta(n)$ is the discrete impulse function, Koc and Liu have shown that the IDST-IIe and the IDCT-IIe transforms of the pseudo-phases $g_m^s(k)$ and $g_m^c(k)$, respectively, are defined as a sum of discrete impulse functions:

$$\begin{aligned}\text{IDST-IIe}\left\{\xi(k)\,g_m^s(k)\right\}&=\frac{2}{N}\sum_{k=1}^{N}\xi^2(k)\,g_m^s(k)\sin\left(\frac{k\left(n+\frac{1}{2}\right)\pi}{N}\right)\\&=\delta(m-n)-\delta(m+n+1)\end{aligned} \tag{3.142}$$

$$\begin{aligned}\text{IDCT-IIe}\left\{\xi(k)\,g_m^c(k)\right\}&=\frac{2}{N}\sum_{k=0}^{N-1}\xi^2(k)\,g_m^c(k)\cos\left(\frac{k\left(n+\frac{1}{2}\right)\pi}{N}\right)\\&=\delta(m-n)+\delta(m+n+1)\end{aligned} \tag{3.143}$$

It was also shown that the opposite signs of the impulse functions $\delta(m-n)$ and $\delta(m+n+1)$ in eq. 3.142 can be used to detect the shift direction. In fact, if the IDST-IIe inverse transform of the pseudo-phase g_m^s is computed, the observable window of the index space in the inverse DST domain will be limited to $\{0,\ldots,N-1\}$. As a consequence, in the case of a right shift ($m>0$), one spike (generated by the positive δ function) will point upwards at location $n=m$ in the observable window, while the other δ function will point downwards at $n=-(m+1)$, outside the observable window. This is illustrated in fig. 3.28(a), where the observable window is represented as a gray shaded area. On the other hand, when a left shift is considered ($m<0$), a negative spike, at $n=-(m+1)>0$, will fall in the observable window, and the positive impulse $\delta(n)$, at $n=m<0$, will fall outside the observable space (see fig. 3.28(b)). The exact displacement of this shift (m) can thus be obtained from the position of the peak. Hence, this scheme provides the means to determine the direction and displacement of the shift from the sign of the impulse functions $\delta(n)$ in eq. 3.142.

The whole algorithm for shift estimation can be summarized as stated in fig 3.29. According to Koc and Liu, the above methodology may even be applied when, due to the presence of noise resulting from prediction mismatches, there is a slight discrepancy between the two considered shifted signals [12, 38, 39, 41].



(a) Right shift ($m > 0$). **(b)** Left shift ($m < 0$).

Figure 3.28: Application of the sinusoidal orthogonal principle to DST pseudo-phases, to evaluate the direction and magnitude of the shift between two 1-D signals.

1. Compute the DCT-Ie and the DST-Ie coefficients of $x_1(n)$ and the DCT-IIe and the DST-IIe coefficients of $x_2(n)$;
2. Compute the pseudo-phase $g_m^s(k)$ for $k = 1, \ldots, N$, by solving the following equation:

$$g_m^s(k) = \begin{cases} \frac{Z_1^c(k).X_2^s(k) - Z_1^s(k).X_2^c(k)}{\left[Z_1^c(k)\right]^2 + \left[Z_1^s(k)\right]^2}, & \text{for } k \neq N, \\ 1 & \text{for } k = N; \end{cases} \tag{3.144}$$

3. Compute the IDST-IIe of the pseudo-phase $\{\xi(k) g_m^s(k);\ k = 1, \ldots, N\}$ to produce the output $\{d(n);\ n = 0, \ldots, N-1\}$;
4. Search for the peak value inside the observable window; the estimated displacement m can be found as:

$$m = \begin{cases} i_p, & \text{if } d(i_p) > 0, \\ -(i_p + 1), & \text{if } d(i_p) < 0, \end{cases} \tag{3.145}$$

where $i_p = \arg\max_n |d(n)|$ is the index at which the peak value is located.

Figure 3.29: Shift estimation using the 1-D pseudo-phases technique.

Shift estimation in 2-D signals

The described pseudo-phases technique for extracting the shift values from the DCT pseudo-phases can be extended to the 2-D case and be directly applied to the problem of motion estimation. A detailed mathematical formulation of the corresponding procedure is presented in appendix A. The corresponding procedure is entirely similar to the 1-D case, where 2-D trigonometric transforms have to be computed in order to extract the shift information.

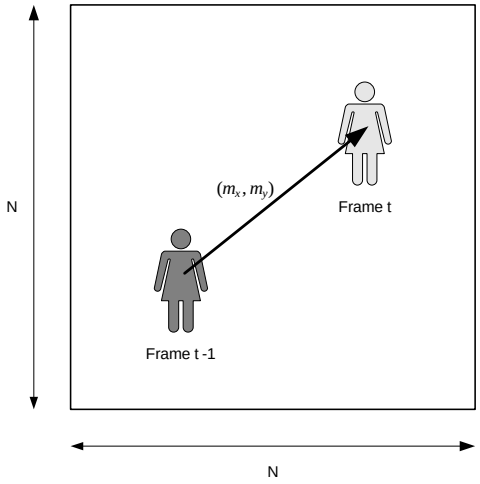


Figure 3.30: Adopted 2-D translation motion model.

The considered displacement estimation procedure assumes a simple translational motion model, in which a given object moves by m_x, in the horizontal direction, and by m_y, in the vertical direction, between the time instances corresponding to two consecutive frames, taken at instants $t-1$ and t, respectively (see fig. 3.30). In fig. 3.31 it is summarized the formulation of the corresponding algorithm.

E - Application to motion re-estimation in the DCT-domain

Under the 2-D translational motion model, Koc and Liu proposed the described motion estimation algorithm [40] by applying the pseudo-phases technique to estimate the displacement between two images, directly in the DCT-domain. However, as it will be observed, this algorithm presents some characteristics that significantly difficult its application in a real-life video transcoding system:

- **Lack of adequacy to standard video coding systems**
  The algorithm makes use of a significant set of 2-D trigonometric transforms of both the first kind (DCCT-Ie, DCST-Ie, DSCT-Ie and DSST-Ie) and of

1. Compute the trigonometric transform coefficients of the second kind (DCCT-IIe, DCST-IIe, DSCT-IIe and DSST-IIe) of a $(N \times N)$ block of pixels of the current frame $\mathbf{x}_t$;

2. Compute the trigonometric transform coefficients of the first kind (DCCT-Ie, DCST-Ie, DSCT-Ie and DSST-Ie) of a $(N \times N)$ block of pixels of the previous frame $\mathbf{x}_{t-1}$;

3. Compute the pseudo-phases $f(k,l)$ and $g(k,l)$, by solving eqs. A.19 and A.20 and by taking eq. A.9 into account to obtain $\{g^{cs}(k,l);\ k=0,\ldots,N-1;\ l=1,\ldots,N\}$ and $\{g^{sc}(k,l);\ k=1,\ldots,N;\ l=0,\ldots,N-1\}$;

4. Obtain the normalized pseudo-phases $f'(k,l)$ and $g'(k,l)$ outside the block boundaries, in order to cope with any eventual ill-conditioned equations:

$$f'(k,l) = \begin{cases} f(k,l), & \text{for } |f(k,l)| \leq 1, \\ 0 & \text{otherwise,} \end{cases} \quad ; \quad g'(k,l) = \begin{cases} g(k,l), & \text{for } |g(k,l)| \leq 1, \\ 0 & \text{otherwise.} \end{cases}$$

5. Compute the 2-D inverse trigonometric transforms of the second kind (IDCST-IIe and IDSCT-IIe) of the pseudo-phases $f'(k,l)$ and $g'(k,l)$ as $\mathrm{DCS}(m,n)$ and $\mathrm{DSC}(m,n)$, for $m,n \in \{0,\ldots,N-1\}$, which represent the impulse functions whose peak positions indicate the shift magnitude and the corresponding peak signs reveal the direction of the movement:

$$\mathrm{DCS}(m,n) = \frac{4}{N^2} \sum_{k=0}^{N-1} \sum_{l=1}^{N} \xi^2(k)\xi^2(l) f'(k,l) \cos\left[\frac{k\pi}{N}\left(m+\frac{1}{2}\right)\right] \sin\left[\frac{l\pi}{N}\left(n+\frac{1}{2}\right)\right]$$

$$\mathrm{DSC}(m,n) = \frac{4}{N^2} \sum_{k=1}^{N} \sum_{l=0}^{N-1} \xi^2(k)\xi^2(l) g'(k,l) \sin\left[\frac{k\pi}{N}\left(m+\frac{1}{2}\right)\right] \cos\left[\frac{l\pi}{N}\left(n+\frac{1}{2}\right)\right]$$

6. Search for the peaks of $\mathrm{DCS}(m,n)$ and $\mathrm{DSC}(m,n)$ inside the observable window $\mathbf{\Phi} = \{0,\ldots,N-1\}^2$, such that:

$$(i_{\mathrm{DCS}}, j_{\mathrm{DCS}}) = \arg \max_{m,n \in \mathbf{\Phi}} |\mathrm{DCS}(m,n)|\,,$$
$$(i_{\mathrm{DSC}}, j_{\mathrm{DSC}}) = \arg \max_{m,n \in \mathbf{\Phi}} |\mathrm{DSC}(m,n)|\,.$$

7. Estimate the displacement magnitude $d = (m_x, m_y)$ from the signs and positions of the peaks of $\mathrm{DCS}(m,n)$ and $\mathrm{DSC}(m,n)$:

$$m_x = \begin{cases} i_{\mathrm{DSC}} = i_{\mathrm{DCS}}, & \text{if } \mathrm{DSC}(i_{\mathrm{DSC}}, j_{\mathrm{DSC}}) > 0 \\ -(i_{\mathrm{DSC}}+1) = -(i_{\mathrm{DCS}}+1), & \text{if } \mathrm{DSC}(i_{\mathrm{DSC}}, j_{\mathrm{DSC}}) < 0 \end{cases}$$

$$m_y = \begin{cases} j_{\mathrm{DCS}} = j_{\mathrm{DSC}}, & \text{if } \mathrm{DCS}(i_{\mathrm{DCS}}, j_{\mathrm{DCS}}) > 0 \\ -(j_{\mathrm{DCS}}+1) = -(j_{\mathrm{DSC}}+1), & \text{if } \mathrm{DCS}(i_{\mathrm{DCS}}, j_{\mathrm{DCS}}) < 0 \end{cases}$$

Figure 3.31: Motion estimation using the 2-D pseudo-phases technique.

the second kind (DCCT-IIe, DCST-IIe, DSCT-IIe and DSST-IIe). However, such transform coefficients cannot be directly obtained from the decoded DCT-H.26x/MPEG-x video streams. To circumvent this problem, Koc and Liu presented a set of point-to-point relationships between the trigonometric transforms of the first kind and the corresponding transforms of the second kind [12, 40, 41]. However, not only do these relations imply the computation of a significant amount of operations, but they also require the storage of a significant number of adaptation constant matrices. Moreover, the presented relations do not solve the problem of computing the required trigonometric transforms of the second kind (DCST-IIe, DSCT-IIe and DSST-IIe) from the DCT coefficients (DCCT-IIe) decoded from the received stream, which would imply the computation of more arithmetic operations and the storage of more adaptation matrices.

Another important issue is concerned with the block size that is adopted by this algorithm. Contrary to the traditional motion estimation algorithms, which require a larger search area than the reference block, the proposed algorithm limits the search area to the size of the candidate block under processing. By restricting the search area to a single block, this algorithm is significantly more prone to the influences of the boundary effect. This is observed when the displacement is large when compared with the block size. As a result, the displaced object may partially or completely move out of the block, making the contents of the two temporally consecutive blocks very different. Therefore, it is usually accepted that the larger the block, the better the estimation. As a consequence, two different approaches may be followed to cope with this restriction:

- adopt the $(N \times N)$ block structure traditionally used by standard video coding algorithms (with $N = 8$), for both the candidate block and the search area; such alternative is not only more prone to the influence of the boundary effect (when the displacement is larger than the block size), but it also gives rise to motion vectors that are not compliant with the majority of the video standards, which adopt the motion estimation procedure at the macroblock level;
- adopt the $(M \times M)$ macroblock size for the block size in the proposed algorithm; however, such alternative would introduce some additional difficulties, since the $(N \times N)$ DCT coefficients block grid used by video coding algorithms is smaller than the $(M \times M)$ macroblock grid (usually, with $M = 2N$), which would imply that four $(N \times N)$ coefficient blocks

had to be merged in order to obtain the larger $(M \times M)$ DCT coefficients block, required by this algorithm.

In their published work, Koc and Liu have followed the second alternative. They proposed an adaptive overlapping scheme [12, 40, 41] to enlarge the block area and diminish the boundary effect when the displacement is large, compared to the block size. Since the reference block size and the search area size have to be equal, instead of using the reference block, they used a block of the same size and position in the current frame as the search area of the previous frame, defined by the search range $\boldsymbol{\mathcal{D}} = \{(u, v) : -p \leq u, v \leq +p\}$. The peak values of DSC and DCS are searched in a zig-zag way over this index range $\boldsymbol{\Phi} = \{0, \ldots, p\}^2$. As a consequence, in addition to the requirement that the new peak value must be larger than the current peak value by a given threshold, it is also necessary to check if the motion estimate determined by the new peak index lies within the search region $\boldsymbol{\mathcal{D}}$. Nevertheless, this approach should be regarded as a significantly expensive method in what concerns its computational cost, since it requires the implementation of the concatenation operation of neighboring DCT blocks in the DCT-domain [43].

- **High computational cost**
  The referred lack of adequacy of this algorithm to standard video transcoding systems and its requirement to obtain several trigonometric transforms, other than the received DCCT-IIe, confers the algorithm a significant computational cost, in order to obtain the required transforms.
  Such computational effort is further increased by the significant amount of computations required by the calculation of the pseudo-phases $f(k, l)$ and $g(k, l)$ (eqs. A.9, A.19 and A.20), as well as by the application of the 2-D IDCT-IIe inverse transform to compute the DCS(m, n) and DSC(m, n) functions, and by the subsequent search procedure, to find the index corresponding to the largest peak value.
  Moreover, since all the required 2-D transformations must be computed for an image area corresponding to the macroblock under processing, it still implies an extra and significant computational effort in order to calculate the elements of the DCT coefficients blocks that are obtained from the concatenation of the four decoded adjacent blocks [43].

- **Preprocessing requirements**
  To alleviate some difficulties and improve the performance of the presented algorithm, the usage of a preprocessing step has also been proposed [12, 41].

Such procedure should be employed in order to enhance the features of moving objects and remove strong and unwanted background features that can affect the accuracy of the estimation. The proposed preprocessing step consists on the application of a simple edge extraction scheme, by horizontally and vertically convolving the block under processing with a (3×3) Sobel operator; furthermore, a frame differentiation procedure is also applied, to restrict the processing to the difference of two consecutive frames.

- **High precision requirements and ill-conditioned solutions**
  The authors have also observed that if the absolute computed value of the pseudo-phases $f(k,l)$ and $g(k,l)$ (see eqs. A.19 and A.20) is greater than 1, such value tends to be ill-conditioned and should be discarded, by assigning it the null value [12]. Such situation occurs when the denominator in eqs. A.19 and A.20 is close to zero in comparison to the finite arithmetic precision, or when it is set to zero after the quantization step. It happens more likely when k and l are both large, since high frequency coefficients tend to be lower than low frequency coefficients. Some other inaccuracies are also identified when the algorithm is applied to signals at the presence of noise or quantization errors. As a consequence, the accuracy of the obtained motion vectors highly depends on the adopted arithmetic precision.

- **Performance**
  A performance assessment of this algorithm was also conducted [12, 41], when compared with traditional optimal and sub-optimal pixel-domain block-matching motion estimation algorithms. The Full-Search Block-Matching (FSBM), the Logarithmic Search method (LOG) [35], the Three Step Search method (TSS) [42] and the Subsampled Search approach (SUB) [55] were used in this assessment. The performance of the different schemes was evaluated and compared in terms of the MSE and the total number of bits required to encode the motion compensated residuals. The presented results evidence that the proposed algorithm performed worse than all other considered block-matching algorithms in terms of MSE for most of the considered test sequences. For some sequences, though, the proposed algorithm performed slightly better than the sub-optimal block-matching algorithms in terms of the total number of bits required to encode the motion compensated residual.

As a consequence of all these aspects, this algorithm has not been adopted by many recent video transcoding systems, which usually tend to adopt motion estimation algorithms more suited to be applied with current video standards.

3.3.6 Time scaling

As it was previously referred, high bit rate reduction ratios are often required to transmit precoded video streams over low bandwidth channels. However, such reduction ratios often result in unacceptable picture quality levels when the video is transcoded to provide the original full frame rate. As a consequence, frame rate reduction is often considered as an efficient scheme to provide the allocation of more bits to the remaining frames. This allows to maintain the quality of the non-dropped frames as high as possible. Similar frame rate conversion schemes are also needed whenever the target terminals only support video streams with lower frame rates.

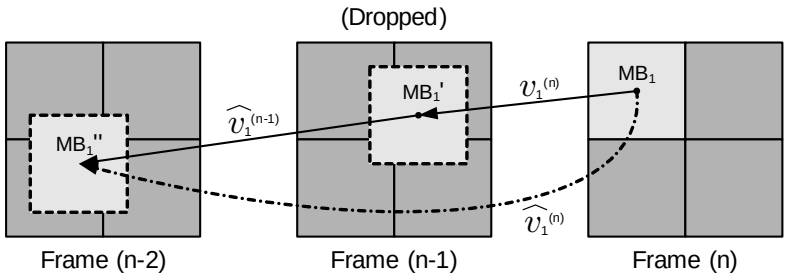
Just like the previously described transcoding issues, a straightforward approach to implement frame rate scaling is to use a cascade of a decoder and an encoder, so that one frame is retained out of a given set of original frames. Various techniques can be devised to accomplish such task more efficiently [13, 115]. In particular, transcoding techniques entirely operating in the compressed DCT-domain have been proposed in order to reduce the involved computational cost. Some of such techniques will be described in the following paragraphs.

A - Macroblock tracking over dropped frames

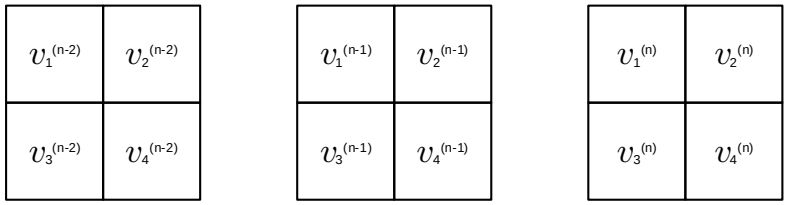
The main difficulty to efficiently implement frame dropping techniques that reduce the incoming video sequence frame rate arises from the predictive nature of most current video standards. In particular, when a subset of the frames received from the incoming video sequence is dropped, the motion vectors corresponding to the remaining frames are no longer valid, since they may point to prediction regions located in the dropped frames that do not exist in the output video sequence any more. As a consequence, new motion vectors need to be derived. The same situation occurs with the prediction difference signal, which must also be re-evaluated whenever frame dropping techniques are implemented to reduced the required video bandwidth.

One example of such situation is illustrated in fig. 3.32. In this figure, it is assumed that $\mathrm{MB}_1^{'}$ represents the best matching macroblock of MB_1 and $\mathrm{MB}_1^{''}$ represents the best matching macroblock of $\mathrm{MB}_1^{'}$. In the event of dropping the frame $(n-1)$, it is necessary to find a new motion vector, pointing to one of the macroblocks in frame $(n-2)$, which best matches macroblock MB_1.

Several schemes have been proposed to fulfill this task. In the following, it will be described the methods based on: *i)* backward bilinear interpolation of the motion vectors, *ii)* forward dominant motion vector selection [115], and *iii)* activity



(a) Tracking macroblocks in backward order.



(b) Incoming motion vectors.

Figure 3.32: Backward motion vector composition.

dominant motion vector selection [13]. In all these schemes, it is desirable the reusage of the motion vectors received from the incoming video sequence, followed by a refinement post-processing step. This additional refinement can significantly improve the motion estimation accuracy, thus justifying the inherent additional computational cost.

Motion vector bilinear interpolation method

One possible way to obtain the required motion vector is to perform an entire full-search block-matching motion estimation procedure using a search area defined within the last non-dropped frame $(n-2)$. To avoid the high computational cost involved by such procedure, a new motion vector can be obtained by summing up the decoded motion vector $v_1^{(n)}$ with $\widehat{v}_1^{(n-1)}$, corresponding to the macroblock $\mathrm{MB}_1^{'}$, in fig. 3.32. However, since $\mathrm{MB}_1^{'}$ is not defined within the adopted macroblock grid, the corresponding motion vector $\widehat{v}_1^{(n-1)}$ is not available from the decoded bit stream. To circumvent this situation, an estimate of such vector may be obtained by applying a bilinear interpolation scheme considering the motion vectors $\left\{v_1^{(n-1)},\ v_2^{(n-1)},\ v_3^{(n-1)},\ v_4^{(n-1)}\right\}$ of the four neighboring macroblocks of $\mathrm{MB}_1^{'}$ (see fig. 3.32). However, this bilinear interpolation method usually presents several drawbacks that often make this procedure unreliable and not feasible in practical systems, namely due to the large memory requirements and inaccuracy. In fact, for consecutive dropped frames, the interpolation has to be performed in the backward order, starting from the last dropped frame to the first dropped frame, requiring that

all motion vectors corresponding to those dropped frames must be stored for further processing. Moreover, the resulting motion vectors tend to be non-optimal, since the four motion vectors, corresponding to the considered adjacent macroblocks, may be too divergent to be properly described by a single motion vector.

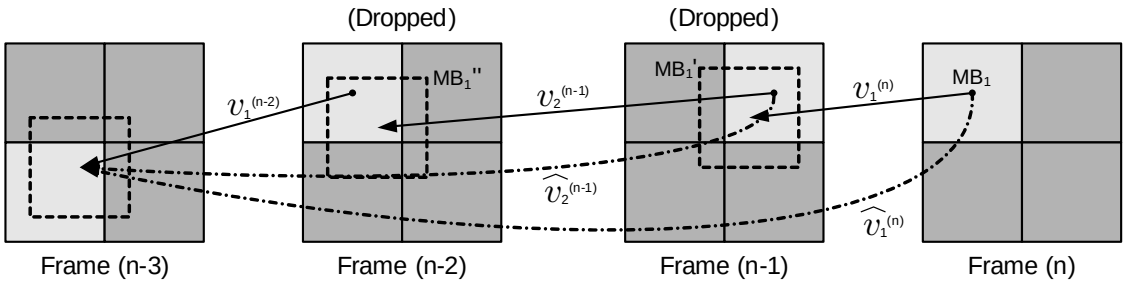
To minimize these drawbacks, overlapping area weighted average compositing schemes, entirely similar to the ones presented in section 3.3.4, have been proposed. Nevertheless, these bilinear interpolation schemes tend to offer interpolated motion vectors whose reliability is often unsatisfactory and not accurate enough to be adopted in high performance transcoding systems.

Forward dominant motion vector selection method

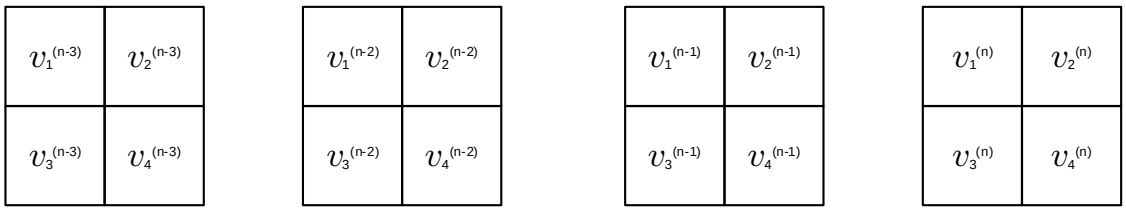
To overcome the drawbacks presented by the bilinear interpolation methods, Youn, Sun, and Lin [115] proposed a different approach denominated by Forward Dominant Vector Selection (FDVS). This method selects the motion vector from the set of four adjacent macroblocks that has the largest overlapping area with the macroblock pointed by the incoming motion vector. According to this approach, for the example illustrated in fig. 3.32, the motion vector of macroblock $\mathrm{MB}_1^{'}$ would be: $\widehat{v}_1^{(n-1)} = v_2^{(n-1)}$ (see fig. 3.32(b)).

It was claimed that this algorithm can achieve higher performances than the backward bilinear interpolation method, both in terms of the PSNR quality level and of the output bit rate, while having a lower computation effort. Moreover, when multiple consecutive frames are dropped, this algorithm can be processed in the forward order, thus eliminating the need to store the incoming motion vectors of all dropped frames in the system memory.

In fig. 3.33 it is illustrated an example where two consecutive frames are dropped. According to this algorithm, when the first skipped frame $(n-2)$ is dropped, the corresponding motion vectors are stored in a motion vectors table, within the system memory. Such table will be used to composite the motion vectors at the next frame dropping. Hence, when frame $(n-1)$ is dropped, the FDVS algorithm searches the dominant macroblock within frame $(n-2)$ for each current macroblock. As an example, the first macroblock $\mathrm{MB}_1^{(n-2)}$ in frame $(n-2)$ becomes the dominant macroblock of the second macroblock $\mathrm{MB}_2^{(n-1)}$ in frame $(n-1)$. This dominant motion vector $v_1^{(n-2)}$ is selected from the motion vectors table at the location addressed by the first macroblock and is then added to the current incoming motion vector $v_2^{(n-1)}$, corresponding to the current macroblock $\mathrm{MB}_2^{(n-1)}$. The motion vector table is then



(a) Tracking macroblocks by selecting the dominant motion vector.



(b) Incoming motion vectors.

Figure 3.33: Forward dominant motion vector selection method.

updated with this new composited motion vector value:

$$\widehat{v}_2^{(n-1)} = v_1^{(n-2)} + v_2^{(n-1)}. \tag{3.146}$$

Finally, when frame (n) is processed, the composited motion vector for the first macroblock $\mathrm{MB}_1^{(n)}$ will be set at:

$$\widehat{v}_1^{(n)} = \left[v_1^{(n-2)} + v_2^{(n-1)}\right] + v_1^{(n)}, \tag{3.147}$$

since the value stored in the motion vectors table for the dominant macroblock pointed by $v_1^{(n)}$ will be the dominant motion vector of $\mathrm{MB}_1^{'}$. Hence, the main advantage of this scheme is the need of only one table to store the composited motion vectors corresponding to the several dropped frames.

Activity dominant motion vector selection method

More recently, Chen, Chu, and Pan [13] have claimed that when the overlapping areas of the current macroblock over the four macroblocks of the previous frame are very close, the motion vector obtained with the FDVS algorithm may be not meaningful. As a consequence, they have proposed an alternative Activity Dominant Vector Selection (ADVS) algorithm, which selects the motion vector with the largest prediction error among the set of motion vectors corresponding to the overlapped macroblocks of the prediction frame. To obtain such an estimate of the prediction error, they proposed to use the DCT coefficients energy of the corresponding residual

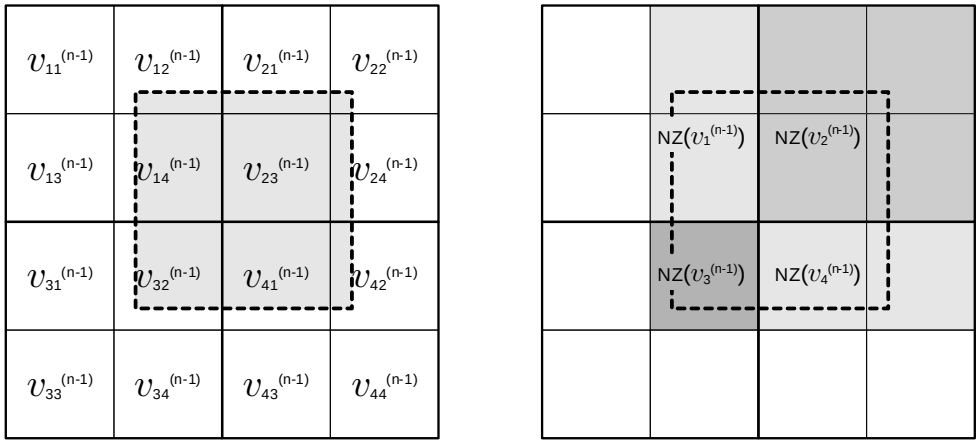


Figure 3.34: Activity dominant motion vector selection method.

blocks, directly obtained from the decoded bit stream. With such an approach, the activity information of a given macroblock is obtained by counting the number of non-zero quantized DCT coefficients of the (8×8) residual blocks that are overlapped by the macroblock area under processing. Similar alternative schemes to estimate the amount of spatial activity can also be obtained using other statistics, such as by summing the absolute values of the DCT coefficients.

In fig. 3.34 it is illustrated the application of the ADVS method, as proposed by Chen, Chu, and Pan [13]. The motion vector corresponding to the macroblock with the maximum number of Non-Zero quantized DCT coefficients (NZ) is selected as the dominant motion vector. Hence, the greater the macroblock activity (NZ), the more significant the respective motion:

$$\mathrm{NZ}(\, v_1^{(n-1)}\,) = \mathrm{NZ}(\, v_{12}^{(n-1)}\,) + \mathrm{NZ}(\, v_{14}^{(n-1)}\,) \tag{3.148}$$

$$\mathrm{NZ}(\, v_2^{(n-1)}\,) = \mathrm{NZ}(\, v_{21}^{(n-1)}\,) + \mathrm{NZ}(\, v_{22}^{(n-1)}\,) + \\ + \mathrm{NZ}(\, v_{23}^{(n-1)}\,) + \mathrm{NZ}(\, v_{24}^{(n-1)}\,) \tag{3.149}$$

$$\mathrm{NZ}(\, v_3^{(n-1)}\,) = \mathrm{NZ}(\, v_{32}^{(n-1)}\,) \tag{3.150}$$

$$\mathrm{NZ}(\, v_4^{(n-1)}\,) = \mathrm{NZ}(\, v_{41}^{(n-1)}\,) + \mathrm{NZ}(\, v_{42}^{(n-1)}\,) \tag{3.151}$$

In this example, $\mathrm{NZ}(v_1^{(n-1)})$ could actually be greater than $\mathrm{NZ}(v_2^{(n-1)})$, despite the fact that $\mathrm{NZ}(v_1^{(n-1)})$ only covers two (8×8) blocks, as compared to the four (8×8) blocks covered by $\mathrm{NZ}(v_2^{(n-1)})$, which would be chosen if the FDVS method was applied.

Since the quantized DCT coefficients of the prediction macroblocks are available from the decoded bit stream, the computational cost of this method is very low.

B - Motion vector refinement post-processing

It is important to note that the composited motion vectors obtained with the described methods tend to be non-optimal, since each vector is obtained as an estimate value of the true optimal motion vector. Such approximation will result in an inherent degradation effect due to the mismatch between the prediction and the residual components. This degradation may increase the resulting bit rate and, for smaller quantization steps, will decrease the final quality level.

To overcome this quality loss, current transcoding systems make use of the composited motion vectors, obtained after the application of the described schemes, as the basis for the application of subsequent MV refinement procedures [13]. Such algorithms make use of the MV estimate and implement a search procedure for the best matching macroblock over the last reconstructed reference frame, centered at the point given by this MV estimate. Typically, the search window used for motion vector refinement is relatively small, when compared with the original search window, e.g., $[-2, +2]$. Not only does this keep the added computational cost low, but it still provides a significant amount of achievable gains. Some of such refinement procedures were already described in section 3.3.5. Their usage increases the transcoder performance almost to the level that is obtained with the application of the full-search block-matching algorithm.

An algorithm that determines an appropriate search range based on the motion vectors magnitudes and on the number of skipped frames was proposed in [25]. To dynamically determine the number of skipped frames and to maintain a smooth playback, frame rate control schemes based on the characteristics of the video content have also been proposed [25, 45].

It should be noted that, when the new motion vectors are re-estimated, the corresponding residual data needs to be re-computed accordingly. The computation of these new residuals, using the current frame and the new reference frame data, can be directly performed in the transformed domain by using the DCT-domain motion compensation techniques, previously described in section 3.3.4 [13, 20, 109].

C - Discussion

Independently of the several possible strategies that may be proposed to estimate the motion vector that points to the best temporal predictor of a given macroblock, when adjacent inter-frames are dropped, a difficult trade-off must be undertaken in order to obtain the best balance between several aspects that considerably affect the performance and feasibility of the transcoding system:

- the amount of memory that is required to store the MVs corresponding to the dropped frames;

- the accuracy of the estimation approach;

- the computational cost of the subsequent motion re-estimation mechanism.

In particular, a difficult trade-off is often associated to the last two aspects. In fact, the resulting accuracy of the estimated MVs will definitely affect the selection of the search range and the computational cost of the motion re-estimation mechanism. Unfortunately, high accuracy levels are only achievable at the expense of a significant amount of computations, thus imposing an inherent balance between the cost of these two operations.

3.4 Conclusions

An overview of the main contributions on video transcoding of precoded video streams over the last few years was presented in this chapter. Such description started by illustrating the main advantages, both in terms of the computational efficiency and of the output video quality, of video transcoding techniques in the compressed domain. These techniques directly operate with the DCT coefficients received from the decoded video stream, thus avoiding the need to implement both the DCT and IDCT processing blocks.

A comparison of well established transcoding architectures was also presented, by illustrating the main differences between the traditional cascaded pixel-domain architectures and the more efficient transform-domain processing structures, that have been presented in the past few years.

The rest of the chapter was devoted to the presentation of the main processing techniques and algorithms that have been proposed by several authors, entirely or partially operated in the compressed DCT-domain. Such techniques include motion compensation prediction algorithms, bit rate and quality adaptation schemes, spacial video scaling algorithms, motion vector composition techniques, motion estimation procedures and time scaling algorithms.

As it was referred along the description, many processing structures require the mutual application of several of these techniques, giving rise to highly computational efficient architectures, entirely operating in the compressed DCT-domain.

References

[1] I. Ahmad, X. Wei, Y. Sun, and Y.-Q. Zhang, "Video transcoding: An overview of various techniques and research issues," *IEEE Transactions on Multimedia*, vol. 7, no. 5, pp. 793–804, Oct. 2005.

[3] Y. Arai, T. Agui, and M. Nakajima, "A fast DCT-SQ scheme for images," *Transactions of the Institute of Electronics, Information and Communication Engineers (IEICE)*, vol. E71, no. 11, pp. 1095–1097, 1988.

[4] P. Assunção and M. Ghanbari, "Buffer analysis and control in CBR video transcoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, no. 1, pp. 83–92, Feb. 2000.

[5] P. Assunção and M. Ghanbari, "Post-processing of MPEG2 coded video for transmission at lower bit rates," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Gorgia USA, May 1996, pp. 1998–2001.

[6] P. Assunção and M. Ghanbari, "A frequency-domain video transcoder for dynamic bit-rate reduction of MPEG-2 bitstreams," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 8, pp. 953–967, Dec. 1998.

[7] V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards: Algorithms and Architectures*, 2nd ed. Kluwer Academic Publishers, Jun. 1997.

[11] S.-F. Chang and D. G. Messerschmitt, "Manipulation and compositing of MC-DCT compressed video," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 1, pp. 1–11, Jan. 1995.

[12] J. Chen, U.-V. Koc, and K. J. R. Liu, *Design of Digital Video Coding Systems - A Complete Compressed Domain Approach.* Marcel Dekker, 2002.

[13] M.-J. Chen, M.-C. Chu, and C.-W. Pan, "Efficient motion-estimation algorithm for reduced frame-rate video transcoder," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 4, pp. 269–275, Apr. 2002.

[16] R. Dugad and N. Ahuja, "A fast scheme for image size change in the compressed domain," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 4, pp. 461–474, Apr. 2001.

[17] A. Eleftheriadis and D. Anastassiou, "Constrained and general dynamic rate shaping of compressed digital video," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, Arlington, Virginia, U.S.A., Oct. 1995.

[20] K.-T. Fung, Y.-L. Chan, and W.-C. Siu, "Dynamic frame skipping for high-performance transcoding," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, Oct. 2001, pp. 425–428.

[24] Q. Hu and S. Panchanathan, "Image/video spatial scalability in compressed domain," *IEEE Transactions on Industrial Electronics*, vol. 45, no. 1, pp. 23–31, Feb. 1998.

[25] J.-N. Hwang, T.-D. Wu, and C.-W. Lin, "Dynamic frame-skipping in video transcoding," in *Proceedings of the IEEE International Workshop on Multimedia Signal Processing (MMSP)*, 1998, pp. 616–621.

[26] *MPEG-1: ISO/IEC JTC1 CD 11172 - "Coding of moving pictures and associated audio for digital storage media up to 1.5 Mbit/s – Part 2: Video"*, ISO, 1992.

[27] *MPEG-2: ISO/IEC JTC1 CD 13818 - "Generic coding of moving pictures and associated audio – Part 2: Video"*, ISO, 1994.

[29] *ITU-T Recommendation H.261 - "Video Codec for Audiovisual Services at p×64 Kbit/s"*, ITU-T, Mar. 1993.

[30] *ITU-T Recommendation H.263 - "Video Coding for Low Bitrate Communication"*, ITU-T, Feb. 1998.

[32] *JPEG: ITU-T Recommendation T.81 - "Digital compression and coding of continuous-tone still images"*, ITU-T, 1993.

[35] J. R. Jain and A. K. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Transactions on Communications*, vol. COM-29, no. 12, pp. 1799–1808, Dec. 1981.

[38] U.-V. Koc and K. J. R. Liu, "Low-complexity motion estimation scheme utilizing sinusoidal orthogonal principle," in *Proceedings of the IEEE International Workshop on Visual Signal Processing and Communications (VSPC)*, New Brunswick, NJ, Sep. 1994, pp. 57–62.

[39] U.-V. Koc and K. J. R. Liu, "Discrete-cosine/sine-transform based motion estimation," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, vol. 3, Austin, Texas, Nov. 1994, pp. 771–775.

[40] U.-V. Koc and K. J. R. Liu, "Adaptive overlapping approach for DCT-based motion estimation," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, Washington, DC, 1995.

[41] U.-V. Koc and K. J. R. Liu, "DCT-based motion estimation," *IEEE Transactions on Image Processing*, vol. 7, no. 7, pp. 948–965, Jul. 1998.

[42] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion-compensated interframe coding for video conferencing," in *Proceedings of the National Telecommunications Conference*, New Orleans, LA, Nov. 1981, pp. G5.3.1–G5.3.5.

[43] W. Kou and T. Fjällbrant, "A direct computation of DCT coefficients for a signal block taken from two adjacent blocks," *IEEE Transactions on Signal Processing*, vol. 39, no. 7, pp. 1692–1695, Jul. 1991.

[45] A. Y. Lan and J.-N. Hwang, "Scene context dependent reference frame placement for MPEG videocoding," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 4, Munich - Germany, Apr. 1997, pp. 2997–3000.

[46] Y.-R. Lee and C.-W. Lin, "DCT-domain spatial transcoding using generalized DCT decimation," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, Genoa - Italy, Sep. 2005.

[47] Y.-R. Lee, C.-W. Lin, and C.-C. Kao, "A DCT-domain video transcoder for spatial resolution downconversion," in *Proceedings of the International Conference on Recent Advances in Visual Information Systems*, Mar. 2002, pp. 207–218.

[48] Y.-R. Lee, C.-W. Lin, S.-H. Yeh, and Y.-C. Chen, "Low-complexity DCT-domain video transcoders for arbitrary-size downscaling," in *Proceedings of the IEEE International Workshop on Multimedia Signal Processing (MMSP)*, Sep. 2004, pp. 31–34.

[51] H. Li and H. Shi, "A fast algorithm for reconstructing motion compensated blocks in compressed domain," *Journal of Visual Languages and Computing*, vol. 10, no. 6, pp. 607–623, Dec. 1999.

[52] Y. Liang, L.-P. Chau, and Y.-P. Tan, "Arbitrary downsizing video transcoding using fast motion vector re-estimation," *IEEE Signal Processing Letters*, vol. 9, no. 11, pp. 352–355, Nov. 2002.

[54] C.-W. Lin and Y.-R. Lee, "Fast algorithms for DCT-domain video transcoding," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, Thessaloniki - Greece, Oct. 2001, pp. 421–424.

[55] B. Liu and A. Zaccarin, "New fast algorithms for the estimation of block motion vectors," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 3, no. 2, pp. 148–157, Apr. 1993.

[56] S. Liu and A. C. Bovik, "Local bandwidth constrained fast inverse motion compensation for DCT domain video transcoding," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Salt Lake City, UT, May 2001.

[57] S. Liu and A. C. Bovik, "Local bandwidth constrained fast inverse motion compensation for DCT-domain video transcoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 5, pp. 309–319, May 2002.

[58] S. A. Martucci, "Symmetric convolution and discrete sine and cosine transforms," *IEEE Transactions on Signal Processing*, vol. SP-42, no. 5, pp. 1038–1051, May 1994.

[59] S. A. Martucci, "Image resizing in the discrete cosine transform domain," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, vol. 2, Washington D.C. - USA, Oct. 1995, pp. 244–247.

[62] N. Merhav and V. Bhaskaran, "A fast algorithm for DCT-domain inverse motion compensation," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 4, Atlanta, GA, USA, May 1996, pp. 2307–2310.

[63] N. Merhav and V. Bhaskaran, "Fast algorithms for DCT-domain image downsampling and for inverse motion compensation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, no. 3, pp. 468–476, Jun. 1997.

[66] B. K. Natarajan and B. Vasudev, "A fast approximate algorithm for scaling down digital images in the DCT domain," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, vol. 2, Washington D.C. - USA, Oct. 1995, pp. 241–243.

[72] Y. S. Park and H. W. Park, "Arbitrary-ratio image resizing using fast DCT of composite length for DCT-based transcoder," *IEEE Transactions on Image Processing*, vol. 15, no. 2, pp. 494–500, Feb. 2006.

[73] V. Patil, R. Kumar, and J. Mukherjee, "A fast arbitrary factor video resizing algorithm," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 9, pp. 1164–1171, Sep. 2006.

[83] R. Reeves, "Image matching in the compressed domain," Ph.D. dissertation, Queensland University of Technology, Australia, 1999.

[84] R. Reeves and K. Kubik, "Compressed domain image matching using symmetric convolution," in *Proceedings of IEEE Region 10 Annual Conference on Speech and Image Technologies for Computing and Telecommunications - TENCON'97*. Brisbane, Queensland: Queensland QUT Publications, Dec. 1997.

[85] J. Ridge, "Efficient transform-domain size and resolution reduction of images," *Signal Processing: Image Communication*, vol. 18, no. 8, pp. 621–639, Sep. 2003.

[88] N. Roma and L. Sousa, "Fast transcoding architectures for insertion of non-regular shaped objects in the compressed DCT-domain," *Signal Processing: Image Communication*, vol. 18, no. 8, pp. 659–683, Sep. 2003.

[89] N. Roma and L. Sousa, "Least squares motion estimation algorithm in the compressed DCT domain for H.26x/MPEG-x video sequences," in *Proceedings of the IEEE International Conference on Advanced Video and Signal-Based Surveillance (AVSS)*. Como - Italy: IEEE, Sep. 2005, pp. 576–581.

[90] N. Roma and L. Sousa, "Efficient hybrid DCT-domain algorithm for any arbitrary integer re-size video downscaling," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, no. 57291, pp. 1–16, Sep. 2007.

[92] C. L. Salazar and T. D. Tran, "On resizing images in the DCT domain," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, vol. 4, Oct. 2004, pp. 2797–2800.

[93] K.-D. Seo and J.-K. Kim, "Motion vector refinement for video downsampling in the DCT domain," *IEEE Signal Processing Letters*, vol. 9, no. 11, pp. 356–359, Nov. 2002.

[94] K.-D. Seo and J.-K. Kim, "Fast motion vector re-estimation for transcoding MPEG-1 into MPEG-4 with lower spatial resolution in DCT-domain," *Signal Processing: Image Communication*, vol. 19, no. 4, pp. 299–312, Apr. 2004.

[95] T. Shanableh and M. Ghanbari, "Heterogeneous video transcoding to lower spatio-temporal resolution and different encoding formats," *IEEE Transactions on Multimedia*, vol. 2, no. 2, pp. 101–110, Jun. 2000.

[96] T. Shanableh and M. Ghanbari, "Transcoding of video into different encoding formats," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 4, Jun. 2000, pp. 1927–1930.

[98] T. Shanableh and M. Ghanbari, "Hybrid DCT/pixel domain architecture for heterogeneous video transcoding," *Signal Processing: Image Communication*, vol. 18, no. 8, pp. 601–620, Sep. 2003.

[99] B. Shen and I. K. Sethi, "Block-based manipulations of transformed-compressed images and videos," *ACM Multimedia System Journal*, vol. 6, no. 2, pp. 113–124, Mar. 1998.

[101] B. Shen, I. Sethi, and B. Vasudev, "Adaptive motion-vector resampling for compressed video downscaling," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 6, pp. 929–936, Sep. 1999.

[102] H. Shu and L.-P. Chau, "An efficient arbitrary downsizing algorithm for video transcoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 6, pp. 887–891, Jun. 2004.

[103] H. Shu and L.-P. Chau, "A resizing algorithm with two-stage realization for DCT-based transcoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 2, pp. 248–253, Feb. 2007.

[106] H. Sun, W. Kwok, and J. Zdepski, "Architectures for MPEG compressed bitstream scaling," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 2, pp. 191–199, Apr. 1996.

[107] H. Sun, X. Chen, and T. Chiang, *Digital Video Transcoding for Transmission and Storage*. CRC Press, 2004.

[108] Y.-P. Tan, Y. Liang, and H. Sun, "On the methods and performances of rational downsizing video transcoding," *Signal Processing: Image Communication*, vol. 19, pp. 47–65, 2004.

[109] A. Vetro, P. Yin, B. Liu, and H. Sun, "Reduced spatio-temporal transcoding using an INTRA refreshing technique," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, 2002, pp. IV723–IV726.

[112] J. W. C. Wong and O. C. Au, "Modified predictive motion estimation for reduced-resolution video from high-resolution compressed video," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 4, 1999, pp. 524–527.

[113] J. Xin, C.-W. Lin, and M.-T. Sun, "Digital video transcoding," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 84–97, Jan. 2005.

[114] P. Yin, A. Vetro, B. Liu, and H. Sun, "Drift compensation for reduced spatial resolution transcoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 11, pp. 1009–1020, Nov. 2002.

[115] J. Youn, M.-T. Sun, and C.-W. Lin, "Motion vector refinement for high performance transcoding," *IEEE Transactions on Multimedia*, vol. 1, no. 1, pp. 30–40, Mar. 1999.

[116] W. Zhu, K. H. Yang, and M. J. Beacken, "CIF-to-QCIF video bitstream down-conversion in the DCT domain," *Bell Labs Technical Journal*, vol. 3, no. 3, pp. 21–29, Jul. 1998.

4

Static Video Composition

Contents

4.1 Introduction

With the widespread dissemination of video data over the several communication means that have become increasingly available over the last decades, static composition of video objects became one of the most common processing operations that are carried out on video sequences broadcasted by conventional television networks, or even on sequences that are transmitted to the last generations of mobile and portable terminal devices. At this respect, a *video object* can be defined as a group of correlated pixels, corresponding to either a natural or synthetic representation of a real-world object, scene or symbol. Each video object is represented by two types of information associated with it: its shape and its texture. Hence, by using simple video composition or segmentation operations, each video sequence may be represented by a number of different video objects.

In the scope of this chapter, *static video composition* is defined as a set of stationary or quasi-stationary manipulations on precoded video streams, in order to insert extra visual information in the video sequence that is delivered or broadcasted to the receiver. Although the video objects that are considered in these static composition operations may have any arbitrary shape, being denoted by Non-Regular Shaped Objects (NRSOs), their representation and position within the composited scene do not significantly change with the time: in the event of any change, it cannot occur between INTRA type frames, in order to keep the temporal prediction mechanism as unaffected as possible. This characteristic contrasts with *dynamic video composition* operations, discussed in chapter 5, which will allow the composition of video objects with changes on their texture component, from frame to frame.

Among the several possible manipulations, the insertion of logos, subtitles, fixed images or graphical symbols in encoded video sequences represent the most common application of these algorithms. Such objects usually occupy a small area of the encoded frame and tend to be static for reasonably long periods of time.

One important application of these video transcoding techniques is related with intellectual property management and protection. As a consequence, the insertion of visible objects, such as logos and open subtitles (henceforward simply designated by *"logos"*) in the compressed-domain has faced a growing interest by broadcasting television networks as well as digital video producers and distributors, in order to provide the possibility of inserting their own logos and subtitles in pre-encoded video streams [61]. To take into account for the several DCT-based video standards currently in use (DCT-H.26x/MPEG-x), not only should this mechanism provide the insertion capability in object oriented video standards (such as MPEG-4 Visual [28,

74]), but it should also be easily implemented in the earlier MPEG-1 Video and MPEG-2 Video standards, as well as in other simpler real-time services based on the H.261 and the H.263 video standards.

As it was described in chapter 3, recent developments on video transcoders have shown that significant advantages concerning the computational cost of the algorithms can be achieved by fully operating in the transform-domain [6]. In fact, not only does it avoid the implementation of both the forward DCT and its inverse (IDCT), but it also takes advantage of the presence of a large number of null quantized DCT coefficients to heavily reduce the data manipulation rate [57, 99].

As a consequence, a set of new DCT-domain static composition techniques will be presented in this chapter. When compared with their pixel-domain counterparts, such transform-domain architectures provide the means not only to significantly reduce the involved computational cost but also to increase the obtained video quality. Nevertheless, it should be emphasized that although the described algorithms and architectures will be focused on the insertion of fixed logos or subtitles, they can equally be applied to the composition of precoded video sequences with any other type of fixed objects, such as fixed images or graphical symbols.

4.2 Objects insertion[§]

Up until now, most insertion algorithms have been based on the compositing operation proposed by Chang and Messerschmitt [11], Porter and Duff [77]. However, despite its simplicity, when an NRSO (such as a letter or a logo) is inserted, it gives rise to an undesired semi-transparent rectangular region around the object, which corresponds to the area that is actually processed by the insertion algorithm. In this chapter, a different insertion technique is proposed, by restricting the application of the Porter and Duff's algorithm [77] to the logo area. The application of this technique in the compressed DCT-domain will require the usage of the multiplication-convolution relationship for the even type-II DCT (described in section 2.5.2), since the simple linear combination adopted in the Porter and Duff's algorithm will no

[§]Some portions of this section appeared in:

[86] - N. Roma and L. Sousa, "Insertion of irregular-shaped logos in the compressed DCT domain," in *Proceedings of the IEEE International Conference on Digital Signal Processing (DSP)*, vol. 1. Santorini, Greece: IEEE, Jul. 2002, pp. 125–128.

[87] - N. Roma and L. Sousa, "Transcoding architectures for object insertion in compressed video," INESC-ID – Lisboa, Portugal, Tech. Rep. RT/006/2002, Oct. 2002.

[88] - N. Roma and L. Sousa, "Fast transcoding architectures for insertion of non-regular shaped objects in the compressed DCT-domain," *Signal Processing: Image Communication*, vol. 18, no. 8, pp. 659–683, Sep. 2003.

longer be applicable.

Several transcoding architectures will also be derived for the proposed NRSO insertion algorithms, in both the pixel-domain and in the compressed DCT-domain. Such architectures offer different characteristics in what concerns the obtained video quality (PSNR) and of the computational load required to perform the insertion algorithm.

4.2.1 Insertion of irregular shaped objects in the pixel-domain

Object insertion in the pixel-domain can be performed by combining the pixels of the background image $b(n_1, n_2)$ with the object $\ell(n_1, n_2)$ to obtain the output image $f(n_1, n_2)$. This operation is usually expressed as a linear combination in the form [11, 77]:

$$f(n_1, n_2) = \alpha.\ell(n_1, n_2) + (1 - \alpha).b(n_1, n_2), \tag{4.1}$$

where the α factor determines the transparency level of the object. In particular, when $\alpha = 1$ all pixels of the background image are replaced by the object, giving rise to an opaque overlapping of the object over the input image.

The main disadvantage of this method is emphasized when NRSOs are inserted. In fact, since most video coding algorithms perform their processing on blocks with $N \times N$ pixels (usually $N = 8$), the insertion of objects whose shape and position do not coincide with the defined block geometry and grid implies the extension of the original object area to an integer multiple of $(N \times N)$ pixels blocks. This extension is usually performed by defining a transparency color (C_T), whose value is assigned to all pixels of this set of blocks that do not belong to the original NRSO. Three different regions in the output image usually arise from this extension:

- the pixels corresponding to the original object, where $f_1(n_1, n_2) = \alpha.\ell(n_1, n_2) + (1 - \alpha).b(n_1, n_2)$;
- the transparent pixels of the extended blocks that do not belong to the original object, where $f_2(n_1, n_2) = \alpha.C_T + (1 - \alpha).b(n_1, n_2)$;
- the blocks that do not contain any pixel of the object to be inserted, where $f_3(n_1, n_2) = b(n_1, n_2)$.

Independently of the considered value for C_T, this scheme gives rise to an undesired semi-transparent rectangular region around the object, where $f(n_1, n_2) = f_2(n_1, n_2) \neq b(n_1, n_2)$. An example of this phenomenon is illustrated in fig. 4.1(a), where eq. 4.1 was applied to insert the object using $\alpha = 0.5$.

This undesired semi-transparent region can be avoided if eq. 4.1 is restricted to the pixels of the original object. However, this will imply the usage of segmentation techniques to isolate the pixels corresponding to the original object from the rest of the pixels of the block [86].

Even so, the previously described formalism for the linear combination can still be applied if the concept of transparency mask is introduced and defined as:

$$m(n_1, n_2) = \begin{cases} 1 & , (n_1, n_2) \in \text{NRSO} \\ 0 & , (n_1, n_2) \notin \text{NRSO} \end{cases} \tag{4.2}$$

Altough the definition of this mask may be restricted to the blocks where the NRSO should be inserted, its extension to the whole image area can provide useful information concerning the location of the pixels that must be processed by the insertion algorithm. As it will be described in section 4.3.3, this infomation can be used to significantly increase the efficiency of the insertion algorithm.

Hence, eq. 4.1 becomes:

$$\begin{aligned} f(n_1, n_2) &= [\alpha.m\,(n_1, n_2)] \odot \ell\,(n_1, n_2) + [1 - \alpha.m\,(n_1, n_2)] \odot b\,(n_1, n_2) && (4.3) \\ &= \phi\,(n_1, n_2) + \psi\,(n_1, n_2) \odot b\,(n_1, n_2)\,, && (4.4) \end{aligned}$$

where $\odot$ denotes pixel-wise multiplication. In this equation, $\phi\,(n_1, n_2)$ and $\psi\,(n_1, n_2)$ represent constant matrices that are solely dependent on the considered object. Consequently, they can be pre-computed and stored in memory. In fig. 4.1(b) it is illustrated the result of the application of this technique using $\alpha = 0.5$. As it can be seen, the undesired semi-transparent region is no longer present around the NRSO.

(a) Porter and Duff technique [77]. **(b)** Proposed technique.

Figure 4.1: Pixel-domain insertion of an NRSO ($\alpha = 0.5$).

4.2.2 Insertion of objects in the compressed DCT-domain

The pixel-domain insertion algorithm presented in eq. 4.1 can be applied in the compressed-domain by using the orthogonality properties of the DCT. Since α and $(1-\alpha)$ are scalars and the DCT is linear, eq. 4.1 becomes:

$$F(k_1,k_2) = \alpha.L(k_1,k_2) + (1-\alpha).B(k_1,k_2), \tag{4.5}$$

where $X(k_1,k_2)$ generically denotes the DCT transform of signal $x(n_1,n_2)$, computed, in matrix form, as $\mathbf{X} = \mathbf{T}\cdot\mathbf{x}\cdot\mathbf{T}^{\mathrm{T}}$.

However, most video coding algorithms perform the computation of the DCT on blocks with $(N \times N)$ pixels, which makes this relation difficult to be applied if the undesired semi-transparent region around the NRSO is to be avoided [86]. In fact, although α and $(1-\alpha)$ are scalar constants, the terms $[\alpha.m\,(n_1,n_2)]$ and $[1-\alpha.m\,(n_1,n_2)]$ of eq. 4.4 represent matrices. Consequently, each pixel-wise multiplication will have to be replaced by a convolution in the DCT-domain (see section 2.5.2). Hence, the application of eq. 4.4 in the compressed DCT-domain is stated as follows:

$$F(k_1,k_2) = \Phi\,(k_1,k_2) + \Psi\,(k_1,k_2) \circledast B\,(k_1,k_2)\,, \tag{4.6}$$

where $F(k_1,k_2) = \mathrm{DCT}\,[f\,(n_1,n_2)]$, $\Phi(k_1,k_2) = \mathrm{DCT}\,[\phi\,(n_1,n_2)]$, $\Psi(k_1,k_2) = \mathrm{DCT}\,[\psi\,(n_1,n_2)]$ and $B(k_1,k_2) = \mathrm{DCT}\,[b\,(n_1,n_2)]$. As it was previously seen in section 2.5.2, the multiplication-convolution property of the DCT is based on a symmetric convolution operation over $(2N \times 2N)$ extended sequences. For the particular case of the 2-D even type-II cosine transform (DCCT-II transform), these $(2N \times 2N)$ extended sequences should have a WSWA symmetric extension (as described in section 2.2.1), expressed as:

$$\tilde{X}\,(k_1,k_2) = \begin{cases} 0 & ,\ k_1 = 0 \text{ or } k_2 = 0 \\ \hat{X}(N-k_1,N-k_2) & ,\ k_1 = 1\ldots(N-1),\ k_2 = 1\ldots(N-1) \\ \hat{X}(k_1-N,N-k_2) & ,\ k_1 = N\ldots(2N-1),\ k_2 = 1\ldots(N-1) \\ \hat{X}(N-k_1,k_2-N) & ,\ k_1 = 1\ldots(N-1),\ k_2 = N\ldots(2N-1) \\ \hat{X}(k_1-N,k_2-N) & ,\ k_1,\ k_2 = N\ldots(2N-1). \end{cases} \tag{4.7}$$

Consequently, the above 2-D convolution should be computed as follows:

$$\Psi(k_1,k_2) \circledast B(k_1,k_2) = W_{N\times N}\left(\tilde{\Psi}(k_1,k_2)\ \textcircled{\scriptsize S}\ \tilde{B}(k_1,k_2)\right), \tag{4.8}$$

where $W_{N\times N}(k_1,k_2)$ is a $N \times N$ rectangular window used to extract the representative samples out of the base period of the result of the convolution and $\tilde{\Psi}(k_1,k_2)$ and

$\tilde{B}(k_1, k_2)$ are WSWA $(2N \times 2N)$ symmetric extensions of $\Psi(k_1, k_2)$ and $B(k_1, k_2)$, as described in eq. 4.7, with

$$\hat{X}(k_1, k_2) = \frac{X(k_1, k_2)}{\xi(k_1)\xi(k_2)} \tag{4.9}$$

and $\xi(k)$ defined in eq. 2.13.

The symbol Ⓢ denotes the skew-circular convolution, defined as:

$$\tilde{\Psi}(k_1, k_2) \text{Ⓢ} \tilde{B}(k_1, k_2) = \frac{1}{2N}\, \xi(k_1)\, \xi(k_2) \left[\sum_{m_1=0}^{2N-1} \sum_{m_2=0}^{2N-1} \tilde{\Psi}(m_1, m_2) \cdot \right.$$
$$\left. \tilde{B}\Big(\text{mod}_{2N}(k_1 - m_1), \text{mod}_{2N}(k_2 - m_2)\Big) \cdot S(k_1 - m_1) \cdot S(k_2 - m_2) \right] \tag{4.10}$$

where:

$$S(k-m) = \begin{cases} 1 & , \; k - m \in [0,\; (2N-1)] \\ -1 & , \; \text{otherwise} \end{cases} . \tag{4.11}$$

The described operation requires a significant amount of multiplications and sums ($\propto 16N^4$). Recently, Shen et al. [100] proposed a different approach to compute the DCT-domain convolution, by exploiting the symmetry and the orthogonality properties of the DCT to reduce the overall computational cost of this operation (see section 2.5.3). With such algorithm, it is possible to reduce the total amount of operations to only 25% [100].

The application of the proposed insertion method in the compressed DCT-domain to insert two NRSOs, corresponding to the logo presented in fig. 4.2(a) and the subtitle shown in fig. 4.2(b), in the Common Intermediate Format (CIF) *Carphone* video sequence is presented in fig. 4.3. These objects were positioned at coordinates $(241, 10)$ and $(256, 130)$, respectively. Such positions were carefully selected in order to affect the minimum number of MBs as possible.

The frame presented in fig. 4.3(a) was obtained using a transparency factor $\alpha = 0.5$. In this processed image, it is still possible to perceive the presence of some details of the background image in the area corresponding to the inserted object. Fig. 4.3(b) presents the same frame using a transparency factor $\alpha = 1.0$. Contrasting with the previous setup, in this case the insertion is 100% opaque. The corresponding transparency mask $m(n_1, n_2)$, which was applied to the whole image, is presented in fig. 4.3(c).

(a) Logo.

(b) Subtitle.

Figure 4.2: Considered set of NRSOs ($C_T = 0$).

(a) $\alpha = 0.5$. **(b)** $\alpha = 1.0$. **(c)** Transparency mask: $m(n_1, n_2)$.

Figure 4.3: Object insertion in the compressed DCT-domain using the *Carphone* video sequence and the transparency mask of fig. 4.3(c).

4.3 Transcoding architectures for insertion of non-regular shaped objects

In this section, a set of different architectures of both pixel-domain and compressed-domain transcoders for insertion of NRSOs in compressed video sequences will be presented. The considered primordial objective was to achieve the optimum trade-off between the obtained *image quality* (PSNR) and the involved *computational cost* (evaluated by counting the number of arithmetic operations, namely, sums and multiplications).

4.3.1 Pixel-domain transcoder with re-estimation of motion vectors

The most straightforward architecture of a pixel-domain transcoder for object insertion is illustrated in fig. 4.4. This decoder-encoder cascaded pair is usually regarded as the most trivial architecture, since it comprises one full decoder followed by one full encoder. The input video bit stream is first fully decoded and then re-encoded after the application of the pixel-domain insertion algorithm, resulting in the output bit stream. In this case, the compositing scheme previously described in eq. 4.4 is directly applicable.

Despite its simplicity, the main disadvantages of this architecture are concerned with its high computational cost. In fact, since the insertion algorithm is performed in the pixel-domain and there is no re-usage of the original coding parameters, a significant amount of operations is required to perform both the direct and inverse discrete cosine transforms, as well as the block-matching motion estimation algorithm at the encoder side of the transcoding system. Moreover, this decoding-

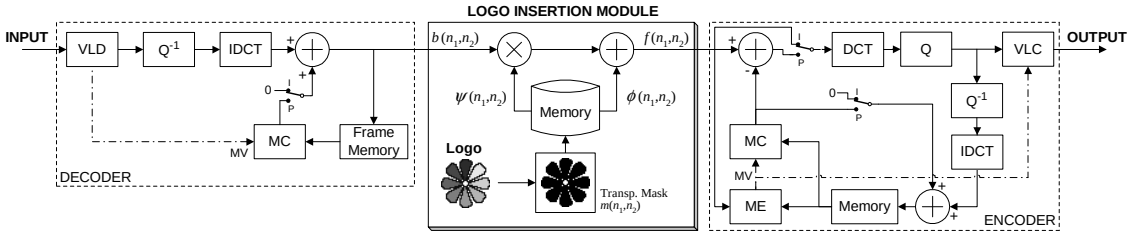


Figure 4.4: Pixel-domain transcoder for object insertion with re-estimation of motion vectors.

encoding process of the whole picture usually leads to a non-negligible degradation of the video quality. As it was shown in [6, 36] and briefly described in section 3.1.2, transcoding systems using this cascaded transmission chain lead to an extra amount of distortion, as a result of the twofold dequantization-quantization operation: one at the encoder and another one at the transcoder. According to the results presented in [36], this cascaded quantization introduces losses of about $0.5 - 1.0$ dB. Besides the noticeable degradation introduced in the processed frames, it also affects the performance of the motion estimation operation that is implemented at the encoder side of the transcoder. In fact, while the MVs of the incoming bit stream were computed with full-precision pixel values, the search procedure is now performed in decoded pictures that were already affected by the quantization distortion (data loss) of the coding algorithm [6].

4.3.2 Pixel-domain transcoder without re-estimation of motion vectors

Considering that motion estimation is undoubtedly the most computational demanding block of the video coding algorithm, an usual approach to significantly reduce the computational cost of video transcoders is to remove the motion estimation function from the encoder side. By doing so, the MVs of the incoming bit stream are re-used, instead of calculating new ones. In this case, the architecture of the pixel-domain transcoder is the one illustrated in fig. 4.5.

As it was already referred (and will be shown later), not only is this architecture characterized by a significantly lower computational cost, but it usually leads, in practice, to better quality performances than the previous described architecture (with re-estimation of MVs at the encoder side). Even so, this architecture still demands a significant amount of operations, since the computation of both the forward and the inverse discrete cosine transforms are still required by this scheme. Hence, this computational cost incited the proposals of DCT-domain video transcoders [6, 54, 97, 111].

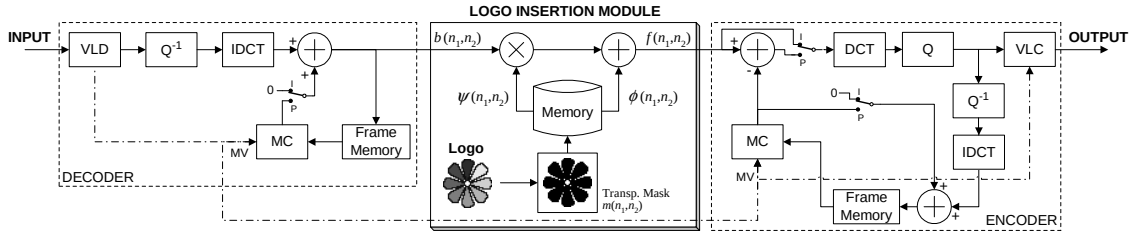


Figure 4.5: Pixel-domain transcoder for object insertion without re-estimation of motion vectors.

Independently of the adopted motion estimation procedure, the two pixel-domain insertion algorithms previously described can be formulated as stated in fig. 4.6. In this figure, $b(n_1, n_2)$ denotes the input $(N \times N)$ pixels block under processing, while $f(n_1, n_2)$ represents the output block, resulting from the application of the insertion algorithm.

I - Pre-compute and store in memory the two constant matrices that characterize the NRSO to be inserted:

$$\phi(n_1, n_2) = [\alpha.m(n_1, n_2)] \odot \ell(n_1, n_2)$$
$$\psi(n_1, n_2) = [1 - \alpha.m(n_1, n_2)]$$

II - For each decoded input block $\mathbf{b(n_1, n_2)}$:

- if $b(n_1, n_2) \cap \text{NRSO} = \emptyset$:
  - Do nothing: $f(n_1, n_2) = b(n_1, n_2)$.
- if $b(n_1, n_2) \cap \text{NRSO} \neq \emptyset$:
  - The NRSO portion in $b(n_1, n_2)$ must be inserted: $f(n_1, n_2) = \phi(n_1, n_2) + \psi(n_1, n_2) \odot b(n_1, n_2)$.

Figure 4.6: Pixel-domain insertion algorithm.

4.3.3 Compressed DCT-domain transcoder

The most straightforward approach to implement the presented transcoder in the compressed-domain is to perform the compositing operation of eq. 4.4 in the DCT-domain, as it was described in eq. 4.6. However, this will imply that certain operations, such as MC, will also have to be performed in the compressed-domain. As it was described in section 3.3.1, devoted to the study of MC in the compressed DCT-domain, this operation presents a computational cost that is somewhat higher than its pixel-domain counterpart. This is mainly due to the fact that, in the general

case, four neighbor blocks will have to be processed in order to obtain the prediction of the current block (see section 3.3.1). Even so, several efficient algorithms have recently been proposed to reduce the computational load of DCT-domain motion compensation algorithms [51, 56, 57]. Their main advantages are obtained by exploiting the sparseness property of the DCT representation of the several blocks as well as their spatial continuity between adjacent blocks.

The block diagram of the implemented DCT-domain transcoder is presented in fig. 4.7. As in the previous pixel-domain transcoder, the motion vectors decoded from the incoming bit stream are re-used, instead of performing their re-estimation at the encoder part of the transcoder. This provides the ability to avoid the introduction of degradation related to the usage of reference images already affected by an additional quantization process.

The insertion algorithm is performed in a block-by-block basis: for each block of the input image, it convolves its discrete cosine transform with $\Psi(k_1, k_2)$, corresponding to the transparency mask, using the 2-D symmetric convolution in the DCT-domain (see section 2.5.2) and adds the result with the data corresponding to the NRSO $\Phi(k_1, k_2)$ (see eq. 4.6). Since $\Psi(k_1, k_2)$ and $\Phi(k_1, k_2)$ only depend on the object that is being inserted, they can be pre-computed and stored in memory, thus leading to a significant reduction of the required number of operations (see fig. 4.7). The formulation of this algorithm can be stated as presented in fig. 4.8.

To increase the efficiency of this architecture, some important issues concerning the insertion procedure and the encoding of video data may be taken into account. As an example, the computational cost of the insertion algorithm can be greatly reduced by using the transparency mask to provide useful information about the position of the NRSO. Hence, every MB whose values of the corresponding transparency mask $m(n_1, n_2)$ are all equal to zero ($\forall(n_1, n_2) \in \mathrm{MB}_x,\ m(n_1, n_2) = 0$) can be skipped from the whole insertion procedure.

More than the desired position of the object or subtitle, it is also convenient to know the width and the height of the hypothetical box that tightly bounds the

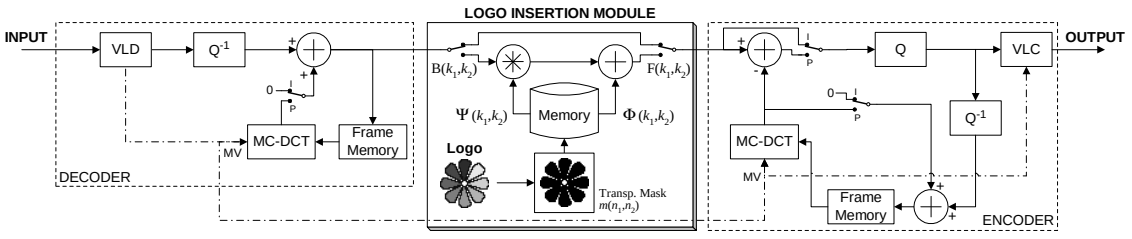


Figure 4.7: Compressed DCT-domain transcoder for object insertion.

I - Pre-compute and store in memory the two constant matrices that characterize the NRSO to be inserted:

$\Phi(k_1, k_2) = \mathrm{DCT}\left[\phi\left(n_1, n_2\right)\right]$, with $\phi\left(n_1, n_2\right) = \left[\alpha.m\left(n_1, n_2\right)\right] \odot \ell\left(n_1, n_2\right)$

$\Psi(k_1, k_2) = \mathrm{DCT}\left[\psi\left(n_1, n_2\right)\right]$, with $\psi\left(n_1, n_2\right) = \left[1 - \alpha.m\left(n_1, n_2\right)\right]$

II - For each decoded input block $\mathbf{B(k_1, k_2)}$:

- if $B(k_1, k_2) \cap \mathrm{NRSO} = \emptyset$:
  - Do nothing:
    $F(k_1, k_2) = B(k_1, k_2)$.
- if $B(k_1, k_2) \cap \mathrm{NRSO} \neq \emptyset$:
  - The NRSO portion in $B(k_1, k_2)$ must be inserted:
    $F(k_1, k_2) = \Phi\left(k_1, k_2\right) + \Psi\left(k_1, k_2\right) \circledast B\left(k_1, k_2\right)$.

Figure 4.8: Compressed DCT-domain insertion algorithm.

NRSO. Some authors have suggested that, whenever it is possible, the position of this box should be adjusted so that the number of MBs affected by the NRSO insertion algorithm is minimized [71].

One other important issue is concerned with the effect of the insertion on the encoding of the several MBs. In fact, despite all strategies that have been proposed to re-use, in the coder side of the transcoder, as much information decoded from the incoming bit stream as possible, one now has to take into account that this information may be no longer valid for the MBs in the vicinity of the insertion. As an example, to reduce the involved computational cost, video transcoders usually re-use the decoded MVs, which may no longer precisely point to the best matching MB. Furthermore, it is also highly possible that decoded MVs that pointed to the MBs where the object has been inserted may have to be re-evaluated, since the object content is no longer the expected prediction that should be inserted in those regions of the image.

Panusopone et al. [71] have recently studied the problem of inserting translucent objects (see eq. 4.1) in pixel-domain transcoders. They considered two distinct approaches to adapt the motion vectors (and the quantization steps), so that the resulting impact on the coding algorithm is reduced [71]:

Method 1 - Simple re-usage of the MVs decoded from the input bit stream. The coding efficiency of this method will be inevitably reduced, as a consequence of the MVs inaccuracy for the MBs where objects have been inserted, as well as for those MBs that have MVs pointing to the inserted object, since they will certainly have wrong reference MBs.

Method 2 - Usage of two different sets of MVs, by choosing the original MVs decoded from the input bit stream for those MBs which are dominated by the background content, or by adopting a null MV for those MBs that are dominated by the content of the inserted object. The following threshold rule was proposed, to determine the MVs to adopt in the MBs where the object is inserted:

- $v = (0, 0)$ when the transparency factor (α) is greater than or equal to some adjustable threshold, e.g., $\alpha = 0.5$;
- v will be kept unchanged, otherwise.

This scheme can equally be applied to select the MV value of those MBs whose original MVs were pointing to the region where the object was inserted.

However, not only is Panusopone et al. insertion method entirely implemented in the pixel-domain, but their proposed insertion techniques are also clearly different from those that are now presented and from the set of fast compressed DCT-domain techniques that will be proposed in the following sections. Nevertheless, despite such quite significant differences, their proposals concerning the re-usage of the MVs can still be applied to the proposed architectures with minor changes.

4.3.4 Computational-reduced compressed DCT-domain transcoder

From the analysis of the previously described compressed DCT-domain transcoder, one can realize that the proposed insertion algorithm in the transform-domain involves a significant amount of operations (even when re-usage of motion vectors is considered). There are two functional blocks which contribute the most for this computational cost:

- **DCT-domain motion compensation (MC-DCT)**:
  By following a straightforward approach, for each $N \times N$ pixels block it is necessary to perform 6 matrix products and 3 matrix sums (see section 3.3.1). Since N^3 multiplications and $N^2(N-1)$ sums are involved in the computation of each matrix product, $6N^3$ multiplications and $6N^3 - 3N^2$ sums are required to process each block. However, under certain circumstances, the bandwidth constrained methods described in subsection D of section 3.3.1, (page 74) can reduce the required computational load and take advantage of the sparseness property of the DCT representations of the several blocks.

- **2-D symmetric convolution**:
  For each $N \times N$ pixels block, it is necessary to perform $4N^4$ multiplications and sums (see section 2.5.2).

If the whole set of manipulations required by the compressed DCT-domain transcoder is taken into account, the number of operations (multiplications and additions) required to process each $(N \times N)$ pixels block will be those represented in table 4.1. On the other hand, the computational load required by the traditional pixel-domain transcoder without re-estimation of motion vectors (described in section 4.3.2) corresponds to the set of operations represented in table 4.2.

Thus, while the compressed DCT-domain transcoder is characterized by an $\mathcal{O}(N^4)$ complexity level, the traditional pixel-domain transcoder exhibits a computational load proportional to only $\mathcal{O}(N^3)$. This substantial extra amount of com-

Table 4.1: Required number of operations to process each $(N \times N)$ pixels block using the DCT-domain transcoder.

Functional Block	Multiplications	Additions
MC-DCT decoder	$6N^3$	$6N^3 - 3N^2$
Differential decoder	-	N^2
Convolution with $\Psi(k_1, k_2)$	$4N^4$	$4N^4$
Sum with $\Phi(k_1, k_2)$	-	N^2
MC-DCT encoder	$6N^3$	$6N^3 - 3N^2$
Differential encoder	-	N^2
Differential decoder	-	N^2
TOTAL	$4N^4 + 12N^3$	$4N^4 + 12N^3 - 2N^2$

Table 4.2: Required number of operations to process each $(N \times N)$ pixels block using the pixel-domain transcoder.

Functional Block	Multiplications	Additions
IDCT	$2N^3$	$2N^3 - 2N^2$
MC decoder	-	-
Differential decoder	-	N^2
Logo insertion	N^2	N^2
MC encoder	-	-
Differential encoder	-	N^2
DCT	$2N^3$	$2N^3 - 2N^2$
Differential decoder	-	N^2
TOTAL	$4N^3 + N^2$	$4N^3$

putations required by the described DCT-domain transcoder incited the presented research for more efficient architectures.

Meanwhile, to clarify the presentation of this and of other architectures proposed in this chapter, the following nomenclature will be adopted:

$\mathbf{i}_n$ - INTRA type image;

$\mathbf{p}_n$ - INTER type image;

$\overline{\mathbf{i}_n}$ - INTRA type image after the insertion of the NRSO;

$\overline{\mathbf{p}_n}$ - INTER type image after the insertion of the NRSO.

Similarly to previously defined notations, upper-case letters denote the discrete cosine transform coefficients corresponding to pixel-domain signals: $\mathbf{X} = \mathrm{DCT}(\mathbf{x})$. moreover. the sequential number of each image is appended to its representation as a subscript index (e.g.: $\mathbf{I}_n\ \mathbf{P}_{n+1}\ \mathbf{P}_{n+2}\ \ldots \mathbf{P}_{n+G-1}\ \mathbf{I}_{n+G}\ \mathbf{P}_{n+G+1}\ \ldots$), where G is the number of frames in each GOP that was adopted in the encoding setup of the considered video sequence.

The architecture presented in this section tries to achieve a reduction of the overall computational load by eliminating the DCT motion-compensation operation from the processing of certain macroblocks in INTER type images. By considering the general expression that is used to decode INTER type images at the decoder end of the coding system:

$$\mathbf{p}_t = \mathrm{MC}(\mathbf{p}_{t-1}) + \mathbf{e}_t \tag{4.12}$$

which, in the DCT-domain, is stated as:

$$\mathbf{P}_t = \text{MC-DCT}(\mathbf{P}_{t-1}) + \mathbf{E}_t, \tag{4.13}$$

the objective of this new architecture is to process INTER type images using solely the information corresponding to the current frame that is received from the communication channel: the differences signal ($\mathbf{E}_t$), the motion vectors and the quantization levels. The expression for the decoding operation of INTER type images, to be performed by the ultimate decoder of the whole video communication system, should have the form:

$$\overline{\mathbf{p}_t} = \mathrm{MC}\left(\overline{\mathbf{p}_{t-1}}\right) + \mathbf{e}_t^*, \tag{4.14}$$

where $\mathrm{MC}\left(\overline{\mathbf{p}_{t-1}}\right)$ is obtained by applying the motion compensation algorithm to the previous decoded image with the NRSO already inserted on it. The operand $\mathbf{e}_t^*$ represents the differences signal after the application of the insertion algorithm to the original differences frame: $\mathbf{e}_t^* = f\left(\mathbf{e}_t\right)$.

By considering a fraction of a given video sequence composited by only two consecutive frames ($\ldots \mathbf{i}_{t-1}\ \mathbf{p}_t \ldots$) and an NRSO defined by $\ell(n_1, n_2)$, the insertion

algorithm described in eq. 4.1 for a given transparency factor α comes as follows:

$$\begin{aligned}\overline{\mathbf{i}_{t-1}} &= \alpha\boldsymbol{\ell} + (1-\alpha)\mathbf{i}_{t-1} && (4.15)\\ &= \mathbf{i}_{t-1} + \alpha(\boldsymbol{\ell} - \mathbf{i}_{t-1}) && (4.16)\\ &= \mathbf{i}_{t-1} + \mathbf{r}_{t-1} && (4.17)\end{aligned}$$

$$\begin{aligned}\overline{\mathbf{p}_t} &= \alpha\boldsymbol{\ell} + (1-\alpha)\mathbf{p}_t && (4.18)\\ &= \mathbf{p}_t + \alpha(\boldsymbol{\ell} - \mathbf{p}_t) && (4.19)\\ &= \mathbf{p}_t + \mathbf{r}_t && (4.20)\end{aligned}$$

where $\mathbf{r}_{t-1} = \alpha(\boldsymbol{\ell} - \mathbf{i}_{t-1})$ and $\mathbf{r}_t = \alpha(\boldsymbol{\ell} - \mathbf{p}_t)$, respectively. According to eq. 4.12 and eq. 4.20, it follows that:

$$\begin{aligned}\overline{\mathbf{p}}_t &= \mathrm{MC}\left(\mathbf{p}_{t-1}\right) + \mathbf{e}_t + \mathbf{r}_t && (4.21)\\ &= \mathrm{MC}\left(\overline{\mathbf{p}_{t-1}} - \mathbf{r}_{t-1}\right) + \mathbf{e}_t + \mathbf{r}_t && (4.22)\\ &= \mathrm{MC}\left(\overline{\mathbf{p}_{t-1}}\right) + \mathbf{e}_t - \mathrm{MC}\left(\mathbf{r}_{t-1}\right) + \mathbf{r}_t && (4.23)\\ &= \mathrm{MC}\left(\overline{\mathbf{p}_{t-1}}\right) + \mathbf{e}_t^*, && (4.24)\end{aligned}$$

where:

$$\mathbf{e}_t^* = \mathbf{e}_t - \mathrm{MC}\left(\mathbf{r}_{t-1}\right) + \mathbf{r}_t. \quad (4.25)$$

In the compressed DCT-domain, eq. 4.17 and eq. 4.24 are stated as:

$$\overline{\mathbf{I}_{t-1}} = \mathbf{I}_{t-1} + \mathbf{R}_{t-1} \quad (4.26)$$

$$\overline{\mathbf{P}_t} = \text{MC-DCT}\left(\overline{\mathbf{P}_{t-1}}\right) + \mathbf{E}_t^*, \quad (4.27)$$

where

$$\mathbf{E}_t^* = \mathbf{E}_t - \text{MC-DCT}\left(\mathbf{R}_{t-1}\right) + \mathbf{R}_t, \quad (4.28)$$

and

$$R_t(k_1, k_2) = \alpha(k_1, k_2) \circledast \left[L(k_1, k_2) - P_t(k_1, k_2)\right]. \quad (4.29)$$

From the previous description, it can be observed that eq. 4.27 corresponds to the desired expression for the DCT-domain insertion algorithm, presented in eq. 4.14. In fact, it can be shown, from eq. 4.28, that such procedure can be decomposed in two distinct operations:

1. insertion of the data corresponding to the fraction of the NRSO that should be placed in the current position: $+\mathbf{R}_t$;

2. removal of the data corresponding to the fraction of the NRSO that was inserted in the previously processed image; according to the encoding algorithm, it is necessary to take into account the possible displacement between the current macroblock and its prediction macroblock: $-\text{MC-DCT}\,(\mathbf{R}_{t-1})$.

In the formulation of the above equations, the motion compensation function was assumed to be a linear operation. However, that is not entirely true. As it was previously noted, even when the considered motion vectors are the same, the inherent integer truncation and round-off errors may introduce some slight distortion in the overall DCT-domain processing.

The expression stated in eq. 4.28 conducts to the formulation of the insertion algorithm that should be applied to each block of the image under processing, as stated in fig. 4.9. In this figure, $\mathbf{A}_n$ denotes the DCT coefficients of the $(N \times N)$ pixels block under processing, while $\mathbf{B}_{n-1}$ denotes the coefficients of the prediction block, obtained after the motion compensation operation over the previously processed image, using the corresponding motion vector v_n.

The block diagram of this computational reduced frequency-domain insertion transcoder is presented in fig. 4.10. According to this architecture, the NRSO data

I - INTRA type images:

- if $\mathbf{A}_t \cap \text{NRSO} \neq \emptyset$:
  - $\overline{\mathbf{I}_t} = \mathbf{I}_t + \mathbf{R}_t$;
- if $\mathbf{A}_t \cap \text{NRSO} = \emptyset$:
  - Do nothing: $\overline{\mathbf{I}_t} = \mathbf{I}_t$.

II - INTER type images:

- if $\mathbf{A}_u \cap \text{NRSO} = \emptyset$ and $\mathbf{B}_{u-1} \cap \text{NRSO} = \emptyset$:
  - Do nothing: $\mathbf{E}^*_u = \mathbf{E}_u$;
- if $\mathbf{A}_u \cap \text{NRSO} = \emptyset$ and $\mathbf{B}_{u-1} \cap \text{NRSO} \neq \emptyset$:
  - The NRSO portion in $\mathbf{B}_{u-1}$ must be removed: $\mathbf{E}^*_u = \mathbf{E}_u - \text{MC-DCT}(\mathbf{R}_{u-1})$;
- if $\mathbf{A}_u \cap \text{NRSO} \neq \emptyset$ and $\mathbf{B}_{u-1} \cap \text{NRSO} = \emptyset$:
  - The NRSO portion in $\mathbf{A}_u$ must be inserted: $\mathbf{E}^*_u = \mathbf{E}_u + \mathbf{R}_u$;
- if $\mathbf{A}_u \cap \text{NRSO} \neq \emptyset$ and $\mathbf{B}_{u-1} \cap \text{NRSO} \neq \emptyset$:
  - The NRSO portion in $\mathbf{B}_{u-1}$ must be removed and the NRSO portion in $\mathbf{A}_u$ must be inserted: $\mathbf{E}^*_u = \mathbf{E}_u - \text{MC-DCT}(\mathbf{R}_{u-1}) + \mathbf{R}_u$.

Figure 4.9: Computational-reduced insertion algorithm in the compressed DCT-domain.

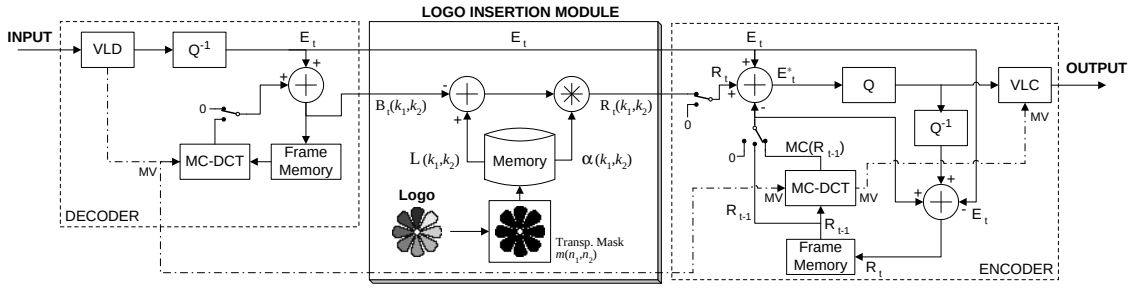


Figure 4.10: Computational-reduced compressed DCT-domain transcoder for object insertion.

is inserted in INTER type images directly in the differences signal ($\mathbf{E}_t$). This contrasts with the previous architecture, where the NRSO was inserted in the partially decoded image ($\mathbf{P}_t$). Moreover, a significant reduction of the computational load can now be achieved in the encoder part of the transcoder. In fact, while in the previously described compressed DCT-domain architecture the motion compensation operation was applied to the entire image area, in this architecture this operation is only performed in the area covered by the NRSO. Therefore, the degree of this reduction will naturally depend on the particular characteristics of the NRSO that is being inserted.

It should be noted, however, that a special care must be taken into account when the MC procedure is performed in the encoder phase of this algorithm. In fact, to achieve a perfect removal of the previously inserted NRSO data (see point 2 on the preceding page), the MC operation must be accurate enough in order to avoid any evidence of the removed data. Consequently, this precision requirement may impede the usage of the bandwidth constrained methods described in subsection D of section 3.3.1, (page 74), since the significant computational saving that is offered by such algorithm comes at a cost of a slight distortion level, which is often enough to prevent a perfect removal of the NRSO data. Nevertheless, this slight degradation does not significantly affect the perfect decoding of the received video, which makes this computational reduced method particularly well suited to be applied in the decoder part of the insertion architecture.

Nevertheless, despite the described computational saving strategy, one realizes that this architecture still requires a significant amount of computations. Namely, in the DCT-domain motion compensation operation, that is performed over the entire image at the decoder part; in the 2-D symmetric convolution, that is performed over the area covered by the NRSO at the insertion module; and in the DCT-domain motion compensation operation, that is performed over the area covered by the NRSO at the encoder part of the transcoding system.

4.3.5 Open-loop compressed DCT-domain transcoder

To cope with the high computational load of the previously presented compressed-domain insertion algorithms, alternative schemes with lower computational requisites have been considered. Such architectures take into account some information concerning the GOP structure of the coded video under processing and are somewhat more permissive to the introduction of a certain distortion level in the image area corresponding to the insertion. In fact, if one takes into account that NRSOs (such as logos or subtitles) usually do not change with time (or only change between a significant amount of frames), they can be often considered as quasi-stationary entities. Consequently, one can easily raise the question about the need to insert them in all encoded frames. As an example, INTER type images are encoded with temporal prediction schemes, where the DCT is applied to the difference between the current image and the image obtained from the application of the motion-compensation mechanism to the previously decoded frame. Considering that logos (or other types of NRSOs) do not change with time, it wouldn't be difficult to accept that very little information concerning them would be present in the differences signal corresponding to the encoding of INTER type images. Consequently, only INTRA type images would have to be processed by the insertion algorithm.

Unfortunately, this assumption is not completely true, not only due to the motion compensation mechanism, but also due to the fact that the considered insertion algorithm deals with semi-transparent type NRSOs. In fact, it should be taken into account that the prediction of a given macroblock in a INTER type image is obtained by conveniently displacing the previously decoded macroblocks, according to the corresponding motion vectors. If one considers the simplest case, where one of the macroblocks corresponding to the NRSO has a non-null motion vector, one should accept that its prediction might not contain the NRSO, or even if it does, it will be probably not placed in the correct position. Therefore, some important issues will have to be considered in order to solve these problems.

From the previous sections, one realizes that the implementation of the insertion algorithm in the compressed DCT-domain requires one 2-D symmetric convolution and one or more motion-compensation operations in the DCT-domain. These operations are the most computationally expensive ones, with a complexity level of $\mathcal{O}(N^4)$ and $\mathcal{O}(N^3)$, respectively. Hence, an alternative approach may raise the possibility of tolerating the introduction of a certain amount of distortion in order to decrease this required computational burden. To achieve such objective, a rather simplified architecture is assumed for the insertion module, which is based on the

two following principles:

1. The NRSOs are solely inserted in INTRA type images;

2. Only the differences signal (E_t) is processed in INTER type images.

To comply the insertion procedure with the existing video standards, and to keep unchanged the general expression adopted in the decoding operation of INTER type images at the decoder end of the system (see eq. 4.12), the processing of INTER type images will have to be performed using only the information corresponding to the image being processed and that is received from the communication channel: the discrete cosine transform of the differences signal ($\mathbf{E}_t$). This expression should have the form:

$$\widehat{\overline{\mathbf{P}_t}} = \text{MC-DCT}\left(\widehat{\overline{\mathbf{P}_{t-1}}}\right) + \mathbf{E}_t^*. \tag{4.30}$$

The $\widehat{\mathbf{X}}$ nomenclature was adopted in eq. 4.30 and will be used in the following expressions to represent signals where a certain level of degradation or distortion is tolerated, as a result of the application of the simplified insertion algorithm.

According to the first principle previously stated, the information that is considered in the processing of all $(G-1)$ INTER type images of a given GOP refers solely to the previously processed INTRA frame:

$$\begin{aligned} \overline{\mathbf{i}_t} &= \alpha\boldsymbol{\ell} + (1-\alpha)\mathbf{i}_t & (4.31)\\ &= \mathbf{i}_t + \alpha\left(\boldsymbol{\ell} - \mathbf{i}_t\right) & (4.32)\\ &= \mathbf{i}_t + \mathbf{r}_t, & (4.33) \end{aligned}$$

where $\mathbf{r}_t = \alpha\left(\boldsymbol{\ell} - \mathbf{i}_t\right)$. This $\mathbf{r}_t$ factor should be considered constant within a GOP. Consequently, in the processing of the INTER type image $\mathbf{p}_u$ of such GOP, corresponding to time instant u (where $u - t \leq G$), the same $\mathbf{r}_t$ factor should also be used. Hence, instead of having:

$$\begin{aligned} \overline{\mathbf{p}_u} &= \alpha\boldsymbol{\ell} + (1-\alpha)\mathbf{p}_u & (4.34)\\ &= \mathbf{p}_u + \alpha\left(\boldsymbol{\ell} - \mathbf{p}_u\right) & (4.35) \end{aligned}$$

one will have the slightly distorted $\widehat{\overline{\mathbf{p}_u}}$:

$$\begin{aligned} \widehat{\overline{\mathbf{p}_u}} &= \mathbf{p}_u + \alpha\left(\boldsymbol{\ell} - \mathbf{i}_t\right) & (4.36)\\ &= \mathbf{p}_u + \mathbf{r}_t. & (4.37) \end{aligned}$$

Eq. 4.34 and eq. 4.36 can be used to estimate the amount of distortion ($\boldsymbol{\epsilon}_t$) that is introduced by this simplified scheme. In fact, considering that $\mathbf{r}_t = \alpha\left(\boldsymbol{\ell} - \mathbf{i}_t\right)$, the

slightly distorted $\widehat{\overline{\mathbf{p}_u}}$ is given by:

$$\begin{aligned}
\widehat{\overline{\mathbf{p}_u}} &= \mathbf{p}_u + \alpha\left(\boldsymbol{\ell} - \mathbf{i}_t\right) && (4.38)\\
&= \alpha\boldsymbol{\ell} + \mathbf{p}_u - \alpha\mathbf{i}_t && (4.39)\\
&= \alpha\boldsymbol{\ell} + (1-\alpha)\mathbf{p}_u + \alpha(\mathbf{p}_u - \mathbf{i}_t) && (4.40)\\
&= \alpha\boldsymbol{\ell} + (1-\alpha)\mathbf{p}_u + \boldsymbol{\epsilon}_t, && (4.41)
\end{aligned}$$

where:

$$\boldsymbol{\epsilon}_t = \alpha\left(\mathbf{p}_u - \mathbf{i}_t\right). \tag{4.42}$$

From eq. 4.37, the processing of $\widehat{\overline{\mathbf{p}_u}}$ is conducted as follows:

$$\begin{aligned}
\widehat{\overline{\mathbf{p}_u}} &= \mathbf{p}_u + \mathbf{r}_t && (4.43)\\
&= \mathrm{MC}\left(\mathbf{p}_{u-1}\right) + \mathbf{e}_u + \mathbf{r}_t && (4.44)\\
&= \mathrm{MC}\left(\widehat{\overline{\mathbf{p}_{u-1}}} - \mathbf{r}_t\right) + \mathbf{e}_u + \mathbf{r}_t && (4.45)\\
&= \mathrm{MC}\left(\widehat{\overline{\mathbf{p}_{u-1}}}\right) + \mathbf{e}_u - \mathrm{MC}\left(\mathbf{r}_t\right) + \mathbf{r}_t && (4.46)\\
&= \mathrm{MC}\left(\widehat{\overline{\mathbf{p}_{u-1}}}\right) + \mathbf{e}^*_u. && (4.47)
\end{aligned}$$

Hence, in the compressed DCT-domain, eqs. 4.33 and 4.47 are stated as:

$$\overline{\mathbf{I}_t} = \mathbf{I}_t + \mathbf{R}_t, \tag{4.48}$$

$$\widehat{\overline{\mathbf{P}_u}} = \text{MC-DCT}\left(\widehat{\overline{\mathbf{P}_{u-1}}}\right) + \mathbf{E}^*_u. \tag{4.49}$$

Once more, eq. 4.49 is entirely similar to eq. 4.30, representing the decoding operation of INTER type images at the decoder end of the video coding system. $\mathbf{E}^*_u$ represents the processed differences signal and is obtained from the application of the insertion algorithm to the original differences signal:

$$\mathbf{E}^*_u = \mathbf{E}_u - \text{MC-DCT}\left(\mathbf{R}_t\right) + \mathbf{R}_t \tag{4.50}$$

By considering this expression, the algorithm corresponding to this simplified insertion scheme can be formalized as shown in fig. 4.11. In this figure, $\mathbf{A}_n$ represents the DCT coefficients of the $(N \times N)$ pixels block under processing, while $\mathbf{B}_{n-1}$ denotes the coefficients of its prediction block, obtained after the motion compensation operation over the previously decoded image using the current motion vector v_n.

In fig. 4.12 it is illustrated the block diagram corresponding to this insertion architecture. The main advantages of this method are concerned with the reduced

I - INTRA type images:

- if $\mathbf{A}_t \cap \text{NRSO} \neq \emptyset$:
  - $\overline{\mathbf{I}_t} = \mathbf{I}_t + \mathbf{R}_t$;
  - Store $\mathbf{R}_t$ in memory.
- if $\mathbf{A}_t \cap \text{NRSO} = \emptyset$:
  - Do nothing: $\overline{\mathbf{I}_t} = \mathbf{I}_t$.

II - INTER type images:

- if $v_u = (0,0)$:
  - Do nothing: $\mathbf{E}_u^* = \mathbf{E}_u$;
- if $v_u \neq (0,0)$:
  - if $\mathbf{A}_u \cap \text{NRSO} = \emptyset$ and $\mathbf{B}_{u-1} \cap \text{NRSO} = \emptyset$:
    - Do nothing: $\mathbf{E}_u^* = \mathbf{E}_u$;
  - if $\mathbf{A}_u \cap \text{NRSO} = \emptyset$ and $\mathbf{B}_{u-1} \cap \text{NRSO} \neq \emptyset$:
    - The NRSO portion in $\mathbf{B}_{u-1}$ must be removed: $\mathbf{E}_u^* = \mathbf{E}_u - \text{MC-DCT}(\mathbf{R}_t)$;
  - if $\mathbf{A}_u \cap \text{NRSO} \neq \emptyset$ and $\mathbf{B}_{u-1} \cap \text{NRSO} = \emptyset$:
    - The NRSO portion in $\mathbf{A}_u$ must be inserted: $\mathbf{E}_u^* = \mathbf{E}_u + \mathbf{R}_t$;
  - if $\mathbf{A}_u \cap \text{NRSO} \neq \emptyset$ and $\mathbf{B}_{u-1} \cap \text{NRSO} \neq \emptyset$:
    - The NRSO portion in $\mathbf{B}_{u-1}$ must be removed and the NRSO portion in $\mathbf{A}_u$ must be inserted: $\mathbf{E}_u^* = \mathbf{E}_u - \text{MC-DCT}(\mathbf{R}_t) + \mathbf{R}_t$;

Figure 4.11: Open-loop compressed DCT-domain insertion algorithm.

computational load that is required to insert the NRSOs in the input video sequence. Not only are the required operations performed solely in the area affected by the NRSO insertion (which is usually much smaller than the whole image area), but the amount of operations required to process each image block is significantly

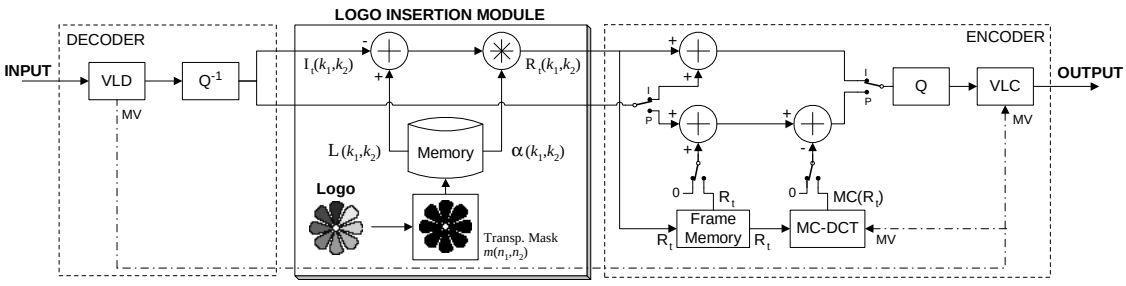


Figure 4.12: Open-loop compressed DCT-domain transcoder for object insertion.

lower. Such reduction is owed to the fact that while the blocks that are processed in INTRA type images only require a 2-D symmetric convolution in the compressed DCT-domain, the blocks that need some processing in INTER type images only require one addition and, when the current MV is non-null, a compressed DCT-domain motion-compensation operation, to remove the fraction of the NRSO that was inserted during the processing of the previously encoded image.

It should also be noted that the required amount of memory was also significantly reduced. While in the presented scheme it is only necessary to store the data corresponding to the fraction of the previously processed INTRA type image where the NRSO was inserted ($\mathbf{R}_t$), in the closed-loop compressed DCT-domain transcoders, presented in sections 4.3.3 and 4.3.4, the insertion algorithm required two image memories. However, while both frame memories of the closed-loop DCT-domain transcoder, as well as the encoder-side memory of the computational-reduced transcoder, have to be updated for each new precessed frame, the frame memory of the open-loop transcoder only has to be updated with the $\mathbf{R}_t$ data when a new INTRA type image is processed.

Furthermore, contrasting with the other transcoder architectures, presented in the previous sections, the transparency factor (α) plays, in this structure, a somewhat different role. More than the transparency degree of the inserted NRSO, it is also of great influence on the possibility of performing the inverse operation (removal of the NRSO that was inserted in the previously processed image), that is required in some macroblocks of INTER type images when the motion vector is non-null. From one side, such situation arises from the fact that the values of the output pixels in the processed area are obtained by a linear combination of the NRSO data (weighted by the α factor) and of the background data (weighted by the $(1-\alpha)$ factor). In the limit situation, when $\alpha = 1$, the result of this linear combination is only composited by the pixels of the NRSO, making it impossible to perform the inverse operation and to recover the background image. In fact, even when $\alpha < 1$, such perfect removal is still compromised, due to the degradation effect introduced by the quantization mechanism. On the other hand, it was already shown that the introduced distortion $\boldsymbol{\epsilon}_t = \alpha\,(\mathbf{p}_u - \mathbf{i}_t)$, resulting from the gradual differences between the encoded images, increases significantly with α. As it will be seen in section 6.2, concerning the experimental results, these limitations play a major rule on the video quality of the output video sequence that is obtained with this open-loop insertion algorithm. At the presence of significant movement or at highly textures areas, it gives rise to a gradual introduction of an observable distortion effect in the neighboring regions close to the pixels areas where the NRSOs

have been inserted. Such effect will prevail along the whole GOP and can only be removed when a new INTRA type frame is encoded.

4.4 Conclusions

A set of new video composition techniques for static object insertion in the compressed DCT-domain was proposed in this chapter. The presented schemes are based on the pixel-domain compositing technique and imply the usage of a symmetric convolution operator in the DCT-domain. This operator provided the means to propose a set of new and efficient processing algorithms. Such algorithms not only operate directly with the decoded DCT-coefficients, received from the incoming video stream, but also restrict their processing to the pixels area corresponding to the objects that are intended to be inserted in the video scene. By using these techniques, the problem concerning the introduction of undesired semi-transparent rectangular regions around irregular-shaped objects (such as logos or subtitles) was circumvented.

The presentation of all these algorithms was carried out with the description of several pixel-domain and DCT-domain transcoding architectures, that were proposed to implement the NRSO insertion system.

References

[6] P. Assunção and M. Ghanbari, "A frequency-domain video transcoder for dynamic bit-rate reduction of MPEG-2 bitstreams," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 8, pp. 953–967, Dec. 1998.

[11] S.-F. Chang and D. G. Messerschmitt, "Manipulation and compositing of MC-DCT compressed video," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 1, pp. 1–11, Jan. 1995.

[28] *MPEG-4: ISO/IEC 14496-2:2004. Information technology – Coding of audio-visual objects – Part 2: Visual*, ISO, 2004.

[36] G. Keesman, R. Hellinghuizen, F. Hoeksema, and G. Heideman, "Transcoding of MPEG bitstreams," *Signal Processing: Image Communication*, vol. 8, pp. 481–500, 1996.

[51] H. Li and H. Shi, "A fast algorithm for reconstructing motion compensated

blocks in compressed domain," *Journal of Visual Languages and Computing*, vol. 10, no. 6, pp. 607–623, Dec. 1999.

[54] C.-W. Lin and Y.-R. Lee, "Fast algorithms for DCT-domain video transcoding," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, Thessaloniki - Greece, Oct. 2001, pp. 421–424.

[56] S. Liu and A. C. Bovik, "Local bandwidth constrained fast inverse motion compensation for DCT domain video transcoding," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Salt Lake City, UT, May 2001.

[57] S. Liu and A. C. Bovik, "Local bandwidth constrained fast inverse motion compensation for DCT-domain video transcoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 5, pp. 309–319, May 2002.

[61] J. Meng and S.-F. Chang, "Embedding visible video watermarks in the compressed domain," *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, vol. 1, pp. 474–477, 1998.

[71] K. Panusopone, X. Chen, and F. Ling, "Logo insertion in MPEG transcoder," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Salt Lake City - USA, May 2001.

[74] F. Pereira and T. Ebrahimi, Eds., *The MPEG-4 Book*. Prentice Hall PTR, 2002.

[77] T. Porter and T. Duff, "Compositing digital images," *Proceedings of the ACM International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, vol. 18, no. 3, pp. 253–259, Jul. 1984.

[86] N. Roma and L. Sousa, "Insertion of irregular-shaped logos in the compressed DCT domain," in *Proceedings of the IEEE International Conference on Digital Signal Processing (DSP)*, vol. 1. Santorini, Greece: IEEE, Jul. 2002, pp. 125–128.

[87] N. Roma and L. Sousa, "Transcoding architectures for object insertion in compressed video," INESC-ID – Lisboa, Portugal, Tech. Rep. RT/006/2002, Oct. 2002.

[88] N. Roma and L. Sousa, "Fast transcoding architectures for insertion of non-regular shaped objects in the compressed DCT-domain," *Signal Processing: Image Communication*, vol. 18, no. 8, pp. 659–683, Sep. 2003.

[97] T. Shanableh and M. Ghanbari, "Transcoding architectures for DCT-domain heterogeneous video transcoding," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, Thessaloniki - Greece, Oct. 2001.

[99] B. Shen and I. K. Sethi, "Block-based manipulations of transformed-compressed images and videos," *ACM Multimedia System Journal*, vol. 6, no. 2, pp. 113–124, Mar. 1998.

[100] B. Shen, I. K. Sethi, and V. Bhaskaran, "DCT convolution and its application in compressed domain," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 8, pp. 947–952, Dec. 1998.

[111] S. Wee and B. Vasudev, "Splicing MPEG video streams in the compressed domain," in *Proc. IEEE Workshop on Multimedia Signal Processing*, Princeton, Jun. 1997.

5

Dynamic Video Composition

Contents

5.1 Introduction

Besides the previously described family of stationary manipulations of precoded video sequences, characterized by fixed insertion operations of one or more static objects, other more complex operations are often required to manipulate and adapt the video sequence that is being transmitted at a given instant. Among the several possible operations that may be offered to the end-user, a particular set of video compositing systems have become increasingly popular along the past few years, to combine two or more video sequences in a single scene. However, contrary to what was observed in the previous chapter for the static video compositing operations, these techniques, herein denoted by dynamic compositing schemes, require the simultaneous processing and combination of the several, and possibly distinct, encoding strategies that are adopted by the video sequences under processing. As an example, one of the most challenging operations that is often required is the adaptation of the temporal prediction schemes of the input sequences, in order to provide an efficient encoding of the output video stream.

Hence, in the scope of this chapter, *dynamic video composition* is defined as a set of techniques that provide the simultaneous processing and composition of several non-stationary video objects. In particular, these objects may be characterized by arbitrary changes on their texture component, from frame to frame, which impose an individual processing of their own temporal prediction mechanisms. Moreover, contrary to what was observed for the *static video composition* techniques, described in the previous chapter, the video objects that are now considered are also characterized by an arbitrary dimension but regular rectangular shape. Nevertheless, the flexibility that is offered by these techniques does not require that such dimension is necessary related to the fixed block or macroblock grid, adopted by the most recent video standards.

With such characteristics and requisites in mind, this chapter will be focused on the proposal of a set of particularly reliable and efficient processing operations, intended to manipulate video sequences in the compressed-domain. The first of these operations, presented in section 5.2, concerns the space-scaling of compressed video sequences by an arbitrary integer scaling factor. The proposed approach is based on an averaging and downsampling method, performed in a hybrid pixel-transform domain, in order to minimize the introduction of any inherent distortion. The proposed method also provides a minimization of the computational cost, by avoiding spurious operations and by only performing those that actually contribute to the evaluation of the output values.

In section 5.3 it will be presented a new motion estimation algorithm that may be implemented in the compressed DCT-domain. This algorithm provides an efficient alternative to refine, or even re-compute, the set of motion vectors that are required to implement the motion compensated prediction mechanism of the output video sequence. The presented algorithm makes use of the DCT coefficients obtained directly from the input video stream and is based on the application of an iterative scheme that estimates the new motion vectors by applying a Least Squares Estimation (LSE) technique.

These two important contributions are applied in the proposal of a new and highly efficient video compositing architecture, that fully operates in the compressed DCT-domain. This architecture will be presented in section 5.4 and directly operates with the partially decoded DCT coefficients of the involved video sequences, thus potentially providing significant advantages in what concerns the output video quality. Moreover, the presented approach relies on a careful treatment of the implemented temporal prediction mechanism, by adopting the DCT-domain motion re-estimation algorithm proposed in section 5.3. Contrary to other refinement schemes previously proposed by other authors, this structure may consider arbitrary search area dimensions to significantly reduce the resulting bit rate, thus increasing the transmission or storage efficiency. Furthermore, contrary to other approaches, the presented DCT-domain technique does not impose any limitation on the composition setup, allowing each foreground video sequence to be placed over any location of the background video scene.

Along the description of the several techniques presented in this chapter, a particular attention will be devoted to properly tailor all the processing steps of these compressed-domain algorithms, so that all operations are performed using the same $(N \times N)$ coefficients block structure that is received from the input video sequences (usually, with $N = 8$). This characteristic is crucial to assure the conformance of the involved operations with most image and video standards, and simultaneously optimize the involved computational effort.

In addition, an optional and possible combination of the presented algorithms with discarding techniques of the highest frequency DCT coefficients will also be considered. These techniques provide a flexible and often required complexity scalability feature, thus giving rise to an adaptable trade-off between the involved scalable computational cost and the resulting video quality and bit rate, in order to meet any system requirements.

5.2 Space scaling algorithm by an arbitrary integer scale factor[§]

In section 3.3.3 it was presented a brief overview of the main space scaling techniques that have been proposed over the past few years. As it was stated, the characteristics of these algorithms do not always comply with the requisites of current space-scaling transcoding systems. In particular, many such schemes are only directly applied to scaling operations using a scale factor ($\mathcal{S}_\mathcal{F}$) given by an integer power of 2 ($\mathcal{S}_\mathcal{F} = 2,\ 4,\ 8,\ 16,$ etc.). Nevertheless, downscaling procedures using any other arbitrary integer scaling factor are often required. As it was also previously referred, in the last few years some proposals have arisen in order to implement these algorithms for any integer scale factors [48, 72, 92, 102, 103]. However, although these proposals provide good video quality for integer powers of 2 scaling ratios, their performance significantly degrades when other scaling factors are applied. Moreover, some of these algorithms do not allow the direct processing of pre-coded data using the fixed ($N \times N$) pixels block structure (usually, with $N = 8$) that is adopted by most digital image and video standards. Besides all these aspects, such transcoding algorithms are often required to exhibit a high computational efficiency level, in order to make it possible to execute them in real-time when implemented in a variety of heterogenous target devices. Consequently, other feasible and reliable alternatives have to be adopted, in order to obtain better quality performances for any arbitrary scaling factors and to achieve the block-based organization found in most image and video standards.

Meanwhile, some authors have distinguished the scaling algorithms in what concerns their output domains [98]. While the input and output blocks of some proposed algorithms are both defined in the DCT-domain, other approaches also process encoded input blocks (DCT-domain) but provide their output in the pixel-domain. The processing of such output blocks can then either continue in the pixel-domain, or an extra DCT computation module can be applied, in order to recover such output into the DCT-domain. As a consequence, this latter kind of approach is often referred to as a *hybrid* algorithm [98].

In this section, it will be presented the proposal of a reliable and very efficient video downscaling method for any arbitrary integer scaling factor, with particularly notable performances for scaling factors other than integer powers of 2. The

[§]Some portions of this section appeared in:

[90] - N. Roma and L. Sousa, "Efficient hybrid DCT-domain algorithm for any arbitrary integer re-size video downscaling," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, no. 57291, pp. 1–16, Sep. 2007.

algorithm adopts an averaging and downsampling approach (see subsection B of section 3.3.3 (page 95)) performed in a hybrid pixel-transform domain, in order to minimize the introduction of any inherent distortion. Moreover, the proposed method also restricts the involved operations, to minimize the computational cost. In fact, such approach allows to avoid spurious and useless computations, by only performing those that are really needed to obtain the output values. Furthermore, all the involved steps are properly tailored, so that all operations are performed using the received $(N \times N)$ coefficient blocks structure, independently of the adopted scaling factor ($\mathcal{S}_{\mathcal{F}}$). This characteristic has never been proposed for this kind of algorithms and is of extreme importance, to simultaneously guarantee the conformance with most image and video standards and optimize the involved computational effort.

In addition, an optional and possible combination of the presented algorithm with high order AC frequency DCT coefficients discarding techniques is also considered. These techniques, usually adopted by *DCT decimation* algorithms (see subsection C of section 3.3.3 (page 97)), confer a scalable computational cost to these downscaling algorithms, providing an useful trade-off between the number of required operations and the resulting video quality and bit rate.

5.2.1 Downscaling algorithms by an arbitrary scale factor

Besides the simplest half-scaling setups previously described in section 3.3.3, many applications have arisen which require arbitrary non-integer scaling factors ($\mathcal{S}_{\mathcal{F}}$). From the digital signal processing point of view, an arbitrary-resize procedure using a scaling factor $\mathcal{S}_{\mathcal{F}} = U/D$ (where U and D may take any non-null relative prime integer values) can be accomplished by cascading an integer upscaling module (by a factor U), followed by an integer downscaling module (by a factor D).

Dugad and Ahuja [16] have shown that by using their proposed DCT decimation technique, the upscaling step can be efficiently implemented by padding with zeros, at the high frequencies, the DCT coefficients of the original image sub-blocks, in order to obtain the corresponding target $(N \times N)$ DCT coefficients blocks of the upscaled image. According to Dugad and Ahuja, since each up sampled block will contain all the frequency content corresponding to its original sub-block, this approach provides better interpolation results when compared with the usage of bilinear interpolation algorithms.

However, as it will be shown in the following, the same does not always happen in what concerns the implementation of the downscaling step using this approach. In the last few years, proposals have arisen in order to implement *DCT decimation* algorithms for any integer scale factor [48, 72, 73, 92, 102, 103]. However, not only

are they directly influenced by the degradation effect resulting from the coefficient discard, but they often suffer from computational inefficiency on their processing, either by storing a large amount of data matrices [102] or by operating with large matrices [72, 73, 92, 103]. One of such proposals was recently presented by Patil et al. [73], who proposed a DCT-decimation approach based on simple matrix multiplications that processes each original DCT frame as a whole, without fragmenting the involved processing by the several macroblocks. However, in practical implementations such approach may lead to serious degradations in what concerns the processing efficiency. The manipulation of such wide matrices may hardly be efficiently carried out in most current processing systems, namely, due to the inherently high cache miss-rate that will be necessarily involved. This can result in an even more serious problem when the processing of high resolution video sequences is considered. By using an alternative and somewhat simpler approach, Lee et al. [48] proposed another downscaling technique by generalizing the previously described DCT-decimation approach, in order to implement the space scaling with any arbitrary integer scaling factor ($\mathcal{S}_\mathcal{F}$), other than integer powers of 2 (e.g. 3, 5, 7, etc). Their methodology is illustrated in fig. 5.1 and can be described as follows:

1. For each original block of $(N \times N)$ pixels $\mathbf{B}_{i,j}$, retain the low-frequency $(K_{\mathcal{S}_\mathcal{F}} \times K_{\mathcal{S}_\mathcal{F}})$ DCT coefficients $\mathbf{B}'_{i,j}$, thus discarding the remaining AC frequency DCT coefficients, with $K_{\mathcal{S}_\mathcal{F}}$ defined as: $K_{\mathcal{S}_\mathcal{F}} = \lceil N/\mathcal{S}_\mathcal{F} \rceil$;

2. Inverse transform each sub-block $\mathbf{B}'_{i,j}$ to the pixel domain, using $\mathbf{b}'_{i,j} = \mathbf{T}^{\mathrm{T}}_{K_{\mathcal{S}_\mathcal{F}}} \mathbf{B}'_{i,j} \mathbf{T}_{K_{\mathcal{S}_\mathcal{F}}}$, where $\mathbf{T}_{K_{\mathcal{S}_\mathcal{F}}}$ is the $K_{\mathcal{S}_\mathcal{F}}$-point DCT kernel matrix;

3. Concatenate $(\mathcal{S}_\mathcal{F} \times \mathcal{S}_\mathcal{F})$ sub-blocks, in order to form an $(N_{\mathcal{S}_\mathcal{F}} \times N_{\mathcal{S}_\mathcal{F}})$ pixels block $\mathbf{b}'$, with $N_{\mathcal{S}_\mathcal{F}}$ defined as $N_{\mathcal{S}_\mathcal{F}} = \mathcal{S}_\mathcal{F}.K_{\mathcal{S}_\mathcal{F}}$:

$$\mathbf{b}' = \begin{bmatrix} \mathbf{b}'_{0,0} & \cdots & \mathbf{b}'_{0,1} \\ \vdots & \ddots & \vdots \\ \mathbf{b}'_{1,0} & \cdots & \mathbf{b}'_{1,1} \end{bmatrix}_{\left(N_{\mathcal{S}_\mathcal{F}} \times N_{\mathcal{S}_\mathcal{F}}\right)} \tag{5.1}$$

4. Compute $\mathbf{B}' = \mathrm{DCT}\left(\mathbf{b}'\right) = \mathbf{T}_{N_{\mathcal{S}_\mathcal{F}}} \mathbf{b}' \mathbf{T}^{\mathrm{T}}_{N_{\mathcal{S}_\mathcal{F}}}$ where $\mathbf{T}_{N_{\mathcal{S}_\mathcal{F}}}$ is the $N_{\mathcal{S}_\mathcal{F}}$-point DCT kernel matrix;

5. Extract the $(N \times N)$ low frequency DCT coefficients of $\mathbf{B}'$ (e.g.: $N = 8$), in order to obtain the $(N \times N)$ DCT-domain scaled block $\hat{\mathbf{B}}$.

Although this methodology is often claimed to provide better performance results than bilinear downscaling approaches in what concerns the obtained video

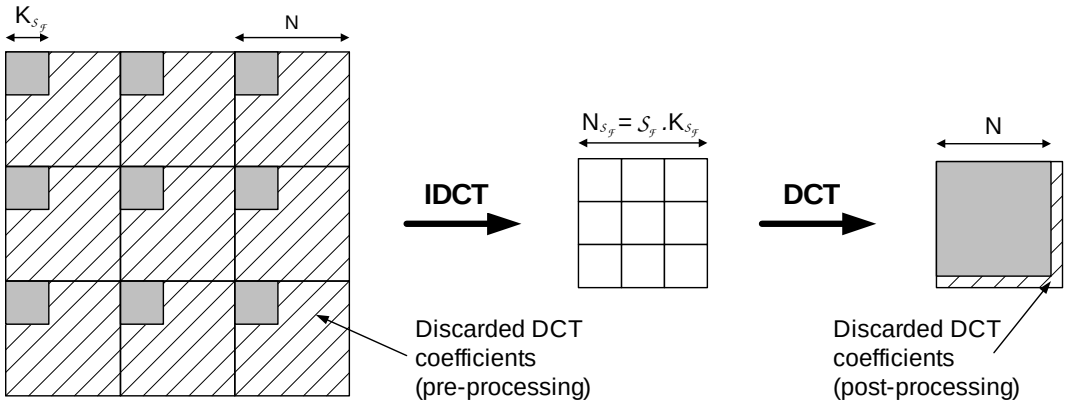


Figure 5.1: Discarded DCT coefficients in arbitrary downscale DCT decimation algorithms (example for $\mathcal{S}_{\mathcal{F}} = 3$).

quality [16, 98], it can be shown that such statement is not always true. In particular, when these generalized DCT decimation downscaling schemes are applied using a scaling factor other than an integer power of 2, it can be shown that the obtained video quality is clearly worse than the provided by the previously described pixel averaging approaches. The reason for the introduction of such degradation comes as a result of the additional discarding procedure of the DCT coefficients, that is referred in step 5 and illustrated in fig. 5.1. Contrary to the first discarding step (performed in step 1), the second discard of high order AC frequency DCT coefficients only occurs for scaling factors other than integer powers of 2 and introduces serious blocky artifacts, mainly in image areas with complex textured regions. To better understand such phenomenon, it is presented in table 5.1 the number of DCT coefficients that is considered along the implementation of this algorithm. As it can be seen, the number of discarded coefficients during the last processing step may be rather significative. The corresponding degradation effect will be thoroughly assessed in section 6.3.1.

Table 5.1: Number of DCT coefficients considered by Lee et al.'s [48] arbitrary downscaling algorithm.

Scaling factor	$\mathcal{S}_{\mathcal{F}}$	2	3	4	5	6	7	8
Number of preserved coefficients in each direction during pre-processing	$K_{\mathcal{S}_{\mathcal{F}}} = \lceil N/\mathcal{S}_{\mathcal{F}} \rceil$	4	3	2	2	2	2	1
Reconstructed downscaled block size	$N_{\mathcal{S}_{\mathcal{F}}} = \mathcal{S}_{\mathcal{F}}.K_{\mathcal{S}_{\mathcal{F}}}$	8	9	8	10	12	14	8
Number of discarded coefficients in each direction during post-processing	$N_{\mathcal{S}_{\mathcal{F}}} - N$	0	1	0	2	4	6	0

To overcome the introduction of such degradation by downscaling algorithms using any arbitrary integer scaling factor, a different approach will be proposed in the next subsection, based on an efficient implementation of a pixel averaging downscaling technique.

5.2.2 Proposed downscaling approach

Considering an arbitrary integer scaling factor $\mathcal{S}_{\mathcal{F}} = \left(\mathcal{S}_{\mathcal{F}x}, \mathcal{S}_{\mathcal{F}y}\right) \in \mathbb{N}^2$, where $\mathcal{S}_{\mathcal{F}x}$ and $\mathcal{S}_{\mathcal{F}y}$ are the horizontal and the vertical downscaling ratios, respectively, the purpose of the presented downscaling algorithm is to compute the $(N \times N)$ DCT encoded block corresponding to a set of $\left(\mathcal{S}_{\mathcal{F}x} \times \mathcal{S}_{\mathcal{F}y}\right)$ original blocks, each one with $(N \times N)$ DCT coefficients.

According to the pixel averaging approach previously described in subsection B of section 3.3.3 (page 95), a generalized arbitrary integer downscaling procedure can be formulated as follows: by denoting $\mathbf{b}$ as the pixels area corresponding to the set of $\left(\mathcal{S}_{\mathcal{F}x} \times \mathcal{S}_{\mathcal{F}y}\right)$ original blocks $\mathbf{b}_{i,j}$, each with $(N \times N)$ pixels,

$$\mathbf{b} = \begin{bmatrix} [\mathbf{b}_{0,0}] & [\mathbf{b}_{0,1}] & \cdots & [\mathbf{b}_{0,\mathcal{S}_{\mathcal{F}x}-1}] \\ [\mathbf{b}_{1,0}] & [\mathbf{b}_{1,1}] & \cdots & [\mathbf{b}_{1,\mathcal{S}_{\mathcal{F}x}-1}] \\ \vdots & \vdots & \ddots & \vdots \\ [\mathbf{b}_{\mathcal{S}_{\mathcal{F}y}-1,0}] & [\mathbf{b}_{\mathcal{S}_{\mathcal{F}y}-1,1}] & \cdots & [\mathbf{b}_{\mathcal{S}_{\mathcal{F}y}-1,\mathcal{S}_{\mathcal{F}x}-1}] \end{bmatrix} \tag{5.2}$$

the downscaled $(N \times N)$ pixels block $(\hat{\mathbf{b}})$ can be obtained by multiplying $\mathbf{b}$ by the downsampling and filtering matrices $\mathbf{f}_{\mathcal{S}_{\mathcal{F}x}}$ and $\mathbf{f}_{\mathcal{S}_{\mathcal{F}y}}$ as follows:

$$\hat{\mathbf{b}} = \left(\frac{1}{\mathcal{S}_{\mathcal{F}x}\mathcal{S}_{\mathcal{F}y}}\right) \times \mathbf{f}_{\mathcal{S}_{\mathcal{F}y}} \cdot \mathbf{b} \cdot \mathbf{f}^{\mathrm{T}}_{\mathcal{S}_{\mathcal{F}x}}, \tag{5.3}$$

where $\mathbf{f}_{\mathcal{S}_{\mathcal{F}q}}$ is a $\left(N \times N\mathcal{S}_{\mathcal{F}q}\right)$ constant matrix with the following structure:

$$f_{\mathcal{S}_{\mathcal{F}q}}(i,j) = \begin{cases} 1 & , \text{ for } i = \left\lfloor \frac{j}{\mathcal{S}_{\mathcal{F}q}} \right\rfloor \text{ and } j \in \{0, \ldots, (N\mathcal{S}_{\mathcal{F}q} - 1)\}, \\ 0 & , \text{ otherwise.} \end{cases} \tag{5.4}$$

These matrices are used to decimate the input image along the two dimensions. To simplify the description, from now on it will be adopted a common scaling factor for both the horizontal and vertical directions ($\mathcal{S}_{\mathcal{F}} = \mathcal{S}_{\mathcal{F}x} = \mathcal{S}_{\mathcal{F}y}$). However, such simplification does not introduce any restriction nor any limitation in the described algorithm. As an example, the $\mathbf{f}_3$ filtering matrix ($\mathcal{S}_{\mathcal{F}} = 3$), considering $N = 5$, is given by eq. 5.5. This matrix may be used to perform image downscaling by a factor of 3: each set of (3×3) blocks, each one composed by (5×5) pixels, is downsampled

in order to obtain a single (5×5) pixels block.

$$\mathbf{f}_3 = \underbrace{\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{f}_3^0} \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{f}_3^1} \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}}_{\mathbf{f}_3^2} \tag{5.5}$$

However, the computation of eq. 5.3 using the filtering matrices defined in eq. 5.4 is usually difficult to handle, since it may involve the manipulation of large matrices. Furthermore, although these filtering matrices may seem reasonably sparse in the pixel-domain, that does not happen when such filtering procedure is transposed into the DCT-domain. Therefore, a significant amount of data, corresponding to the pre-computed filtering matrices, has to be stored. The computation of eq. 5.3 is even harder to accomplish if we take into account that the $(N \times N)$ block structure adopted in image and video coding requires the several involved operations to be performed directly on blocks with $(N \times N)$ elements.

To circumvent all these issues, a different and more efficient approach is now proposed. Firstly, by splitting the $\mathbf{f}_{\mathcal{S}_\mathcal{F}}$ matrix into $\mathcal{S}_\mathcal{F}$ sub-matrices $\mathbf{f}_{\mathcal{S}_\mathcal{F}}^0$, $\mathbf{f}_{\mathcal{S}_\mathcal{F}}^1$, $\ldots \mathbf{f}_{\mathcal{S}_\mathcal{F}}^{\mathcal{S}_\mathcal{F}-1}$, each with $(N \times N)$ elements, the computation of eq. 5.3 can be decomposed in a series of product terms and take a form entirely similar to eq. 3.87:

$$\hat{\mathbf{b}} = \frac{1}{{\mathcal{S}_\mathcal{F}}^2} \left(\mathbf{f}_{\mathcal{S}_\mathcal{F}}^0 \; \mathbf{b}_{00} \; {\mathbf{f}_{\mathcal{S}_\mathcal{F}}^0}^{\mathrm{T}} + \mathbf{f}_{\mathcal{S}_\mathcal{F}}^0 \; \mathbf{b}_{01} \; {\mathbf{f}_{\mathcal{S}_\mathcal{F}}^1}^{\mathrm{T}} + \cdots + \mathbf{f}_{\mathcal{S}_\mathcal{F}}^{(\mathcal{S}_\mathcal{F}-1)} \; \mathbf{b}_{(\mathcal{S}_\mathcal{F}-1)(\mathcal{S}_\mathcal{F}-1)} \; {\mathbf{f}_{\mathcal{S}_\mathcal{F}}^{(\mathcal{S}_\mathcal{F}-1)}}^{\mathrm{T}} \right) \tag{5.6}$$

$$= \frac{1}{{\mathcal{S}_\mathcal{F}}^2} \sum_{i=0}^{\mathcal{S}_\mathcal{F}-1} \sum_{j=0}^{\mathcal{S}_\mathcal{F}-1} \mathbf{f}_{\mathcal{S}_\mathcal{F}}^i \cdot \mathbf{b}_{ij} \cdot {\mathbf{f}_{\mathcal{S}_\mathcal{F}}^j}^{\mathrm{T}}, \tag{5.7}$$

where $\mathbf{b}_{ij}$ are the several input blocks involved in the downscaling operation, directly obtained from the input video sequence. In the bottom of eq. 5.5 it was represented the set of three $(N \times N)$ $\mathbf{f}_{\mathcal{S}_\mathcal{F}}^x$ sub-matrices, for the case with $\mathcal{S}_\mathcal{F} = 3$ and $N = 5$, with $x \in \{0, \ldots, (\mathcal{S}_\mathcal{F} - 1)\}$.

Secondly, the computation of these terms can be greatly simplified if the sparse nature and the high number of zeros of each $\mathbf{f}_{\mathcal{S}_\mathcal{F}}^x$ matrix are taken into account. In particular, it can be shown that each $\mathbf{f}_{\mathcal{S}_\mathcal{F}}^i \cdot \mathbf{b}_{ij} \cdot {\mathbf{f}_{\mathcal{S}_\mathcal{F}}^j}^{\mathrm{T}}$ term only contributes to the computation of a restricted sub-set of pixels of the subsampled block ($\hat{\mathbf{b}}$), within an

area delimited by lines $(l_{min}(i) : l_{max}(i))$ and by columns $(c_{min}(j) : c_{max}(j))$, where:

$$l_{min}(i) = \left\lfloor \frac{i * N}{\mathcal{S}_\mathcal{F}} \right\rfloor \quad \text{and} \quad l_{max}(i) = \left\lfloor \frac{i * N + (N-1)}{\mathcal{S}_\mathcal{F}} \right\rfloor \tag{5.8}$$

$$c_{min}(j) = \left\lfloor \frac{j * N}{\mathcal{S}_\mathcal{F}} \right\rfloor \quad \text{and} \quad c_{max}(j) = \left\lfloor \frac{j * N + (N-1)}{\mathcal{S}_\mathcal{F}} \right\rfloor \tag{5.9}$$

with $i, j = 0 \ldots (\mathcal{S}_\mathcal{F} - 1)$. By denoting the contribution of each block $\mathbf{b}_{i,j}$ to the sampled pixels block $\hat{\mathbf{b}}$ by the $(n_l(i) \times n_c(j))$ matrix $\overline{\mathbf{p}_{i,j}}$, one has:

$$\overline{\mathbf{p}_{i,j}} = \underbrace{\overline{\mathbf{f}^i_{\mathcal{S}_\mathcal{F}}} \cdot \mathbf{b}_{i,j} \cdot \overline{\mathbf{f}^j_{\mathcal{S}_\mathcal{F}}}^{\mathrm{T}}}_{n_l(i) \times n_c(j) \text{ matrix}} \tag{5.10}$$

The terms $\overline{\mathbf{f}^i_{\mathcal{S}_\mathcal{F}}}$ and $\overline{\mathbf{f}^j_{\mathcal{S}_\mathcal{F}}}$ are $(n_l(i) \times N)$ and $(n_c(j) \times N)$ matrices, respectively, with

$$n_l(i) = l_{max}(i) - l_{min}(i) + 1 \tag{5.11}$$

and

$$n_c(j) = c_{max}(j) - c_{min}(j) + 1, \tag{5.12}$$

that are obtained from $\mathbf{f}^i_{\mathcal{S}_\mathcal{F}}$ and $\mathbf{f}^j_{\mathcal{S}_\mathcal{F}}$ by only considering the lines with non-null elements (see the dashed boxes in eq. 5.5).

The resulting $(N \times N)$ pixels sampled block $(\hat{\mathbf{b}})$ is obtained by summing up the contributions of all these terms:

$$\hat{\mathbf{b}} = \frac{1}{{\mathcal{S}_\mathcal{F}}^2} \cdot \left(\sum_{i=0}^{\mathcal{S}_\mathcal{F}-1} \sum_{j=0}^{\mathcal{S}_\mathcal{F}-1} \mathbf{p}_{i,j} \right) \tag{5.13}$$

where:

$$p_{i,j}(l,c) = \begin{cases} \overline{p_{i,j}}(l,c) & , \text{ for } l_{min}(i) \leq l \leq l_{max}(i) \text{ and } c_{min}(j) \leq c \leq c_{max}(j) \\ 0 & , \text{ otherwise} \end{cases} \tag{5.14}$$

with $0 \leq l, c \leq (N-1)$. Such decomposition allows to greatly reduce the overall number of computations, since most of the null terms of the $\mathbf{f}_{\mathcal{S}_\mathcal{F}}$ matrices are no longer considered.

It is also worth noting that some pixels of the sampled block $(\hat{\mathbf{b}})$ may be obtained from several of these product-terms. Such situation will occur whenever the set of $\mathcal{S}_\mathcal{F}$ non-null elements of a given line of the $\mathbf{f}_{\mathcal{S}_\mathcal{F}}$ matrix is split into two distinct $\mathbf{f}^x_{\mathcal{S}_\mathcal{F}}$ sub-matrices (see eq. 5.5). In such case, the value of the output pixel will be the sum of the mutual contribution of adjacent $\mathbf{b}_{i,j}$ $(N \times N)$ pixels blocks. One example of such scenario can be observed in the previously described case with

$\mathcal{S}_\mathcal{F} = 3$ and $N = 5$ (see $\mathbf{f}_3$ matrix in eq. 5.5), illustrated in fig. 5.2. While the pixels of the first row of the sampled $(N \times N)$ output block are obtained with only the subset of original blocks $\{\mathbf{b}_{00}, \mathbf{b}_{01}, \mathbf{b}_{02}\}$, the pixels of the second row are the result of the mutual contribution of the set of blocks $\{\mathbf{b}_{00}, \mathbf{b}_{01}, \mathbf{b}_{02}, \mathbf{b}_{10}, \mathbf{b}_{11}, \mathbf{b}_{12}\}$. The same situation can be verified in what concerns the columns of the output block: while the first column is obtained with blocks $\{\mathbf{b}_{00}, \mathbf{b}_{10}, \mathbf{b}_{20}\}$, the second column is computed with blocks $\{\mathbf{b}_{i0}, \mathbf{b}_{i1}\}$, with $i \in \{0, \ \ldots, \ (\mathcal{S}_\mathcal{F} - 1)\}$.

A particular case also occurs whenever the original frame dimension, in any of its directions, is not an integer multiple of $\mathcal{S}_\mathcal{F}$. In such case, the pixels of the last column (or line) cannot be obtained from ${\mathcal{S}_\mathcal{F}}^2$ input pixels, since only a subset of the original pixels remains to be considered in that line or column. To overcome such situation, the corresponding averaging weights have to be adjusted to the available number of pixels at the end of that line $\left(W_c - \mathcal{S}_\mathcal{F}.\left\lfloor \frac{W_c}{\mathcal{S}_\mathcal{F}} \right\rfloor\right)$ or column $\left(W_l - \mathcal{S}_\mathcal{F}.\left\lfloor \frac{W_l}{\mathcal{S}_\mathcal{F}} \right\rfloor\right)$, where W_c and W_l denote the number of columns and lines of the original image. As an example, the last sampled pixel of a given line should be computed as:

$$\hat{b}\left(:, \left\lfloor \frac{W_c}{\mathcal{S}_\mathcal{F}} \right\rfloor\right) = \frac{1}{\mathcal{S}_\mathcal{F}\left(W_c - \mathcal{S}_\mathcal{F}.\left\lfloor \frac{W_c}{\mathcal{S}_\mathcal{F}} \right\rfloor\right)} \times p\left(:, \left\lfloor \frac{W_c}{\mathcal{S}_\mathcal{F}} \right\rfloor\right) \tag{5.15}$$

This adjustment can be compensated *a posteriori*, by multiplying the pixels of the last column of the sampled block $(\hat{\mathbf{b}})$ by:

$$\hat{b}\left(:, \left\lfloor \frac{W_c}{\mathcal{S}_\mathcal{F}} \right\rfloor\right) = \left[\frac{\mathcal{S}_\mathcal{F}}{W_c - \mathcal{S}_\mathcal{F}.\left\lfloor \frac{W_c}{\mathcal{S}_\mathcal{F}} \right\rfloor}\right] \times \hat{b}\left(:, \left\lfloor \frac{W_c}{\mathcal{S}_\mathcal{F}} \right\rfloor\right) \tag{5.16}$$

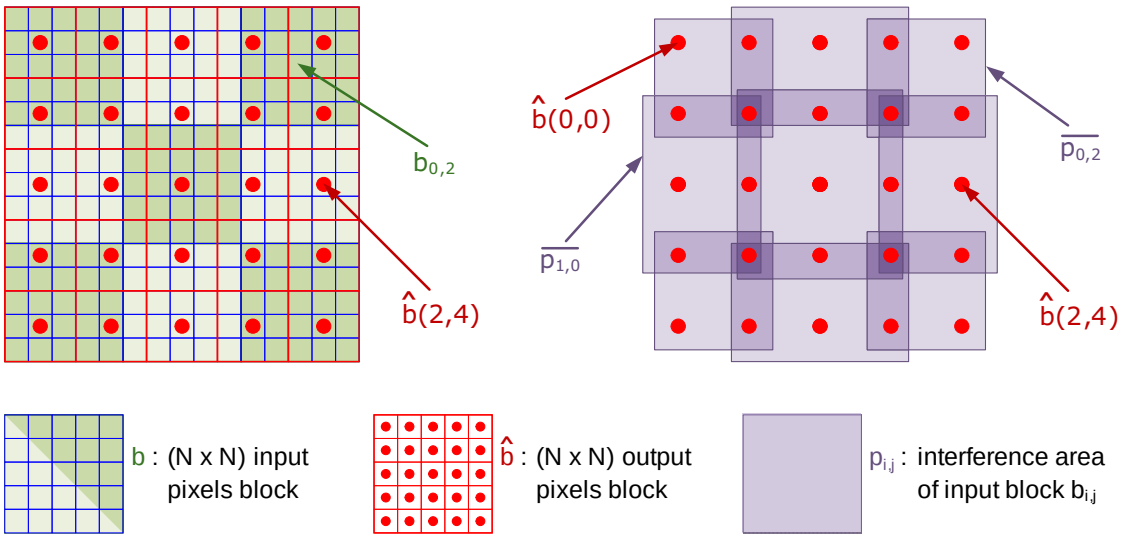


Figure 5.2: Contributions of the several blocks of the original image ($\overline{\mathbf{p}_{i,j}}$) to the final value of each pixel of the sampled block $\hat{\mathbf{b}}$ ($\mathcal{S}_\mathcal{F} = 3$, $N = 5$).

The same applies for the vertical direction of the sampled image.

A - Hybrid downscaling algorithm

As it was referred in section 2.2.5, since the DCT is an unitary orthonormal transform, it is distributive to matrix multiplication. Consequently, the described scaling procedure can be directly performed in the DCT-domain and still provide the previously presented computational advantages. By considering the matrix decomposition formulation to compute the DCT coefficients of a given pixels block $\mathbf{x}$, $\mathbf{X} = \mathbf{T} \cdot \mathbf{x} \cdot \mathbf{T}^{\mathrm{T}}$, eq. 5.13 can be directly computed in the DCT-domain as:

$$\hat{\mathbf{B}} \;=\; \mathbf{T} \cdot \hat{\mathbf{b}} \cdot \mathbf{T}^{\mathrm{T}} \;=\; \frac{1}{{\mathcal{S}_{\mathcal{F}}}^2} \cdot \mathbf{T} \cdot \left(\sum_{i=0}^{\mathcal{S}_{\mathcal{F}}-1} \sum_{j=0}^{\mathcal{S}_{\mathcal{F}}-1} \mathbf{p}_{i,j} \right) \cdot \mathbf{T}^{\mathrm{T}} \tag{5.17}$$

The computation of this expression may be greatly simplified, provided that the definition of the $\mathbf{p}_{i,j}$ matrices, presented in eq. 5.14, is taken into account. In particular, the computation of its $(n_l(i) \times n_c(j))$ non-null elements ($\overline{\mathbf{p}_{i,j}}$) can be carried out as follows:

$$\overline{\mathbf{p}_{i,j}} \;=\; \overline{\mathbf{f}^{i}_{\mathcal{S}_{\mathcal{F}}}} \cdot \mathbf{b}_{i,j} \cdot \overline{\mathbf{f}^{j}_{\mathcal{S}_{\mathcal{F}}}}^{\mathrm{T}} \tag{5.18}$$

$$\;=\; \overline{\mathbf{f}^{i}_{\mathcal{S}_{\mathcal{F}}}} \cdot \mathbf{T}^{\mathrm{T}} \cdot \mathbf{B}_{i,j} \cdot \mathbf{T} \cdot \overline{\mathbf{f}^{j}_{\mathcal{S}_{\mathcal{F}}}}^{\mathrm{T}} \tag{5.19}$$

By denoting the product $\overline{\mathbf{f}^{i}_{\mathcal{S}_{\mathcal{F}}}} \cdot \mathbf{T}^{\mathrm{T}}$ by the $(n_l(i) \times N)$ matrix $\overline{\mathbf{F}^{i}_{\mathcal{S}_{\mathcal{F}}}}$ and the product $\overline{\mathbf{f}^{j}_{\mathcal{S}_{\mathcal{F}}}} \cdot \mathbf{T}^{\mathrm{T}}$ by the $(n_c(j) \times N)$ matrix $\overline{\mathbf{F}^{j}_{\mathcal{S}_{\mathcal{F}}}}$, the above expression can be represented as:

$$\overline{\mathbf{p}_{i,j}} = \underbrace{\overline{\mathbf{F}^{i}_{\mathcal{S}_{\mathcal{F}}}} \cdot \mathbf{B}_{i,j} \cdot \overline{\mathbf{F}^{j}_{\mathcal{S}_{\mathcal{F}}}}^{\mathrm{T}}}_{n_l(i) \times n_c(j) \text{ matrix}} \tag{5.20}$$

where $\mathbf{B}_{i,j}$ is the $(N \times N)$ DCT coefficients block directly obtained from the partially decoded bit stream. Since all the $\overline{\mathbf{F}^{x}_{\mathcal{S}_{\mathcal{F}}}}$ terms (with $0 \leq x \leq \mathcal{S}_{\mathcal{F}} - 1$) are constant matrices, they can be pre-computed and stored in memory.

Furthermore, the overall computational cost of the described procedure can still be further reduced if the usage of *partial DCT information* [51, 54, 57] techniques is considered, as it will be shown in the following subsections.

B - DCT-domain pre-filtering for computational cost reduction

The advantages of the previously described hybrid downscaling scheme in what concerns the computational cost can be regarded as the result of an efficient implementation of the following cascaded processing steps: *inverse DCT*, *low-pass filtering* (averaging), *downsampling* and *direct DCT* (see fig. 5.3). However, the

(a) Proposed procedure.

(b) Equivalent approach.

Figure 5.3: DCT-domain frame scaling procedure.

efficiency of this procedure can be further improved by noting that the signal component corresponding to most of the high order AC frequency DCT coefficients, obtained from the first implicit processing step (*inverse DCT*), is almost completely discarded as the result of the second step (*low-pass filtering*). Hence, the overall computational cost of this scheme can be significantly reduced by introducing an optional low-pass *pre-filtering* stage in the inverse DCT processing step, which is directly implemented by only considering a sub-set of the original DCT coefficients. By denoting K as the maximum bandwidth of this low-pass pre-filter, given by the highest line/column index of the considered DCT coefficients, only the coefficients $\tilde{\mathbf{B}}_{i,j}(m,n) = \{\mathbf{B}_{i,j}(m,n) : m,n \leq K\}$ will be used for the inverse DCT operation. In practice, this pre-filtering can be formulated as follows:

$$\tilde{\mathbf{B}}_{i,j} = \begin{bmatrix} [\mathbf{I}]_{K\times K} & 0 \\ 0 & 0 \end{bmatrix} \cdot \mathbf{B}_{i,j} \cdot \begin{bmatrix} [\mathbf{I}]_{K\times K} & 0 \\ 0 & 0 \end{bmatrix}^{\mathrm{T}} \tag{5.21}$$

$$= \begin{bmatrix} [\mathbf{B}_{i,j}]_{K\times K} & 0 \\ 0 & 0 \end{bmatrix} \tag{5.22}$$

where $[\mathbf{I}]_{K\times K}$ is the $(K \times K)$ identity matrix corresponding to the considered pre-filter and $[\mathbf{B}_{i,j}]_{K\times K}$ is a $(K \times K)$ sub-matrix of $\mathbf{B}_{i,j}$, obtained by extracting the $(K \times K)$ lower-order DCT coefficients. Thus, the representative contribution of $\mathbf{B}_{i,j}$ to the output pixels $\overline{\mathbf{p}_{i,j}}$ (see eq. 5.20) can be obtained as:

$$[\overline{\mathbf{p}_{i,j}}]_{n_l(i)\times n_c(j)} = \left[\overline{\mathbf{F}^i_{\mathcal{S}_\mathcal{F}}}\right]_{n_l(i)\times K} \cdot \left[\tilde{\mathbf{B}}_{i,j}\right]_{K\times K} \cdot \left[\overline{\mathbf{F}^j_{\mathcal{S}_\mathcal{F}}}^{\mathrm{T}}\right]_{K\times n_c(j)}. \tag{5.23}$$

By adopting this scheme, the proposed procedure provides a full control over the final accuracy level towards a perfect fulfillment of any real-time requirement, thus providing an useful trade-off between speed and accuracy. Furthermore, by considering that the $\mathbf{B}_{i,j}$ matrices usually have most of their high order AC frequency coefficients equal to zero, and provided that K is not too small, the distortion resulting from this scheme is often negligible, as it will be shown in section 6.3.1.

5.2.3 Algorithm

In fig. 5.4 it is formally stated the proposed hybrid downscaling algorithm, where $(linS, colS)$ are the block coordinates within the target (scaled) image; (l, c) are the coordinates within the set of ${\mathcal{S}_\mathcal{F}}^2$ blocks being sampled; and l_{min}, l_{max}, c_{min} and c_{max}, defined in eqs. 5.8 and 5.9, respectively, are the bounding coordinates of the target block area affected by each iteration.

I - Initialization:

- Compute and store in memory the set of $\overline{\mathbf{F}^x_{\mathcal{S}_\mathcal{F}}}$ matrices;

II - Computation:

```
for linS=0 to (W_l/S_F − 1), linS+=N do
  for colS=0 to (W_c/S_F − 1), colS+=N do
    for l=0 to (S_F − 1) do
      for c=0 to (S_F − 1) do
        [p̄_{l,c}]_{n_l×n_c} = [F̄^l_{S_F}]_{n_l×K} · [B̃_{l,c}]_{K×K} · [F̄^c_{S_F}]^T_{K×n_c}
        b̂(l_min : l_max, c_min : c_max) += 1/S_F² [p̄_{i,j}]_{n_l×n_c}
      end for
    end for
    [B̂]_{N×N} = [T]_{N×N} · [b̂]_{N×N} · [T^T]_{N×N}
  end for
end for
```

Figure 5.4: Proposed hybrid pixel/transform-domain scaling algorithm.

To evaluate the computational cost of the proposed algorithm, the number of multiplications ($\mathcal{M}$) required to process each of the $(W_c \times W_l)$ pixels of the original frame was considered as the main figure of merit. Furthermore, two additional downscaling algorithms were also considered, to assess the computational advantages provided by the proposed algorithm:

- Cascaded Pixel Averaging Transcoder (CPAT);
- DCT Decimation Transcoder (DDT).

By adopting the same complexity metric, their computational costs were evaluated and compared.

Cascaded Pixel Averaging Transcoder (CPAT)

In the Cascaded Pixel Averaging Transcoder, shown in fig. 5.3(b), the filtering and downsampling processing steps are entirely carried-out in the pixel-domain,

by firstly decoding the whole set of DCT coefficients received from the incoming video stream. Most of the operations that are required to process each scaled block ($\hat{\mathbf{B}}$) are performed in the computation of the ${\mathcal{S}_\mathcal{F}}^2$ IDCTs, each one requiring $2N^3$ multiplications, since a single multiplication is required to compute the average of each set of $(\mathcal{S}_\mathcal{F} \times \mathcal{S}_\mathcal{F})$ pixels:

$$\mathcal{M}\,(\text{CPAT}) \;=\; \frac{1}{W_l W_c}\left[\underbrace{\frac{W_l W_c}{N^2}\cdot 2N^3}_{\text{IDCTs}} \;+\; \underbrace{\frac{W_l W_c}{{\mathcal{S}_\mathcal{F}}^2}\cdot 1}_{\text{averaging}} \;+\; \underbrace{\frac{W_l W_c}{{\mathcal{S}_\mathcal{F}}^2 N^2}\cdot 2N^3}_{\text{DCTs}}\right] \tag{5.24}$$

By considering that $\frac{1}{{\mathcal{S}_\mathcal{F}}^2} \ll 1$, it can be approximately formulated as:

$$\mathcal{M}\,(\text{CPAT}) \;\approx\; 2N. \tag{5.25}$$

DCT Decimation Transcoder (DDT)

As it was described in section 5.2.1 about the DCT Decimation Transcoder with arbitrary integer scaling factors, formulated by Lee et al. [48], most operations that are required to process each scaled block ($\hat{\mathbf{B}}$) are performed in the computation of the ${\mathcal{S}_\mathcal{F}}^2$ IDCTs, each one requiring $2K^3$ multiplications (with $K = \left\lceil \frac{N}{\mathcal{S}_\mathcal{F}} \right\rceil$), and of the final $(K\mathcal{S}_\mathcal{F})$-point DCT:

$$\mathcal{M}\,(\text{DDT}) \;=\; \frac{1}{W_l W_c}\left(\underbrace{\frac{W_l W_c}{N^2}\,2K^3}_{\text{IDCTs}} \;+\; \underbrace{\frac{W_l W_c}{{\mathcal{S}_\mathcal{F}}^2 N^2}\,2\,(K.\mathcal{S}_\mathcal{F})^3}_{\text{DCT}}\right) \tag{5.26}$$

$$\;=\; \frac{2K^3}{N^2}\;(1+\mathcal{S}_\mathcal{F}) \tag{5.27}$$

This computational metric can be approximately formulated as:

$$\mathcal{M}\,(\text{DDT}) \;\approx\; \frac{2K^3\mathcal{S}_\mathcal{F}}{N^2}. \tag{5.28}$$

Hybrid Downscaling Transcoder (HDT)

To estimate the overall computational cost of the proposed Hybrid Downscaling Transcoder, let us first evaluate the cost of computing each $\overline{\mathbf{p}_{i,j}}$ matrix:

$$\mathcal{M}\,(\overline{\mathbf{p}_{i,j}}) = n_l(i).K.K + n_l(i).K.n_c(j), \tag{5.29}$$

where $n_l(i)$ and $n_c(j)$ were defined in eqs. 5.11 and 5.12. Hence, it follows that:

$$\mathcal{M}\left(\text{HDT}\right) = \frac{1}{W_l W_c}\left[\frac{W_l W_c}{{\mathcal{S}_\mathcal{F}}^2 N^2}\left(\underbrace{\sum_{i=0}^{\mathcal{S}_\mathcal{F}-1}\sum_{j=0}^{\mathcal{S}_\mathcal{F}-1}\mathcal{M}\left(\overline{\mathbf{p}_{i,j}}\right)}_{\text{IDCTs + averaging + scaling}} + \underbrace{2N^3}_{\text{DCTs}}\right)\right] \tag{5.30}$$

where:

$$\sum_{i=0}^{\mathcal{S}_\mathcal{F}-1}\sum_{j=0}^{\mathcal{S}_\mathcal{F}-1}\mathcal{M}\left(\overline{\mathbf{p}_{i,j}}\right) = K^2\mathcal{S}_\mathcal{F}\sum_{i=0}^{\mathcal{S}_\mathcal{F}-1} n_l(i) + K\left(\sum_{i=0}^{\mathcal{S}_\mathcal{F}-1} n_l(i)\right)\left(\sum_{j=0}^{\mathcal{S}_\mathcal{F}-1} n_c(j)\right). \tag{5.31}$$

By generically defining

$$n_q = \left\lfloor\frac{qN + (N-1)}{\mathcal{S}_\mathcal{F}}\right\rfloor - \left\lfloor\frac{qN}{\mathcal{S}_\mathcal{F}}\right\rfloor + 1 \tag{5.32}$$

as the number of lines of each $\overline{\mathbf{f}^q_{\mathcal{S}_\mathcal{F}}}$ matrix, it can be shown that:

$$N \leq \sum_{q=0}^{\mathcal{S}_\mathcal{F}-1} n_q < \mathcal{S}_\mathcal{F}\left(\left\lfloor\frac{N}{\mathcal{S}_\mathcal{F}}\right\rfloor + 2\right), \tag{5.33}$$

where the lower limit of the previous expression corresponds to the case when N is an integer multiple of $\mathcal{S}_\mathcal{F}$. On the other hand, the upper limit corresponds to a hypothetical worst case situation, when the set of $\mathcal{S}_\mathcal{F}$ non-null elements of both the upper and the lower lines of each $\overline{\mathbf{f}^q_{\mathcal{S}_\mathcal{F}}}$ matrix are split across different $\overline{\mathbf{f}^q_{\mathcal{S}_\mathcal{F}}}$ matrices (see eq. 5.5). Thus:

$$\sum_{i=0}^{\mathcal{S}_\mathcal{F}-1}\sum_{j=0}^{\mathcal{S}_\mathcal{F}-1}\mathcal{M}\left(\overline{\mathbf{p}_{i,j}}\right) < K^2{\mathcal{S}_\mathcal{F}}^2\left(\left\lfloor\frac{N}{\mathcal{S}_\mathcal{F}}\right\rfloor + 2\right) + K{\mathcal{S}_\mathcal{F}}^2\left(\left\lfloor\frac{N}{\mathcal{S}_\mathcal{F}}\right\rfloor + 2\right)^2. \tag{5.34}$$

By using the above relation, as well as eq. 5.30, one can obtain:

$$\mathcal{M}\left(\text{HDT}\right) < \frac{1}{{\mathcal{S}_\mathcal{F}}^2 N^2}\left[K^2{\mathcal{S}_\mathcal{F}}^2\left(\left\lfloor\frac{N}{\mathcal{S}_\mathcal{F}}\right\rfloor + 2\right) + K{\mathcal{S}_\mathcal{F}}^2\left(\left\lfloor\frac{N}{\mathcal{S}_\mathcal{F}}\right\rfloor + 2\right)^2 + 2N^3\right], \tag{5.35}$$

which can be approximately formulated as:

$$\mathcal{M}\left(\text{HDT}\right) \approx \frac{KN\mathcal{S}_\mathcal{F}\left(K+4\right) + 2\left(N^3 + K^2{\mathcal{S}_\mathcal{F}}^2\right)}{N^2{\mathcal{S}_\mathcal{F}}^2}. \tag{5.36}$$

Comparison

In table 5.2 it is presented the obtained comparison in what concerns the involved computational cost, both in terms of the adopted scaling factor ($\mathcal{S}_\mathcal{F}$) and of the

Table 5.2: Comparison of the several considered downscaling approaches in what concerns the involved computational cost.

Algorithm	DCT coeffs.	$\mathcal{M}$	Comparison
CPAT	N	$2N$	$\frac{\mathcal{M}(\text{HDT})}{\mathcal{M}(\text{CPAT\_N})} \propto \frac{1}{\mathcal{S}_\mathcal{F}}$
DDT	$K = \left\lceil \frac{N}{\mathcal{S}_\mathcal{F}} \right\rceil$	$\frac{2K^3\mathcal{S}_\mathcal{F}}{N^2}$	$\frac{\mathcal{M}(\text{HDT})}{\mathcal{M}(\text{DDT})} \propto \frac{1}{{\mathcal{S}_\mathcal{F}}^2}$
HDT	$K \in [1, N]$	$\frac{KN\mathcal{S}_\mathcal{F}(K+4)+2\left(N^3+K^2{\mathcal{S}_\mathcal{F}}^2\right)}{N^2{\mathcal{S}_\mathcal{F}}^2}$	1

considered number of DCT coefficients (K). The presented comparison relations were extracted by not considering the impact of the discarded constant factors in eqs. 5.24, 5.27 and 5.35.

$$\frac{\mathcal{M}\,(\text{HDT})}{\mathcal{M}\,(\text{CPAT})} \approx \frac{K^2\mathcal{S}_\mathcal{F} + 2N^2}{2N^2{\mathcal{S}_\mathcal{F}}^2} \propto \frac{1}{\mathcal{S}_\mathcal{F}} \tag{5.37}$$

$$\frac{\mathcal{M}\,(\text{HDT})}{\mathcal{M}\,(\text{DDT})} \approx \frac{K}{N}\frac{1}{{\mathcal{S}_\mathcal{F}}^2} \propto \begin{cases} \frac{1}{{\mathcal{S}_\mathcal{F}}^2} & \text{for } K = N \\ \frac{1}{\mathcal{S}_\mathcal{F}} & \text{for } K = \frac{N}{\mathcal{S}_\mathcal{F}} \end{cases} \tag{5.38}$$

This comparison clearly evidences the advantages in what concerns the computational cost provided by the proposed algorithm, when compared with other considered approaches and, in particular, with the DCT decimation transcoder (DDT). Such advantages are even more significant when higher scaling factors are considered, as it will be demonstrated in section 6.3.1.

Before concluding this presentation, it is worth noting that although the presented comparative study was entirely carried out by considering the usage of the matrix decomposition algorithm to compute the DCT coefficients ($\mathbf{X}$) of a given pixels block $\mathbf{x}$: $\mathbf{X} = \mathbf{T} \cdot \mathbf{x} \cdot \mathbf{T}^\text{T}$, similar results could equally be obtained if other faster DCT computation algorithms (such as [14]) were adopted in the pixel-domain architectures that were considered in this evaluation. It can be shown that, although the resulting complexity ratios would be slightly different from those presented above, they would still be entirely favorable to the proposed hybrid pixel/transform-domain scaling scheme.

5.3 Block-based motion re-estimation in the DCT-domain[§]

As it was mentioned in section 3.3.5, the possibility to implement ME procedures in the compressed-domain has deserved a significant interest in the last few years. Most ME techniques that have been proposed up until now for compressed-domain video transcoding systems confine their procedure to a simple re-usage and composition of the MVs already decoded from the received video stream. Such procedure is accomplished by applying one of the several composition schemes described in section 3.3.4, such as the simple mean, the weighted average, the median, etc. [12, 52, 108]. However, most of these techniques often lead to non-optimal motion compensated prediction results, due to the mismatch between the prediction and residual components. Consequently, they often require a post-processing re-estimation stage to refine the composited MVs. In practice, it is often observed that a refinement procedure around the composited MV within a range of a couple of pixels may often offer similar performance levels as those obtained with a full-search motion re-estimation [95]. Moreover, few algorithms have been presented to estimate entirely new MVs in cases of complete absence of any precoded MV in the input video sequences (e.g. INTRA to INTER frame conversion, frame dropping video time scaling [115]) or when those MVs do not have any correlation with the motion activity in the new encoded frame (e.g. video composition [9, 50, 67], logo insertion [88], etc.).

To circumvent such limitation, some proposals have been presented to perform motion estimation directly in the DCT-domain, as it was described in section 3.3.5 and summarized in the following paragraphs:

- *Pixel/DCT-domain cascaded transcoders*, described in subsection A of section 3.3.5 (page 107) - These structures re-use the MVs extracted from the incoming bit stream and obtain a prediction of the new MV by using one of the several possible composition schemes described in section 3.3.4. An optimized MV can then be obtained by performing a new block-matching motion-estimation procedure. The advantage of these schemes is their ability to provide a video quality level that is comparable to the one that could be obtained by going through a new full-search ME procedure, but using consid-

[§]Some portions of this section appeared in:

[89] - N. Roma and L. Sousa, "Least squares motion estimation algorithm in the compressed DCT domain for H.26x/MPEG-x video sequences," in *Proceedings of the IEEE International Conference on Advanced Video and Signal-Based Surveillance (AVSS)*. Como - Italy: IEEE, Sep. 2005, pp. 576–581.

erably less computations.

- *Convolution based transcoders*, described in subsection B of section 3.3.5 (page 109) - These structures are based on the convolution-multiplication relationship for the discrete cosine and sine transforms. By providing a feasible alternative to compute the convolution of two shifted images, this technique makes it possible to easily determine the position of the peak within the convolution result, which will indicate the magnitude of the disparity of the two considered images. However, this symmetric convolution approach presents one important limitation, since it only provides useful results for integral disparities of zero or one. Consequently, the poor accuracy of the results that it provides makes it not suitable for current video transcoding applications.

- *Least squares based transcoders*, described in subsection C of section 3.3.5 (page 112) - These structures, which were first proposed by Reeves [83], were originally applied in photogrammetric techniques for image registration of JPEG compressed aerial photos and tried to apply matching techniques using the convolution in the DCT-domain. Unfortunately, the obtained experimental results demonstrated that not only does this methodology present a poor precision level in the obtained values, but it also suffers from a sign ambiguity in the disparity result.

- *DCT pseudo-phases based transcoders*, described in subsection D of section 3.3.5 (page 113) - These structures were first proposed by Koc and Liu [41] and are based on the application of the DCT pseudo-phases techniques. Such techniques employ the sinusoidal orthogonal principle to extract the shift information from the pseudo-phases hidden in the 2-D DCT coefficients of the second kind (DCCT-IIe), decoded from the received video stream. However, as it was already referred, this algorithm presents some important drawbacks that significantly difficult its usage in practical transcoding systems: not only does it involve a significant computational cost, but it also makes use of other types of the discrete sine and cosine transforms of the processed blocks that usually cannot be directly obtained from the video stream. In fact, not only does it require the computation of the discrete cosine transform of the first kind (DCCT-I) of the blocks under processing, but it also needs to compute the four variations of the discrete sine/cosine trigonometric transforms of the second kind: DCCT-II, DCST-II, DSCT-II and DSST-II. Moreover, all these 2D transformations must be computed for an image area corresponding to the macroblock under processing, which implies a further computational effort in

order to calculate the coefficients obtained from the concatenation of the four involved decoded blocks.

Consequently, a different approach is now presented in this section to perform the estimation of new MVs using the DCT coefficients directly obtained from a video stream. The proposed algorithm is based on an iterative scheme that estimates the new MVs by applying a Least Squares Estimation (LSE) technique and following an approach somewhat similar to the one frequently adopted in image registration procedures [83]. Moreover, to reduce its computational effort, the algorithm may consider only an arbitrary subset of non-null DCT coefficients, by following an approach entirely similar to the *partial DCT information* technique [51], also adopted in the previously proposed spatial downscale algorithm, presented in subsection B of section 5.2.2 (page 174).

5.3.1 Linear least squares estimation

The LSE method is frequently used to fit a set of observed points to a given theoretical model, defined by a set of adjustable parameters. One of the most popular applications of this method is found in linear regression, to fit a set of M data points (x_i, y_i) to a straight-line model:

$$y(x) = y(x; a, b) = a + bx + n(x), \tag{5.39}$$

where $n(x)$ is the noise introduced by the sampling procedure.

The solution of this optimization process is often found by minimizing a merit function that estimates the error between the theoretical model and the observed samples. When $n(x)$ is Gaussian, the obtained solution is optimal and is computed by minimizing:

$$\varepsilon^2(a, b) = \sum_{i=1}^{M} (y_i - a - bx_i)^2 \tag{5.40}$$

At this minimum, the derivatives of $\varepsilon^2(a, b)$ with respect to a and b will vanish:

$$\frac{\partial \varepsilon^2(a, b)}{\partial a} = -2 \sum_{i=1}^{M} (y_i - a - bx_i) = 0 \tag{5.41}$$

$$\frac{\partial \varepsilon^2(a, b)}{\partial b} = -2 \sum_{i=1}^{M} x_i (y_i - a - bx_i) = 0 \tag{5.42}$$

and the optimal values for the parameters a and b can be found by solving a linear system of equations.

Models like this, where there is a linear function in the estimated parameters, are denoted by *linear models*. On the other hand, when the models are not linear in the parameters, the solution can not be found by simply solving a linear system of equations. In such cases, the model is often denoted by:

$$y = f(x;\lambda) + n(x), \tag{5.43}$$

where $f(x;\lambda)$ is a known function, that is non-linear in the parameter λ. The least squares criterion for these cases becomes:

$$\varepsilon^2(\lambda) = \sum_{i=1}^{M} \left(y_i - f(x_i,\lambda)\right)^2 . \tag{5.44}$$

By minimizing the derivatives of ε^2 with respect to the parameters λ_j, one obtains:

$$\frac{\partial \varepsilon^2(\lambda)}{\partial \lambda_j} = 0 \tag{5.45}$$

$$\Leftrightarrow -2\sum_{i=1}^{M} \left(y_i - f(x_i,\lambda)\right) \frac{\partial f(x_i,\lambda)}{\partial \lambda_j}\bigg|_{\lambda_j=\lambda_j} = 0 \tag{5.46}$$

Since $f(x,\lambda)$ is non-linear in the parameter λ_j, these equations are usually quite difficult to solve. In some *non-linear models*, like this, a Taylor series expansion is often used to approximate the non-linear regression model to a simpler model defined with linear terms. The linear LSE method is then applied to estimate the model parameters.

5.3.2 Least squares motion estimation

The ME approach used in video coding has a lot in common with the non-linear LSE technique described above. The main objective of such procedure is to find the best predicting macroblock in a search region defined in a reference image $s(x_1,x_2)$, that best matches the macroblock of the current image $r(x_1,x_2)$ under processing. Since most video standards restrict this displacement to a simple translational model, the output of this algorithm will be a motion vector $\mathbf{v} = (v_1, v_2)$ that defines the distance between the macroblock in the current image $r(x_1,x_2)$ and the best predicting macroblock in the reference image $s(x_1+v_1, x_2+v_2)$.

However, although this translational model can be described by a simple linear equation:

$$\mathbf{y} = \mathbf{x} + \mathbf{v} \tag{5.47}$$

the procedure to find the best motion vector associated to all pixels of a given macroblock is highly non-linear. Nevertheless, the non-linear LSE model, as described in eq. 5.43, proves to be quite suitable to be adopted and applied to the current and reference image macroblocks:

$$r(x_1, x_2) \simeq s(x_1 + v_1, x_2 + v_2). \tag{5.48}$$

By decomposing this expression in a first-order Taylor series expansion:

$$r(x_1, x_2) \simeq \; s(x_1 + v_1^0, x_2 + v_2^0) + \sum_{n=1}^{2} \left.\frac{\partial s}{\partial v_n}\right|_{v=v^0} \left(v_n - v_n^0\right) \tag{5.49}$$

$$\simeq \; s(x_1 + v_1^0, x_2 + v_2^0) + \left.\frac{\partial s}{\partial v_1}\right|_{v=v^0} \left(v_1 - v_1^0\right) + \left.\frac{\partial s}{\partial v_2}\right|_{v=v^0} \left(v_2 - v_2^0\right), \tag{5.50}$$

the desired displacement parameter v can be estimated by following an iterative procedure and by computing the several increments $dv^i = v^i - v^{i-1}$, where $v^0 = (v_1^0, v_2^0)$ is a preliminary prediction of the desired motion vector. As it was previously referred, to accelerate this iterative procedure, an initial MV prediction can be obtained by compositing the MVs decoded from the received video sequences. Nevertheless, whenever such prediction cannot be obtained, a null motion vector $v^0 = (0, 0)$ should be used for this initial estimate.

$$r(x_1, x_2) \simeq \; s(x_1 + v_1^i, x_2 + v_2^i) + \left.\frac{\partial s}{\partial v_1}\right|_{v=v^{i-1}} dv_1^i + \left.\frac{\partial s}{\partial v_2}\right|_{v=v^{i-1}} dv_2^i \tag{5.51}$$

The best matching macroblock is obtained by minimizing the prediction error $e(x_1, x_2)$:

$$e(x_1, x_2) = r(x_1, x_2) - s(x_1 + v_1^i, x_2 + v_2^i) \simeq \left.\frac{\partial s}{\partial v_1}\right|_{v=v^{i-1}} dv_1^i + \left.\frac{\partial s}{\partial v_2}\right|_{v=v^{i-1}} dv_2^i \tag{5.52}$$

This equation can be represented as a linear system with M equations and 2 unknowns (dv_1^i and dv_2^i), where M denotes the number of considered samples:

$$\mathbf{e} = \mathbf{j}_s \cdot \mathbf{dv}^i \tag{5.53}$$

In this equation, the vector $\mathbf{e}$ denotes a $(M \times 1)$ matrix obtained by re-arranging the columns of the difference matrix $\mathbf{e} = \mathbf{r} - \mathbf{s}$:

$$\mathbf{e} = \begin{bmatrix} \vdots \\ r(x_l, x_c) - s(x_l + v_1^i, x_c + v_2^i) \\ \vdots \end{bmatrix}_{M \times 1} \tag{5.54}$$

On the other hand, $\mathbf{j}_s$ is a $(M \times 2)$ Jacobian matrix whose columns contain the partial derivatives of s disposed in major column order:

$$\mathbf{j}_s = \begin{bmatrix} \vdots & \vdots \\ \frac{\partial s}{\partial v_1} & \frac{\partial s}{\partial v_2} \\ \vdots & \vdots \end{bmatrix}_{M \times 2} \tag{5.55}$$

and $\mathbf{dv}$ is the desired partial displacement matrix:

$$\mathbf{dv}^i = \begin{bmatrix} dv_1^i \\ dv_2^i \end{bmatrix}_{2 \times 1}. \tag{5.56}$$

Hence, eq. 5.53 can be solved by simple algebraic manipulation:

$$\mathbf{e} = \mathbf{j}_s \cdot \mathbf{dv}^i \tag{5.57}$$

$$\Leftrightarrow {\mathbf{j}_s}^{\mathrm{T}} \cdot \mathbf{e} = {\mathbf{j}_s}^{\mathrm{T}} \cdot \mathbf{j}_s \cdot \mathbf{dv}^i \tag{5.58}$$

$$\Leftrightarrow \mathbf{dv}^i = \left({\mathbf{j}_s}^{\mathrm{T}} \cdot \mathbf{j}_s\right)^{-1} \cdot {\mathbf{j}_s}^{\mathrm{T}} \cdot \mathbf{e} \tag{5.59}$$

At the end of each iteration, the obtained displacement $\mathbf{v}$ is updated: $\mathbf{v}^i = \mathbf{v}^{i-1} + \mathbf{dv}^i$ and a new displaced image area s is obtained by performing a motion compensation and an interpolation procedures: $s^i(x_1 + v_1^i, x_2 + v_2^i)$. The algorithm is then repeated by computing the value of $\frac{\partial s}{\partial v}\big|_{v=v^i;\ s=s^i}$.

This iterative process will continue until an arbitrary stop condition is met. As an example:

$$\|v^i - v^{i-1}\| < \delta \tag{5.60}$$

for an arbitrarily small δ value.

5.3.3 Least squares motion estimation in the DCT-domain

By recalling the definition of the discrete cosine transform of a given signal $\mathbf{a}$ as $\mathbf{A} = \mathrm{DCT}(\mathbf{a}) = \mathbf{T} \cdot \mathbf{a} \cdot \mathbf{T}^{\mathrm{T}}$, where:

$$\mathbf{T} \triangleq T(m,i) = \sqrt{\frac{2}{N}}\, \xi(m) \cos\left(\frac{m\left(i + \frac{1}{2}\right)\pi}{N}\right), \tag{5.61}$$

and $\xi(m)$ was defined in eq. 2.13, the described ME algorithm can be easily implemented directly in the compressed DCT-domain if one takes into account the orthonormal properties of the DCT:

$$\mathbf{T} = \mathbf{T}^*;\ \mathbf{T}.\mathbf{T}^{\mathrm{T}} = \mathbf{T}.\mathbf{T}^{-1} = \mathbf{I} \ \Rightarrow\ \mathbf{T}^{-1} = \mathbf{T}^{\mathrm{T}} \tag{5.62}$$

In this case, eq. 5.52 will become:

$$E(k_1,k_2) = R(k_1,k_2) - S(k_1,k_2) \simeq \frac{\partial S}{\partial v_1} dv_1^i + \frac{\partial S}{\partial v_2} dv_2^i \tag{5.63}$$

$$\Leftrightarrow \ \mathbf{E} = \mathbf{J}_s \cdot \mathbf{dv}^i \tag{5.64}$$

$$\Leftrightarrow \ \mathbf{dv}^i = \left(\mathbf{J}_s{}^{\mathrm{T}} \cdot \mathbf{J}_s\right)^{-1} \cdot \mathbf{J}_s{}^{\mathrm{T}} \cdot \mathbf{E} \tag{5.65}$$

where $\mathbf{E} = \mathrm{DCT}(\mathbf{e})$ is a $(M \times 1)$ vector obtained by re-arranging the columns of the residual matrix in the transform-domain:

$$\mathbf{E} = \begin{bmatrix} \vdots \\ R(k_1,k_2) - S(k_1,k_2) \\ \vdots \end{bmatrix}_{M\times 1} . \tag{5.66}$$

As before, $\mathbf{J}_s$ is a $(M \times 2)$ matrix whose columns contain the partial derivatives of S disposed in column form:

$$\mathbf{J}_s = \begin{bmatrix} \vdots & \vdots \\ \frac{\partial S}{\partial v_1} & \frac{\partial S}{\partial v_2} \\ \vdots & \vdots \end{bmatrix}_{M\times 2} \tag{5.67}$$

and $\mathbf{dv}$ is the desired displacement matrix:

$$\mathbf{dv}^i = \begin{bmatrix} dv_1^i \\ dv_2^i \end{bmatrix}_{2\times 1} . \tag{5.68}$$

A - Computing the image derivative in the DCT-domain

Despite the apparent simplicity of the described procedure, the practical interest of the proposed algorithm will greatly depend on the possibility and on the easiness of computing the derivatives of S from each received encoded frame.

To handle such problem, the following formulation will consider the displacement of a given image by a small motion vector $v = (v_1, v_2)$. According to the definition of the inverse DCT, the pixel-domain image data $s(y_1, y_2) = s(x_1 + v_1, x_2 + v_2)$ will be given by:

$$s(y_1,y_2) = \sum_{m_1=0}^{N-1} T(m_1,y_1) \sum_{m_2=0}^{N-1} T(m_2,y_2) \cdot S(m_1,m_2). \tag{5.69}$$

By taking into account the chain rule on the computation of the derivatives of $s\,(y_1, y_2)$, with $(y_1, y_2) = (x_1 + v_1, x_2 + v_2)$:

$$\frac{\partial s\,(y_1,y_2)}{\partial v_1} = \frac{\partial s}{\partial y_1} \cdot \frac{\partial y_1}{\partial v_1} = \frac{\partial s}{\partial y_1} \cdot 1 = \frac{\partial s\,(y_1,y_2)}{\partial y_1} \tag{5.70}$$

one obtains:

$$\frac{\partial s(y_1,y_2)}{\partial y_1} = \sum_{m_1=0}^{N-1} \frac{\partial T(m_1,y_1)}{\partial y_1} \sum_{m_2=0}^{N-1} T(m_2,y_2)\cdot S(m_1,m_2) \tag{5.71}$$

$$= \sum_{m_1=0}^{N-1} P(m_1,y_1) \sum_{m_2=0}^{N-1} T(m_2,y_2)\cdot S(m_1,m_2). \tag{5.72}$$

where

$$P(m_1,y_1) \;=\; \frac{\partial T(m_1,y_1)}{\partial y_1} \;=\; \sqrt{\frac{2}{N}}\;\xi\left(y_1\right)\left(-\frac{m_1\pi}{N}\right)\sin\left(\frac{m_1\left(y_1+\frac{1}{2}\right)\pi}{N}\right) \tag{5.73}$$

In matrix form:

$$\left[\frac{\partial s(y_1,y_2)}{\partial y_1}\right] = \mathbf{P}\cdot\mathbf{S}\cdot\mathbf{T}^{\mathrm{T}}\ . \tag{5.74}$$

In a similar way, it can be shown that

$$\left[\frac{\partial s(y_1,y_2)}{\partial y_2}\right] = \mathbf{T}\cdot\mathbf{S}\cdot\mathbf{P}^{\mathrm{T}}\ . \tag{5.75}$$

Consequently, in the transform-domain, one has:

$$\left[\frac{\partial S}{\partial v_1}\right] = \mathbf{T}\left[\frac{\partial s}{\partial v_1}\right]\mathbf{T}^{\mathrm{T}} = \mathbf{T}\ \mathbf{P}\ \mathbf{S}\ \mathbf{T}^{\mathrm{T}}\ \mathbf{T}^{\mathrm{T}} = \mathbf{D_1}\ \mathbf{S}\ \mathbf{D_2}^{\mathrm{T}} \tag{5.76}$$

$$\left[\frac{\partial S}{\partial v_2}\right] = \mathbf{T}\left[\frac{\partial s}{\partial v_2}\right]\mathbf{T}^{\mathrm{T}} = \mathbf{T}\ \mathbf{T}\ \mathbf{S}\ \mathbf{P}^{\mathrm{T}}\ \mathbf{T}^{\mathrm{T}} = \mathbf{D_2}\ \mathbf{S}\ \mathbf{D_1}^{\mathrm{T}} \tag{5.77}$$

where $\mathbf{D_1} = \mathbf{TP}$ and $\mathbf{D_2} = \mathbf{TT}$ are constant $(N \times N)$ matrices that can be pre-computed and stored in memory.

B - Algorithm

According to the formulation stated above, the DCT-domain iterative least squares motion estimation algorithm can be described by the procedure presented in fig. 5.5.

C - Scalable computational cost

As it was previously stated, the main advantage of using the DCT in video coding is concerned with its capability to concentrate most of the pixels energy in the lower frequency band of the encoded blocks. Consequently, most high frequency coefficients are often set to zero after the quantization step. As it was seen for the previously proposed spatial downscale algorithm (presented in subsection B of

Step 0: Read the macroblocks of the current ($\mathbf{R}$) and reference ($\mathbf{S}$) images from the encoded video sequence:

$$\mathbf{R} = \mathrm{DCT}\left(r\left(x_1, x_2\right)\right)$$

$$\mathbf{S} = \mathrm{DCT}\left(s\left(x_1 + v_1^0, x_2 + v_2^0\right)\right)\Big|_{v^0=\left(v_1^0, v_2^0\right)}$$

Step 1: Compute the prediction error in the DCT-domain:

$$\mathbf{E} = \left[\mathbf{R} - \mathbf{S}\right]_{M\times 1} \text{ , by considering } \mathbf{v}^i$$

Step 2: Compute the partial derivatives of $\mathbf{S}$:

$$\mathbf{J}_s = \left[\begin{array}{c:c} \mathbf{D_1}\cdot\mathbf{S}\cdot\mathbf{D_2}^{\mathrm{T}} & \mathbf{D_2}\cdot\mathbf{S}\cdot\mathbf{D_1}^{\mathrm{T}} \end{array}\right]_{M\times 2}$$

Step 3: Compute the displacement increment $\mathbf{dv}^i$:

$$\mathbf{dv}^i = \left[\begin{array}{c} dv_1^i \\ dv_2^i \end{array}\right]_{2\times 1} = \left(\mathbf{J}_s{}^{\mathrm{T}}\cdot\mathbf{J}_s\right)^{-1}\cdot\mathbf{J}_s{}^{\mathrm{T}}\cdot\mathbf{E}$$

Step 4: Update the motion vector $\mathbf{v}^i$:

$$\mathbf{v}^i = \mathbf{v}^{i-1} + \mathbf{dv}^i$$

Step 5: Evaluate the stop condition:

If $\|\mathbf{v}^i - \mathbf{v}^{i-1}\| < \delta$, stops the algorithm and sets $\mathbf{v} = \mathbf{v}^i$;

otherwise, re-compute $\mathbf{S}|_{v=v^i}$ and return to **Step 1**.

Figure 5.5: Iterative LSE algorithm for the computation of the motion vectors in the DCT-domain.

section 5.2.2 (page 174)), this fact may be highly exploited by applying a *partial DCT information* processing approach [51], to obtain significant advantages when the proposed motion estimation algorithm is performed in the DCT-domain.

As it was described in section 5.3.2, when this algorithm is applied in the pixel-domain all $(N \times N)$ pixels of the four luminance blocks that compose each macroblock should be taken into account in the iterative estimation procedure. Consequently, for most usual cases, the number of considered samples will be $M = 4 \times (8 \times 8) = 256$. Hence, most of the computational effort of the proposed algorithm is spent in the computation of the (256×2) sized partial derivatives matrix $\mathbf{j}_s$.

In contrast, when the proposed motion estimation algorithm is applied in the DCT-domain, only the non-null coefficients need to be taken into account, thus leading to a significant smaller number of considered samples M. This fact can also be used to scale the computation effort that should be spent by this algorithm. As it was suggested for the image registration algorithm proposed by Reeves [83],

a subset of the $(N \times N)$ DCT coefficients of each block can be selected using a zig-zag pattern, just keeping the information that is more relevant for the image structure. Alternatively, a low-pass pre-filtering scheme, entirely similar to the one proposed in subsection B of section 5.2.2 (page 174), may be adopted. By denoting K as the maximum bandwidth of this low-pass pre-filter, given by the highest line/column index of the considered DCT coefficients, only the low-frequency coefficients $\tilde{\mathbf{S}}_{i,j}(m,n) = \{\mathbf{S}_{i,j}(m,n) : m,n \leq K\}$ will be used for the computation of the partial derivatives matrix $\mathbf{j}_s$. By following such approach, the number of considered samples will be decreased to $M = 4K^2$, with $K < N$, thus significantly reducing the computational cost involved in each iterative step.

It is still worth noting that despite the described computational advantage, the usage of this pre-filtering scheme may actually improve the overall estimation performance. In fact, since higher frequency bands are often more prone to the presence of noise, the convergence of this method may even be improved by discarding those high frequency DCT coefficients [83]. Furthermore, to avoid the interference of eventual differences of the global luminance level of the two considered frames, the DC coefficient of the blocks under processing should also be disregarded.

5.4 Dynamic video composition in the DCT-domain[§]

As it was referred, video compositing transcoder architectures have become increasingly popular along the past few years, to combine two or more video sequences into a single scene. Among the several possible applications, these techniques have been particularly applied by surveillance systems, multi-point videoconferencing, interactive network video, preview guides and television walls, where multiple video sources are composited and combined into a single video sequence. To implement such video transcoders, the former analog video compositing systems typically required the existence of multiple tuners in the receiver device, as well as specialized compositing modules to embed the several video sequences in real-time. Such requisites usually imposed an additional cost for the terminal devices. In contrast, modern digital video processing systems can implement this kind of video manipulations in a much more efficient approach, which can significantly reduce the implementation cost of the terminal devices.

[§]Some portions of this section appeared in:

[91] - N. Roma and L. Sousa, "Fully compressed-domain transcoder for PIP/PAP video composition," in *Proceedings of the Picture Coding Symposium (PCS)*, Lisbon - Portugal, Nov. 2007, pp. CD–ROM.

Video composition typically involves several processing operations, such as overlapping, scaling, translation, filtering, etc. Such operations can be implemented either at the client side or at the server side. However, server side systems, implemented by means of specialized network transcoding nodes, tend to provide significant advantages. On the one hand, they avoid the need to transmit multiple video bitstreams to the several receivers, thus providing an extra saving of a lot of bandwidth. On the other hand, they also allow the implementation of much simpler and cost effective terminal devices, without any need for additional hardware or software requirements. In fact, with the observed increase of both the bandwidth and the computational cost levels inherent to these video transmission systems, video composition implemented at the client side tend to be hardly suitable for embedded and portable device applications, including handsets and PDAs.

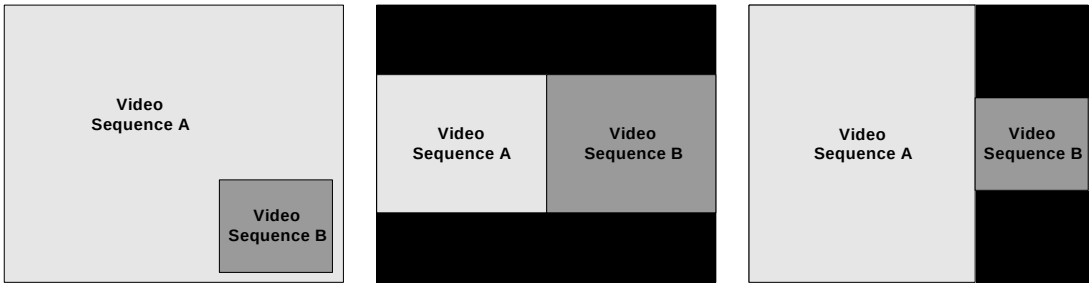
Among the several possible video compositing setups, recent commercial applications have devoted a particular interest in Picture-In-Picture (PIP), Picture-And-Picture (PAP) and Picture-Over-Picture (POP) composition schemes. These setups are illustrated in fig. 5.6 and can be described as follows:

PIP - two or more video sequences are combined into a single scene, by scaling all but one of the sequences (foreground scenes) and inserting them inside the area occupied by the background scene (see fig. 5.6(a));

PAP - two or more video sequences are combined into a single scene, at the same scale, side-by-side (see fig. 5.6(b));

POP - two or more video sequences are combined into a single scene, by scaling all but one of the sequences (foreground scenes) and inserting them in a separate region, over the background scene (see fig. 5.6(c)).

Several different transcoding structures to perform digital video composition have been presented in the past. Among such composition schemes, quite different



(a) PIP setup. **(b)** PAP setup. **(c)** POP setup.

Figure 5.6: Considered video composition setups.

processing approaches have been adopted, both in terms of the processing domain (either in the pixel-domain, in the DCT-domain or even in a hybrid Pixel-Transform-domain), in terms of the offered flexibility to insert arbitrary positioned foreground video objects, or even in terms of the capability to consider arbitrary-shaped video sequences. Furthermore, many of the presented architectures have also considered several different trade-offs and inherent compromises in terms of the bandwidth efficiency and of the involved computational cost.

Pixel-domain processing schemes are usually regarded as the most simple approaches, since they implement the required composition by first decoding all the input video sequences, which are then combined and re-encoded, in order to obtained the output video sequence. In contrast, compressed-domain approaches often tend to be more attractive, but serious obstacles exist that prevent efficient video compositing techniques to be fully implemented in the motion-compensated domain. As an example, the reference image blocks of the background image in an overlapping scene may be replaced by the foreground image. Consequently, video sequences frequently need to be inverse-motion-compensated before composition, thus requiring a subsequent motion compensation operation afterwards. This can make real-time high-speed implementations quite difficult to be achieved.

One of the first proposals of a complete video composition architecture was presented by Chang and Messerschmitt [9]. They presented a quite simple PIP compositing scheme, that processes two block-aligned video sequences, without any resizing of the foreground video sequence. To provide this structure with a motion-compensated temporal prediction mechanism, the video composition processing part was entirely implemented in the pixel-domain, which required the whole decoding of the involved video sequences. Similarly, the proposed structure also included a quite simple pixel-domain motion re-estimation mechanism, implemented solely in the areas of the background video sequence that were actually affected by the object insertion. This re-estimation only considered a quite reduced number of candidate prediction MBs and was based on a sub-optimal 2-D binary search ME algorithm, proposed by Jain and Jain [35]. Such quite simple ME algorithm was based on a monotonically variation assumption of the adopted matching measure, along the horizontal and vertical directions.

To reduce the computational cost, the same architecture was later transposed into the DCT-domain [10, 11]. By implementing the motion compensation mechanism directly in the transform-domain, this structure starts by reverting the whole temporal prediction mechanism of the encoded input sequences, in order to remove any inter-dependency between the received frames. The actual composition of the

involved video sequences is then performed by directly operating with the DCT coefficients blocks of the frames under processing. Nevertheless, this processing approach still considers the existence of a perfect alignment of the block and MB grids of the composited video sequences. By applying this scheme, though, both the DCT and the IDCT processing blocks of the traditional straightforward pixel-domain approach are removed. Moreover, considering that a significant amount of the AC frequency DCT coefficients of each encoded block are zero, the compositing unit often needs to process less data in the DCT-domain than in the pixel-domain. The temporal prediction mechanism of the processed video sequence is then recovered by applying the same DCT-domain motion compensation procedure in the encoder part of this transcoder. However, to prevent this processing block from becoming the computation-dominant in the whole compositing system, many of the new MVs are inferred and composited directly from those obtained from the original input sequences, without any further refinement. As in [9], the motion vectors are evaluated by considering three distinct regions of the composited scene:

- *unaffected background / foreground areas*: the original MVs are still valid and may be used to encode the output video sequence;
- *directly affected areas* - the MVs have to be re-evaluated, since the original predictors area was replaced by a foreground scene;
- *indirectly affected areas* - although the prediction region was not overlapped by a foreground scene, the motion compensation mechanism still has to be re-evaluated, since the corresponding prediction blocks could have been compromised through error propagation and drift.

Hence, in both the directly and indirectly affected areas, the background and foreground video sequences need to be fully decoded, so that new motion compensated prediction data can be calculated, for example, by using a pixel-domain sub-optimal ME algorithm [9]. This over-simplified evaluation of the motion vectors greatly reduces the involved computational cost, at the expense of a natural reduction of the compression performance. Furthermore, it is still worth noting that even those areas corresponding to the foreground video scenes may also have to be re-evaluated. One example of such situation occurs whenever the composition setup comprises scaling procedures of the inserted video objects.

This DCT-domain approach was further improved by Noguchi et al. [67] and applied to PIP compositing of MPEG-1 video sequences. Their technique essentially expands the previously proposed algorithms, in order to comply the decoding stage

of the transcoder with the new featured half-pixel MVs and bidirectional prediction modes, provided by the targeted MPEG-1 video standard [26]. As before, this transcoding algorithm directly infers the new MVs from those of the original MPEG-1 input video sequences, after subsequent scaling and composition, but does not perform any re-estimation nor any refinement procedure.

Meanwhile, with the advent of the most recent video standards, a further research effort has been recently devised around this issue. In particular, Li et al. [49, 50] presented two compositing schemes to be applied with the H.264/AVC video standard [31]: the Rate-Distortion Re-Encoding (RDRE) architecture (also known as Picture-In-Picture Cascaded Transcoder (PIPCT)) and the Partial Re-Encoding Transcoder (PRET) architecture. Due to the increased complexity of the H.264/AVC video standard, both approaches entirely perform the composition of the involved video sequences in the pixel-domain. However, while the RDRE scheme fully decodes the several video streams into the pixel-domain and then compresses the whole composited frame into a new bitstream, the PRET architecture presents some significant computational advantages, by only processing those MBs that were actually modified by the compositing operation, thus skipping the processing of all unaffected blocks. Independently of the considered approach, both schemes perform a pixel-domain re-estimation procedure of the MVs corresponding to the MBs of the area affected by the PIP insertion: *i)* the original MVs of the received video sequences are properly scaled and composited; and *ii)* the obtained MVs are refined using the same pixel-domain sub-optimal ME algorithm that was adopted in [9]. Nevertheless, the performance of this quite simple MV re-estimation is further compromised by the fact that such ME procedure only takes into account the candidate predictors that belong to the same video scene, thus disregarding all blocks where the composition has been applied.

Independently of the processing domain where the compositing operation is carried out, there are some research issues that still deserve further investigation, in order to improve these transcoding architectures. Naturally, one of such issues is concerned with the (re-)estimation of the MVs, in order to improve the involved temporal prediction mechanism. As it was seen, most of these architectures adopt a rather simplistic approach that merely infers the new MVs from those of the original video sequences. These estimates are then either directly used by the encoder-side motion-compensation prediction mechanism [67], or are applied to a quite simple MV re-mapping scheme, such as the one adopted by Chang and Messerschmitt [11], based on the ME algorithm proposed by Jain and Jain [35]. However, these remapped MVs may not yield the minimum residual signal, due to the simplicity

of their derivation. Furthermore, the involved ME operation usually implies the transposition of the DCT encoded MBs into the pixel-domain, in order to compute the matching measure.

Another important issue that is not usually considered by many DCT-domain compositing architectures [9, 11] concerns the manipulation of the video sequences. In order to simplify, as much as possible, the processing of the involved DCT encoded blocks and to avoid an extra transposition into the pixel-domain, many approaches imply a perfect alignment of the foreground sequences with the adopted $(N \times N)$ block grid.

To overcome such limitations, an alternative and highly efficient architecture that fully operates in the compressed DCT-domain is now considered. By directly operating with the partially decoded DCT coefficients of the involved video sequences, the proposed approach potentially provides significant advantages in what concerns the output video quality performance. Moreover, instead of only considering a couple of alternative predictors for each re-estimated MV [11], the presented approach significantly improves the temporal prediction mechanism, by adopting a more efficient DCT-domain motion re-estimation algorithm. Contrary to the refinement scheme proposed by Jain and Jain [35], this algorithm may consider any dimension for the search area, which significantly improves the output video quality, as well as the coding efficiency. Furthermore, the presented DCT-domain approach does not impose any limitation on the adopted composition setup, allowing each foreground video sequence to be placed over any location of the background scene.

5.4.1 Proposed transcoder architecture

One possible approach to implement video compositing transcoders in the compressed-domain, without fully decoding the considered video streams, is to represent and process the input sequences using a common intermediate meta-format representation. Such representation must be generated after the bitstreams parsing (VLD) and syntax decoding stages. By adopting this approach, not only does the image compositing part may be implemented in a fully independent way of the bitstream processing workload, but it is also significantly easier to manipulate video sequences that were encoded using distinct video standards (see fig. 5.7). In fact, by considering that most block-based video standards that have been proposed up to now (such as the H.261 [29], H.263 [30], MPEG-1 Video [26] and MPEG-2 Video [26]) adopt the same N-kernel discrete cosine transform (with $N = 8$), most of such video sequences may be easily represented using this common meta-format representation that include, among others, the following data structures:

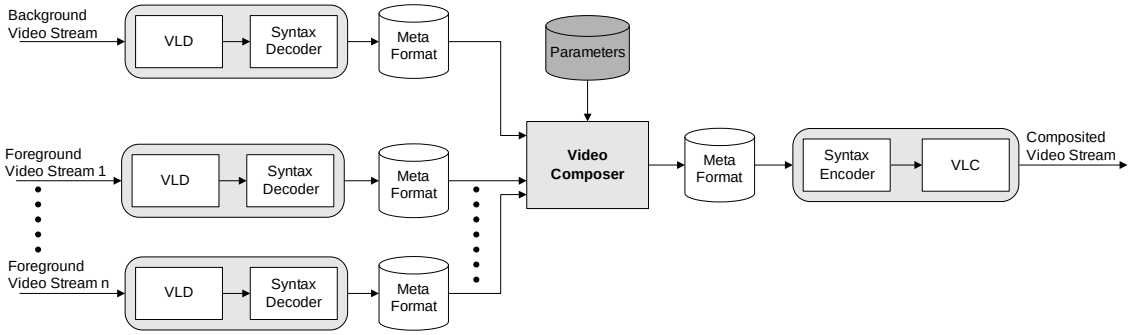


Figure 5.7: Video compositing transcoder based on an intermediary meta-format representation.

- spatial frame dimensions;
- quantization factors;
- DCT coefficients of the partially decoded blocks;
- MVs of the motion-compensated macroblocks;
- encoding modes and other parameters that are specific for each considered video standard.

Similarly, other equivalent data structures will have to be taken into account if the most recent video standards (such as H.264/AVC [31] and MPEG-4 Visual [28, 74]) are considered, where several other coding techniques may be adopted both for the spatial prediction and temporal prediction mechanisms. Some examples of such situation are the usage of INTRA frame spatial predictors and of multiple reference frames prediction modes [31].

As it would be expected, besides this common meta-format information, video compositing transcoders also require another set of parameters that are specifically concerned with the required composition setup of the involved video sequences:

- scaling factor to be applied to each video sequence;
- coordinates of each insertion;
- format of the composition setup.

With this restricted set of information, it is possible to parameterize any supported compositing setup. Furthermore, it should be noted that all these data structures will only affect the processing modules that are actually responsible for the video composition and should not influence the remaining decoding and encoding modules of the video processing system.

The proposed DCT-domain compositing transcoder is presented in fig. 5.8. This architecture naturally reflects the meta-format oriented processing structure, previously presented in fig. 5.7. As it was referred before, serious obstacles exist in

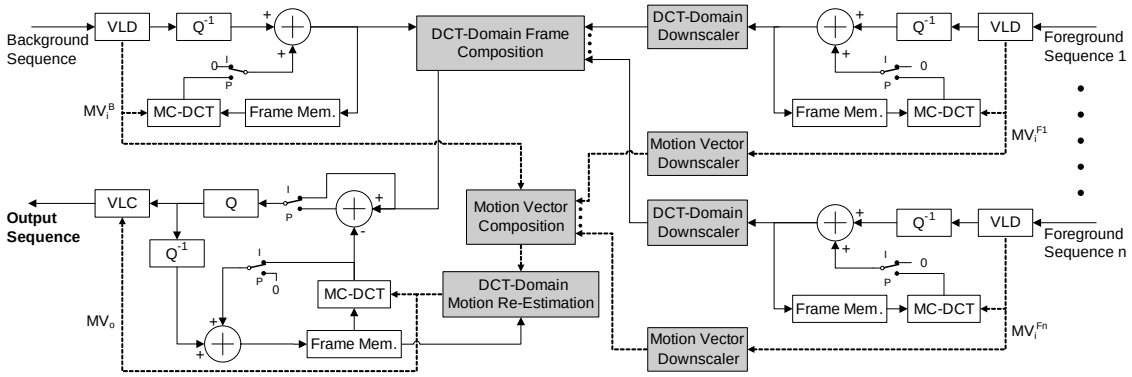


Figure 5.8: DCT-domain video compositing transcoder architecture.

the implementation of DCT-domain operations directly in the motion-compensated domain. Consequently, besides going through the bitstreams parsing (VLD) and the syntax decoding stages, all processed video sequences are inverse motion compensated before any composition operation can be conducted, thus requiring a subsequent motion compensation operation afterwards. All the processing modules that are responsible for the actual implementation of the video composition operation were represented with shaded blocks in fig. 5.8, namely the:

- DCT-domain frame downscaler;
- Motion vector downscaler;
- DCT-domain frame compositer;
- Motion vector compositer;
- DCT-domain motion re-estimator.

The implementation of each of these modules will be described in the following subsections.

5.4.2 Frame scaling

The eventually required frame scaling operation of the several foreground video sequences that are involved in the required video composition setup is implemented by means of two processing steps: *i)* reduction of the spatial resolution of the partially decoded frames, and *ii)* compositing and downscaling of the original MVs.

A - DCT-domain spatial frame scaling

To implement the spatial downscaling of the partially decoded foreground video sequences, it was adopted the averaging and sub-sampling DCT-domain algorithm for any arbitrary integer downscaling factor ($\mathcal{S}_{\mathcal{F}}$), proposed in section 5.2.

By denoting by $\mathbf{T}$ the $(N \times N)$ DCT matrix kernel ($N = 8$), so that $\mathbf{X} = \mathrm{DCT}(\mathbf{x}) = \mathbf{T} \cdot \mathbf{x} \cdot \mathbf{T}^{\mathrm{T}}$, the proposed algorithm is applied to compute each $(N \times N)$ DCT coefficients block $\hat{\mathbf{B}}$, corresponding to the set of $(\mathcal{S}_{\mathcal{F}} \times \mathcal{S}_{\mathcal{F}})$ original pixel blocks $\mathbf{b}_{i,j}$:

$$\hat{\mathbf{B}} \;=\; \frac{1}{{\mathcal{S}_{\mathcal{F}}}^2} \cdot \mathbf{T} \cdot \left(\sum_{i=0}^{\mathcal{S}_{\mathcal{F}}-1} \sum_{j=0}^{\mathcal{S}_{\mathcal{F}}-1} \mathbf{p}_{i,j} \right) \cdot \mathbf{T}^{\mathrm{T}} \tag{5.78}$$

As it was previously defined in eq. 5.14, each matrix $\mathbf{p}_{i,j}$ is computed as:

$$\mathbf{p}_{i,j}\,(l,c) = \begin{cases} \overline{\mathbf{p}_{i,j}}\,(l,c) & , \text{ for } \begin{cases} l_{min}(i) \le l \le l_{max}(i) \\ c_{min}(j) \le c \le c_{max}(j) \end{cases} \\ 0 & , \text{ otherwise} \end{cases} \tag{5.79}$$

where:

$$l_{min}(i) = \left\lfloor \frac{i * N}{\mathcal{S}_{\mathcal{F}}} \right\rfloor \quad \text{and} \quad l_{max}(i) = \left\lfloor \frac{i * N + (N-1)}{\mathcal{S}_{\mathcal{F}}} \right\rfloor \tag{5.80}$$

$$c_{min}(j) = \left\lfloor \frac{j * N}{\mathcal{S}_{\mathcal{F}}} \right\rfloor \quad \text{and} \quad c_{max}(j) = \left\lfloor \frac{j * N + (N-1)}{\mathcal{S}_{\mathcal{F}}} \right\rfloor \tag{5.81}$$

with $i, j = 0 \ldots (\mathcal{S}_{\mathcal{F}} - 1)$. The computation of the non-null elements ($\overline{\mathbf{p}_{i,j}}$) can be implemented as follows:

$$\overline{\mathbf{p}_{i,j}} = \underbrace{\overline{\mathbf{F}^{i}_{\mathcal{S}_{\mathcal{F}}}} \cdot \mathbf{B}_{i,j} \cdot \overline{\mathbf{F}^{j}_{\mathcal{S}_{\mathcal{F}}}}^{\mathrm{T}}}_{n_l(i) \times n_c(j) \text{ matrix}} \tag{5.82}$$

with $n_l(i) = l_{max}(i) - l_{min}(i) + 1$ and $n_c(j) = c_{max}(j) - c_{min}(j) + 1$. In this equation, $\mathbf{B}_{i,j}$ is the $(N \times N)$ DCT coefficients block, directly obtained from the original bit stream. The $(n(x) \times N)$ $\overline{\mathbf{F}^{x}_{\mathcal{S}_{\mathcal{F}}}}$ terms (with $0 \le x \le \mathcal{S}_{\mathcal{F}} - 1$) are constant matrices, that can be pre-computed and stored in memory (see subsection A of section 5.2.2 (page 174)).

By adopting this algorithm, not only will the proposed architecture avoid the influence of the degradation effects allied to blocky artifacts that are introduced whenever the adopted scaling factor is not an integer power of 2 (see section 5.2.1), but it also benefits from the significant computational advantages that it provides (see section 5.2.3).

B - Motion vector downscaling

One of the strategies that was adopted in the proposed video composition architecture to minimize the involved computational cost is to restrict the search area considered in the motion re-estimation procedure at the encoder-side, by computing a prior prediction for each MV. Such prediction is calculated by properly reusing

the set of MVs obtained from the partially decoded foreground and background video sequences. However, considering that some compositing setups involve a reduction of the spatial resolution, the received MVs must also be properly adapted by a scaling and a compositing step.

As it was previously described in section 3.3.4, several possible MV compositing methods have been proposed, to directly predict the new MVs from the set of vectors that are obtained from the precoded video. Among the considered approaches, the compositing methods that rely on the spatial activity of each compositing MB in the interpolation of the new motion vector, as described in subsection C of section 3.3.4 (page 102), have proved to provide specially adequate results. In particular, the proposed video composition architecture adopts a variation of the Maximum Quantization step-size times number of Bits-Area (MQBA) technique [108], which relies on each MB compositing area and on its corresponding spatial activity to weight the influence of each MV. Nevertheless, to avoid the inherent cost of computing the inverse DCT, the spatial activity was directly estimated from the DCT coefficients of each compositing block, by counting the number of non-null AC frequency DCT coefficients, as proposed by Liang et al. [52]. Such spatial activity estimate is then used in the computation of the composited motion vector ($\tilde{v}$), as follows:

$$\tilde{v} = \frac{\sum_i \sum_j v'_{i,j} \cdot \alpha_{i,j} \cdot A_{i,j}}{\sum_i \sum_j \alpha_{i,j} \cdot A_{i,j}} \tag{5.83}$$

In eq. 5.83, $v'_{i,j}$ denotes the scaled MV corresponding to the original video sequence; $\alpha_{i,j}$ is the spatial activity of macroblock (i, j), measured by the number of non-null AC frequency DCT coefficients; and $A_{i,j}$ is a measure of the involved compositing area, evaluated in terms of the number of pixels of the original macroblock (i, j) that is considered in the composition of the transcoded macroblock.

Moreover, to further improve the performance and accuracy levels of the temporal prediction mechanism, both the scaling and compositing procedures make use of a half-pixel precision level.

5.4.3 DCT-domain frame composition

The DCT-domain composition with the background scene of one or more foreground video sequences, at any arbitrary position, often leads to mismatches of the corresponding block grids [11], as it is illustrated in fig. 5.9. In such a situation, each $(N \times N)$ DCT coefficients block of the involved foreground scenes ($\mathbf{Y}_i$) has to be properly segmented and translated with respect to the block structure of the target background scene ($\mathbf{X}$). Similarly, to fulfill the required opaque superposition, the

superimposed area of the background video sequence will also have to be segmented and removed, according to the adopted composition setup. After the removal of all these superimposed regions of the background scene, each composited output block $\mathbf{B}$ is formed by summing up the contributions from all the involved foreground video sequences:

$$\mathbf{B} = \mathbf{X} - \sum_i \mathbf{X}_{\text{seg}_i} + \sum_i \mathbf{Y}_{\text{seg}_i} \tag{5.84}$$

The mathematical model to obtain the DCT coefficients of a given extracted and translated sub-block was already formulated in [11] and is quite similar to the formulation previously described in section 3.3.1, concerning the implementation of the motion compensation module in the compressed-domain. Such operation can be carried out by multiplying each of the involved DCT coefficients blocks by the appropriate filter matrices ($\mathbf{H}_1$ and $\mathbf{H}_2$), that simultaneously perform the segmentation and the translation operations of the required regions [11, 62]:

$$\mathbf{X}_{\text{seg}} = \mathbf{H}_1^x \cdot \mathbf{X} \cdot \mathbf{H}_2^x \tag{5.85}$$

$$\mathbf{Y}_{\text{seg}} = \mathbf{H}_1^y \cdot \mathbf{Y} \cdot \mathbf{H}_2^y \tag{5.86}$$

In these equations, $\mathbf{H}_k = \text{DCT}(\mathbf{h}_k)$, with $k \in \{1, 2\}$, are pre-computed and stored matrices that are required for the several different block composition setups, as defined in table 5.3. $\mathbf{I}_h$ and $\mathbf{I}_w$ are $(h \times h)$ and $(w \times w)$ identity matrices, respectively, where h and w denote the number of rows and columns to be extracted. Similarly, $\mathbf{I}_N$ represents the $(N \times N)$ identity matrix.

Hence, for the particular example shown in fig. 5.9, the required segmentation and translation matrices are the follow:

$$\mathbf{H}_1^x = \text{DCT}(\mathbf{h}_1^x) \text{ , with } \mathbf{h}_1^x = \begin{bmatrix} 0 & 0 \\ 0 & \mathbf{I}_h \end{bmatrix} \tag{5.87}$$

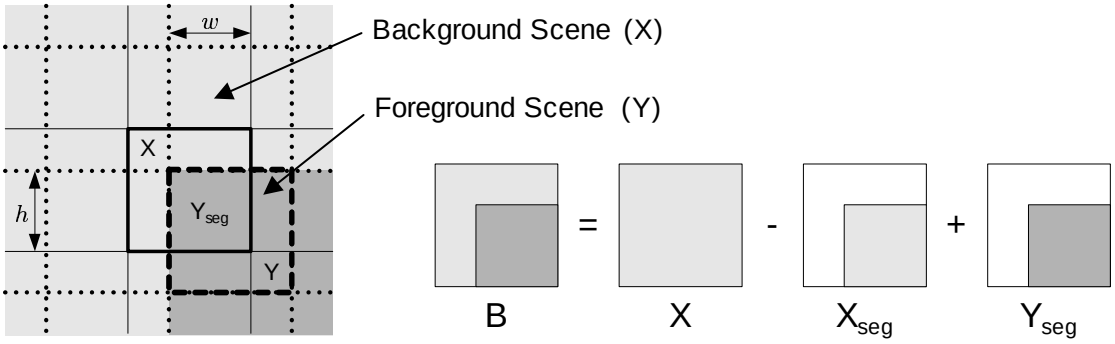
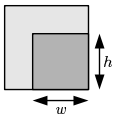
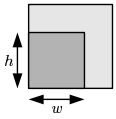
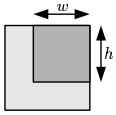
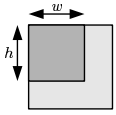
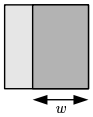
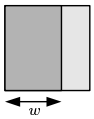
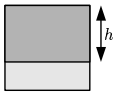
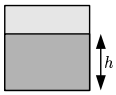
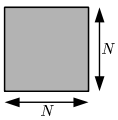


Figure 5.9: Block segmentation and translation for DCT-domain video compositing.

Table 5.3: Filtering matrices used by the translation and segmentation operations for dynamic video composition.

Composition Setup	$\mathbf{h}_1^x$	$\mathbf{h}_2^x$	$\mathbf{h}_1^y$	$\mathbf{h}_2^y$
	$\begin{bmatrix} 0 & 0 \\ 0 & \mathbf{I}_h \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & \mathbf{I}_w \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ \mathbf{I}_h & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & \mathbf{I}_w \\ 0 & 0 \end{bmatrix}$
	$\begin{bmatrix} 0 & 0 \\ 0 & \mathbf{I}_h \end{bmatrix}$	$\begin{bmatrix} \mathbf{I}_w & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ \mathbf{I}_h & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ \mathbf{I}_w & 0 \end{bmatrix}$
	$\begin{bmatrix} \mathbf{I}_h & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & \mathbf{I}_w \end{bmatrix}$	$\begin{bmatrix} 0 & \mathbf{I}_h \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & \mathbf{I}_w \\ 0 & 0 \end{bmatrix}$
	$\begin{bmatrix} \mathbf{I}_h & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} \mathbf{I}_w & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & \mathbf{I}_h \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ \mathbf{I}_w & 0 \end{bmatrix}$
	$\mathbf{I}_N$	$\begin{bmatrix} 0 & 0 \\ 0 & \mathbf{I}_w \end{bmatrix}$	$\mathbf{I}_N$	$\begin{bmatrix} 0 & \mathbf{I}_w \\ 0 & 0 \end{bmatrix}$
	$\mathbf{I}_N$	$\begin{bmatrix} \mathbf{I}_w & 0 \\ 0 & 0 \end{bmatrix}$	$\mathbf{I}_N$	$\begin{bmatrix} 0 & 0 \\ \mathbf{I}_w & 0 \end{bmatrix}$
	$\begin{bmatrix} \mathbf{I}_h & 0 \\ 0 & 0 \end{bmatrix}$	$\mathbf{I}_N$	$\begin{bmatrix} 0 & \mathbf{I}_h \\ 0 & 0 \end{bmatrix}$	$\mathbf{I}_N$
	$\begin{bmatrix} 0 & 0 \\ 0 & \mathbf{I}_h \end{bmatrix}$	$\mathbf{I}_N$	$\begin{bmatrix} 0 & 0 \\ \mathbf{I}_h & 0 \end{bmatrix}$	$\mathbf{I}_N$
	$\mathbf{B} = \mathbf{Y}_{\text{seg}_i} = \text{MC-DCT}(\mathbf{Y})\big\|_{v=(N-h,N-w)}$			

$$\mathbf{H}_2^x = \mathrm{DCT}(\mathbf{h}_2^x) \text{ , with } \mathbf{h}_2^x = \begin{bmatrix} 0 & 0 \\ 0 & \mathbf{I}_w \end{bmatrix} \tag{5.88}$$

$$\mathbf{H}_1^y = \mathrm{DCT}(\mathbf{h}_1^y) \text{ , with } \mathbf{h}_1^y = \begin{bmatrix} 0 & 0 \\ \mathbf{I}_h & 0 \end{bmatrix} \tag{5.89}$$

$$\mathbf{H}_2^y = \mathrm{DCT}(\mathbf{h}_2^y) \text{ , with } \mathbf{h}_2^y = \begin{bmatrix} 0 & \mathbf{I}_w \\ 0 & 0 \end{bmatrix} \tag{5.90}$$

The particular situation corresponding to the case where the output composited block ($\mathbf{B}$) is entirely formed by the pixels of the foreground scene can be implemented in a way wholly similar to the compressed-domain motion compensation operation, described in subsection B of section 3.3.1 (page 69). In fact, such case is entirely equivalent to the usage of a motion vector given by $v = (N - h, N - w)$, where h and w denote the horizontal and vertical displacement, respectively, as defined in fig. 5.9.

5.4.4 Motion vector re-estimation

As it was previously referred, Chang and Messerschmitt [11] have distinguished three different areas of the composited scene: the *unaffected area*; the *directly affected area*, corresponding to all MBs where the composition is performed; and the *indirectly affected area*, whose prediction data may still need to be recalculated, since its prediction blocks could have been modified through error propagation (drift).

Hence, contrary to the previously presented approaches that have been proposed by other authors [9–11, 49, 50, 67], the compressed-domain video compositing architecture herein proposed introduces an efficient DCT-domain motion re-estimation module. With such a module, it is possible to obtain an effective enhancement of the temporal prediction mechanism in the output composited video sequence, with a consequent saving of the required bandwidth that can potentiate an indirect improvement of the output video quality.

Nevertheless, to minimize the involved computational cost, the MVs obtained from the scaler and compositing stage, described in subsection B of section 5.4.1 (page 197), are applied to a second compositing module, in order to combine the set of MVs corresponding to the several video sequences that affect the MB under processing. This additional MV prediction stage is then followed by the actual motion re-estimation module, as it will be described in the following subsections.

A - Motion vector prediction

As it was referred, a first estimate of the MVs that will be used by the temporal prediction mechanism of the output video sequence can be directly obtained from the set of MVs of the original background (v_B) and foreground (v_F) video sequences. However, the composition of the foreground sequences with the background scene at any arbitrary position may naturally lead to mismatches of the corresponding MB grids, as it was already explained. As a consequence, the MB grid of the composited sequence often does not align with the original MB grid of the precoded frames. Hence, depending on the actual position of the foreground sequences, the temporal prediction data, corresponding to the precoded MBs, may be only partially used to composite each MBs of the output sequence.

In such cases, it is necessary to define a selection criteria to choose, among the decoded MVs of the foreground and background video sequences, the particular MV that will be used to compute the prediction error of a given region. In an entirely similar way, the search range that will be adopted by the subsequent refinement procedure may be selected according to the confidence level of the corresponding prediction MV: smaller search ranges will be preferred whenever most pixels of the composited MB belong to the same decoded video sequence as the selected prediction MB.

The proposed video composition transcoder adopted the procedure described in fig. 5.10 to obtain the initial MV predictions ($\hat{v}$) corresponding to the affected areas, as well as a preliminary measure of the search area that will be considered in the MV re-estimation module (p_{mre}). Once again, to restrict the computational cost of the

$\hat{v} = (0,0)$; $p_{mre} = p_{max}$	$\rightarrow$	if the current MB and its prediction belong to different compositing sequences;
$\hat{v} = v_B$; $p_{mre} = p_{max}/2$	$\rightarrow$	if at least 75% of the current MB area belongs to the background sequence;
$\hat{v} = v_F$; $p_{mre} = p_{max}/2$	$\rightarrow$	if at least 75% of the current MB area belongs to the foreground sequence;
$\hat{v} = (0,0)$; $p_{mre} = p_{max}$	$\rightarrow$	*otherwise*.

Figure 5.10: Computation of the MV prediction and of the search area preliminary measure that will be considered in the MV re-estimation module.

subsequent motion re-estimation algorithm, the maximum considered displacement of the search procedure is limited to a reasonably short range (e.g.: $p_{max} = 4$ pixels).

B - DCT-domain motion re-estimation

The set of MVs that were obtained in the previously described prediction module ($\hat{v}$) are then used as coarse estimates of the MVs of the composited frame. In order to obtain more accurate refinements of these MVs, these estimates, as well as the DCT coefficients blocks of the current and previously composited frames, are applied to a motion re-estimation module. To implement such computational block, the DCT-domain iterative Least Squares Motion Estimation (LSME) algorithm [89] that was proposed in section 5.3 was adopted.

Considering that the DCT concentrates most of the pixels energy in the lower frequency coefficients of the encoded blocks, the overall computational cost required by this algorithm may be adjusted to the current restrictions of the processing system by only considering the $(K \times K)$ lower frequency coefficients of each block under processing. The reformulation of the compressed-domain motion re-estimation algorithm presented in fig. 5.5 to take into account this computational simplification is the following:

Step 0: Fetch the MBs corresponding to the current ($\mathbf{R}$) and reference ($\mathbf{S}$) frames of the composited video sequence, by performing a DCT-domain motion compensation operation using the initial predictor MV: $\mathbf{v}^0 = \hat{\mathbf{v}}$

Step 1: Compute the prediction error in the DCT-domain:
$\mathbf{E} = [\mathbf{R} - \mathbf{S}]_{4K \times 1}$, by considering $\mathbf{v}^i$

Step 2: Compute the partial derivatives of $\mathbf{S}$:
$$\mathbf{J}_s = \left[\begin{array}{c|c} \mathbf{D_1} \cdot \mathbf{S} \cdot \mathbf{D_2}^{\mathrm{T}} & \mathbf{D_2} \cdot \mathbf{S} \cdot \mathbf{D_1}^{\mathrm{T}} \end{array} \right]_{4K \times 2}$$

Step 3: Compute the displacement increment $\mathbf{dv}^i$:
$$\mathbf{dv}^i = \left[\begin{array}{c} dv_1^i \\ dv_2^i \end{array} \right]_{2 \times 1} = \left(\mathbf{J}_s{}^{\mathrm{T}} \cdot \mathbf{J}_s \right)^{-1} \cdot \mathbf{J}_s{}^{\mathrm{T}} \cdot \mathbf{E}$$

Step 4: Update the motion vector $\mathbf{v}^i$:
$$\mathbf{v}^i = \mathbf{v}^{i-1} + \mathbf{dv}^i$$

Step 5: Evaluate the stop condition:
If $\|\mathbf{v}^i - \mathbf{v}^{i-1}\| < \delta$ or $\|\mathbf{v}^i - \hat{\mathbf{v}}\| > p_{mre}$,
– stop the algorithm and set $\mathbf{v} = \mathbf{v}^i$;
otherwise,
– re-compute $\mathbf{S}|_{v=v^i}$ and return to **Step 1**.

All the considered matrices are processed in vectorized form. Moreover, as it was explained in section 5.3, $\mathbf{D_1}$ and $\mathbf{D_2}$ are constant $(K \times K)$ matrices that are pre-computed and stored in memory. As it was also referred, to avoid the interference of any eventual difference of the luminance level of the two considered frames, the DC coefficients of the processed blocks should also be disregarded.

5.5 Conclusions

A set of highly efficient operations to process and manipulate precoded video sequences in the compressed DCT-domain was proposed in this chapter. Contrary to the previously presented static video compositing structures, the algorithms and architectures that were proposed in this chapter actually manipulate the data structures affected to the several encoding strategies that are applied to each of the considered video streams.

As a result, an innovative and efficient hybrid transcoding algorithm for video downscaling in the transform-domain, by any arbitrary integer scaling factor, was proposed in section 5.2. This algorithm offers a considerable advantage in what concerns the computational cost, by benefiting from the used scaling mechanism and by only performing the operations that are really needed to compute the desired output values.

In section 5.3 it was proposed a new compressed-domain ME algorithm. This algorithm is based on an iterative scheme that estimates the new MVs by applying a Least Squares Estimation technique. This algorithm makes use of the DCT coefficients directly obtained from the input video streams and provides an efficient alternative to refine, or even re-compute, the set of MVs that are required to implement the motion compensated prediction mechanism of the output video sequence.

These two important algorithms were then applied in the implementation of a compressed-domain video compositing architecture, presented in section 5.4. While the proposed scaling algorithm significantly contributes to provide this compositing architecture with significant video quality performances, the new compressed-domain ME algorithm actively contributes to an enhancement of the coding efficiency, by improving the temporal prediction mechanism of the processed video sequences.

All these manipulations and algorithms were properly tailored, not only to optimize the computational effort but also to simultaneously comply the involved operations with the video standards block structure.

References

[9] S.-F. Chang and D. G. Messerschmitt, "Compositing motion-compensated video within the network," in *Proceedings of the International Workshop on Multimedia Communications (MULTIMEDIA)*. IEEE, Apr. 1992, pp. 40–56.

[10] S.-F. Chang and D. G. Messerschmitt, "A new approach to decoding and compositing motion-compensated DCT-based images," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 5. IEEE, apr 1993, pp. 421–424.

[11] S.-F. Chang and D. G. Messerschmitt, "Manipulation and compositing of MC-DCT compressed video," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 1, pp. 1–11, Jan. 1995.

[12] J. Chen, U.-V. Koc, and K. J. R. Liu, *Design of Digital Video Coding Systems - A Complete Compressed Domain Approach*. Marcel Dekker, 2002.

[14] W.-H. Chen, C. H. Smith, and S. C. Fralick, "A fast computational algorithm for the discrete cosine transform," *IEEE Transactions on Communications*, vol. COM-25, pp. 1004–1009, Sep. 1977.

[16] R. Dugad and N. Ahuja, "A fast scheme for image size change in the compressed domain," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 4, pp. 461–474, Apr. 2001.

[26] *MPEG-1: ISO/IEC JTC1 CD 11172 - "Coding of moving pictures and associated audio for digital storage media up to 1.5 Mbit/s – Part 2: Video"*, ISO, 1992.

[28] *MPEG-4: ISO/IEC 14496-2:2004. Information technology – Coding of audio-visual objects – Part 2: Visual*, ISO, 2004.

[29] *ITU-T Recommendation H.261 - "Video Codec for Audiovisual Services at* $p\times64$ *Kbit/s"*, ITU-T, Mar. 1993.

[30] *ITU-T Recommendation H.263 - "Video Coding for Low Bitrate Communication"*, ITU-T, Feb. 1998.

[31] *ITU-T Recommendation H.264, "Advanced Video Coding for Generic Audiovisual Services"*, ITU-T, May 2003.

[35] J. R. Jain and A. K. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Transactions on Communications*, vol. COM-29, no. 12, pp. 1799–1808, Dec. 1981.

[41] U.-V. Koc and K. J. R. Liu, "DCT-based motion estimation," *IEEE Transactions on Image Processing*, vol. 7, no. 7, pp. 948–965, Jul. 1998.

[48] Y.-R. Lee, C.-W. Lin, S.-H. Yeh, and Y.-C. Chen, "Low-complexity DCT-domain video transcoders for arbitrary-size downscaling," in *Proceedings of the IEEE International Workshop on Multimedia Signal Processing (MMSP)*, Sep. 2004, pp. 31–34.

[49] C.-H. Li, H. Lin, C.-N. Wang, and T. Chiang, "A fast H.264-based picture-in-picture (PIP) transcoder," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, vol. 3. IEEE, Jun. 2004, pp. 1691–1694.

[50] C.-H. Li, C.-N. Wang, and T. Chiang, "A low complexity picture-in-picture transcoder for video-on-demand," in *IEEE International Conference on Wireless Networks, Communications and Mobile Computing*, vol. 2, Jun. 2005, pp. 1382–1387.

[51] H. Li and H. Shi, "A fast algorithm for reconstructing motion compensated blocks in compressed domain," *Journal of Visual Languages and Computing*, vol. 10, no. 6, pp. 607–623, Dec. 1999.

[52] Y. Liang, L.-P. Chau, and Y.-P. Tan, "Arbitrary downsizing video transcoding using fast motion vector re-estimation," *IEEE Signal Processing Letters*, vol. 9, no. 11, pp. 352–355, Nov. 2002.

[54] C.-W. Lin and Y.-R. Lee, "Fast algorithms for DCT-domain video transcoding," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, Thessaloniki - Greece, Oct. 2001, pp. 421–424.

[57] S. Liu and A. C. Bovik, "Local bandwidth constrained fast inverse motion compensation for DCT-domain video transcoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 5, pp. 309–319, May 2002.

[62] N. Merhav and V. Bhaskaran, "A fast algorithm for DCT-domain inverse motion compensation," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 4, Atlanta, GA, USA, May 1996, pp. 2307–2310.

[67] Y. Noguchi, D. G. Messerschmitt, and S.-F. Chang, "MPEG video compositing in the compressed domain," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 2. IEEE, May 1996, pp. 596–599.

[72] Y. S. Park and H. W. Park, "Arbitrary-ratio image resizing using fast DCT of composite length for DCT-based transcoder," *IEEE Transactions on Image Processing*, vol. 15, no. 2, pp. 494–500, Feb. 2006.

[73] V. Patil, R. Kumar, and J. Mukherjee, "A fast arbitrary factor video resizing algorithm," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 9, pp. 1164–1171, Sep. 2006.

[74] F. Pereira and T. Ebrahimi, Eds., *The MPEG-4 Book*. Prentice Hall PTR, 2002.

[83] R. Reeves, "Image matching in the compressed domain," Ph.D. dissertation, Queensland University of Technology, Australia, 1999.

[88] N. Roma and L. Sousa, "Fast transcoding architectures for insertion of non-regular shaped objects in the compressed DCT-domain," *Signal Processing: Image Communication*, vol. 18, no. 8, pp. 659–683, Sep. 2003.

[89] N. Roma and L. Sousa, "Least squares motion estimation algorithm in the compressed DCT domain for H.26x/MPEG-x video sequences," in *Proceedings of the IEEE International Conference on Advanced Video and Signal-Based Surveillance (AVSS)*. Como - Italy: IEEE, Sep. 2005, pp. 576–581.

[90] N. Roma and L. Sousa, "Efficient hybrid DCT-domain algorithm for any arbitrary integer re-size video downscaling," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, no. 57291, pp. 1–16, Sep. 2007.

[91] N. Roma and L. Sousa, "Fully compressed-domain transcoder for PIP/PAP video composition," in *Proceedings of the Picture Coding Symposium (PCS)*, Lisbon - Portugal, Nov. 2007, pp. CD–ROM.

[92] C. L. Salazar and T. D. Tran, "On resizing images in the DCT domain," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, vol. 4, Oct. 2004, pp. 2797–2800.

[95] T. Shanableh and M. Ghanbari, "Heterogeneous video transcoding to lower spatio-temporal resolution and different encoding formats," *IEEE Transactions on Multimedia*, vol. 2, no. 2, pp. 101–110, Jun. 2000.

[98] T. Shanableh and M. Ghanbari, "Hybrid DCT/pixel domain architecture for heterogeneous video transcoding," *Signal Processing: Image Communication*, vol. 18, no. 8, pp. 601–620, Sep. 2003.

[102] H. Shu and L.-P. Chau, "An efficient arbitrary downsizing algorithm for video transcoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 6, pp. 887–891, Jun. 2004.

[103] H. Shu and L.-P. Chau, "A resizing algorithm with two-stage realization for DCT-based transcoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 2, pp. 248–253, Feb. 2007.

[108] Y.-P. Tan, Y. Liang, and H. Sun, "On the methods and performances of rational downsizing video transcoding," *Signal Processing: Image Communication*, vol. 19, pp. 47–65, 2004.

[115] J. Youn, M.-T. Sun, and C.-W. Lin, "Motion vector refinement for high performance transcoding," *IEEE Transactions on Multimedia*, vol. 1, no. 1, pp. 30–40, Mar. 1999.

6

Experimental Results

Contents

6.1 Introduction

This chapter presents the results that were obtained from the set of experimental procedures that were conducted in order to properly assess and evaluate the static and dynamic video processing algorithms, proposed in the scope of this thesis.

The evaluation of the transcoding architectures proposed in chapter 4, providing the insertion of Non-Regular Shaped Objects (NRSOs) in the compressed DCT-domain, is addressed in section 6.2. Section 6.3 presents the experimental results corresponding to the set of dynamic video compositing techniques, proposed in chapter 5. In particular, in section 6.3.1 it is presented the evaluation of the space scaling algorithm by an arbitrary integer scale factor, described in section 5.2; section 6.3.2 addresses the assessment of the block-based motion re-estimation algorithm in the compressed DCT-domain, described in section 5.3; and section 6.3.3 presents the evaluation of the DCT-domain dynamic video composition architecture, proposed in section 5.4.

The performed experiments were based on a collection of benchmark standard test video sequences of classes A, B and C of the MPEG-4 Video Verification Model [65]. Such set includes test video sequences with a frame rate of 30 Frames per Second (FPS), in both the CIF (352×288 pixels) and the QCIF (176×144 pixels) formats, characterized by different spatial detail and amount of movement:

- *Akiyo* (Class A) – characterized by reduced spatial detail and reduced amount of movement (see fig. 6.1(a));
- *Silent-Voice* (Class B) – characterized by medium spatial detail and medium amount of movement (see fig. 6.1(b));
- *Coastguard* (Class B) – characterized by medium spatial detail and medium amount of movement (see fig. 6.1(c));
- *Carphone* (Class C) – characterized by moderate spatial detail and amount of movement, consisting of local displacements of the head and lips of the person in the scene, and regular translational movements of the background (car window) (see fig. 6.1(d));
- *Table-Tennis* (Class C) – characterized by moderate spatial detail and by a significant amount of movement, both of translational type (ball) and zoom-out type (video camera) (see fig. 6.1(e));
- *Mobile & Calendar* (Class C) – characterized by large spatial detail and a considerable amount of movement (see fig. 6.1(f)).

All the conducted experiments were carried out by considering a wide range of quantization steps, in order to assess the influence of the quantization distortion

(a) *Akiyo.*

(b) *Silent-Voice.*

(c) *Coastguard.*

(d) *Carphone.*

(e) *Table-Tennis.*

(f) *Mobile & Calendar.*

Figure 6.1: Considered test video sequences.

error in the performance of the transcoding algorithms under evaluation. For each considered quantization setup (Q_k), the obtained gains, both in terms of the output video quality (PSNR) and bit rate, were properly registered. Furthermore, to emphasize the direct influence of the algorithms under evaluation on the output bit rate, the output buffer controller of the encoding system was disabled in many of the experiments that were carried out. This procedure provides a direct assessment of each algorithm under evaluation, since the obtained differences on the amount of bits required to encode each frame can be directly assigned to the performance of the

probed algorithm. Furthermore, all the experimental procedures also included an estimation of the computational cost of each algorithm. Such evaluation is based on the number of times that the most predominant arithmetic operation was performed during the execution of the algorithm.

6.2 Static video composition[§]

The performance of the proposed NRSO insertion transcoding architectures was thoroughly evaluated by undergoing several experiments, both in the pixel-domain and in the compressed DCT-domain, and by considering four standard QCIF video sequences of classes A, B and C of the MPEG-4 Video Verification Model [65]: *Akiyo*, *Silent-Voice*, *Carphone* and *Table-Tennis*. The first 130 frames of each video sequence were encoded at 30 FPS with a reference H.263 [30] video transcoding system, using a GOP length (G) of 15 frames and by considering two different quantization setups: $Q = 4$ (step size=8) and $Q = 15$ (step size=30).

(a) Logo. **(b)** Subtitle.

Figure 6.2: Considered set of NRSOs ($C_T = 0$).

The logo and subtitle illustrated in figs. 6.2(a) and 6.2(b) were simultaneously inserted in each of these video sequences at positions $(5, 5)$ and $(120, 26)$, respectively, using a transparency factor $\alpha = 0.5$. With this setup, 68 out of the 396 blocks of $(N \times N)$ pixels of each QCIF frame require some processing, corresponding to a processing rate of 17.2% of the total amount of blocks and of 27.3% of the total amount of macroblocks, respectively.

The following characteristics of the considered transcoding architectures were evaluated:

[§]Some portions of this section appeared in:

[86] - N. Roma and L. Sousa, "Insertion of irregular-shaped logos in the compressed DCT domain," in *Proceedings of the IEEE International Conference on Digital Signal Processing (DSP)*, vol. 1. Santorini, Greece: IEEE, Jul. 2002, pp. 125–128.

[87] - N. Roma and L. Sousa, "Transcoding architectures for object insertion in compressed video," INESC-ID – Lisboa, Portugal, Tech. Rep. RT/006/2002, Oct. 2002.

[88] - N. Roma and L. Sousa, "Fast transcoding architectures for insertion of non-regular shaped objects in the compressed DCT-domain," *Signal Processing: Image Communication*, vol. 18, no. 8, pp. 659–683, Sep. 2003.

- the quality of the video sequences after the insertion of the NRSOs, assessed by using the widely adopted PSNR measure and an optimal pixel-domain compositing scheme, to obtain the reference (maximum quality) video sequence;
- the bit rate of the resulting video sequences after the insertion of the NRSOs;
- the overall efficiency of each transcoder and the computational load that is involved in the insertion of the NRSOs in each video sequence, namely, the number of additions and multiplications;
- the distortion effect that is introduced by the several transcoders, with a special emphasis on the drift introduced in INTER type images by the *open-loop compressed DCT-domain transcoder*.

To ease the representation in the several charts presented in the following subsections, it was adopted the following nomenclature to identify the considered transcoding architectures:

`PDIT-MV` – Pixel-domain insertion transcoder with re-estimation of the motion vectors;

`PDIT-nMV` – Pixel-domain insertion transcoder without re-estimation of the motion vectors;

`TDIT-CL` – Closed-loop compressed DCT-domain insertion transcoder;

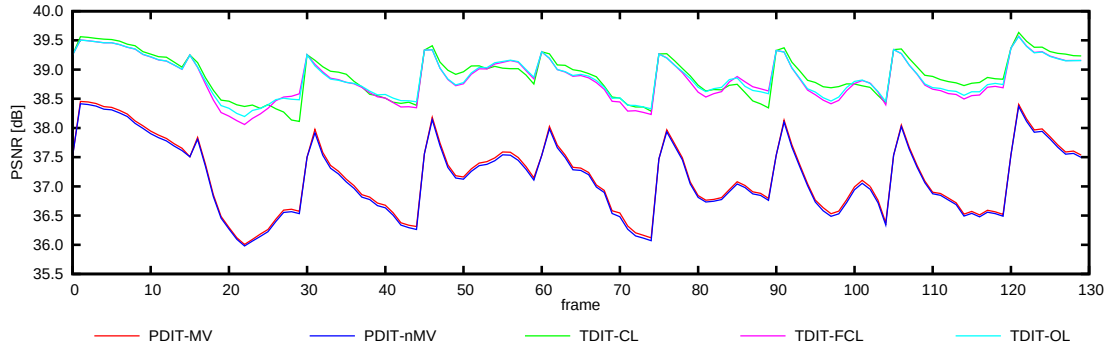
`TDIT-FCL` – Fast computational-reduced closed-loop compressed DCT-domain insertion transcoder;

`TDIT-OL` – Open-loop compressed DCT-domain insertion transcoder.

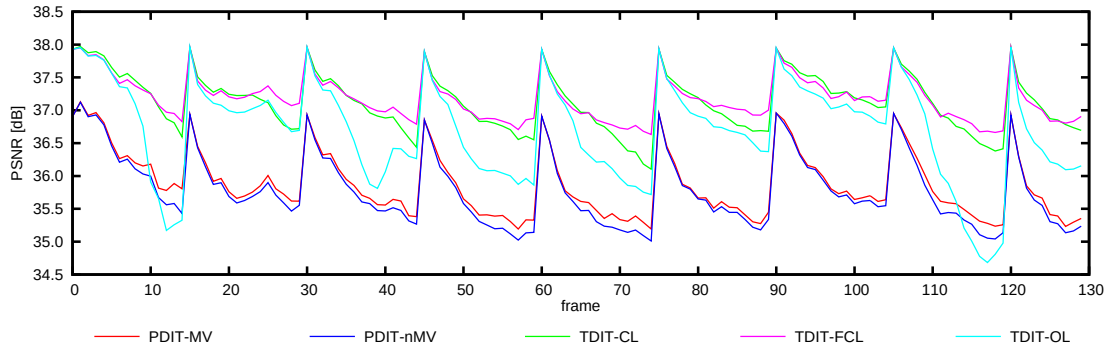
6.2.1 Quality of the encoded video sequences

In fig. 6.3 it is illustrated the variation of the PSNR measure for the four considered video sequences, using a quantizer setup with $Q = 4$. As it can be observed from these charts, the overall quality of the video sequences that were processed by the compressed DCT-domain transcoders is somewhat higher than the quality of the sequences that were obtained using the pixel-domain transcoders. This fact complies with the observations presented in section 3.1.2 for high quality video coding (using small quantization steps), and can be justified by the higher influence of the precision errors that are introduced in pixel-domain transcoding architectures.

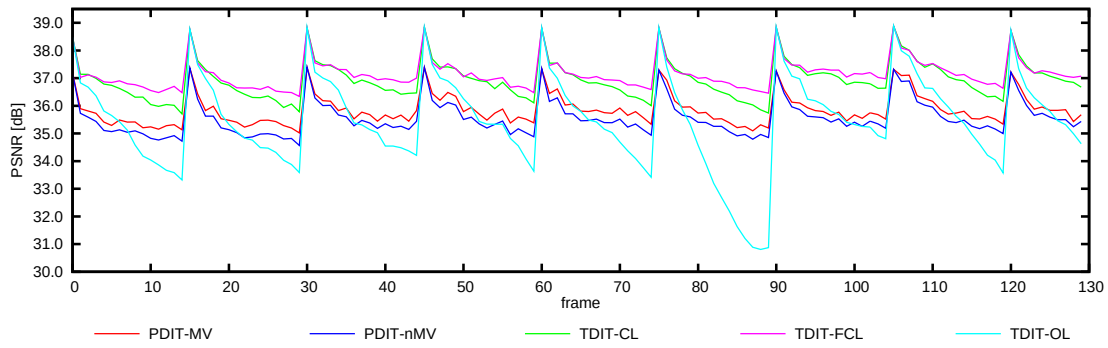
The exception to this dominant advantage of DCT-domain transcoding architectures can be found in the results obtained with the open-loop DCT-domain transcoder for video sequences with greater amounts of movement (*Carphone* and *Table-Tennis*). Although the PSNR measures corresponding to INTRA type images are still higher than those obtained with the pixel-domain transcoders, the



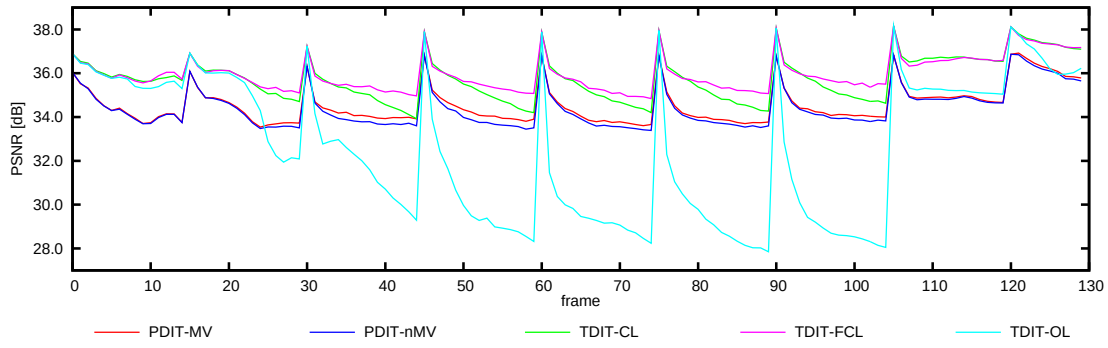
(a) *Akiyo* video sequence.



(b) *Silent-Voice* video sequence.

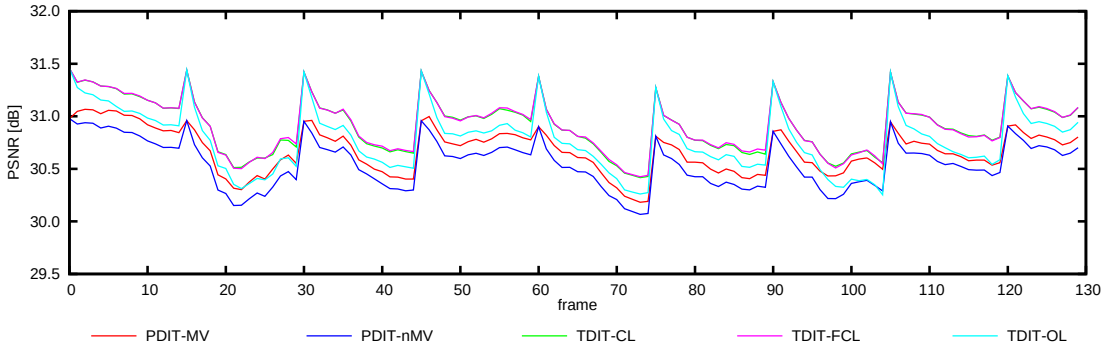


(c) *Carphone* video sequence.

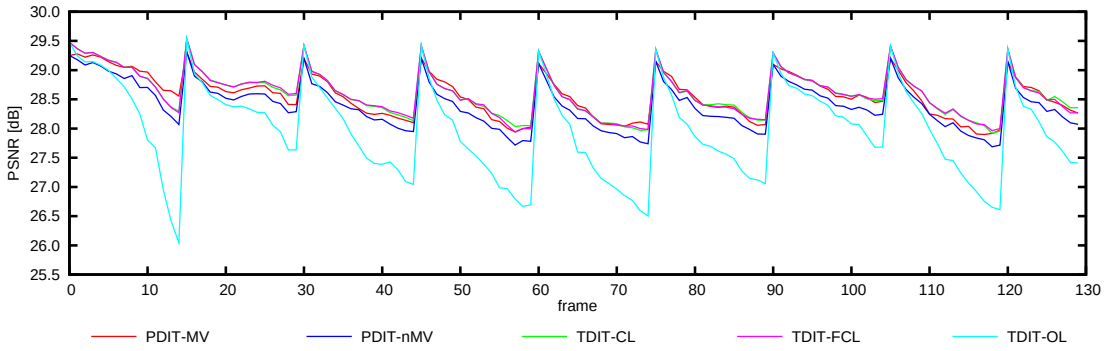


(d) *Table-Tennis* video sequence.

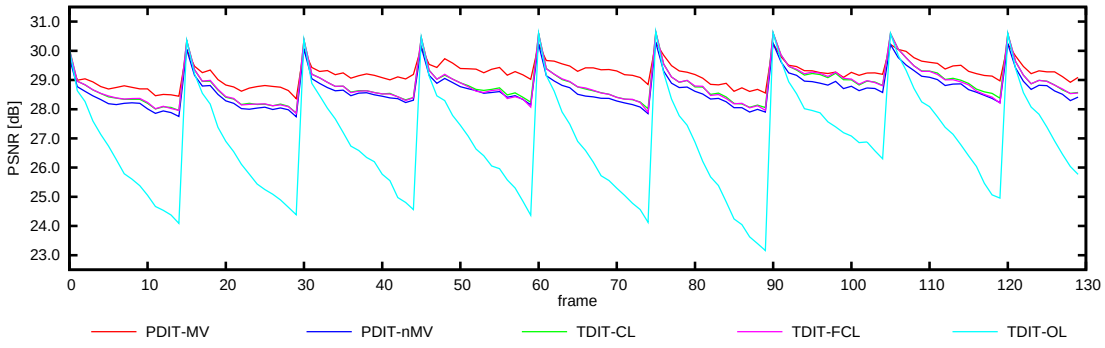
Figure 6.3: Obtained PSNR level after the NRSOs insertion for sequences *Akiyo*, *Silent-Voice*, *Carphone* and *Table-Tennis* using $Q = 4$.



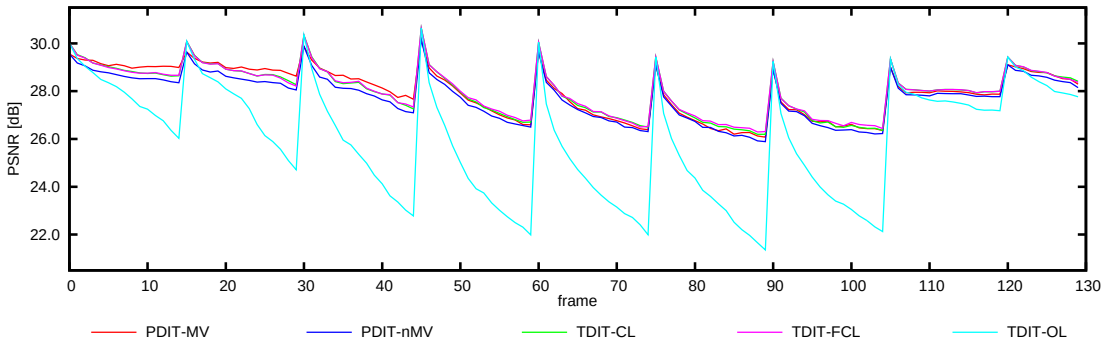
(a) *Akiyo* video sequence.



(b) *Silent-Voice* video sequence.



(c) *Carphone* video sequence.



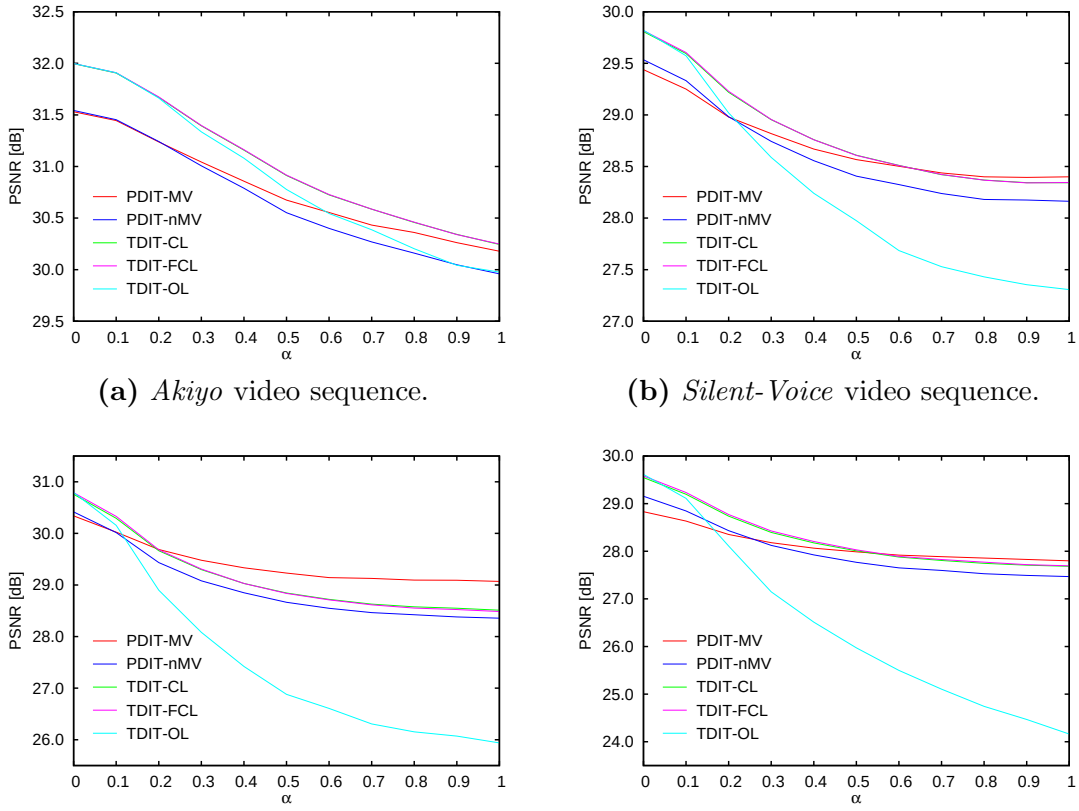
(d) *Table-Tennis* video sequence.

Figure 6.4: Obtained PSNR level after the NRSOs insertion for sequences *Akiyo*, *Silent-Voice*, *Carphone* and *Table-Tennis* using $Q = 15$.

accumulated drift caused a significant degradation in the processing of INTER type images.

In fig. 6.4 it is presented the set of results that were obtained using the same video sequences, when adopting a quantization setup with $Q = 15$. For the *Akiyo* video sequence, all DCT-domain transcoders presented better results than the pixel-domain transcoding architectures. For all other video sequences, the closed-loop DCT-domain transcoding architectures (TDIT-CL and TDIT-FCL) performed better than the PDIT-nMV pixel-domain transcoder, without re-estimation of the motion vectors. However, for certain segments of some considered video sequences (e.g.: *Carphone*), these DCT-domain transcoders presented slightly worse results than the pixel-domain transcoding architecture that re-estimates the motion vectors (PDIT-MV).

Besides these observations, it is also interesting to note the influence of the quantization parameter Q on the variation of the resulting PSNR value (see figs. 6.3



(a) *Akiyo* video sequence.

(b) *Silent-Voice* video sequence.

(c) *Carphone* video sequence.

(d) *Table-Tennis* video sequence.

Figure 6.5: Variation of the PSNR level with the transparency factor α after the NRSOs insertion in sequences *Akiyo*, *Silent-Voice*, *Carphone* and *Table-Tennis*, using $Q = 15$.

and 6.4). In fact, besides the expected decrease of the output video quality when greater quantization values were used, it should also be noted that the PSNR values obtained with the several transcoding architectures for $Q = 15$ not only are more similar to each other, but they also denote a slower degradation influence of the introduced drift. Such phenomenon can be observed by noting the smaller dynamic range of the variation of the video quality levels, between the first and the last INTER type frame of each GOP.

The exception to this quality performance pattern can be found in the experiments performed with the *open-loop DCT-domain transcoder*, whose results will be further discussed in section 6.2.4.

In fig. 6.5 it is presented the variation of the average PSNR level (considering the whole set of frames of each video sequence) with the transparency factor α. These charts provide us the ability to observe that the PSNR measure decreases when the opacity level of the inserted NRSOs is increased. Hence, these results allow us to use the α factor as a measure of the overall disturbance introduced in the original video sequence. This behavior complies with the estimation of the amount of distortion introduced by the open-loop insertion architecture, which was previously presented in eq. 4.42:

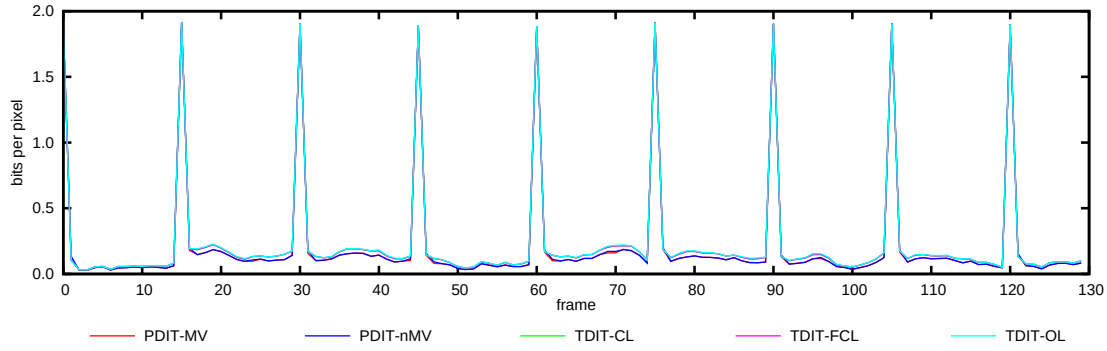
$$\boldsymbol{\epsilon}_t = \alpha \left(\mathbf{p}_u - \mathbf{i}_t \right). \tag{6.1}$$

6.2.2 Bit rate of the encoded video sequences

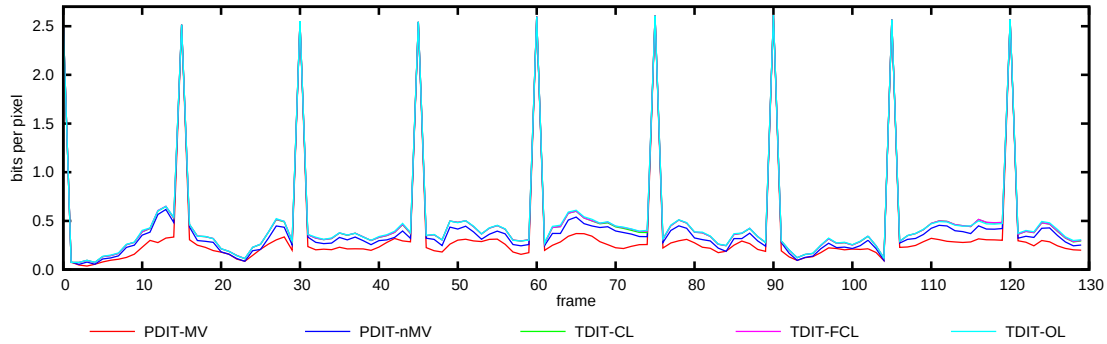
The average amount of bits required to encode each pixel of the considered video sequences with the proposed transcoding architectures is presented in figs. 6.6 and 6.7, for the quantization setups with $Q = 4$ and $Q = 15$, respectively.

The presented results show that the pixel-domain architecture with re-estimation of the motion vectors (PDIT-MV) provides the best performance in terms of the required bandwidth. Such difference is more significant in video sequences with greater amount of movement. This advantage can be justified by this architecture capability to better perform the motion compensated prediction encoding in INTER type images.

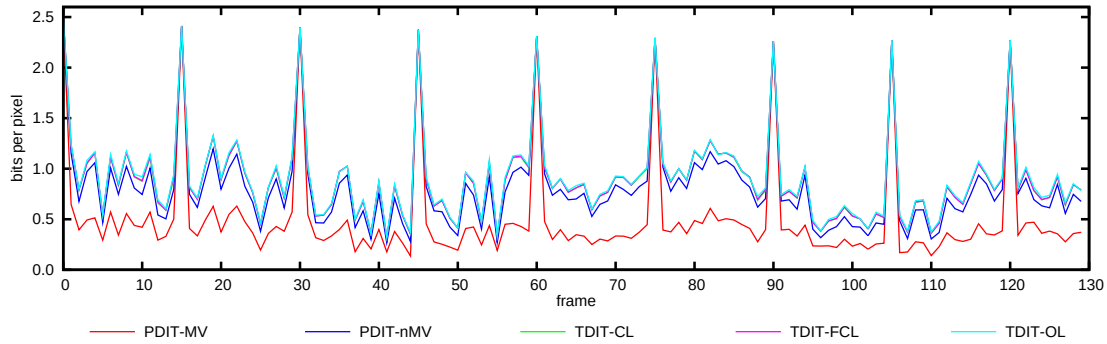
Among the remaining transcoder architectures, where the decoded motion vectors are re-used in the transcoding scheme, it is possible to observe that the amount of bits required to encode each pixel is quite similar, with a slight advantage for the pixel-domain architecture. In fact, it was observed that the DCT-domain transcoders require about 10% more bits than the PDIT-nMV transcoder to encode the considered frames with a smaller quantization step ($Q = 4$), and about 2% more bits to encode the same frames with a greater quantization step ($Q = 15$).



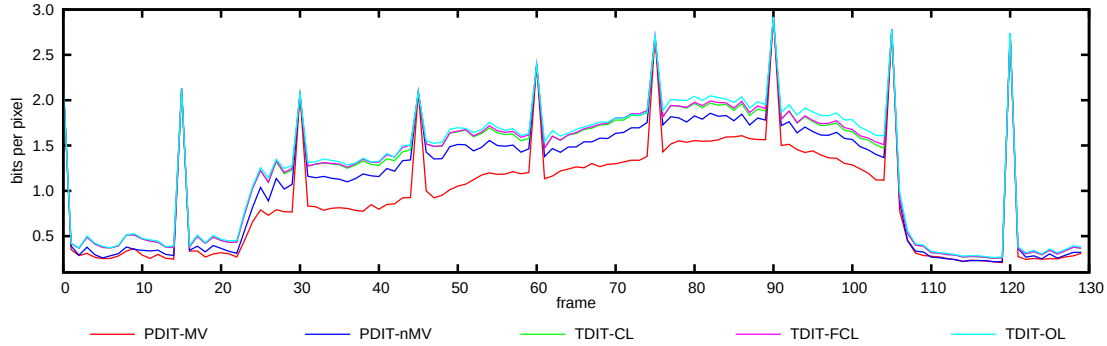
(a) *Akiyo* video sequence.



(b) *Silent-Voice* video sequence.

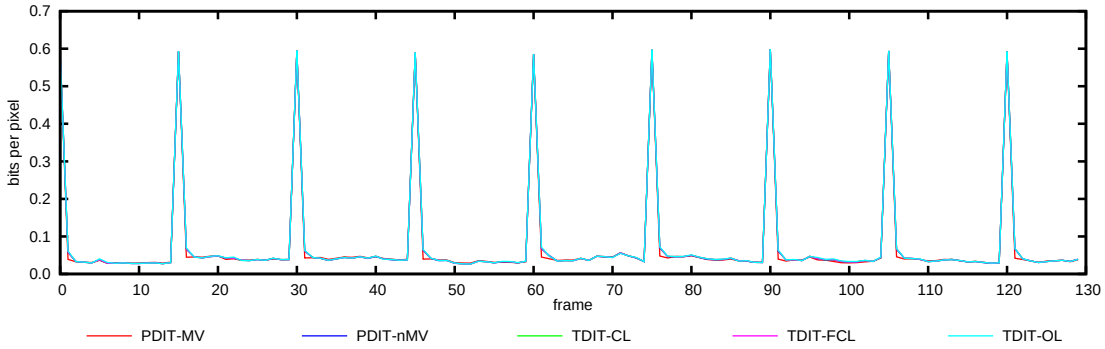


(c) *Carphone* video sequence.

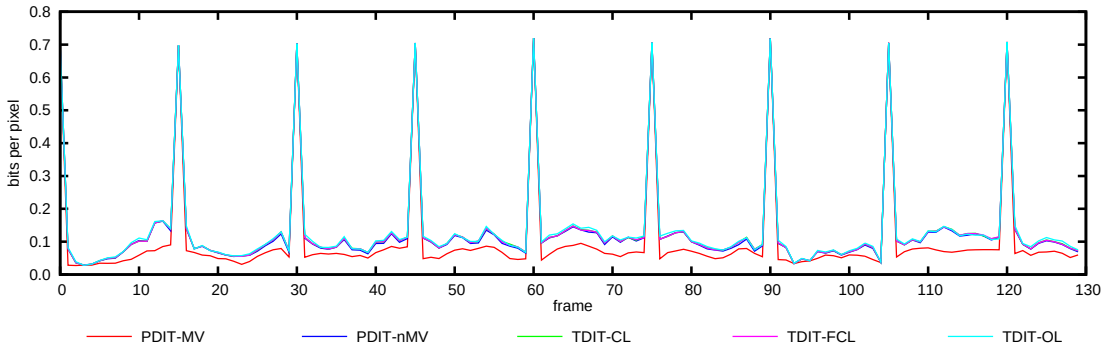


(d) *Table-Tennis* video sequence.

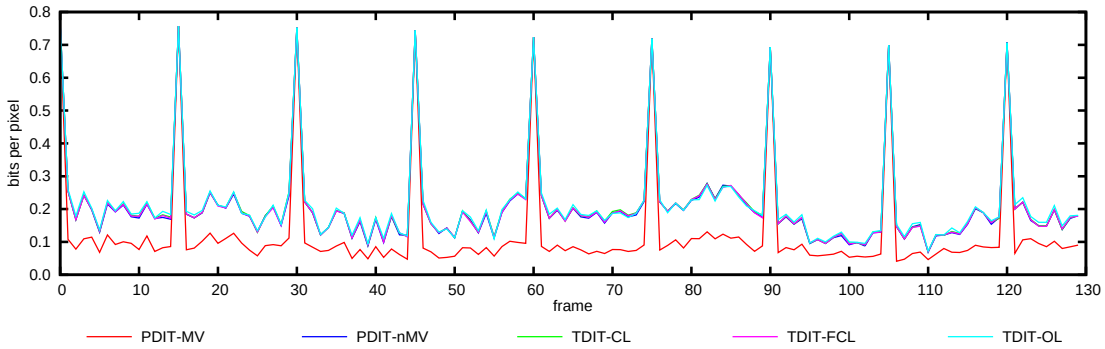
Figure 6.6: Average number of bits required to encode each pixel for the video sequences *Akiyo*, *Silent-Voice*, *Carphone* and *Table-Tennis* ($Q = 4$).



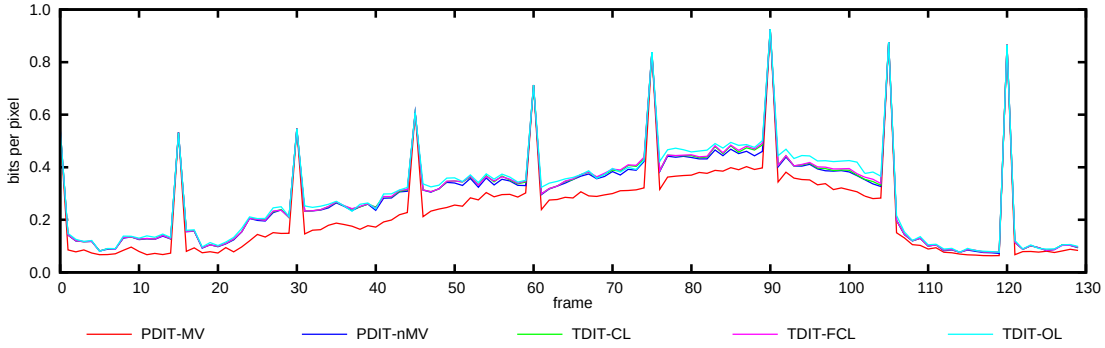
(a) *Akiyo* video sequence.



(b) *Silent-Voice* video sequence.



(c) *Carphone* video sequence.



(d) *Table-Tennis* video sequence.

Figure 6.7: Average number of bits required to encode each pixel for the video sequences *Akiyo*, *Silent-Voice*, *Carphone* and *Table-Tennis* ($Q = 15$).

The charts presented in fig. 6.6 and in fig. 6.7 also evidence the significant difference between the amount of bits required to encode INTRA type images and the amount of bits required to encode INTER type images. Such difference is easily justified by the absence of any prediction mechanism in the encoding algorithm of INTRA type images, leading to bit rates much higher (one order of magnitude greater) than those obtained for INTER type images.

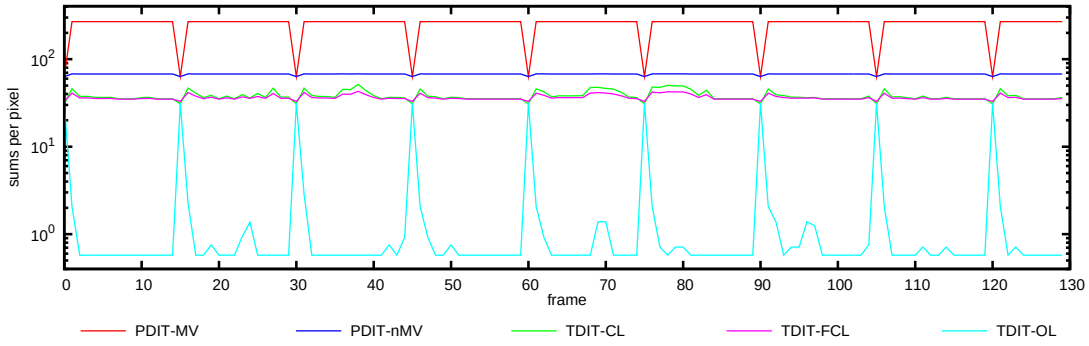
On the other hand, it is also interesting to note the influence of the quantization parameter Q on the variation of the resulting bit rate (see figs. 6.6 and 6.7). The observed decrease of the amount of bits required to encode each pixel of the considered video sequences is the result of a greater amount of AC frequency DCT coefficients that become null within each encoded block. This arises as a consequence of using greater quantization parameters, which implies a consequent reduction of the spatial details that are present in the encoded video sequences.

Finally, these charts also illustrate the influence of the amount of movement that is present in the considered video sequences on the obtained bit rate. Such phenomenon is easily observed in the *Carphone* and in the *Table-Tennis* video sequences. In the last one, it is even possible to observe a significant increase of the required number of bits, starting from frame number 25. Such increase can be justified by the motion activity contained in this sequence, not only due to the presence of a zoom-out effect, but also due to a global displacement (to the left) of the camera device used in the acquisition of this scene.

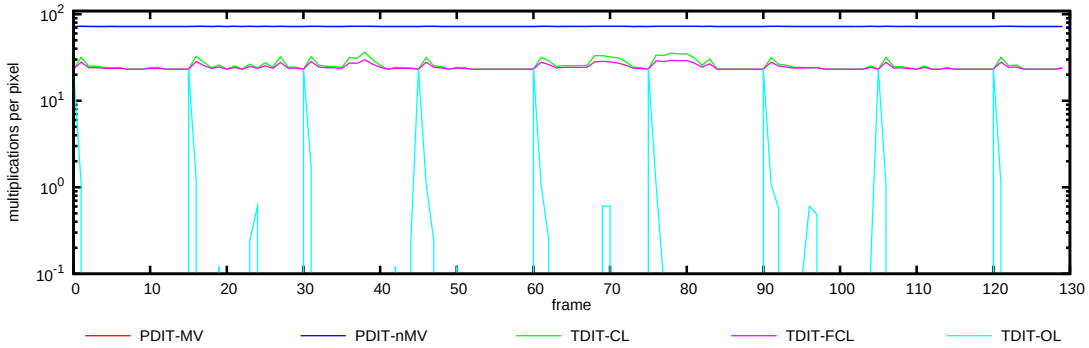
6.2.3 Efficiency of the NRSO insertion transcoders

In this section, it is presented the computational load required by the proposed architectures to insert the logo and subtitle in the considered video sequences, both for $Q = 4$ and $Q = 15$. In figs. 6.8 through 6.11 it is presented the observed variations of the number of required additions and multiplications for the *Akiyo* and *Table-Tennis* video sequences. Entirely similar results, obtained with the *Silent-Voice* and *Carphone* video sequences, are presented in figs. B.3 through B.6, in appendix B. To accommodate the high dynamic range of the amount of performed operations presented in all these figures, these charts were represented using a logarithmic scale in the y axis.

Tables 6.1, 6.2 and 6.3 present some figures of merit concerning the considered compressed DCT-domain architectures, namely, the minimum, the average and the maximum relative number of operations required to process the considered video sequences using the two quantizer setups. For comparison purposes, the values presented in these tables were normalized in relation to the corresponding compu-

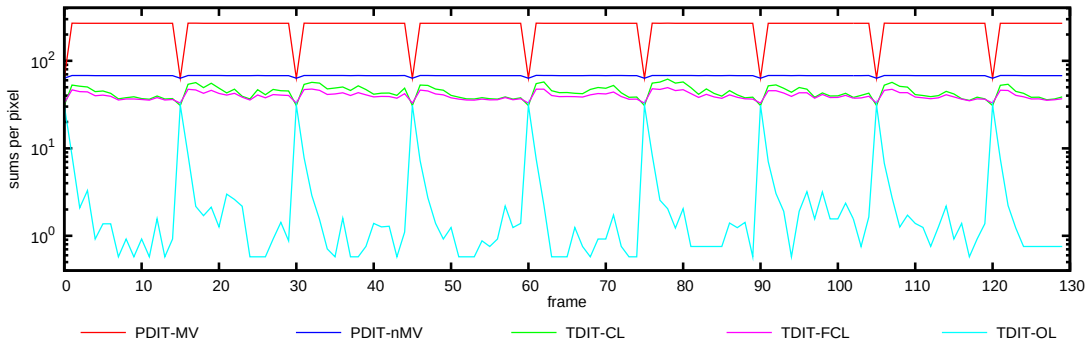


(a) Additions.

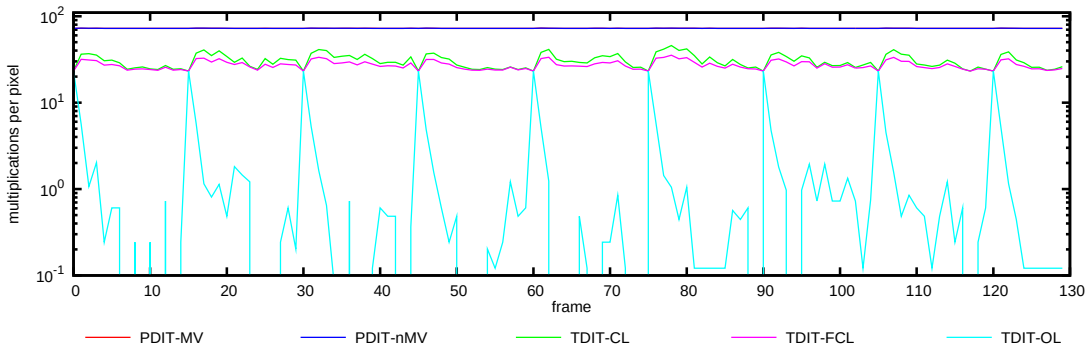


(b) Multiplications.

Figure 6.8: Number of operations required to insert the considered NRSOs in the *Akiyo* video sequence ($Q = 4$).

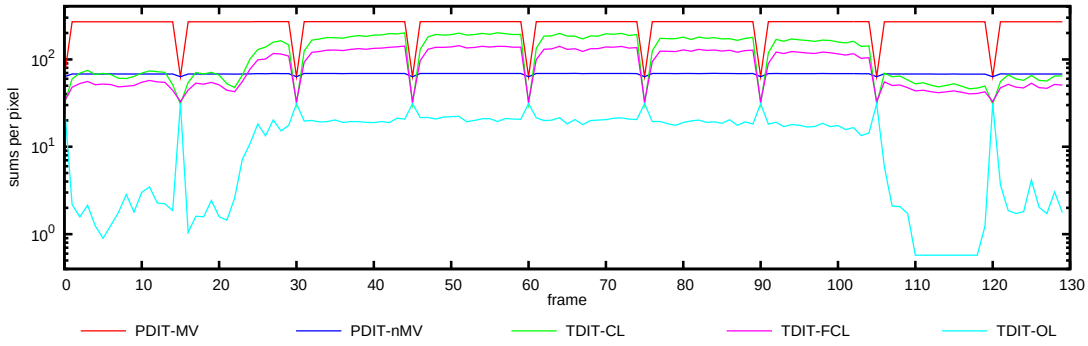


(a) Additions.

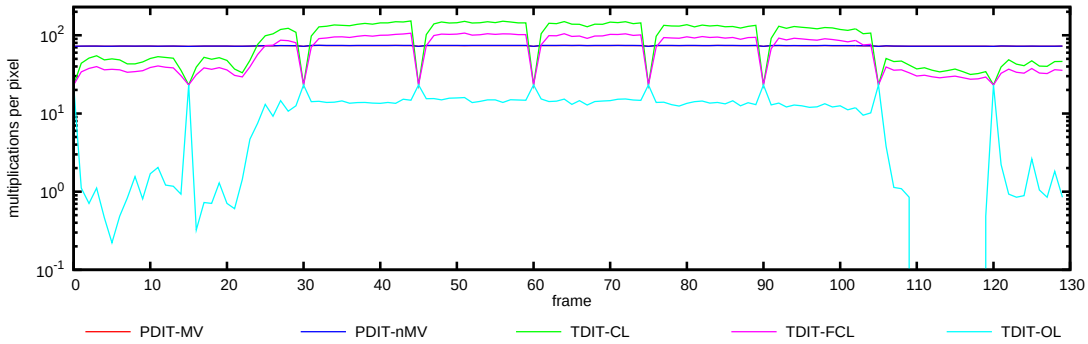


(b) Multiplications.

Figure 6.9: Number of operations required to insert the considered NRSOs in the *Akiyo* video sequence ($Q = 15$).

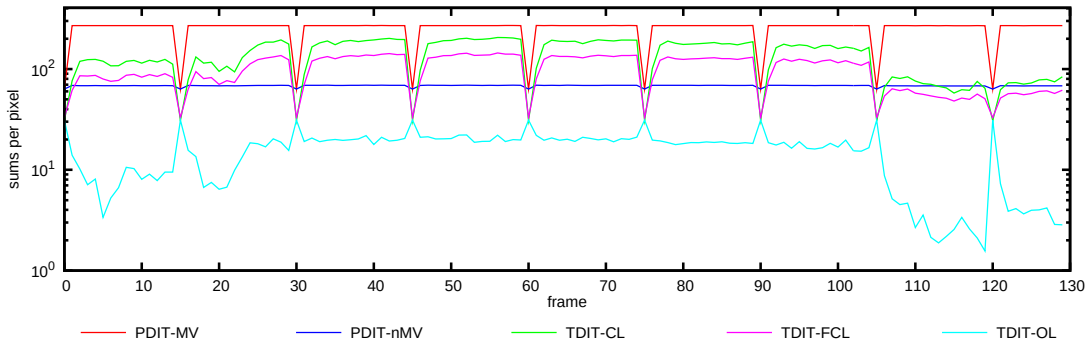


(a) Additions.

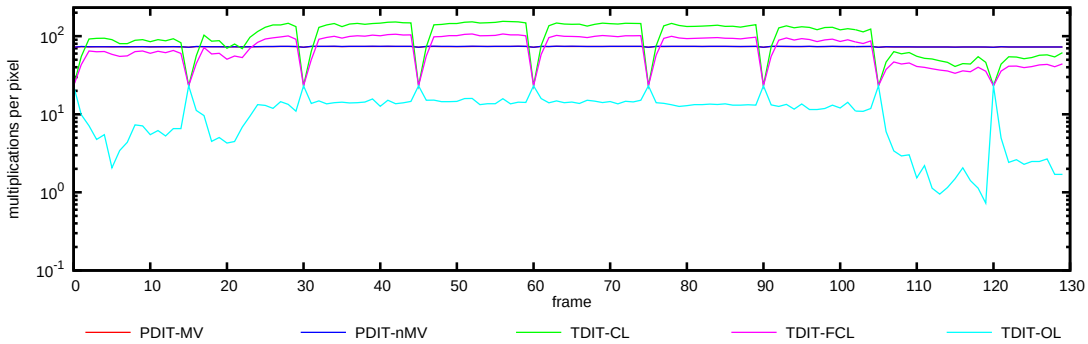


(b) Multiplications.

Figure 6.10: Number of operations required to insert the considered NRSOs in the *Table-Tennis* video sequence ($Q = 4$).



(a) Additions.



(b) Multiplications.

Figure 6.11: Number of operations required to insert the considered NRSOs in the *Table-Tennis* video sequence ($Q = 15$).

Table 6.1: Relation between the minimum, the average and the maximum number of operations required by the closed-loop DCT-domain transcoder (TDIT-CL) and those required by the pixel-domain architecture without re-estimation of the motion vectors (PDIT-nMV).

Video Sequence	$Q = 4$						$Q = 15$					
	Additions			Multiplications			Additions			Multiplications		
	min	**avg**	max	min	**avg**	max	min	**avg**	max	min	**avg**	max
Akiyo	0.49	**0.56**	0.76	0.32	**0.36**	0.53	0.49	**0.65**	0.91	0.32	**0.42**	0.63
Silent	0.49	**0.81**	1.17	0.32	**0.53**	0.85	0.49	**0.94**	1.32	0.32	**0.63**	1.00
Carphone	0.49	**1.68**	2.76	0.32	**1.17**	2.10	0.49	**1.71**	2.53	0.32	**1.20**	2.08
Table-tennis	0.49	**1.82**	2.92	0.32	**1.27**	2.03	0.49	**2.02**	2.98	0.32	**1.41**	2.06

Table 6.2: Relation between the minimum, the average and the maximum number of operations required by the computational-reduced closed-loop DCT-domain transcoder (TDIT-FCL) and those required by the pixel-domain architecture without re-estimation of the motion vectors (PDIT-nMV).

Video Sequence	$Q = 4$						$Q = 15$					
	Additions			Multiplications			Additions			Multiplications		
	min	**avg**	max	min	**avg**	max	min	**avg**	max	min	**avg**	max
Akiyo	0.52	**0.54**	0.63	0.32	**0.34**	0.43	0.52	**0.59**	0.72	0.32	**0.38**	0.48
Silent	0.52	**0.69**	0.90	0.32	**0.45**	0.68	0.52	**0.77**	1.00	0.32	**0.51**	0.78
Carphone	0.52	**1.24**	1.86	0.32	**0.84**	1.38	0.52	**1.26**	1.72	0.32	**0.86**	1.39
Table-tennis	0.52	**1.35**	2.08	0.32	**0.92**	1.42	0.52	**1.46**	2.09	0.32	**1.00**	1.42

Table 6.3: Relation between the minimum, the average and the maximum number of operations required by the open-loop DCT-domain transcoder (TDIT-OL) and those required by the pixel-domain architecture without re-estimation of the motion vectors (PDIT-nMV).

Video Sequence	$Q = 4$						$Q = 15$					
	Additions			Multiplications			Additions			Multiplications		
	min	**avg**	max	min	**avg**	max	min	**avg**	max	min	**avg**	max
Akiyo	0.01	**0.04**	0.46	0.00	**0.02**	0.32	0.01	**0.06**	0.46	0.00	**0.03**	0.32
Silent	0.01	**0.08**	0.46	0.00	**0.05**	0.32	0.01	**0.09**	0.46	0.00	**0.06**	0.32
Carphone	0.05	**0.18**	0.45	0.03	**0.12**	0.31	0.07	**0.19**	0.45	0.04	**0.13**	0.31
Table-tennis	0.01	**0.21**	0.45	0.00	**0.13**	0.31	0.02	**0.23**	0.45	0.01	**0.15**	0.31

tational load of the pixel-domain architecture without re-estimation of the motion vectors (PDIT-nMV).

Similarly to the previously presented charts, representing the variation of the video quality (PSNR) and of the required bandwidth, these charts have a pseudo-periodic characteristic, corresponding to the processing of the several GOPs, with distinctive maximums/minimums corresponding to each INTRA type frame. This fact can be easily justified by the different processing requirements of INTRA and INTER type images (e.g.: motion estimation/compensation, convolution, etc.).

As it would be expected, the pixel-domain transcoder with re-estimation of the motion vectors (PDIT-MV) is the most computationally demanding scheme. Since this architecture does not perform any simplification of the encoding, decoding and insertion algorithms, the required number of operations to process each block is always the same, depending only on the type (INTRA/INTER) of the considered frame. Consequently, by comparing the charts corresponding to the two considered pixel-domain transcoding architectures (PDIT-MV and PDIT-nMV), it is possible to conclude that the difference that is observed in the number of required additions (on average, 200.3 additions/pixel) is mainly due to the huge amount of additions/subtractions performed by the full-search block-matching motion estimation algorithm [75], to compute the SAD matching criteria for every candidate MB of INTER type images. All other processing blocks of these two architectures are exactly the same, leading to the same amount of multiplications that are required by the two insertion algorithms.

In what concerns the compressed-domain transcoding structures, it is possible to observe that in most video sequences the computational resources required by DCT-domain transcoding algorithms are lower than those required by any of the pixel-domain transcoders. This fact, together with the presented results concerning the obtained video quality (see section 6.2.1), justify the preference of the closed-loop DCT-domain transcoding algorithms over the equivalent pixel-domain transcoder architecture (without re-estimation of the motion vectors). Such computational advantage is even more significant if the open-loop DCT-domain transcoder is considered. The number of operations required by this architecture is often one order of magnitude lower than the number of operations required by the remaining architectures.

Contrasting with the pixel-domain transcoding architectures, it can also be observed from the charts presented in figs. 6.8 through 6.11 that the amount of computations required by the DCT-domain transcoders is not constant and varies significantly (even for INTER type images) within a certain GOP. In fact, it is possible

to realize that the computational cost of the two proposed closed-loop DCT-domain architectures (TDIT-CL and TDIT-FCL) has its absolute minimum values (corresponding to 49% of the number of additions and 32% of the number of multiplications of the PDIT-nMV architecture) at the time instants corresponding to INTRA type frames. This fact can be easily justified by recalling that for INTRA-type images it is not necessary to perform the motion compensation prediction, neither the removal of previously inserted objects from certain MBs of the image being processed. Even in INTER type images, the number of MBs that require the insertion or removal of the NRSOs is not always the same, depending on the insertion locations and on the motion activity of the scene along the time, thus leading to the presented variation. This phenomenom is emphasized in the charts representing the amount of computations required to process the *Table-Tennis* sequence (figs. 6.10 and 6.11). In fact, three different temporal segments can be distinguished in this sequence: the segment corresponding to frames 1 through 25; the segment corresponding to frames 25 through 105; and the segment corresponding to frames 105 through 130. While in the first and in the third segments the movements presented in the video sequence are local and restricted to certain regions of the image (e.g.: ping-pong ball), during the second segment there is a global movement of the whole scene caused by a zooming-out effect performed by the video acquisition camera. This zooming-out effect is responsible for a significant increase of the number of MBs that require some processing, thus justifying the abrupt growth observed in the charts illustrated in figs. 6.10 and 6.11. Consequently, the presented results show that this variable computational load should be taken into consideration whenever this type of transcoding architectures is adopted, since it is greatly dependent on the motion activity in the scene and on the amount of MBs that are involved in the insertion of the considered NRSOs.

From the analysis of the obtained results, it can be observed that while the TDIT-CL architecture required a computational load that varies between 0.36 and 2.02 of the amount of operations required by the PDIT-nMV transcoder, the TDIT-FCL structure requires an amount of computations that ranges between 0.34 and 1.46 of the load required by the reference architecture. Nevertheless, it still achieves a video quality very close to the TDIT-CL architecture. Consequently, from the results presented in tables 6.1 and 6.2 one should conclude that, depending on the video sequence under processing, compressed DCT-domain insertion algorithms may offer significant advantages in what concerns the computational load, requiring, in certain cases, an average number of operations that is one half of the number of operations required by the traditional pixel-domain insertion algorithms.

To conclude, a final attention should be given to the number of operations required by the open-loop DCT-domain architecture (TDIT-OL). As it can be seen from figs. 6.8 through 6.11, the lines in some of the charts corresponding to the number of multiplications required in the processing of the considered sequences by this architecture are not visible. Such cases occur whenever the number of multiplications required to process a given frame was zero. These situations were already expected and happen whenever there is no motion activity in the regions where the NRSOs should be inserted. In such circumstances there is no need to perform the motion compensation algorithm in the transcoder, thus leading to a null number of multiplications.

Furthermore, from the values presented in table 6.3 and from the charts illustrated in figs. 6.8 through 6.11, and by taking into account the description presented in section 4.3.5, one can conclude that the TDIT-OL architecture provides a significant saving in the number of operations that are required to perform this insertion algorithm. As in the other DCT-domain architectures, this saving greatly depends on the motion activity in the scene and corresponds to a computational load that ranges between 2% and 23% of the number of operations required by the pixel-domain architecture (PDIT-nMV). As it was referred before, this computational saving is mainly due to the absence of the 2-D symmetric convolution block in the processing of INTER type images. Consequently, and contrasting with the previously described DCT-domain insertion architectures, the overall computational cost of this scheme has its maximum when it processes INTRA type images, as it can be seen in the charts presented in figs. 6.8 through 6.11. As it will be seen in the following subsection, this insertion architecture may offer some interesting computational advantages over the other two architectures in the processing of video sequences with long GOP structures and characterized by very reduced amounts of movement.

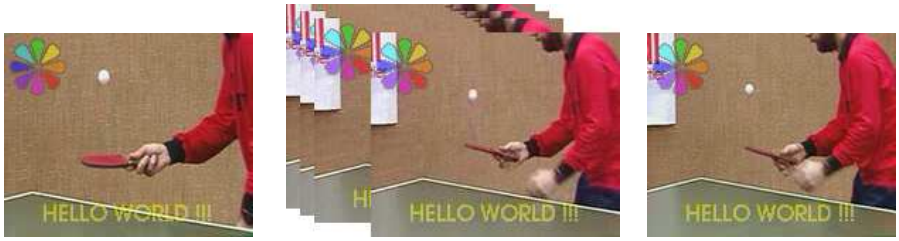
6.2.4 Drift introduced in INTER type images

One of the main causes of distortion in video coding arises from the usage of quantization. Despite the great compression capability that this functional block offers, it is the most responsible for the accumulation of quantization errors (drift) in the encoding of video sequences (see section 3.1.2). Moreover, since the INTER type frames of a given GOP are encoded using a temporal prediction scheme based on the other reference frames of that GOP, this degradation effect tends to suffer a gradual aggravation along the time, as it was already shown in figs. 6.3 and 6.4.

In fig. 6.12 it is illustrated a temporal segment of a decoded GOP, composed by $G = 15$ frames, that was obtained by applying the insertion algorithms to the

Table-Tennis video sequence. The reason for choosing this particular video sequence to illustrate this phenomenom emerges from the fact that it offers the set of worst case conditions to apply the proposed algorithms (see figs. 6.3(d), 6.4(d), 6.6(d) and 6.7(d)). The image shown in fig. 6.12(a) is the first frame of that GOP (INTRA type) and will be used as the reference for the following INTER type images. In fig. 6.12(b) it is illustrated the last frame of the same GOP (INTER type). Fig. 6.12(c) illustrates the first frame (INTRA type) of the following GOP. This set of images was obtained with a quantizer setup with $Q = 4$ and using the pixel-domain transcoder (PDIT-MV). The usage of such a small quantizer parameter led to a minimization of the drift introduced along the GOP, as it was already shown in the chart of fig. 6.3(d).

In fig. 6.13 it is illustrated the last image (frame no.44) of the considered GOP using the same quantizer and all other proposed architectures for the insertion algo-



(a) Frame no.30 ([36.31 dB]). **(b)** Frame no.44 ([33.92 dB]). **(c)** Frame no.45 ([36.81 dB]).

Figure 6.12: First frames (fig. 6.12(a) and fig. 6.12(c) - INTRA type) and last frame (fig. 6.12(b) - INTER type) of two consecutives GOPs of the *Table-Tennis* video sequence, using $Q = 4$ and the pixel-domain transcoder (with re-estimation of the motion vectors).

(a) Frame no.44 ([33.60 dB]). **(b)** Frame no.44 ([33.93 dB]). **(c)** Frame no.44 ([34.97 dB]). **(d)** Frame no.44 ([29.29 dB]).

Figure 6.13: Last frame (INTER type) of the considered GOP of the *Table-Tennis* video sequence, processed using: the pixel-domain transcoder with re-usage of the motion vectors (fig. 6.13(a)); the closed-loop DCT-domain transcoder (fig. 6.13(b)); the computational-reduced DCT-domain transcoder (fig. 6.13(c)) and the open-loop DCT-domain transcoder (fig. 6.13(d)) ($Q = 4$).

rithm: PDIT-nMV (fig. 6.13(a)), TDIT-CL (fig. 6.13(b)), TDIT-FCL (fig. 6.13(c)) and TDIT-OL (fig. 6.13(d)). As it was previously illustrated in fig. 6.3(d), the PSNR obtained using the pixel-domain transcoder with re-usage of the motion vectors (PDIT-nMV) is only slightly lower than the PSNR measure obtained using the pixel-domain insertion algorithm with re-estimation of the motion vectors (PDIT-MV). In figs. 6.13(b) and 6.13(c) it is illustrated the same image, but obtained with the compressed DCT-domain transcoders: TDIT-CL and TDIT-FCL, respectively. As it was previously shown in section 3.1.2, the absence of the degradation effect directly introduced by the usage of fixed-precision in the arithmetic operations that are performed in the computation of both the direct and inverse DCTs leads to greater PSNR quality measures, obtained with the closed-loop DCT-domain transcoders, than the quality levels that are obtained with the pixel-domain insertion algorithm with re-estimation of the motion vectors (PDIT-MV). The significant difference that was obtained for the particular case of the computational-reduced DCT-domain transcoder (TDIT-FCL) of about 1.05 dB can be justified by the fact that this insertion algorithms only processes the MBs where the NRSO is supposed to be inserted (and, eventually, the surrounding MBs). Consequently, the degradation effects that are introduced can be considered as a localized form of distortion that only affects a restricted sub-region in the image (which tends to enlarge along the GOP), corresponding to the fraction that is actually affected by the insertion of the NRSO. If one takes into account that the considered NRSOs do not significantly change with time, one can naturally consider that, from a mere subjective point of view, the degradation that is introduced in such areas will not significantly affect the perception of the rest of the scene. Consequently, they are not easily perceived by the viewer.

In fig. 6.13(d) it is illustrated the same frame but obtained with the open-loop DCT-domain transcoder (TDIT-OL). A careful observation of this image provides the means to further understand the source of the significant degradation effect that was shown in the chart of fig. 6.3(d). In fact, by comparing this frame with the first image of the considered GOP (INTRA type image, illustrated in fig. 6.12(a)), one can easily realize that the gap-line that crosses the characters of the word "WORLD" in the subtitle of fig. 6.13(d) is almost parallel to the edge of the table. If the description of the insertion mechanism that is adopted by this scheme is taken into account (see section 4.3.5), one can easily conclude that this mismatch is mainly caused by a deficient cancellation of the insertion algorithm, as it was previously discussed in the last paragraph of section 4.3.5. This phenomenom is particularly perceived at the presence of significant movement or of highly textured areas, and it gives rise to

a gradual introduction of an observable distortion effect in the neighboring regions close to the pixels areas where the NRSOs have been inserted. Such effect will prevail along the whole GOP and can only be removed when a new INTRA type frame is encoded. In this case, the combination of the presence of the table edge and of the zooming-out effect caused by the video camera provided the worst processing conditions which led to an easy perception of this phenomenom and whose distortion effect was easily noticeable in the chart presented in fig. 6.3(d). Moreover, it also resulted in an observable increment of the computational load, as it was previously noted. Were these conditions not satisfied and this degradation would not be easily perceived by the observer.

Despite this degradation effect, there is still another noticeable phenomenom in the region covered by the logo that also contributes to the distortion level presented in the chart of fig. 6.3(d). In fact, if one compares the *"flower"* logo inserted in this frame with the logos inserted using the other transcoder architectures, it is possible to perceive a certain reduction of its luminance level. This fact can be justified by a deficient processing of the received differences signal ($\mathbf{E}_t$) and of the adopted transparency factor (α) in the processing of INTER type images (see eq. 4.42).

The images shown in figs. 6.14 and 6.15 were obtained in a similar manner as those presented in figs. 6.12 and 6.13, respectively, but using, in this case, a quantizer setup with $Q = 15$. By observing figs. 6.14(b), 6.15(a), 6.15(b) and 6.15(c), it is possible to realize that both the pixel-domain insertion algorithms and the closed-loop DCT-domain architectures roughly provide the same video quality levels, as it was previously illustrated in the charts of fig. 6.4(d). On the other hand, from the observation of fig. 6.15(d), corresponding to the same processed frame but obtained with the open-loop DCT-domain transcoder (TDIT-OL), it is possible to perceive a much more serious degradation effect surrounding the NRSO area. As it was referred before, this degradation is mainly caused by a deficient removal and insertion of the NRSO data in the processed macroblocks of INTER type images (see eq. 4.50), resulting from the significant amount of motion activity during this temporal segment of the video sequence, characterized by a global movement of the whole scene due to a zooming-out effect performed by the video camera. This degradation is even worsened by the accumulation of the greater quantization errors along the encoding of the considered GOP, which significantly affects the possibility to perfectly remove the NRSO data that was inserted in previously encoded frames.

Even so, it is worth noting that the open-loop DCT-domain insertion architecture still should be considered for certain specific insertion applications, due to the significant computational savings that it offers. Namely, in those applications where

the particular scene characteristics, in terms of the amount of motion and spatial details, provide the required conditions to keep the distortion level under acceptable limits. One example of such applications is the processing of sequences containing a person, or a group of few people, who keep speaking in front of the camera with a static background. In fact, such type of scene is very common in certain kinds of television programs, such as *"News Reports"*, *"Weather Forecast"* or even *"Live Interviews"*. The considered video sequences *Akiyo* and *Silent-Voice* are good examples of this kind of scenes. In fig. 6.16(a) and 6.16(b) it is presented the last frame of the first GOP of these video sequences. In this processing setup, it was considered a smaller GOP length ($G = 8$), in order to restrict the amount of distortion due to the accumulation of drift. As it can be observed, for these particular conditions, the amount of distortion that is introduced by this algorithm is still confined within tolerable levels, from the subjective point of view.

(a) Frame no.30 ([29.87 dB]). **(b)** Frame no.44 ([27.66 dB]). **(c)** Frame no.45 ([30.16 dB]).

Figure 6.14: First frames (fig. 6.14(a) and fig. 6.14(c) - INTRA type) and last frame (fig. 6.14(b) - INTER type) of two consecutives GOPs of the *Table-Tennis* video sequence, using $Q = 15$ and the pixel-domain transcoder (with re-estimation of the motion vectors).

(a) Frame no.44 ([27.09 dB]). **(b)** Frame no.44 ([27.26 dB]). **(c)** Frame no.44 ([27.34 dB]). **(d)** Frame no.44 ([22.78 dB]).

Figure 6.15: Last frame (INTER type) of the considered GOP of the *Table-Tennis* video sequence, processed using: the pixel-domain transcoder with re-usage of the motion vectors (fig. 6.15(a)); the closed-loop DCT-domain transcoder (fig. 6.15(b)); the computational-reduced DCT-domain transcoder (fig. 6.15(c)) and the open-loop DCT-domain transcoder (fig. 6.15(d)) ($Q = 15$).

(a) Frame no.7 [34.75 dB]. **(b)** Frame no.7 [32.40 dB].

Figure 6.16: Last frame (INTER type) of the *Akiyo* (fig. 6.16(a)) and *Silent-Voice* (fig. 6.16(b)) video sequences, processed using the open-loop DCT-domain transcoder, with a GOP length $G = 8$ and $Q = 8$.

6.3 Dynamic video composition

This section presents the set of experimental results that were obtained to assess the set of dynamic video compositing techniques, proposed in chapter 5. In particular, in section 6.3.1 it is presented the evaluation of the space scaling algorithm by an arbitrary integer scale factor, described in section 5.2; section 6.3.2 addresses the assessment of the block-based motion re-estimation algorithm in the compressed DCT-domain, described in section 5.3; and section 6.3.3 presents the evaluation of the DCT-domain dynamic video composition architecture, proposed in section 5.4.

6.3.1 Space scaling algorithm by an arbitrary integer scale factor[§]

Video transcoding structures for spatial downscale comprise several different stages that must be implemented in order to re-size the incoming video sequence. In fact, while in INTRA type images only the spatial-domain information has to be downscaled, in INTER type frames the downscaling transcoder must also take into account several other processing tasks, as a result of the adopted temporal prediction mechanism. Some of such tasks involve the re-usage and composition of the decoded motion vectors, the scaling of the composited motion vectors, the refinement of the scaled motion vectors, the computation of the new prediction values obtained by motion compensation, etc. All of such processing steps have been

[§]Some portions of this section appeared in:

[90] - N. Roma and L. Sousa, "Efficient hybrid DCT-domain algorithm for any arbitrary integer re-size video downscaling," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, no. 57291, pp. 1–16, Sep. 2007.

jointly or separately studied along the last few years [1, 113].

As many other proposals, the research that was carried out in this field (see section 5.2) focuses solely on the proposal of an efficient computational scheme to downscale the DCT coefficients blocks decoded from the incoming video stream by any arbitrary integer scaling factor. The evaluation of its performance was carried out by integrating the proposed downscaling algorithm in a reference closed-loop H.263 [30] video transcoding system, as shown in fig. 6.17. In this transcoding architecture, both the MC-DCT and the Transform-Domain Motion Estimation (ME-DCT) modules were implemented in the DCT-domain. In particular, the motion estimation module of the encoding part of the transcoder was implemented using the DCT-domain least squares motion re-estimation algorithm proposed in section 5.3, by considering a ± 1 pixel search range [89]. By adopting this structure, the encoder loop may compute a new reduced-resolution prediction residual, providing a realignment of the predictive and residual components and thus minimizing the involved drift [114]. Nevertheless, to isolate the proposed algorithm from other encoding mechanisms that could interfere in this assessment (such as motion estimation/compensation), a first evaluation considering only the provided static video quality, by using solely INTRA type images, is presented in subsection B. A performance evaluation of the real performance of the algorithm, when processing video sequences that apply the traditional temporal prediction mechanisms, is presented in subsection C.

The implemented system was applied in the scaling of a set of several CIF benchmark video sequences with different characteristics (*Akiyo*, *Silent-Voice*, *Carphone*, *Table-Tennis* and *Mobile & Calendar*) and using different scaling factors ($\mathcal{S}_{\mathcal{F}}$). The majority of the results that are presented in this section were obtained using the *Mobile & Calendar* video sequence and a quantization setup with $Q = 4$. The preference for this particular video sequence arises from its peculiar characteristics in

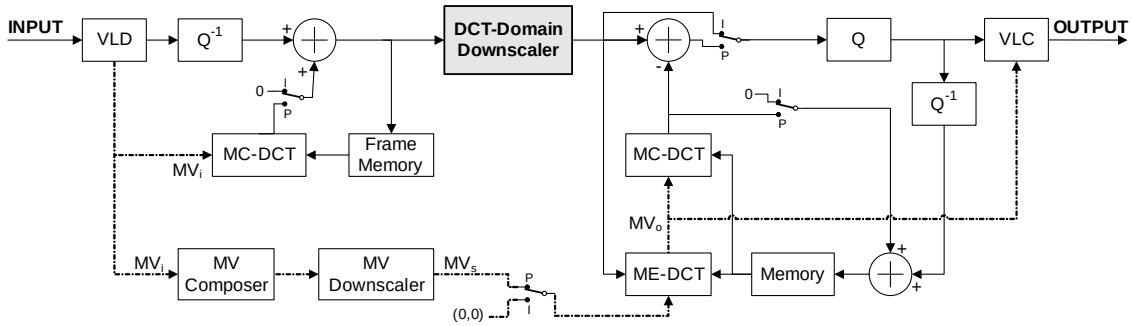


Figure 6.17: Integration of the proposed DCT-domain downscaling algorithm in a H.263 video transcoder.

(a) **(b)** **(c)** **(d)** **(e)**

Figure 6.18: Space scaling of the CIF *Mobile & Calendar* video sequence ($Q = 4$): (a) original frame; (b) $\mathcal{S}_{\mathcal{F}} = 2$; (c) $\mathcal{S}_{\mathcal{F}} = 3$; (d) $\mathcal{S}_{\mathcal{F}} = 4$; (e) $\mathcal{S}_{\mathcal{F}} = 5$.

terms of both the spatial detail and the amount of movement. Nevertheless, the algorithm was equally assessed with all the remaining considered video sequences and using a wide range of quantization steps, leading to entirely equivalent results. For all these experiments, it was considered the block size adopted by most image and video standards [30]: $N = 8$.

In fig. 6.18 it is represented the first frame of both the input and output video streams, considering the *Mobile & Calendar* video sequence and $\mathcal{S}_{\mathcal{F}} = 2,\ 3,\ 4$ and 5. To evaluate the influence of the video scaling on the output bit stream, the same format (CIF) was adopted for both video sequences, by filling the remaining area of the output frame with null pixels. By doing so, not only do the two video streams share a significant amount of the VLC parameters, thus simplifying their comparison, but it also provides an easy encoding of the scaled sequences, since their dimensions are often non-compliant with current video standards. Nevertheless, only the representative area corresponding to the scaled image was actually considered to evaluate the output video quality (PSNR) and drift. To do so, several different alternative approaches could have been adopted to evaluate this PSNR performance. A methodology that has been adopted by several authors consists in implementing and cascading an up-scaling and a downscaling transcoders, in order to compare the reconstructed images at the full-scale resolution [16]. However, since such approach also introduces a non-negligible degradation effect associated with the auxiliary up-scaling stage, it was not adopted in the presented experimental setup. As a consequence, the PSNR quality assessment was carried out by comparing each scaled frame (obtained with each algorithm under evaluation), with a corresponding reference scaled frame, that was carefully computed in order to avoid the influence of any lossy processing step related to the encoding algorithm.

An accurate quantization-free pixel-domain filtering and downsampling scheme was specially implemented for this specific purpose. This solution has proved to be a quite satisfactory alternative, when compared with other possible approaches to compute the scaled reference frame (such as DCT decimation), since it may provide a precise control over the inherent filtering process.

In the following, the proposed algorithm will be compared with the remaining considered downscaling algorithms, by considering several different evaluation metrics, namely: the *computational cost*, the *static video quality*, the introduced *drift* and the resulting *bit rate*.

A - Computational cost

In table 6.4(a), the proposed HDT algorithm is compared with the pixel-domain transcoder (CPAT) and the DCT decimation transcoder (DDT), in what concerns the involved computational cost. As it was mentioned before, such computational cost was evaluated by counting the total amount of multiplications ($\mathcal{M}$) that are required to implement the downscaling procedure. In order to obtain comparison results as fair as possible, all the considered algorithms adopted the same number of DCT coefficients (K) for each of these comparisons and were implemented for several integer scaling factors ($\mathcal{S}_{\mathcal{F}}$).

In the first row of the presented results it is emphasized the expected and significant computational advantage provided by the proposed scheme over the trivial pixel-domain approach, using the whole set of DCT coefficients (CPAT). On the other hand, the results presented in the second row evidence the clear computational advantages provided by the proposed scheme when compared with the DCT decimation transcoder (DDT), to downscale the input video sequences by any arbitrary integer scaling factor. In particular, it can be clearly seen that the HDT approach presents more significant advantages for scaling factors other than integer powers of 2, leading to a reduction of the computational cost as high as 5 ($\mathcal{S}_{\mathcal{F}} = 7$). Such phenomenom was already expected and is a direct consequence of the computational inefficiency inherent to the post-processing discarding stage of the DDT algorithm, illustrated in fig. 5.1. This computational advantage is even more significant for higher values of the difference: $\mathcal{S}_{\mathcal{F}} - 2^{\lfloor \log_2 \mathcal{S}_{\mathcal{F}} \rfloor}$.

In table 6.4(b) it is presented the variation of the computational cost of the considered schemes when a different number of DCT coefficients (K) is used. For such considered experimental setups, the pixel-domain transcoder (CPAT) adopted the whole set of DCT coefficients ($K = N$), while the DCT decimation transcoder (DDT) adopted $K = \lceil N/\mathcal{S}_{\mathcal{F}} \rceil$ coefficients, as defined in [48]. As it was predicted be-

Table 6.4: Computational cost comparison of the several considered downscaling algorithms (CIF *Mobile* & *Calendar* video sequence, $Q = 4$).

(a) Comparison of the proposed algorithm with the other considered schemes in what concerns the number of multiplications required to process each pixel, for several scaling factors ($\mathcal{S}_\mathcal{F}$).

$\mathcal{S}_\mathcal{F}$	2	3	4	5	6	7	8	9	10	K
$\frac{\mathcal{M}(\text{HDT})}{\mathcal{M}(\text{CPAT})}$	0.5	0.3	0.2	0.2	0.2	0.2	0.1	0.1	0.1	$K_{\text{HDT}} = K_{\text{CPAT}} = N$
$\frac{\mathcal{M}(\text{HDT})}{\mathcal{M}(\text{DDT})}$	0.9	0.7	0.9	0.5	0.3	0.2	0.9	0.7	0.5	$K_{\text{HDT}} = K_{\text{DDT}} = \left\lceil \frac{N}{\mathcal{S}_\mathcal{F}} \right\rceil$

(b) Variation of the number of multiplications required to process each pixel with the number of considered DCT coefficients (K).

Computational Cost	$\mathcal{S}_\mathcal{F}$	K							
		8	7	6	5	4	3	2	1
$\mathcal{M}$ (CPAT)	2	30.4	–	–	–	–	–	–	–
$\mathcal{M}$ (HDT)		14.8	13.0	11.4	10.1	8.9	7.9	7.1	6.4
$\mathcal{M}$ (DDT)		–	–	–	–	9.8	–	–	–
$\mathcal{M}$ (CPAT)	3	27.0	–	–	–	–	–	–	–
$\mathcal{M}$ (HDT)		9.3	8.0	6.8	5.7	4.8	4.1	3.5	3.1
$\mathcal{M}$ (DDT)		–	–	–	–	–	5.6	–	–
$\mathcal{M}$ (CPAT)	4	25.7	–	–	–	–	–	–	–
$\mathcal{M}$ (HDT)		5.3	4.5	3.8	3.2	2.7	2.3	2.0	1.7
$\mathcal{M}$ (DDT)		–	–	–	–	–	–	2.2	–
$\mathcal{M}$ (CPAT)	5	25.2	–	–	–	–	–	–	–
$\mathcal{M}$ (HDT)		5.4	4.4	3.6	2.9	2.3	1.9	1.5	1.3
$\mathcal{M}$ (DDT)		–	–	–	–	–	–	2.7	–
$\mathcal{M}$ (CPAT)	6	24.8	–	–	–	–	–	–	–
$\mathcal{M}$ (HDT)		4.1	3.4	2.7	2.2	1.7	1.3	1.0	0.8
$\mathcal{M}$ (DDT)		–	–	–	–	–	–	3.0	–
$\mathcal{M}$ (CPAT)	7	24.7	–	–	–	–	–	–	–
$\mathcal{M}$ (HDT)		4.0	3.3	2.6	2.1	1.6	1.2	0.9	0.8
$\mathcal{M}$ (DDT)		–	–	–	–	–	–	4.1	–
$\mathcal{M}$ (CPAT)	8	24.5	–	–	–	–	–	–	–
$\mathcal{M}$ (HDT)		2.1	1.8	1.4	1.2	0.9	0.7	0.6	0.5
$\mathcal{M}$ (DDT)		–	–	–	–	–	–	–	0.6
$\mathcal{M}$ (CPAT)	9	24.3	–	–	–	–	–	–	–
$\mathcal{M}$ (HDT)		3.2	2.6	2.0	1.5	1.1	0.8	0.5	0.4
$\mathcal{M}$ (DDT)		–	–	–	–	–	–	–	0.6

fore (see table 5.2), it can be seen that the computational cost of the proposed HDT algorithm significantly decreases when the number of considered DCT coefficients decreases.

The presented results also evidence a direct consequence of the computational

advantages provided by the proposed algorithm: for the same amount of multiplications ($\mathcal{M}$) and a given scaling factor ($\mathcal{S}_\mathcal{F}$), the proposed algorithm is able to process a greater amount of decoded DCT coefficients (K) than the DCT decimation transcoder (DDT). This fact can be easily observed for the transcoding setup using $\mathcal{S}_\mathcal{F} = 3$, illustrated in table 6.4(b). By approximately using the same number of operations, the DCT decimation transcoder processes only $K^2 = 9$ DCT coefficients of each block, while the proposed transcoder may process $K^2 = 25$ coefficients. As it will be shown in subsection E, such advantage will allow this algorithm to obtain scaled images with greater PSNR values in transcoding systems with restricted computational resources.

B - Static video quality

As it was referred before, to isolate the evaluation of the proposed algorithm from other processing issues (such as motion vector scaling and refinement, drift compensation, predictive motion compensation, etc.), a first evaluation and assessment of the considered algorithms was performed using solely INTRA type images. The comparison of such static video quality performances allows to better understand the advantages of the proposed approach, by focusing the attention on the most important aspects under analysis: the accuracy and the computational cost of the spatial downscaling algorithms. A dynamic evaluation of the obtained video quality is presented in the following subsection, by also considering the inherent drift that is introduced when temporal prediction schemes are applied.

Table 6.5 presents the PSNR measure that was obtained after the space scaling operation for several scaling factors ($\mathcal{S}_\mathcal{F}$) and considering different amounts of DCT coefficients (K). Similar results were also obtained for all the remaining considered video sequences and quantization steps, evidencing that the overall quality of the resulting video sequences is better when the proposed HDT algorithm is applied. These performance results were also validated by undergoing a perceptual evaluation of the resulting video sequences using several different observers, who have confirmed the obtained quality levels.

The first observation that should be retained from these results is the fact that the proposed algorithm is consistently better than the trivial cascaded pixel-domain architecture (CPAT) for the whole range of considered scaling factors. However, it should be noted that this improvement is not directly owed to the scaling algorithm itself. In fact, when the whole set of decoded DCT coefficients is considered ($K = N$), these two algorithms actually make use of quite similar downsampling filters. Nevertheless, by directly processing the incoming blocks of DCT coefficients

Table 6.5: Comparison of the PSNR quality level [dB] obtained with the considered downscaling algorithms (CIF *Mobile* & *Calendar* video sequence, $Q = 4$).

Algorithm	$\mathcal{S}_\mathcal{F}$	K							
		8	7	6	5	4	3	2	1
CPAT		36.0	–	–	–	–	–	–	–
HDT	2	36.5	36.4	35.2	31.3	31.3	24.6	21.5	18.6
DDT		–	–	–	–	31.4	–	–	–
CPAT		36.1	–	–	–	–	–	–	–
HDT	3	36.7	36.6	36.3	35.6	32.8	28.4	24.8	20.7
DDT		–	–	–	–	–	27.9	–	–
CPAT		36.2	–	–	–	–	–	–	–
HDT	4	36.7	36.6	36.6	36.0	36.0	32.5	32.5	22.0
DDT		–	–	–	–	–	–	32.6	–
CPAT		36.1	–	–	–	–	–	–	–
HDT	5	36.7	36.7	36.5	35.9	34.8	33.8	29.5	23.6
DDT		–	–	–	–	–	–	28.6	–
CPAT		36.2	–	–	–	–	–	–	–
HDT	6	36.8	36.8	36.8	36.5	36.5	34.8	32.0	24.6
DDT		–	–	–	–	–	–	30.2	–
CPAT		36.3	–	–	–	–	–	–	–
HDT	7	36.7	36.7	36.7	36.4	35.4	34.1	31.5	25.2
DDT		–	–	–	–	–	–	28.6	–
CPAT		36.3	–	–	–	–	–	–	–
HDT	8	37.0	37.0	37.0	37.0	37.0	37.0	37.0	37.0
DDT		–	–	–	–	–	–	–	37.0
CPAT		36.3	–	–	–	–	–	–	–
HDT	9	37.0	37.0	36.9	36.6	36.0	35.4	34.2	27.0
DDT		–	–	–	–	–	–	–	28.9

in the DCT-domain, the proposed algorithm reduces the total number of arithmetic operations involved in the scaling, thus reducing the inherent degradation influence of round-off and truncation errors.

The second observation that is worth noting about the HDT algorithm concerns the expected decrease of the PSNR measures, when the number of discarded coefficients increases. Although such decrease may be negligible for greater scaling factors, its importance is highly significant for smaller scalings of the original sequence.

Finally, a careful observation should be devoted to the comparison of the performances obtained with the proposed algorithm and with the DCT decimation approach (DDT). As it was previously predicted, although both algorithms provide quite similar quality performances for scaling factors given by integer powers of 2, the same does not happen when other scaling factors are considered. In such cases,

the results obtained with the proposed HDT approach prove that this algorithm provides significantly better performances than the DDT algorithm. Moreover, as it will be observed in subsection E, these better performance results may even be obtained with fewer arithmetic operations. As a consequence, more decoded DCT coefficients may be processed with the proposed algorithm than with the DCT decimation approach, thus potentially providing much better quality results.

C - Drift

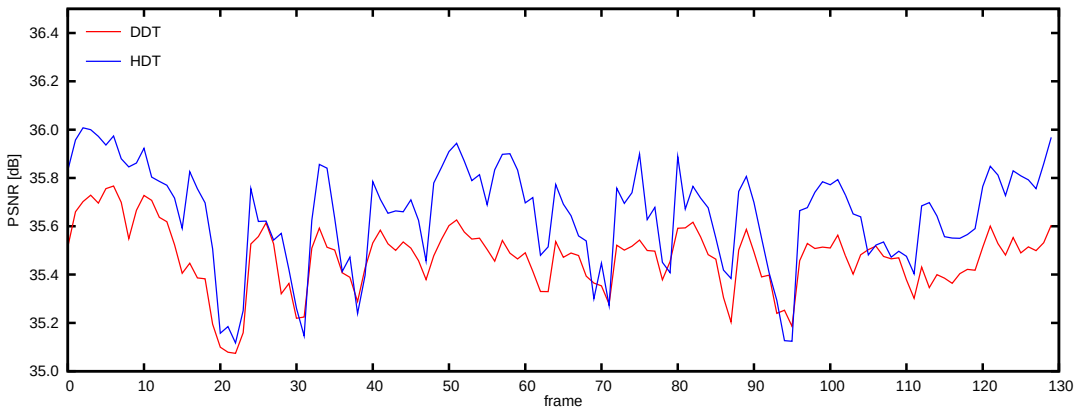
After a first evaluation of the static video quality provided by the considered algorithms, a thorough assessment of their performances when processing video sequences that apply the traditional temporal prediction mechanisms was carried out. Such evaluation was conducted by downscaling encoded video sequences with CIF resolution (352×288 pixels) and GOPs composed by 8 frames, considering both the proposed hybrid approach (HDT) and the DCT decimation transcoding algorithm (DDT). To obtain comparison results as fair as possible, both approaches used the same amount of decoded DCT coefficients: $K_{\mathrm{HDT}} = K_{\mathrm{DDT}} = \lceil N/\mathcal{S}_{\mathcal{F}} \rceil$.

In figs. 6.19 and 6.20 it is presented the variation of the PSNR measure obtained from the downscaling of the first 130 frames of the *Akiyo* and *Mobile & Calendar* video sequences, respectively. These operations considered the scaling factors $\mathcal{S}_{\mathcal{F}} = 3$ and $\mathcal{S}_{\mathcal{F}} = 5$, and a quantization parameter $Q = 4$. These two video sequences have distinct content characteristics: while the *Akiyo* sequence is characterized by a reduced amount of spatial and motion activity, the *Mobile & Calendar* video sequence has a significant amount of spatial detail and movement. From the obtained results it can be observed that the proposed hybrid algorithm (HDT) consistently provides better quality levels than the DCT decimation approach (DDT), thus confirming the conclusions that were previously driven from their static behavior.

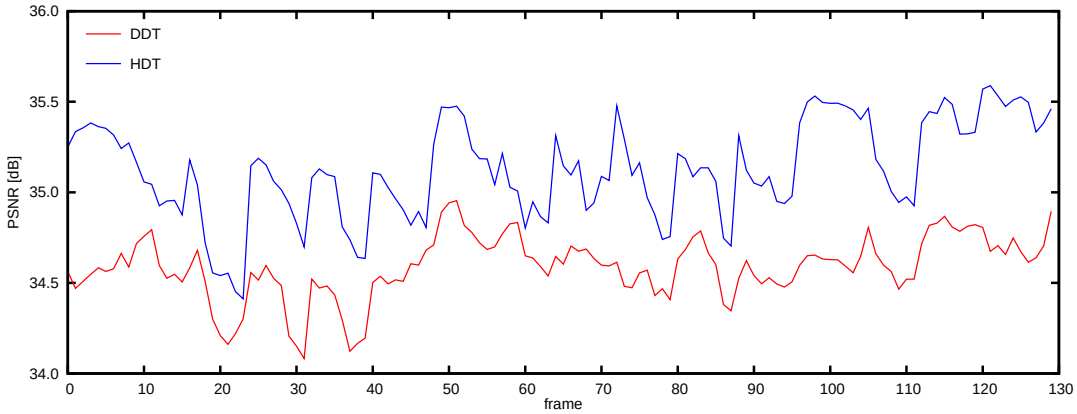
In table 6.6 it is represented the average PSNR gain provided by the proposed HDT approach over the DCT decimation scheme, for several other different video sequences and scaling factors ($\mathcal{S}_{\mathcal{F}}$). Such gain was evaluated by computing the average of the corresponding PSNR difference, for a time period corresponding to 300 frames. Once again, the obtained values demonstrate that while for scaling factors given by integer powers of 2 the two considered approaches provide quite similar quality levels (with a slight advantage for the DDT scheme), for scaling factors other than integer powers of 2 the proposed HDT algorithm can provide significantly better quality performances. In particular, the results that were obtained with the *Silent-Voice* video sequence revealed a notable advantage of the proposed scheme when processing this sequence. Such advantage comes as a result of the significant

amount of spatial detail in the background of this sequence, which is particularly affected by the degradation effect introduced by the post-processing discarding step of the higher frequency DCT coefficients, inherent to the DCT decimation approach. Hence, these results fully comply with the previously presented static video quality behavior.

Moreover, the charts presented in figs. 6.19 and 6.20 also evidence that the effect of the inherent drift on the proposed scheme is not significantly different from the one observed for the DCT decimation approach. In fact, by adopting this closed-loop transcoding architecture (see fig. 6.17) to evaluate the proposed hybrid downscaling algorithm, a new reduced resolution prediction residual signal is computed in the encoder loop. As a consequence, a realignment of the predictive and residual components is provided, leading to a minimization of the involved drift [114]. Such drift mainly arises from re-quantization, elimination of some non-zero DCT



(a) *Akiyo*, $\mathcal{S}_{\mathcal{F}} = 3$.

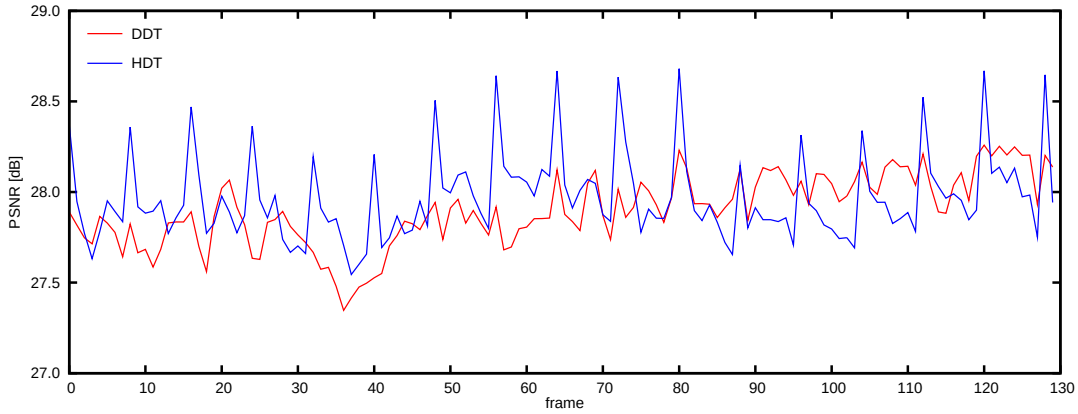


(b) *Akiyo*, $\mathcal{S}_{\mathcal{F}} = 5$.

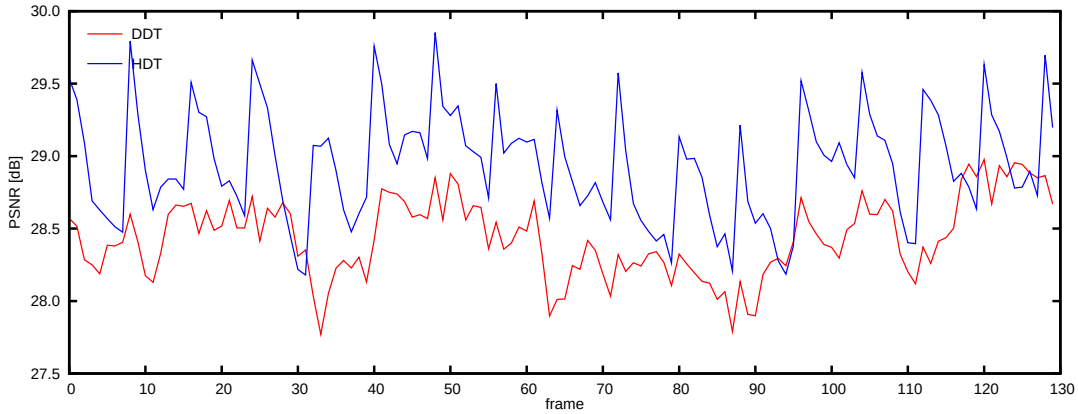
Figure 6.19: PSNR measure obtained by downscaling the *Akiyo* video sequence, considering $Q = 4$ and GOP $= 8$ frames.

coefficients and arithmetic fixed precision errors caused by integer truncation, which will degrade the reference picture used in the temporal prediction mechanism.

To compensate for this gradual degradation along the scaling process, Yin et al. [114] proposed four drift compensation architectures based on a drift error analysis, that attempt to reduce the influence of such degradation. Although some of such proposals are mainly targeted to be applied in open-loop downscaling architectures (which are naturally more prone to the influence of this degradation effect), some of the presented approaches could equally be applied to the closed-loop transcoding architecture that is considered in this section (e.g. *Intra_Refresh*). However, since the main scope of the research presented in this section is not the actual video transcoding architecture, but is the proposal of a computational efficient and more accurate resizing algorithm for any arbitrary integer scaling factor, such compensation architectures were not considered. In fact, the proposed downscaling algorithm could



(a) *Mobile & Calendar*, $\mathcal{S}_{\mathcal{F}} = 3$.



(b) *Mobile & Calendar*, $\mathcal{S}_{\mathcal{F}} = 5$.

Figure 6.20: PSNR measure obtained by downscaling the *Mobile & Calendar* video sequence, considering $Q = 4$ and GOP $= 8$ frames.

Table 6.6: Video quality (PSNR) gains provided by the proposed HDT algorithm over the DDT approach, for different scaling factors ($\mathcal{S}_\mathcal{F}$) and considering the same number of DCT coefficients: $K_{\text{HDT}} = K_{\text{DDT}} = \lceil N/\mathcal{S}_\mathcal{F} \rceil$.

Video Sequence	**ΔPSNR** [dB]						
	$\mathcal{S}_\mathcal{F} = 2$	$\mathcal{S}_\mathcal{F} = 3$	$\mathcal{S}_\mathcal{F} = 4$	$\mathcal{S}_\mathcal{F} = 5$	$\mathcal{S}_\mathcal{F} = 6$	$\mathcal{S}_\mathcal{F} = 7$	$\mathcal{S}_\mathcal{F} = 8$
Akiyo	−0.28	+0.19	−0.34	+0.51	+0.19	+3.68	−0.03
Silent	−0.09	+4.29	−0.54	+8.35	+4.58	+4.17	−0.22
Carphone	−0.23	−0.11	−0.28	+0.25	+1.19	+3.81	−0.10
Table-tennis	−0.15	+0.34	−0.32	+1.03	+1.24	+2.32	−0.01
Mobile	−0.61	−0.06	−0.36	+0.33	+1.35	+2.24	−0.10

equally be implemented in the downsample conversion modules of all architectures proposed in [114].

D - Bit rate

In table 6.7 it is represented the average bit rate gain provided by the proposed HDT approach over the DCT decimation scheme for all the considered video sequences and scaling factors ($\mathcal{S}_\mathcal{F}$), where:

$$\Delta\text{Bit Rate}[\%] = 100 \times \frac{bits(\text{HDT}) - bits(\text{DDT})}{bits(\text{DDT})}. \tag{6.2}$$

As before, such gain was evaluated by averaging the differences between the amount of bits required to encode each frame with the two considered algorithms over a time period corresponding to 300 frames, considering $Q = 4$ and $K_{\text{HDT}} = K_{\text{DDT}} = \lceil N/\mathcal{S}_\mathcal{F} \rceil$.

The obtained results evidence a clear advantage of the proposed algorithm over the DDT approach, since it requires fewer bits (up to 15% less) to encode each frame

Table 6.7: Bit rate gains provided by the proposed HDT algorithm over the DDT approach, for different scaling factors ($\mathcal{S}_\mathcal{F}$) and considering the same number of DCT coefficients: $K_{\text{HDT}} = K_{\text{DDT}} = \lceil N/\mathcal{S}_\mathcal{F} \rceil$.

Video Sequence	**ΔBit Rate [%]**						
	$\mathcal{S}_\mathcal{F} = 2$	$\mathcal{S}_\mathcal{F} = 3$	$\mathcal{S}_\mathcal{F} = 4$	$\mathcal{S}_\mathcal{F} = 5$	$\mathcal{S}_\mathcal{F} = 6$	$\mathcal{S}_\mathcal{F} = 7$	$\mathcal{S}_\mathcal{F} = 8$
Akiyo	−5.85	−7.05	−2.44	+1.70	−0.63	+5.40	+0.84
Silent	−7.70	−10.67	−3.84	−4.05	−5.05	+0.17	−0.80
Carphone	−8.67	−14.13	−4.55	−8.10	−3.70	−1.17	+2.85
Table-tennis	−9.50	−13.73	−4.03	−5.14	−4.20	+0.62	−2.73
Mobile	−12.30	−21.46	−7.68	−14.79	−7.68	−2.77	+1.18

of the considered video sequences. Such advantage is mainly due to the use of more accurate reduced-resolution reference frames in the encoder loop, which provide the implementation of a much better temporal prediction mechanism, thus resulting in smaller residuals. In fact, the observed advantage is more significative in video sequences that present greater amounts of movement, such as the *Carphone*, the *Table-Tennis* and the *Mobile & Calendar*, where such prediction scheme influences the efficiency of the video encoder the most.

E - Pre-filtering parameterization

The low-pass pre-filtering stage that is often performed in the inverse DCT processing step, described in subsection B of section 5.2.2 (page 174), has been widely adopted by several authors to reduce the computational cost required by several transcoding algorithms that are performed directly in the DCT-domain. In this particular application context, such pre-filtering usually relates the minimum set of DCT coefficients that should be considered for a given scaling factor ($\mathcal{S}_{\mathcal{F}}$), in order to balance a certain trade-off of the computational cost and avoid the introduction of noticeable degradation in the output video sequence.

In the following, two entirely different pre-filtering approaches will be proposed, in order to achieve quite distinct compromises: the minimization of the computational cost and the maximization of the output video quality. Such trade-offs were experimentally assessed and a new empirical cost function has been formulated.

Balancing the computational cost

By analyzing the results previously presented in table 6.4(b) and in table 6.5, it can be observed that the proposed HDT downscaling algorithm is able to provide significantly better performances than the DCT decimation approach (DDT) with fewer operations. Consequently, for downscaling operations implemented in restricted computational environments, where the available amount of arithmetic operations that can be performed to process each pixel in real-time is limited, the proposed hybrid algorithm offers the possibility to process more decoded DCT coefficients than the DCT decimation algorithm, thus potentially providing better quality results. Table 6.8 illustrates such situation. For each scaling factor ($\mathcal{S}_{\mathcal{F}}$), it is presented the number of DCT coefficients that is considered by the DCT decimation algorithm (DDT), as well as the number of coefficients that may be processed by the proposed hybrid algorithm (HDT), when both approaches roughly make use of the same number of operations. For each of these experimental setups, it is also presented the corresponding PSNR gain, provided by the proposed HDT approach.

Table 6.8: PSNR gains provided by the proposed approach over the DDT algorithm, when the number of considered DCT coefficients (K) is adjusted so that both schemes make use of the same computational resources.

$\mathcal{S}_\mathcal{F}$	2	3	4	5	6	7	8	9
K_{DDT}	4	3	2	2	2	2	1	1
K_{HDT}	5	5	3	5	6	8	2	2
ΔPSNR	−0.1 dB	+7.7 dB	−0.1 dB	+7.3 dB	+6.6 dB	+8.1 dB	+0.0 dB	+5.3 dB

As it can be observed, while for scaling factors given by integer powers of 2 the performances of these two algorithms are quite similar (with a slight advantage for the DDT algorithm), for scaling factors other than integer powers of 2 and under similar computational constraints, the proposed HDT algorithm is capable of providing much better quality results than the DCT decimation approach.

Balancing the output video quality

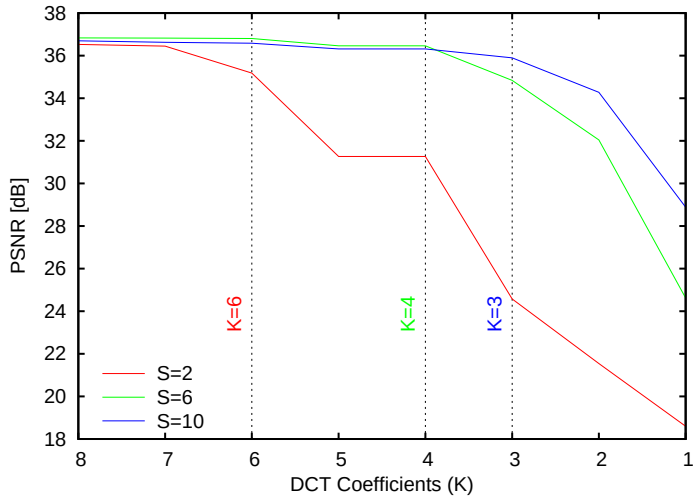
An alternative perspective to evaluate the proposed algorithm can also be considered by observing that, besides the significant computational efficiency and the higher video quality that is obtained with the proposed approach, neither the obtained PSNR measure significantly degrades nor the resulting bit rate considerably changes when only a moderate number of high order AC frequency DCT coefficients is discarded.

In figs. 6.21(a) and 6.21(b) it is represented the variation of the PSNR measure and of the amount of bits required to encode each frame with the adopted maximum DCT coefficient index (K), respectively, for three different scaling factors: $\mathcal{S}_\mathcal{F} = 2,\ 6$ and 10. From such plots, it can be observed that both the obtained distortion level and the variation on the output bit rate are negligible, provided that the maximum index of the considered DCT coefficients (K) does not become smaller than a certain level (K^{min}). Consequently, it can be experimentally stated that the best performance results are obtained when such index (K) lies within a certain interval $K \in [K^{min}, N]$. By considering the set of results that were experimentally obtained in this research, it was empirically estimated a possible expression to compute K^{min} for a given scaling factor $\mathcal{S}_\mathcal{F}$:

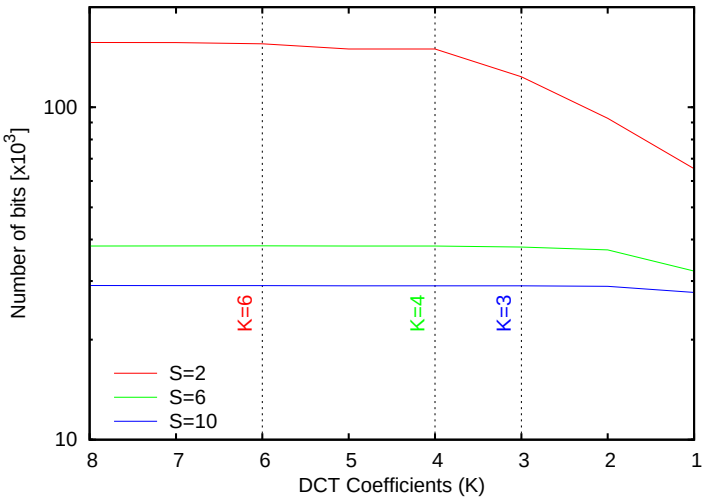
$$K^{min} = \left\lceil \sqrt{N^2/\mathcal{S}_\mathcal{F}} \right\rceil \tag{6.3}$$

This threshold, whose application was verified for several different quantization parameters (Q), is somewhat more conservative than the minimum index value proposed by Lee et al. [48] ($K^{min}_{Lee} = \lceil N/\mathcal{S}_\mathcal{F} \rceil$), giving rise to better quality results.

Table 6.9 presents the values corresponding to the adopted figures of merit when the usage of the DCT-domain pre-filtering stage is considered in the proposed HDT algorithm for reduction of the computational cost, with $K_{\mathrm{HDT}} = K^{min} = \left\lceil \sqrt{N^2/\mathcal{S}_{\mathcal{F}}} \right\rceil$. Such values were obtained by comparing the proposed algorithm with the DCT decimation approach (DDT). From the presented results, it can be concluded that the proposed experimental estimation of the optimal pre-filtering setup



(a) Variation of the obtained video quality (PSNR) with the maximum index (K) of the considered DCT coefficients.



(b) Variation of the bit rate with the maximum index (K) of the considered DCT coefficients.

Figure 6.21: Experimental estimation of the optimal number of DCT coefficients considered by the proposed space scaling algorithm for different scaling factors: $\mathcal{S}_{\mathcal{F}} \in \{2, 6, 10\}$ (CIF *Mobile & Calendar* video sequence, $Q = 4$).

Table 6.9: Considered figures of merit when the usage of the DCT-domain pre-filtering stage is considered in the proposed algorithm for reduction of the computational cost, with $K_{\mathrm{HDT}} = \left\lceil \sqrt{\frac{N^2}{\mathcal{S}_{\mathcal{F}}}} \right\rceil$.

$\mathcal{S}_{\mathcal{F}}$	2	3	4	5	6	7	8	9	10
K_{DDT}	4	3	2	2	2	2	1	1	1
K_{HDT}	6	5	4	4	4	4	3	3	3
ΔPSNR [dB]	+3.75	+7.73	+3.37	+6.23	+6.23	+6.85	−0.03	+6.50	+6.69
ΔBit-Rate [%]	−4.75	−3.30	−2.83	+2.52	−1.69	+6.20	+1.59	−0.35	+3.86

(K^{min}) leads to highly efficient transcoding architectures, which still provide good performance characteristics in what concerns the obtained video quality and the resulting amount of coded data.

6.3.2 Block-based motion re-estimation in the DCT-domain[§]

In section 3.3.5 it was presented a brief overview of some previously proposed approaches to implement motion estimation procedures in the compressed DCT-domain. However, as it was also stated, the characteristics of these algorithms do not always comply with the requisites of the target transcoding system. Either because they: *i*) involve a significant computational cost, such as the *cascaded transcoders*, proposed by Chen et al. [13], Youn et al. [115]; or *ii*) provide accuracy levels that do not make them suitable enough for current video transcoding applications, such as the *convolution based transcoders* proposed by Reeves and Kubik [84]; or *iii*) make use of other kinds of the discrete sine and cosine transforms of the blocks under processing, that cannot be directly obtained from the decoded DCT-H.26x/MPEG-x video stream and have to be computed for an image area that is larger than the usually adopted ($N \times N$) block structure, such as the *DCT pseudo-phases based transcoder*, proposed by Koc and Liu [41].

Consequently, in section 5.3 it was proposed an alternative motion re-estimation algorithm that may be implemented in the compressed DCT-domain. Such algorithm provides an alternative and efficient way to refine the MVs required by the motion compensated prediction mechanism. This scheme makes use of the DCT co-

[§]Some portions of this section appeared in:

[89] - N. Roma and L. Sousa, "Least squares motion estimation algorithm in the compressed DCT domain for H.26x/MPEG-x video sequences," in *Proceedings of the IEEE International Conference on Advanced Video and Signal-Based Surveillance (AVSS)*. Como - Italy: IEEE, Sep. 2005, pp. 576–581.

efficients directly obtained from the DCT-H.26x/MPEG-x video stream and is based on the application of an iterative scheme that estimates the new MVs by applying a Least Squares Estimation (LSE) technique.

To assess the performance of the proposed Transform-Domain Least Squares Motion Estimation (TD-LSME) algorithm, a simpler version of an H.263 video transcoding system for DCT-domain video downscaling, as described in section 5.2, was implemented. This transcoder performs a $\frac{1}{2} \times \frac{1}{2}$ spatial downscaling operation, as described by Chang and Messerschmitt [11], by representing the image area corresponding to each set of four luminance blocks $\mathbf{b}_{i,j}$ (with $i, j \in \{0, 1\}$) that compose each macroblock of the original frame by a single block $\hat{\mathbf{b}}$ in the scaled frame, as shown on fig. 6.22.

The block diagram of the implemented transcoding system is presented in fig. 6.23. The computational cost of this architecture was significantly reduced by restricting the search range of the motion re-estimation procedure. To achieve such objective, the information related to the MVs that are decoded from the input video sequence is used to compute a first prediction of the desired MVs. Hence, in order to obtain an initial estimate of the MV of each scaled macroblock, Motion Vector Compositing and Scaling (MVCS) techniques, such as those described in section 3.3.4, are applied to the decoded MVs. Among the several different approaches that have been proposed and considered, it was adopted the Simple Motion Estimation Scaling (SMES) algorithm proposed by Liang et al. [52]. This algorithm was already presented in subsection B of section 5.4.2 (page 197) and is based on a spatial activity-area weighted average scheme, previously described in subsection C of section 3.3.4 (page 102). This algorithm makes use of the number of non-null AC frequency coefficients of each DCT encoded block, to estimate the spatial activity of the original macroblocks. Such spatial activity is then used in the computation

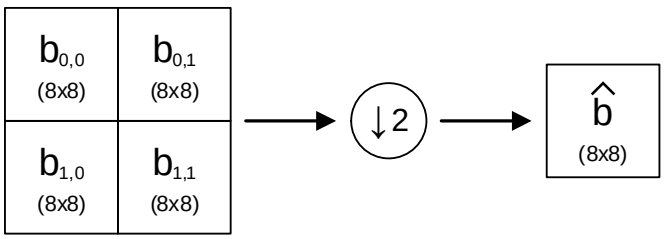


Figure 6.22: Downsampling four adjacent blocks to obtain a single (8×8) pixels block.

of the composited motion vector $\tilde{v}$:

$$\tilde{v} = \frac{\sum_{i=0}^{I} \sum_{j=0}^{J} v_{i,j} \cdot \alpha_{i,j} \cdot A_{i,j}}{\sum_{i=0}^{I} \sum_{j=0}^{J} \alpha_{i,j} \cdot A_{i,j}} \tag{6.4}$$

In this equation, $\alpha_{i,j}$ is the spatial activity of macroblock (i, j), measured by the number of non-null AC frequency coefficients; and $A_{i,j}$ is a measure of the involved area, evaluated in terms of the number of pixels of the original macroblock that is considered in the composition of the transcoded scaled macroblock. For the considered case, it was assumed that $I = J = 1$ and $A_{i,j} = N^2$, $\forall_{i,j}$.

The adopted scaling factor ($\mathcal{S}_{\mathcal{F}} = 2$) is then applied to the composited motion vectors $\tilde{v}$:

$$\hat{v} = \frac{1}{\mathcal{S}_{\mathcal{F}}} \cdot \tilde{v} \tag{6.5}$$

To optimize the performance of the refinement stage that follows, sub-pixel precision levels may be adopted in this MV downscaler block. Hence, the outcome of this processing module will be a first coarse estimate ($\hat{v}$) of the MV corresponding to each macroblock of the downscaled frame. These MVs, as well as the scaled versions of the current and reference frames resulting from the application of the spatial downscaling algorithm, are then applied to the proposed TD-LSME algorithm (fig. 5.5) to obtain an accurate refinement of the desired MV.

Considering that this motion re-estimation algorithm is based on an iterative scheme, it may be advantageous to impose a limit on the number of iteration steps that are performed, in order to keep the involved computational cost under control. In the considered transcoding system, this control of the refinement algorithm

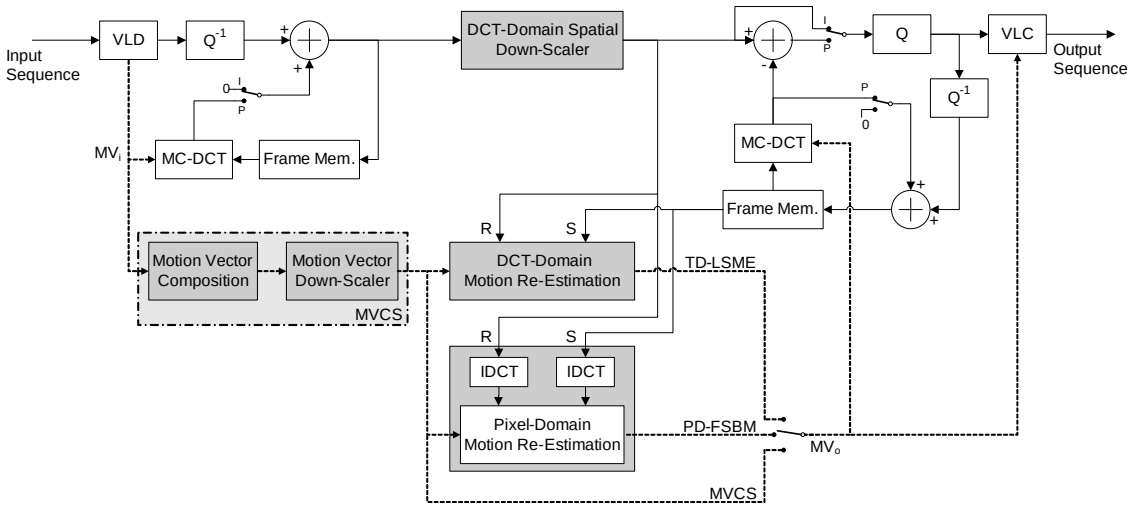


Figure 6.23: Block diagram of the DCT-domain video downscaler with MV re-estimation in the DCT-domain.

was implemented by limiting its operation to a maximum of 3 iterations towards convergence to the final MV, unless an additional constraint, corresponding to the adopted stopping condition, is met: $\|v^i - v^{i-1}\| < \delta$, where $\delta = 0.1$ and v^n is the MV estimate obtained after iteration n. Hence, the implemented stopping condition can be formulated as follows:

$$\text{Stop the TD-LSME if: } \begin{cases} \#\text{iteration} > \text{MaxIter} & \text{, with MaxIter} = 3; \\ \|v^i - v^{i-1}\| < \delta & \text{, with } \delta = 0.1. \end{cases} \tag{6.6}$$

To evaluate the performance of the implemented system, the transcoding architecture presented in fig. 6.23 considers and compares three different motion re-estimation schemes:

MVCS - the MV predictors that are obtained at the output of the Motion Vector Compositing and Scaling (MVCS) module are directly applied by the motion compensation block of the encoder side of the transcoder, thus discarding any refinement step;

TD-LSME - the proposed Transform-Domain Least Squares Motion Estimation (TD-LSME) iterative algorithm is applied to the MV predictors obtained at the output of the MVCS module;

PD-FSBM - the MV predictors are used to obtain the search center of an optimal Pixel-Domain Full-Search Block-Matching (PD-FSBM) motion estimation algorithm, in order to obtain a reference solution entirely similar to the one that is obtained with traditional cascaded pixel-domain approaches.

Hence, the experimental results obtained with the MVCS configuration will provide the means to evaluate the performance of the proposed algorithm, when compared with quite common and simpler approaches that do not apply any refinement of the MVs obtained from the composition and downscaler blocks [12, 52, 108]. On the other hand, the set of MVs obtained with the PD-FSBM configuration will provide a comparison of the refined vectors with the MVs that would be obtained by a traditional cascaded pixel-domain approach, based on the minimization of a given disparity measure (in this case, the SAD, defined in eq. 3.116).

The evaluation of the considered structures was conducted by applying the transform-domain downscaler transcoder architecture presented in fig. 6.23, to convert a set of CIF video sequences into the QCIF format, namely, the *Carphone*, the *Table-Tennis*, the *Stefan*, the *Mobile & Calendar* and the *Football*. To better assess

the motion re-estimation capabilities offered by each of the considered algorithms, the selection of this set of test video sequences took into account their particular characteristics, both in terms of the presented spatial details and amount of movement. The conducted experiments were carried out by considering a search range for the motion refinement procedure of ± 3 pixels and half-pixel precision to the output re-estimated MVs. To assess the influence of the degradation effect, inherent to the adopted quantization setup, on the quality of the temporal prediction resulting from the computed MVs, three different quantization setups ($Q \in \{4, 8, 12\}$) were considered in these experiments, by registering the obtained gains, both in terms of the output video quality (PSNR) and bit rate. Furthermore, to emphasize the direct influence of the considered motion re-estimation schemes on the output bit rate, the output buffer controller of the encoding system was disabled, so that the obtained differences of the amount of bits required to encode each frame can be recognized as a direct consequence of the considered motion re-estimation modules. To conclude, a brief discussion about the involved computational cost will also be provided.

A - Motion vectors

The ultimate output of any ME algorithm consists of the set of computed MVs that are to be used by the motion-compensated temporal prediction mechanism. Hence, to illustrate the operation of the different ME algorithms that were considered, in fig. 6.24 it is presented the superposition, in the corresponding pixels area, of the obtained MV fields, computed along the encoding procedure of a P-type frame of the *Carphone* QCIF video sequence. To enhance the MV fields readability, the amplitude of the represented MVs was enlarged, using a scale factor of 16 pixels.

The first observation that should be retained from fig. 6.24 concerns the disparity of the MVs obtained, for certain MBs, with the different considered approaches. Such differences, in particular those that concern the MVs obtained with the simple MVCS approach, were already expected and are the main motivation for the application of the refinement procedure described in this section. In fact, considering that each MV obtained with the MVCS scheme is just a first and coarse estimate of the vector that best approximates the global motion of a given pixels area of the input frame, it should not be a surprise to see that a better and more accurate estimate can be computed with the optimal and exhaustive PD-FSBM approach.

On the other hand, the observed differences between the MVs computed with the PD-FSBM and the TD-LSME algorithms deserve further attention. In fact, while the MVs obtained with the PD-FSBM scheme represent the closest approximation to the optimal temporal-predictors computed in the pixel-domain, using the SAD

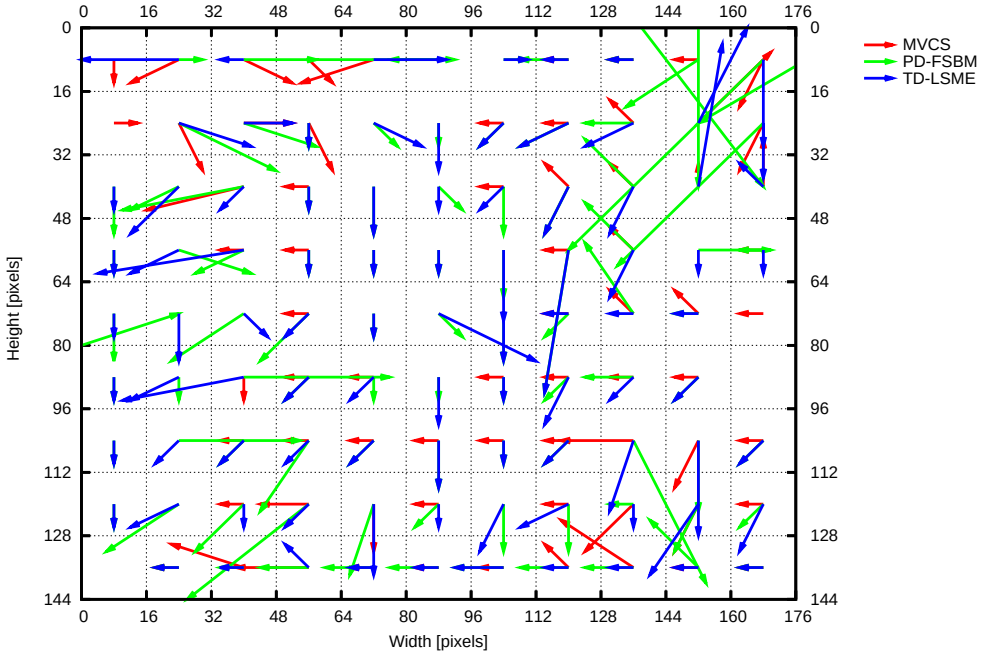


Figure 6.24: Superposition of the MV fields obtained with the considered ME algorithms with the corresponding pixels area of the output downscaled frame. To enhance the plots readability, the amplitude of the MVs was enlarged by a factor of 16.

criteria, the MVs obtained with the TD-LSME algorithm represent the solution of the same estimation problem using a somewhat different criteria: the minimization of a squared error signal, that is entirely computed in the compressed DCT-domain. Furthermore, considering that this iterative scheme was implemented using the stop condition presented in eq. 6.6, the obtained solution should not be expected to be the optimal MV whenever the iterative procedure is prematurely stopped. Consequently, as it can be observed in certain MBs, the obtained MVs may present quite different amplitude values or even point to distinct directions. In the following subsections, it is presented the main consequences of adopting these distinct solutions provided by the considered schemes, in what concerns the obtained video quality (PSNR) and bit rate.

B - Video quality

In table 6.10 it is presented the average PSNR measures that were obtained with the considered algorithms. To best evaluate the advantages of performing an additional motion re-estimation step using the composited and scaled MVs, the PSNR results obtained with the two considered motion re-estimation approaches

Table 6.10: Average PSNR measures obtained with the considered motion re-estimation approaches.

Video Sequence	Number of Frames	Q	PSNR	$\Delta\text{PSNR}^{\text{MVCS}}$	
			MVCS	PD-FSBM	TD-LSME
Carphone	300	4	36.90 dB	+0.25 dB	+0.12 dB
		8	32.96 dB	+0.35 dB	+0.16 dB
		12	30.80 dB	+0.46 dB	+0.22 dB
Table-Tennis	300	4	36.22 dB	+0.04 dB	−0.07 dB
		8	32.44 dB	+0.20 dB	+0.00 dB
		12	30.21 dB	+0.29 dB	+0.07 dB
Stefan	300	4	35.04 dB	−0.20 dB	−0.19 dB
		8	29.95 dB	−0.01 dB	−0.04 dB
		12	27.03 dB	+0.17 dB	+0.08 dB
Mobile	300	4	33.85 dB	+0.06 dB	+0.05 dB
		8	28.54 dB	+0.10 dB	+0.11 dB
		12	25.70 dB	+0.14 dB	+0.13 dB
Football	90	4	36.13 dB	−0.14 dB	−0.23 dB
		8	32.43 dB	−0.05 dB	−0.23 dB
		12	30.44 dB	+0.01 dB	−0.21 dB

(TD-LSME and PD-FSBM) were presented in terms of the PSNR difference, when compared with the simple MVCS approach ($\Delta\text{PSNR}^{\text{MVCS}}$).

$$\Delta\text{PSNR}^{\text{MVCS}}(\textit{algorithm}) = \text{PSNR}(\textit{algorithm}) - \text{PSNR}(\text{MVCS}) \tag{6.7}$$

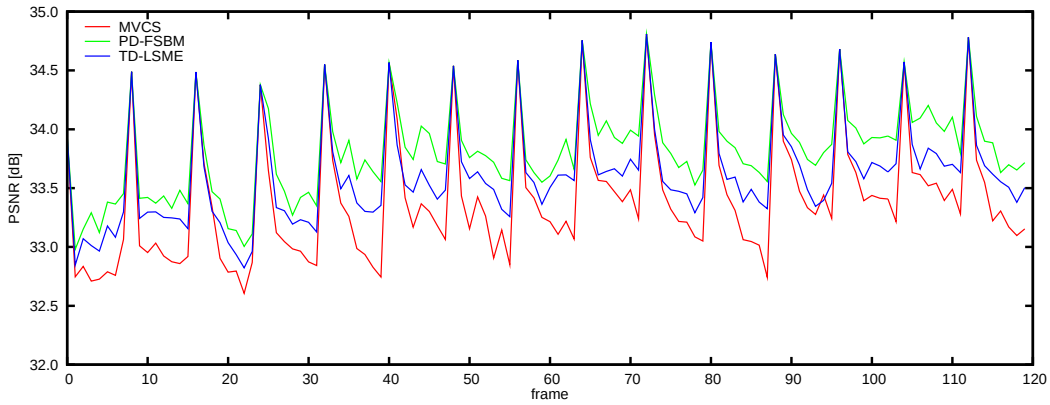
The first observation that is worth noting is concerned with the similar PSNR measures that were obtained with the two refinement schemes for each of the considered video sequences and quantization setups, with both positive and negative variations with magnitude less than 0.5 dB. Such small variation range was already expected and is a direct consequence of the inhibition of the output buffer controller (as referred before), since all prediction differences could be encoded with the maximum possible resolution for the considered quantization steps. In fact, in a hypothetical ideal and lossless configuration, the PSNR of the output video sequences obtained with each of the considered re-estimation algorithms would be exactly the

same, independently of the MVs considered in the motion-compensation step, since the resulting (and possible different) prediction residuals would be encoded at the maximum precision.

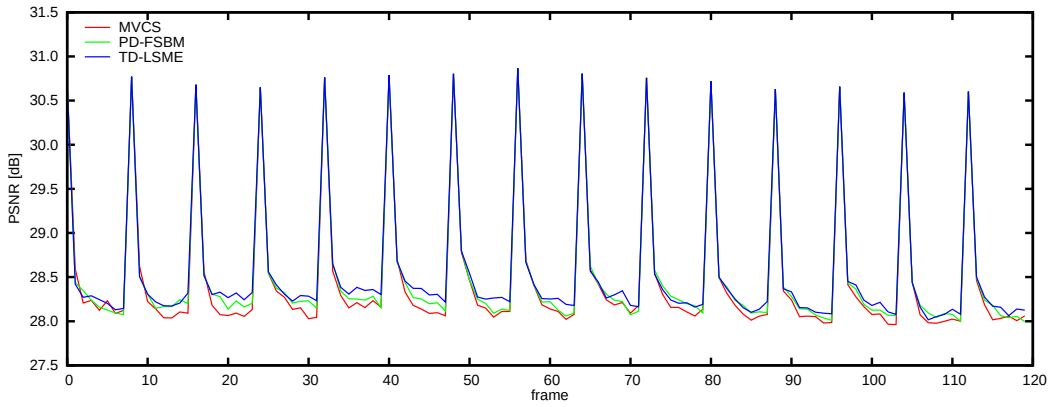
Another interesting observation is concerned with the negative PSNR gains that were obtained for certain video sequences (mainly, in *Stefan* and *Football*). According to what was stated in the previous paragraph, such degradation should be seen as an indirect consequence of the new refined MVs, obtained from the motion re-estimation step. In fact, such distinct set of MVs led to different motion compensated residual signals that were more affected by the quantization and arithmetic round-off errors than those that were obtained by skipping the whole re-estimation step and by directly using the MVs obtained from the MVCS block. Naturally, those video sequences with more motion activity and spatial detail are more prone to these degradation effects, since their DCT encoded blocks present more non-null DCT coefficients that may be either canceled or distorted by the quantization procedure.

In addition, it is also worth noting the fact that even the optimal and exhaustive PD-FSBM algorithm may lead, under these circumstances, to worse results than the simple MVCS approach. Again, such fact is owed to the several arithmetic and round-off errors that are introduced by the IDCT processing steps, during the conversion of the blocks corresponding to both the reference and the search area into the pixel-domain (see fig. 6.23). In fact, the consequences of such degradation were already studied in section 3.1.2 and represent the main motivations to develop and implement video processing algorithms directly in the transform-domain. In this particular situation, such degradation results in an error signal that will be added to the pixel values and will deteriorate the block-matching search procedure.

Finally, in fig. 6.25 it is illustrated the variation of the PSNR measure for two considered video sequences, using a quantization setup with $Q = 8$. Entirely similar results were obtained for the remaining video sequences and quantization setups. As it can be observed, the resulting quality of the video sequences obtained with the three considered motion re-estimation schemes is quite similar. For these particular video sequences and quantization setups, the usage of the TD-LSME and PD-FSBM re-estimation schemes led to better quality performances than those obtained with the simpler MVCS scheme. Nevertheless, while the PD-FSBM scheme presented better quality performances in the *Carphone* video sequence, it was observed that the TD-LSME algorithm performed better in the *Mobile* & *Calendar* video sequence.



(a) *Carphone* video sequence.



(b) *Mobile & Calendar* video sequence.

Figure 6.25: Obtained PSNR level for the video sequences *Carphone* and *Mobile & Calendar*, using $Q = 8$.

C - Bit rate

In table 6.11 it is presented the average bit rate performances that were obtained with the considered algorithms. As before, to better emphasize the advantages of performing the MV refinement of the composited and scaled MVs, the bit rate results obtained with the two considered motion re-estimation approaches (TD-LSME and PD-FSBM) are presented in terms of the bit rate gain, when compared with the simple MVCS approach:

$$\Delta\text{Bit-Rate}^{\text{MVCS}}(\textit{algorithm}) = 100 \times \frac{\text{Bit-Rate}(\textit{algorithm}) - \text{Bit-Rate}(\text{MVCS})}{\text{Bit-Rate}(\text{MVCS})} \quad (6.8)$$

As it was referred before, to better illustrate the direct influence of the considered motion re-estimation schemes on the output bit rate, a variable bit rate configuration was adopted at the encoder side of the transcoder structure, by disabling the output buffer controller. Consequently, the obtained differences of the amount of

Table 6.11: Bit rate measures obtained with the considered motion re-estimation approaches.

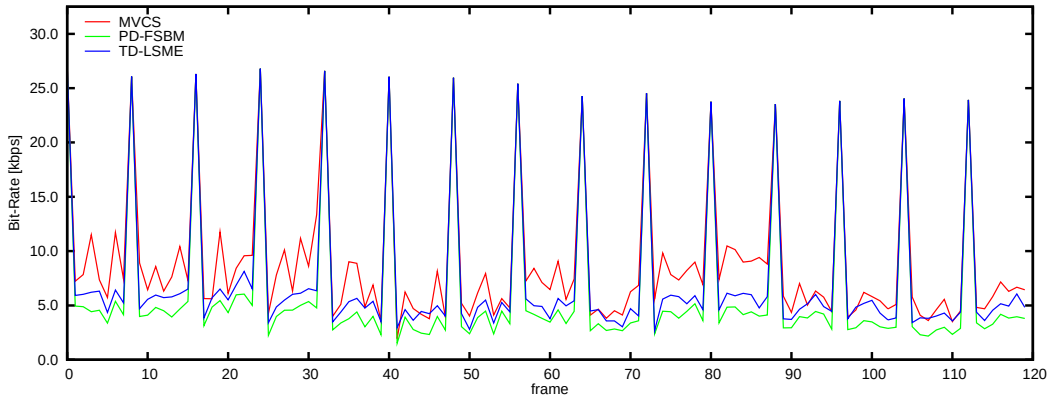
Video Sequence	Number of Frames	Q	Bit Rate	ΔBit-Rate^{MVCS}	
			MVCS	PD-FSBM	TD-LSME
Carphone	300	4	23.2 kbps	−23.9 %	−13.5 %
		8	10.8 kbps	−24.9 %	−13.9 %
		12	6.6 kbps	−23.3 %	−13.6 %
Table-Tennis	300	4	24.3 kbps	−22.4 %	−14.0 %
		8	12.2 kbps	−26.2 %	−15.9 %
		12	8.1 kbps	−26.0 %	−15.1 %
Stefan	300	4	82.8 kbps	−26.8 %	−19.3 %
		8	42.4 kbps	−29.7 %	−21.4 %
		12	26.3 kbps	−30.9 %	−21.9 %
Mobile	300	4	75.5 kbps	−5.3 %	−2.2 %
		8	36.1 kbps	−7.7 %	−4.0 %
		12	21.3 kbps	−9.3 %	−5.3 %
Football	90	4	43.1 kbps	−21.4 %	−12.5 %
		8	19.8 kbps	−22.9 %	−13.4 %
		12	12.0 kbps	−21.8 %	−13.1 %

bits required to encode each frame can be regarded as a direct result of the considered motion re-estimation module, thus minimizing the effect on the output video quality (as explained before).

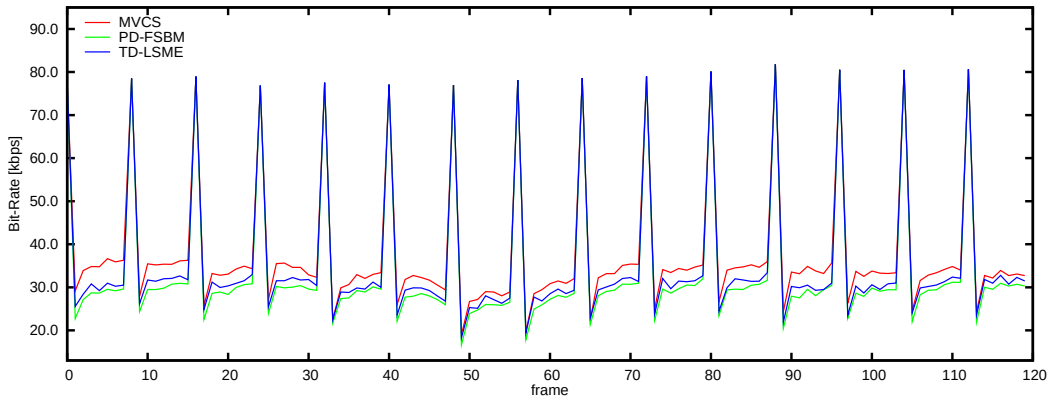
However, contrasting with the small variation range that was observed with the PSNR measures, the bit rate values that were obtained from the conducted experiments with the three different refinement approaches for each of the considered video sequences and quantization setups denote quite significant gains, provided by both the TD-LSME and the PD-FSBM schemes, when compared with the bit rate values obtained with the simpler MVCS approach. Consequently, considering that the prime purpose of any motion estimation algorithm is the computation of the best set of MVs that will provide the subsequent motion-compensated temporal prediction mechanism with the ability to encode a given sequence of images with the smallest amount of bits as possible, the observed bit rate gains, presented in ta-

ble 6.11, plenty justify the usage of the motion re-estimation block in the transcoding structure under consideration. In fact, it can be observed that all the conducted experiments proved effective advantages on performing such refinement step, with gains that range from 2 % up to 30 %.

Nevertheless, it was also observed that the PD-FSBM algorithm consistently provides greater bit rate gains than the proposed TD-LSME approach. In fact, despite the degradation effect resulting from the previously referred arithmetic and round-off errors, introduced by the IDCT computation blocks that are required by the PD-FSBM scheme, its exhaustive full-search strategy often provides a better solution than the one computed with the proposed TD-LSME approach, therefore leading to better performance results. The main reason for this fact comes as a direct consequence of a closer approximation of the minimization criteria adopted by the FSBM, based on the SAD disparity measure, to the differential encoding inherent to the adopted motion-compensated prediction mechanism. On the contrary, the



(a) *Carphone* video sequence.



(b) *Mobile & Calendar* video sequence.

Figure 6.26: Obtained bit rate for the video sequences *Carphone* and *Mobile & Calendar*, using $Q = 8$.

proposed scheme is based on an iterative minimization of a squared error signal, entirely computed in the compressed DCT-domain.

Even so, the results provided by the proposed TD-LSME scheme clearly show that it is possible to implement efficient motion estimation algorithms in the compressed DCT-domain, by directly using the DCT coefficients data structure obtained from the received video stream. However, such non-optimal approach still provides accuracy levels that are high enough to improve the efficiency of the video encoding data stream. However, contrary to other proposals, in order to proceed with the search procedure there is no need to implement the inverse DCT computation module to transpose the DCT coefficients blocks of both the current and the reference frames into the pixel-domain.

In fig. 6.26 it is illustrated the variation of the measured bit rate for two considered video sequences, using a quantization setup with $Q = 8$. Entirely similar results were obtained for the remaining video sequences and quantization setups. As it can be observed, the resulting bit rate corresponding to the video sequences obtained with the proposed TD-LSME scheme is slightly greater than the obtained with the PD-FSBM approach. Nevertheless, it is consistently smaller than the bit rate that would be obtained without the implementation of any motion re-estimation block, by simply considering the MVs obtained with the MVCS scheme.

D - Computational cost

A fair evaluation of the characteristics of the considered motion re-estimation algorithms in terms of the involved computational cost requires some preliminary considerations in terms of their computational characteristics. In fact, contrary to what was observed in the previous sections about other transcoding processing tasks, these two MV refinement schemes present quite distinct characteristics in terms of the most predominant arithmetic operations that are involved. Owing to the computation of the SAD similarity measure, the addition and subtraction arithmetic functions are the most prevailing operations in the implementation of the exhaustive FSBM pixel-domain ME algorithm (PD-FSBM). In contrast, multiplications are the most prevailing operations in the TD-LSME algorithm. As a consequence, to best characterize the computational cost of these transcoder architectures, it was adopted, as the main figure of merit, the sum of the number of *multiplications* and *additions/subtractions* operations that are required to process each pixel of the original frame. It is worth noting that although such approximation may seem to be somewhat unbalanced, since the multiplication operation is generally more complex than the addition and subtraction operations, most current general purpose proces-

Table 6.12: Average number of operations (*multiplications* and *additions/subtractions*) required to process each pixel of the original CIF format frame.

Video Sequence	Number of Frames	Q	Op-Count	ΔOp-Count$^{\text{MVCS}}$	
			MVCS	PD-FSBM	TD-LSME
Carphone	300	4	154.7	+64.6	+50.8
		8	152.1	+66.2	+52.3
		12	152.3	+66.0	+52.0
Table-Tennis	300	4	92.0	+105.6	+76.3
		8	100.7	+98.8	+68.5
		12	110.3	+90.9	+64.5
Stefan	300	4	168.3	+62.8	+56.7
		8	165.1	+64.0	+57.2
		12	163.6	+64.4	+57.2
Mobile	300	4	159.1	+58.2	+42.5
		8	152.0	+61.6	+43.0
		12	147.3	+64.1	+44.0
Football	90	4	181.9	+59.9	+58.1
		8	177.3	+61.5	+55.5
		12	175.4	+61.9	+52.9

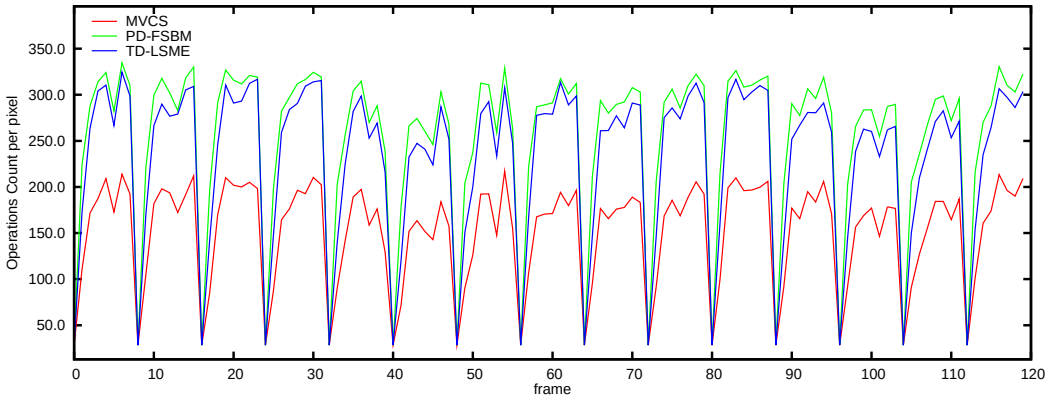
sors are capable of performing any of these arithmetic operations in just a single clock cycle. Consequently, such approximation may still be considered reasonably fair, for most current processor architectures.

In table 6.12 it is presented the average operation-count values that were obtained with the implemented transcoding structures to process the considered CIF test video sequences, at 30 FPS. To emphasize the additional computational cost of the motion re-estimation operation, the number of arithmetic functions corresponding to the two considered motion re-estimation approaches (TD-LSME and PD-FSBM) is presented in terms of its relative extra computational cost, when compared with the number of operations that is required by the most simple approach (MVCS), without any re-estimation of the MV:

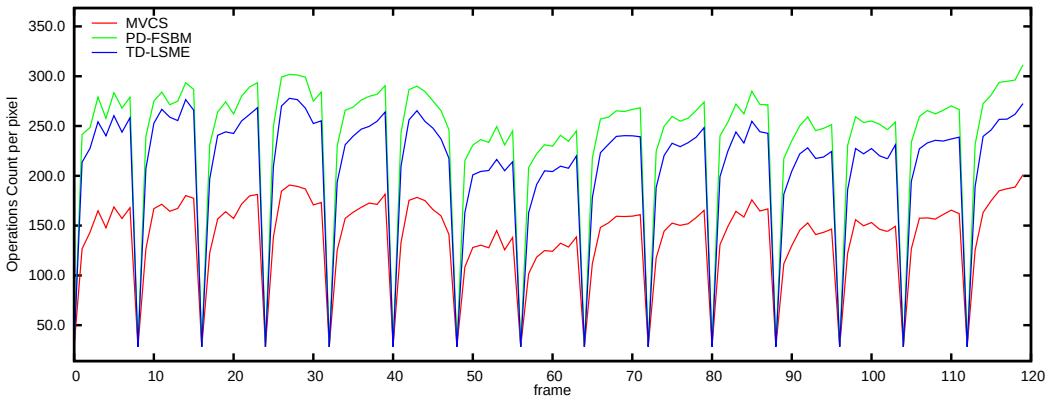
$$\Delta\text{Op-Count}^{\text{MVCS}}(\textit{algorithm}) = \text{Op-Count}(\textit{algorithm}) - \text{Op-Count}(\text{MVCS}) \quad (6.9)$$

From the presented values, it can be observed that the proposed transform-domain LSME approach is characterized by a smaller computational cost than the exhaustive pixel-domain FSBM algorithm. Hence, besides the important offered capability to implement a motion estimation procedure directly in the compressed DCT-domain, the presented scheme may also provide an additional computational advantage. Such aspect may be regarded as a significant implementation factor, mainly when this transcoding operation has to be implemented in processing platforms with limited computational capabilities.

In fig. 6.27 it is illustrated the variation of the computational cost that was observed for the *Carphone* and *Mobile & Calendar* CIF video sequences, using a quantization setup with $Q = 8$. Entirely similar results were obtained for the remaining video sequences and quantization setups. As it can be observed, the computational cost imposed by the proposed TD-LSME scheme is consistently smaller than the



(a) *Carphone* video sequence.



(b) *Mobile & Calendar* video sequence.

Figure 6.27: Average number of operations (*multiplications* and *additions/subtractions*) required to process each pixel of the *Carphone* and *Mobile & Calendar* CIF video sequences ($Q = 8$).

cost of the PD-FSBM approach.

6.3.3 Dynamic video composition in the DCT-domain[§]

In section 5.4 it was presented a brief overview of a set of different previously proposed transcoding architectures to perform digital video composition. After a thorough analysis of the main characteristics of such structures, it was proposed a new compressed-domain transcoding architecture, which tries to overcome some of the limitations presented by the former structures in order to improve the resulting transcoding performance. One of the tackled aspects addresses the implementation of the temporal prediction mechanism. In fact, most of the previously proposed structures adopt rather simplistic compositing and re-mapping approaches, that simply infer the new MVs from those of the original video sequences. In contrast, by adopting a more efficient DCT-domain motion re-estimation algorithm, that may consider any dimension for the search area, the presented architecture attempts to introduce some important improvements in the implementation of the temporal prediction mechanism. Such algorithm potentially contributes to a significant increase of both the output video quality and the bandwidth efficiency. On the other hand, by directly operating with the partially decoded DCT coefficients of the involved video sequences, the proposed approach provides additional advantages in what concerns the output video quality performance, since the absence of both the IDCT and DCT processing blocks will naturally make it less prone to round-off and fixed-precision arithmetic errors. Furthermore, contrary to what happens with most of the previously proposed approaches, the presented DCT-domain processing scheme does not impose any limitation on the adopted composition setup and allows each foreground video sequence to be placed over any location of the background video scene.

To assess the performance of the proposed architecture, this improved DCT-domain video processing structure was applied in the implementation of several different composition layouts. Such layouts considered two distinct sets of CIF test video sequences, composited by the *Mobile & Calendar* + *Carphone* sequences, and by the *Coastguard* + *Silent-Voice* video sequences:

PIP setup - the foreground video sequence was scaled by $\mathcal{S}_{\mathcal{F}} = 3$ and positioned inside the background sequence region, at coordinates $(l, c) = (11, 223)$ (see fig. 6.28(a) and fig. 6.29(a));

[§]Some portions of this section appeared in:

[91] - N. Roma and L. Sousa, "Fully compressed-domain transcoder for PIP/PAP video composition," in *Proceedings of the Picture Coding Symposium (PCS)*, Lisbon - Portugal, Nov. 2007, pp. CD–ROM.

(a) PIP setup. **(b)** PAP setup. **(c)** POP setup.

Figure 6.28: Experimental results obtained with the considered compositing setups, using the *Mobile* & *Calendar* + *Carphone* CIF video sequences ($Q = 4$).

(a) PIP setup. **(b)** PAP setup. **(c)** POP setup.

Figure 6.29: Experimental results obtained with the considered compositing setups, using the *Coastguard* + *Silent-Voice* CIF video sequences ($Q = 4$).

PAP setup - both the foreground and the background video sequences were scaled by $\mathcal{S}_{\mathcal{F}} = 2$ and combined into a single scene, side-by-side (see fig. 6.28(b) and fig. 6.29(b));

POP setup - the foreground video sequence was scaled by $\mathcal{S}_{\mathcal{F}} = 3$ and positioned over the right side of the background sequence (see fig. 6.28(c) and fig. 6.29(c));

The set of parameters adopted by these particular setups were carefully chosen not only because they correspond to quite common layouts used by many video applications, but also because the adopted insertion coordinates lead to a maximization of the misalignment between the foreground and background block grids. Therefore, they provide a practical demonstration of this additional feature, which has been quite often avoided by other previously proposed approaches. Furthermore, the adopted Least Squares Motion Estimation (LSME) refinement algorithm, described in section 5.3, was implemented by only considering the lowest $(K \times K)$

AC frequency DCT coefficients, with $K = 4$; and a maximum of 3 iterations to converge to the final MV, unless an additional constraint was met, corresponding to the adopted stopping condition: $\|v^i - v^{i-1}\| < \delta$, where $\delta = 0.1$ (see subsection B of section 5.3.2 (page 187)).

To better assess the performance of the proposed Transform-Domain Video Compositing Transcoder with Motion Re-estimation (TDVCT-MRE), this structure was compared against two other approaches: a Pixel-Domain Video Compositing Transcoder (PDVCT), equivalent to [9, 49, 50], but using an additional FSBM motion refinement block; and a Transform-Domain Video Compositing Transcoder (TDVCT), equivalent to [67], and that simply adopts the usual straightforward MV compositing scheme. The conducted experiments were carried out by considering three different quantization setups ($Q \in \{4, 8, 12\}$) and by registering the obtained gains, both in terms of the output video quality (PSNR) and bit rate. As it was done with the previously presented experimental procedures, to emphasize the direct influence of the included motion re-estimation module on the output bit rate, the output buffer controller of the encoding system was disabled, so that the obtained differences of the amount of bits required to encode each frame can be regarded as a direct influence of this refinement block. To conclude, a brief discussion about the involved computational cost is also provided.

A - Video quality

The average PSNR values obtained with the considered video composition setups for several quantization steps (Q), are presented in table 6.13. The variation of this video quality measure along the time for the *Mobile & Calendar + Carphone* and *Coastguard + Silent-Voice* composited video sequences is also presented in figs. 6.30 and 6.31.

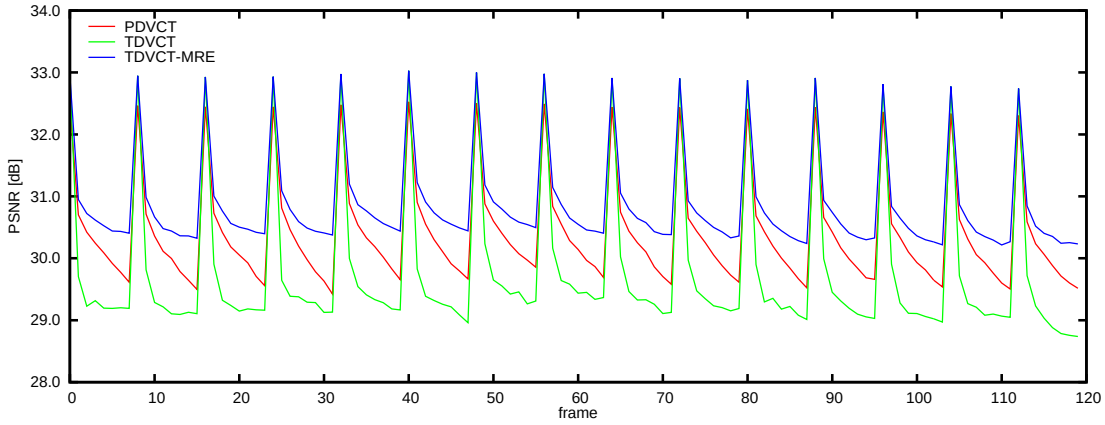
The obtained results evidence that the proposed DCT-domain transcoder can provide significant PSNR gains when compared with the other two approaches. In particular, when compared with the PDVCT architecture, the proposed structure presents quality gains as high as 2.2 dB. As it was previously referred, such quality improvements are mainly due to the absence of arithmetic and round-off errors introduced by the DCT and IDCT computational blocks, as well as to the contribution of the improved temporal prediction mechanism, provided by the included motion re-estimation computational block. The real contribution of such refinement block can be particularly observed by comparing the results of the proposed architecture with the transform-domain transcoder without MV re-estimation (TDVCT). In fact, these two architectures implement the same transcoding algorithm, apart from

Table 6.13: Average PSNR results obtained with the considered video compositing setups.

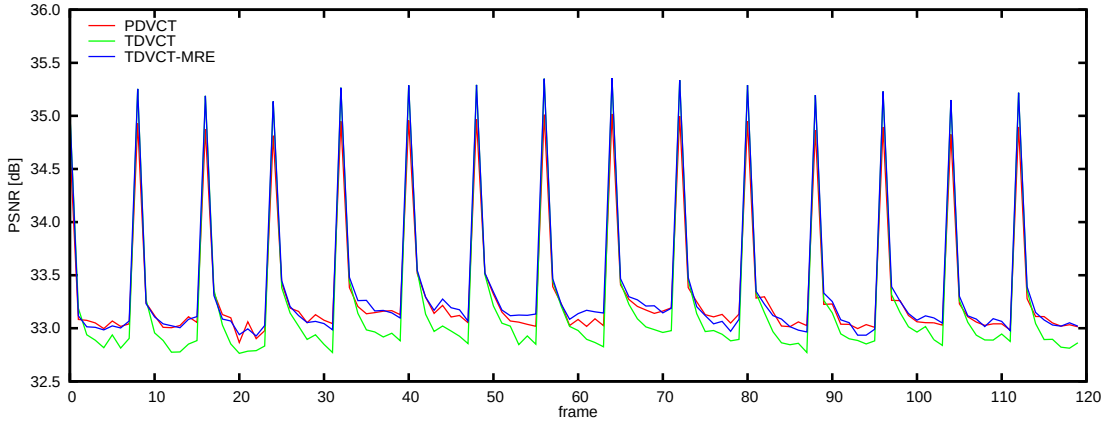
Video Sequences	VC Setup	Q	PSNR [dB]		
			PDVCT	TDVCT	TDVCT-MRE
Mobile+Carphone	PIP	4	33.62	34.57	35.84
		8	30.37	29.66	30.83
		12	27.87	26.89	28.06
	PAP	4	38.13	38.09	38.16
		8	33.29	33.18	33.30
		12	30.69	30.50	30.66
	POP	4	34.76	35.69	36.97
		8	31.53	30.78	31.99
		12	29.04	28.00	29.22
Coastguard+Silent	PIP	4	34.56	35.19	36.41
		8	31.56	31.22	32.18
		12	29.77	29.13	29.98
	PAP	4	38.26	38.43	38.37
		8	34.38	34.37	34.45
		12	32.43	32.28	32.42
	POP	4	35.65	36.19	37.42
		8	32.58	32.17	33.15
		12	30.72	30.04	30.94

the motion re-estimation block. Nevertheless, the proposed TDVCT-MRE structure still provides quality gains as high as 1.2 dB.

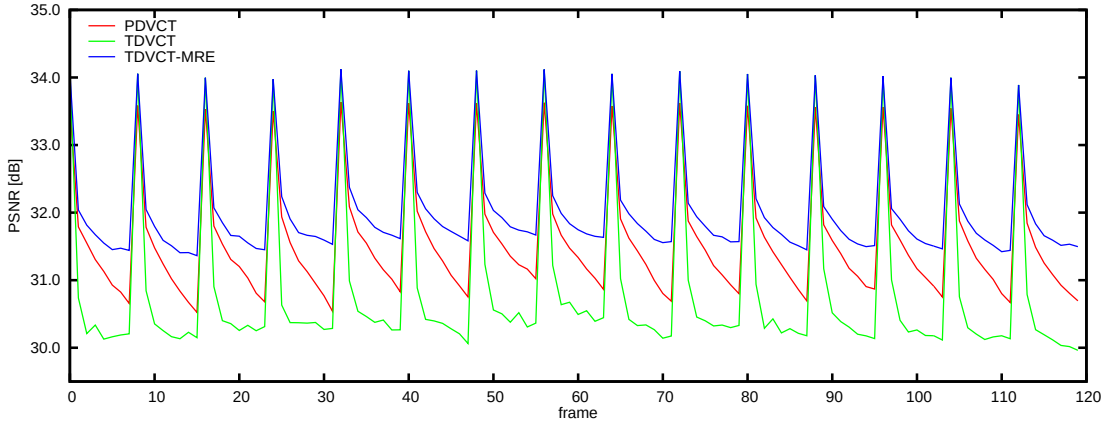
By relating the obtained results with the corresponding compositing setup, it is also worth noting that these quality gains are particularly notorious in the cases of the PIP and POP compositions. The reason for such performance differences may be justified by: *i)* a greater *directly affected area* of the PAP setup, leading to the effective processing of the blocks of the whole output frame area and thus potentially increasing the introduction of arithmetic and round-off distortion errors; *ii)* usage of a scaling factor given by an integer power of 2 ($\mathcal{S}_{\mathcal{F}} = 2$), for which the proposed



(a) PIP setup.



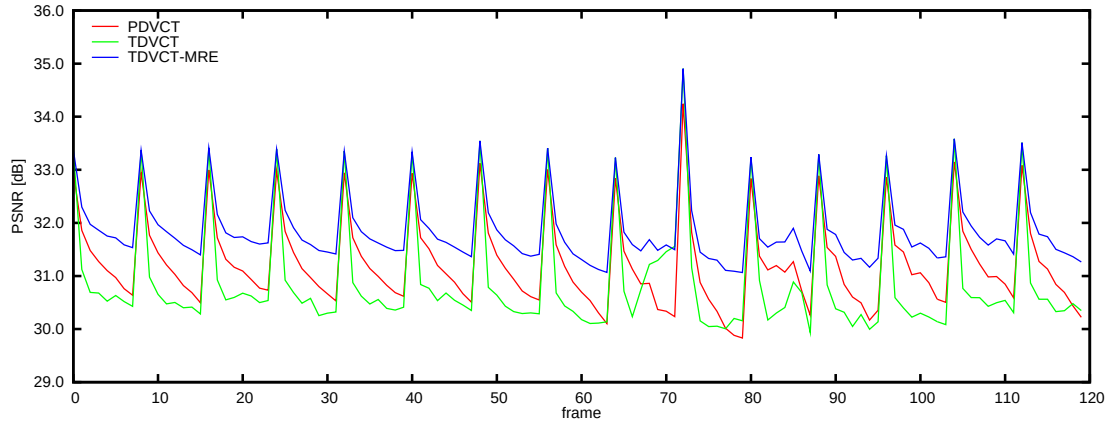
(b) PAP setup.



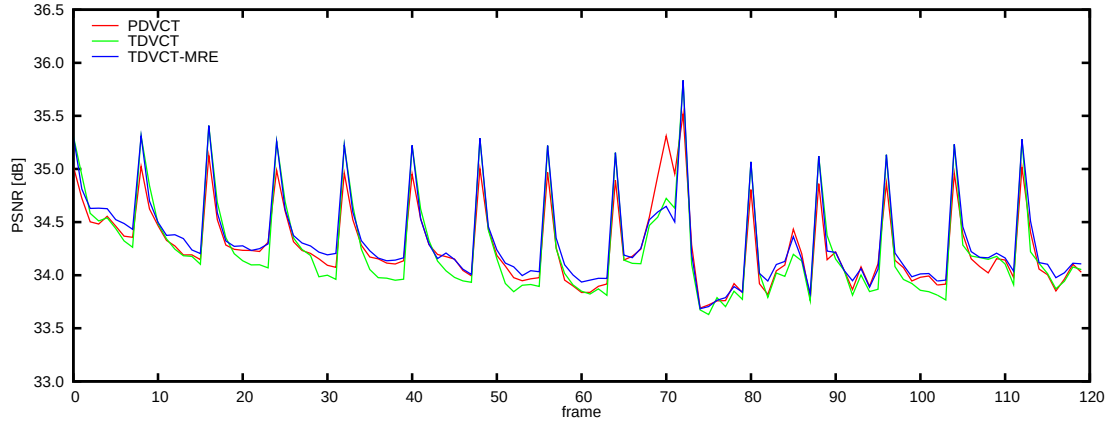
(c) POP setup.

Figure 6.30: Obtained PSNR level for the considered compositing setups using the *Mobile & Calendar + Carphone* video sequences ($Q = 8$).

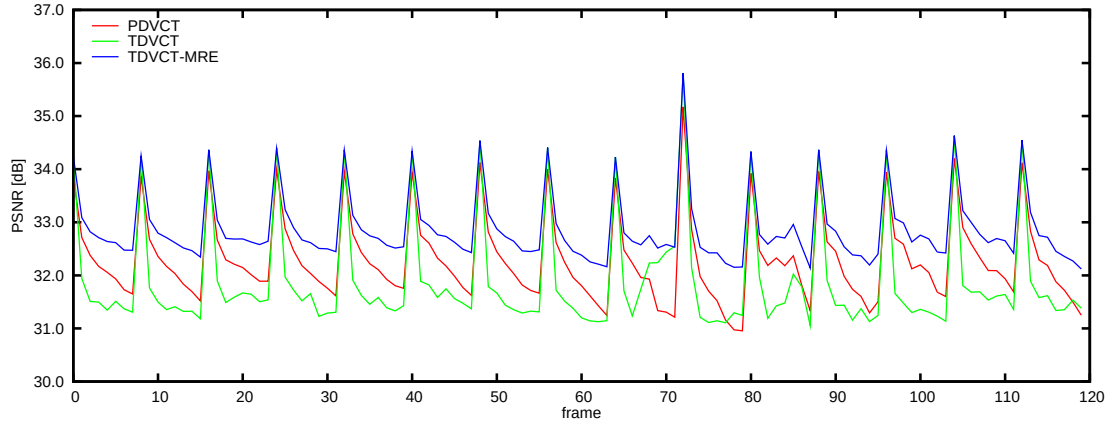
scaling algorithm does not provide a significative advantage (see subsection B of section 6.3.1 (page 236)).



(a) PIP setup.



(b) PAP setup.



(c) POP setup.

Figure 6.31: Obtained PSNR level for the considered compositing setups using the *Coastguard* + *Silent-Voice* video sequences ($Q = 8$).

B - Bit rate

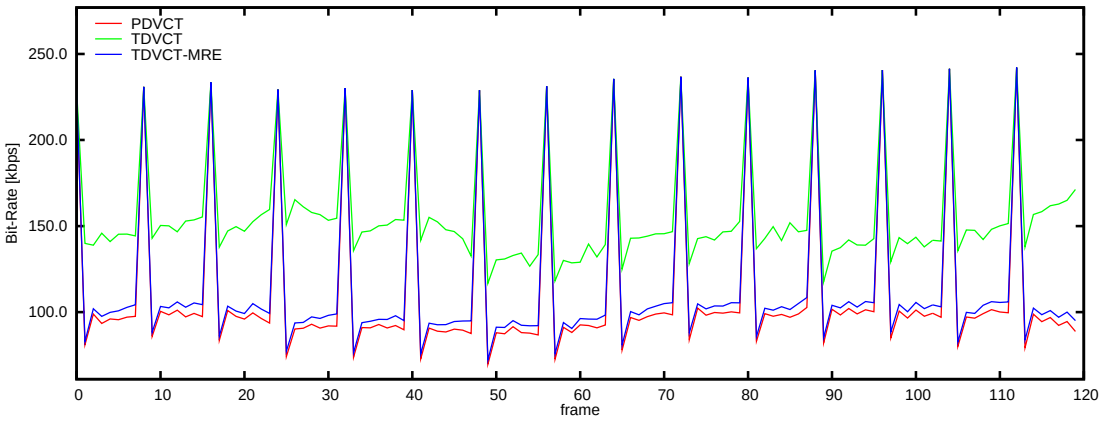
In table 6.14 it is presented the average bit rate performances that were obtained with the implemented algorithms for the several considered composition se-

Table 6.14: Average bit rate results, obtained with the considered video compositing setups.

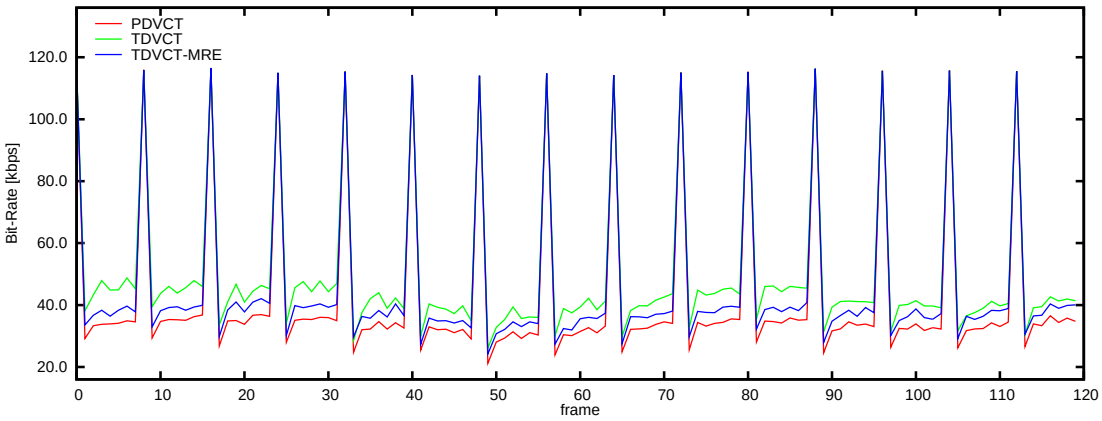
Video Sequences	VC Setup	Q	Bit Rate [kbps]		
			PDVCT	TDVCT	TDVCT-MRE
Mobile+Carphone	PIP	4	203.3	308.2	221.6
		8	105.3	159.5	109.3
		12	66.4	101.6	67.5
	PAP	4	91.9	102.5	99.2
		8	43.6	50.0	47.5
		12	26.7	30.9	29.0
	POP	4	159.1	246.8	172.9
		8	82.6	128.5	85.6
		12	52.3	82.3	53.2
Coastguard+Silent	PIP	4	85.4	144.6	98.1
		8	40.8	65.2	44.0
		12	26.3	39.3	27.0
	PAP	4	39.4	53.9	45.5
		8	18.2	25.1	21.3
		12	12.1	15.9	13.9
	POP	4	69.4	121.8	79.0
		8	33.7	57.0	36.3
		12	22.0	35.2	22.6

tups and quantization steps (Q). The variation of the output bit rate along the time for the *Mobile & Calendar + Carphone* and the *Coastguard + Silent-Voice* composited video sequences is presented in figs. 6.32 and 6.33, respectively. Contrary to what was observed for the video quality gains, the bit rate resulting from the proposed transcoding architecture slightly exceeded the bit rate obtained with the pixel-domain approach (PDVCT). Nevertheless, such result was already expected. As it was previously observed in subsection A of section 6.3.2 (page 249), while the MVs obtained with the pixel-domain FSBM algorithm represent the optimal solution of the desired temporal prediction scheme (using the SAD criteria), the

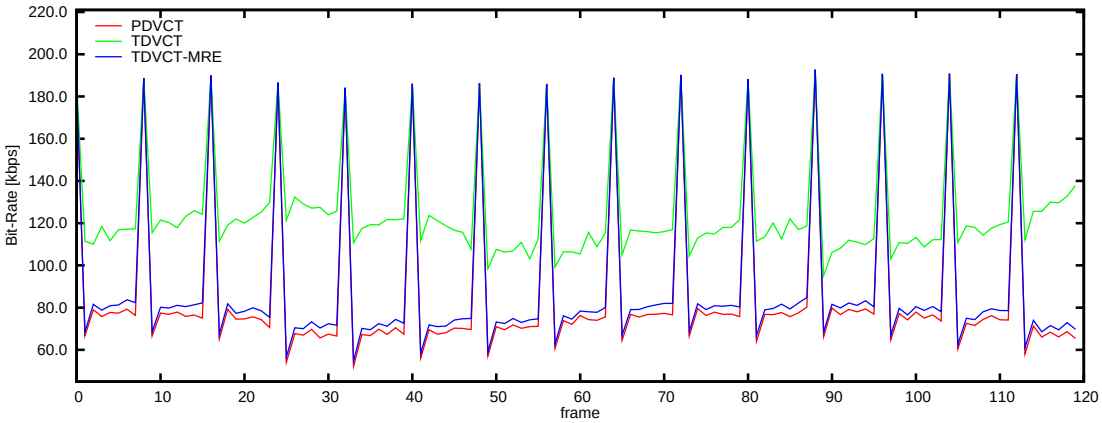
MVs obtained with the adopted LSME algorithm represent the solution of the same estimation problem using a somewhat different criteria. Once again, such criteria



(a) PIP setup.



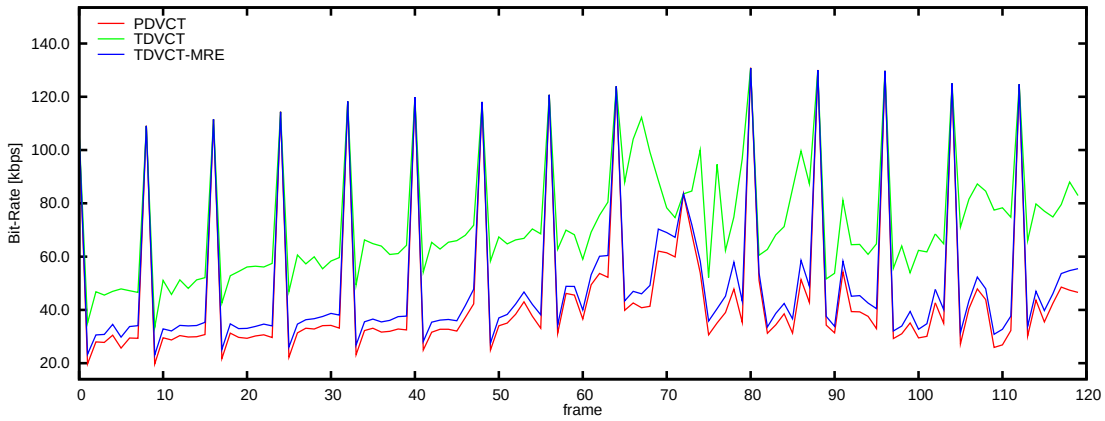
(b) PAP setup.



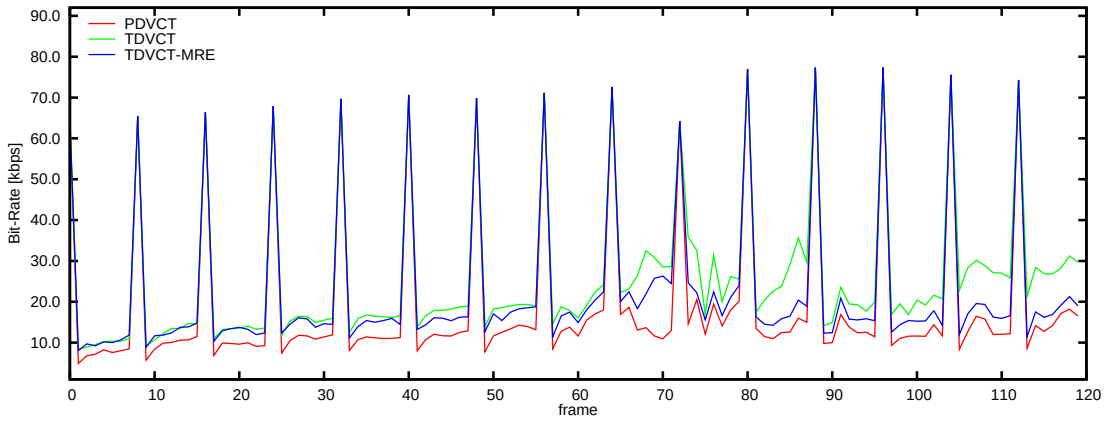
(c) POP setup.

Figure 6.32: Obtained bit rate for the considered compositing setups using the *Mobile & Calendar + Carphone* video sequences ($Q = 8$).

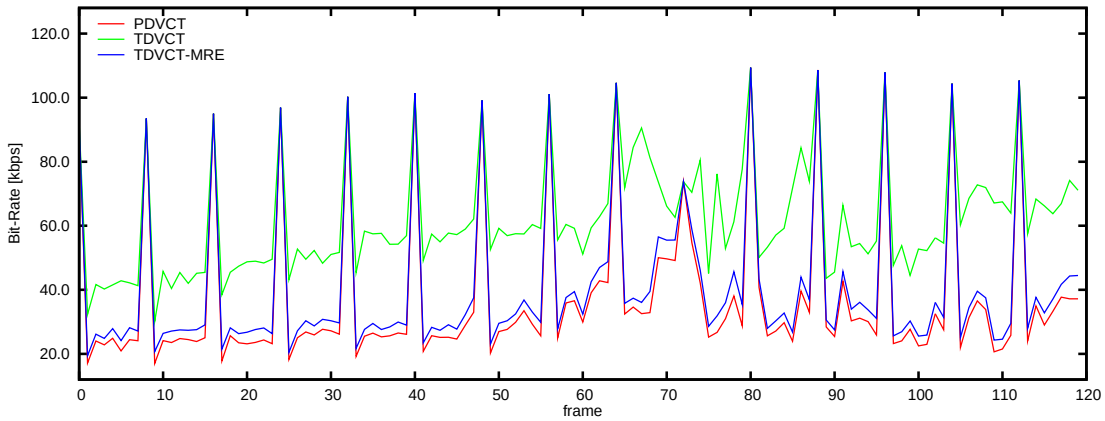
is based on the minimization of a squared error signal, that is entirely computed in the compressed DCT-domain. Furthermore, as it was also previously observed,



(a) PIP setup.



(b) PAP setup.



(c) POP setup.

Figure 6.33: Obtained bit rate for the considered compositing setups using the *Coastguard* + *Silent-Voice* video sequences ($Q = 8$).

this iterative scheme was implemented using a stopping condition that may prevent it to achieve the optimal MVs. Even so, from the results presented in table 6.14 and in figs. 6.32 and 6.33, it can be observed that the proposed TDVCT-MRE approach provides bit rate performance levels quite close to the optimal performances provided by the FSBM-based pixel-domain approach (PDVCT).

However, the same does not happen when the proposed video composition architecture (TDVCT-MRE) is compared with the simplest approach, without re-estimation of the MVs (TDVCT). The average bit rate results, presented in table 6.14, as well as its variation along the time, illustrated in figs. 6.32 and 6.33, clearly evidence that the enhanced temporal prediction mechanism, provided by the proposed MV re-estimation module, leads to a significant reduction of the output bit rate. Such reduction may lead to bandwidth gains as high as 35%.

C - Computational cost

Similarly to what was performed in the evaluation of the computational cost of the proposed LSME algorithm (see subsection D of section 6.3.2 (page 256)), the absence of a predominant arithmetic operation in the implementation of the considered compositing architectures leads to the usage of the sum of the number of *multiplications* and *additions/subtractions* operations that are required to process each pixel of the original frame, as the main figure of merit that best characterizes the computational cost of each transcoding architecture. As a consequence, in table 6.15 it is presented the average operation-count values that were obtained with the implemented algorithms for the several considered composition setups and quantization steps (Q). The variation of this computational cost estimate along the time for the *Mobile & Calendar + Carphone* and the *Coastguard + Silent-Voice* composited video sequences is presented in figs. 6.34 and 6.35.

From the obtained results, it may be observed that for all the conducted experiments, the TDVCT architecture presented the lowest values of the operation-count per pixel metric. Such fact arises from the absence of any motion re-estimation procedure in the transcoder architecture, which led to the lowest video quality levels, as well as the highest bit rate results, as it was previously referred. As a consequence, the following analysis will be mainly focused on the comparison of the computational cost of the PDVCT and of the TDVCT-MRE architectures, since the motion refinement modules included in their structure potentially provide them with better video quality and coding efficiency performances.

Besides the considered transcoding architecture, the number of performed arithmetic operations that were obtained for each configuration evidence that the com-

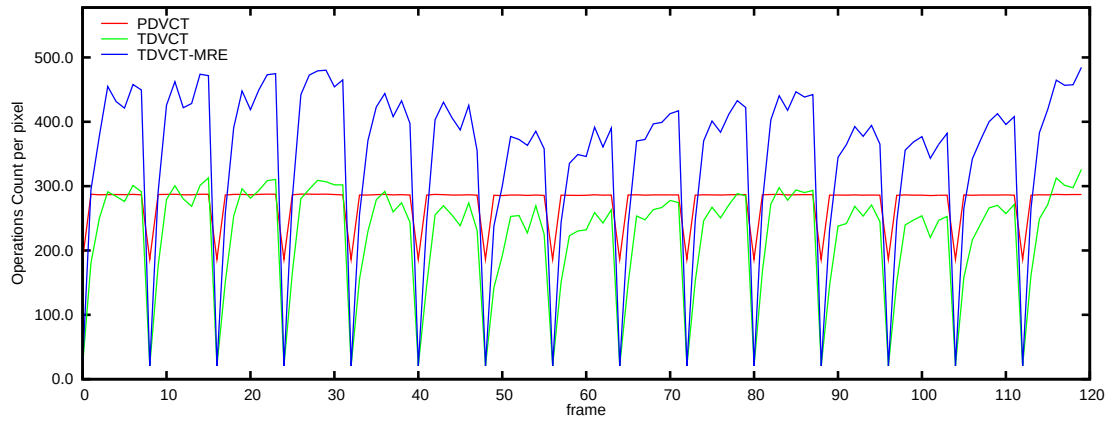
Table 6.15: Experimental average operation-count results, obtained from the considered video compositing setups.

Video Sequences	VC Setup	Q	Op-Count		
			PDVCT	TDVCT	TDVCT-MRE
Mobile+Carphone	PIP	4	274.2	246.6	377.1
		8	274.0	236.8	360.4
		12	273.8	232.0	349.5
	PAP	4	517.3	321.0	482.6
		8	517.1	311.6	461.2
		12	517.0	307.1	447.7
	POP	4	284.4	243.4	353.3
		8	284.2	233.5	337.3
		12	284.1	228.7	328.1
Coastguard+Silent	PIP	4	272.1	168.2	285.8
		8	271.8	164.0	275.3
		12	271.8	165.2	277.6
	PAP	4	515.1	216.2	382.7
		8	514.8	212.5	365.7
		12	514.8	215.2	356.7
	POP	4	282.5	165.4	257.8
		8	282.1	161.1	249.9
		12	282.1	162.2	251.3

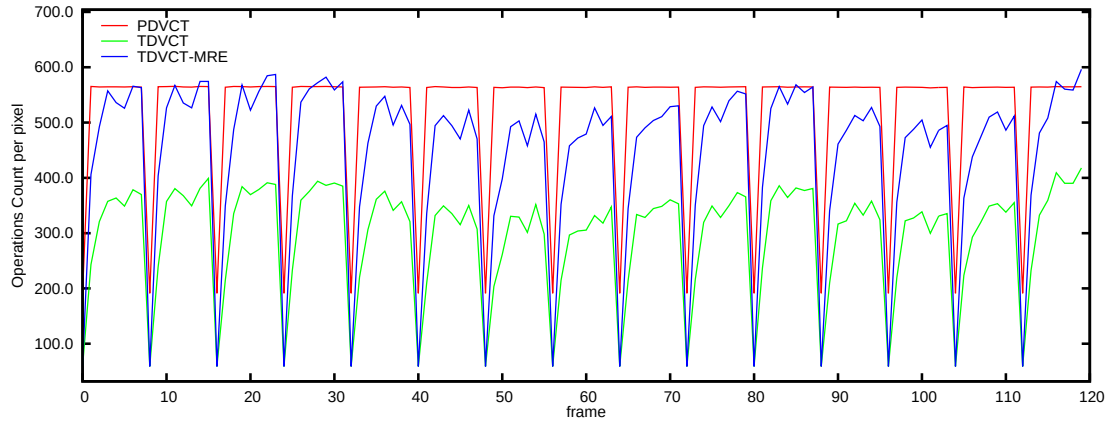
putational cost clearly depends on the processed video sequences and on the implemented compositing setup. In fact, while the PIP and POP configurations presented a significant advantage of the PDVCT structure over the proposed TDVCT-MRE architecture in the processing of the *Mobile* & *Calendar* + *Carphone* composition, the processing of the *Coastguard* + *Silent-Voice* composition revealed that these two architectures present a quite similar computational cost.

Moreover, besides this dependence on the video sequences contents and on the composition setup, the computational cost estimates obtained for all the considered PAP compositions demonstrate that the proposed DCT-domain approach may pro-

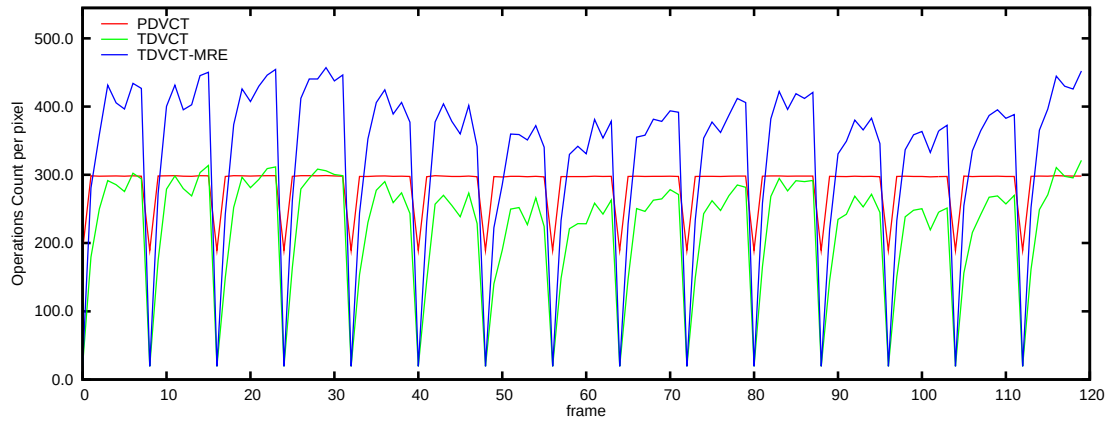
vide significant computational advantages over the pixel-domain transcoder. Such situation may be justified by the larger *directly affected area* of this configuration,



(a) PIP setup.



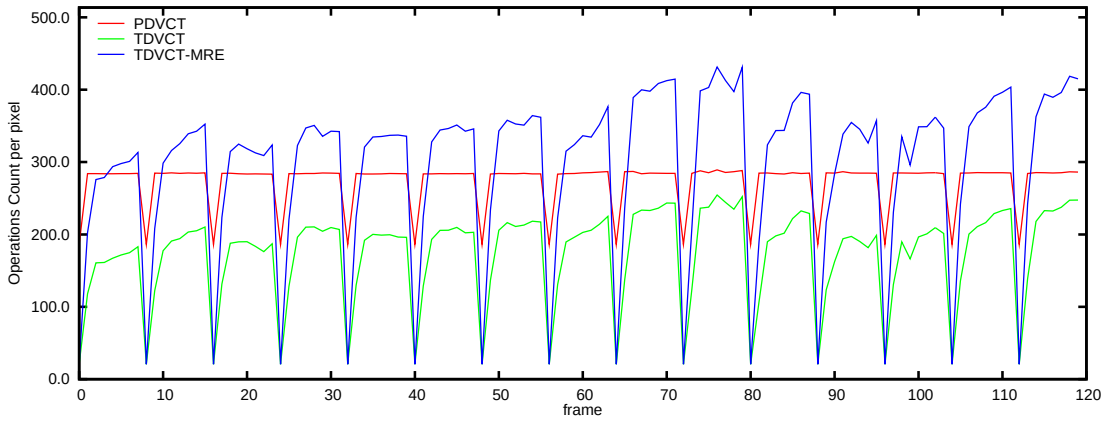
(b) PAP setup.



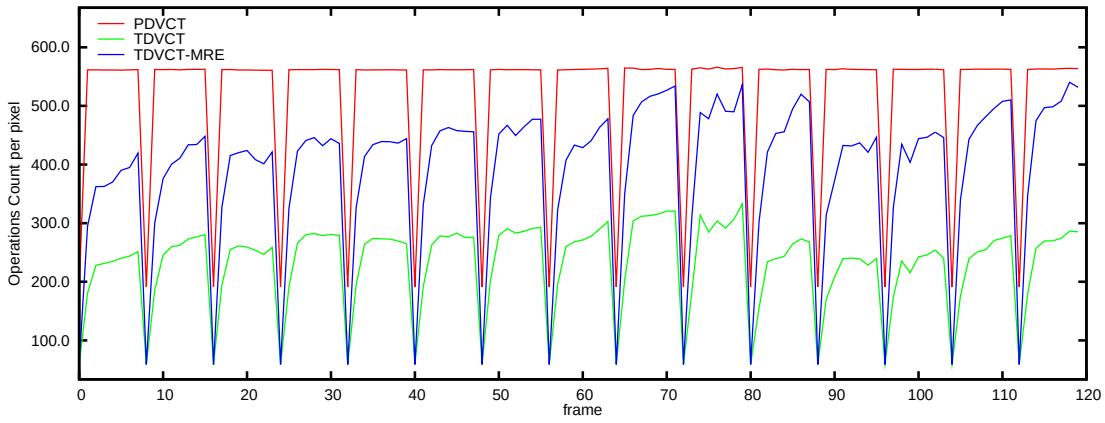
(c) POP setup.

Figure 6.34: Obtained operation-count for the considered compositing setups using the *Mobile & Calendar + Carphone* video sequences ($Q = 8$).

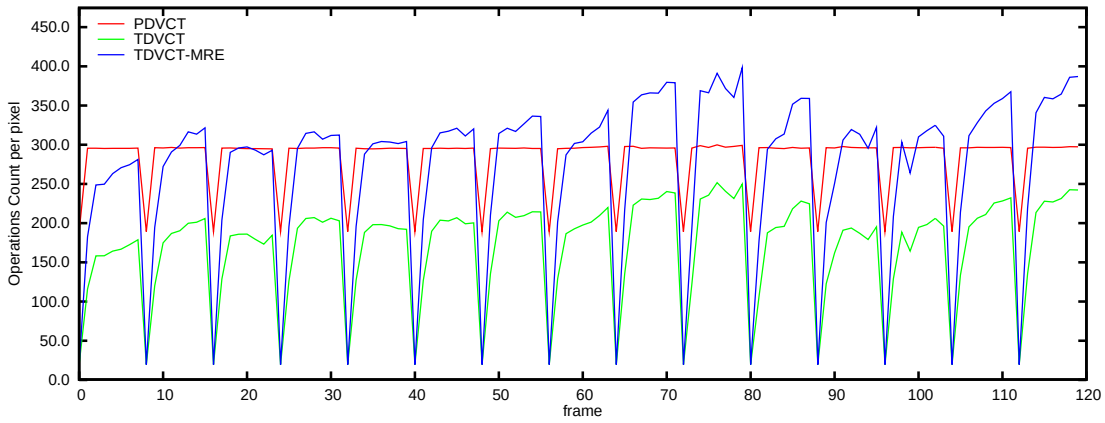
which implies the effective processing of the whole composited frame. In fact, besides the computational advantages provided by the proposed LSME algorithm over



(a) PIP setup.



(b) PAP setup.



(c) POP setup.

Figure 6.35: Obtained operation-count for the considered compositing setups using the *Coastguard* + *Silent-Voice* video sequences ($Q = 8$).

the FSBM algorithm implemented by the PDVCT structure (see subsection D of section 6.3.2 (page 256)), it should be recalled that the implemented DCT-domain scaling algorithm only requires one half of the number of operations performed by the equivalent pixel-domain approach (see table 6.4).

Consequently, depending on the particular compositing video manipulation that is to be implemented by a given video processing system, these potential computational advantages, together with the significant video quality performance results and a coding efficiency level that provides a bit rate quite close to the optimal one (provided by an exhaustive ME refinement algorithm), justify the consideration of the proposed compressed-domain transcoding architecture to implement video compositing in the DCT-domain.

6.4 Conclusions

The several experimental procedures that were conducted in the scope of the presented research were described in this chapter. A thorough discussion of the obtained results was also presented, providing a detailed evaluation of the objective and subjective video quality levels that are achieved by the proposed static and the dynamic video processing algorithms. This evaluation also considered the resulting bit rate of the processed video sequences, as well as the computational cost associated with these transcoding architectures.

In particular, in section 6.2 it was shown that DCT-domain insertion architectures can offer significant advantages, both in terms of image quality and computational cost. They also provide the possibility to easily adapt the computational requirements of the insertion architecture to the particular characteristics of the target application, adjusting the desired trade-off between image quality and computational cost.

In section 6.3.1 it was presented the evaluation of the proposed transcoding algorithm for video downscaling in the transform-domain by any arbitrary integer scaling factor. The obtained experimental results have shown that the proposed algorithm provides significant advantages over the usual DCT decimation approaches, both in terms of the involved computational cost, of the output video quality and of the resulting bit rate. Such advantages are even more significant for scaling factors other than integer powers of 2, leading to a reduction of the computational cost as high as 5. Quite significant PSNR gains are also achieved, when compared with the usual DCT decimation techniques.

The assessment of the proposed block-based motion re-estimation algorithm,

implemented in the compressed DCT-domain, was presented in section 6.3.2. The obtained experimental results have shown that the proposed algorithm can be regarded as a viable alternative to significantly enhance the quality of the temporal prediction adopted in processed video sequences, by providing the means to compute or refine the MVs in the compressed DCT-domain. The experimental procedures that were carried out have shown that the proposed refinement algorithm may provide reductions of the obtained bit rate up to 30%, when compared with a simple MV re-usage and compositing approach.

Finally, in section 6.3.3 it was presented an evaluation of the proposed transcoder architecture to perform video composition in the compressed DCT-domain. The proposed structure not only allows the insertion of the foreground video sequences at any arbitrary location and independently of the usual $(N \times N)$ block grid, but it also includes a quite efficient DCT-domain least-squares motion re-estimator module that significantly improves the temporal prediction mechanism. The experimental results that were obtained with this architecture have shown that it may provide PSNR gains as high as 2.2 dB, when compared with a traditional DCT-domain approach that does not perform any re-estimation of the composited MVs.

References

[1] I. Ahmad, X. Wei, Y. Sun, and Y.-Q. Zhang, "Video transcoding: An overview of various techniques and research issues," *IEEE Transactions on Multimedia*, vol. 7, no. 5, pp. 793–804, Oct. 2005.

[9] S.-F. Chang and D. G. Messerschmitt, "Compositing motion-compensated video within the network," in *Proceedings of the International Workshop on Multimedia Communications (MULTIMEDIA)*. IEEE, Apr. 1992, pp. 40–56.

[11] S.-F. Chang and D. G. Messerschmitt, "Manipulation and compositing of MC-DCT compressed video," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 1, pp. 1–11, Jan. 1995.

[12] J. Chen, U.-V. Koc, and K. J. R. Liu, *Design of Digital Video Coding Systems - A Complete Compressed Domain Approach*. Marcel Dekker, 2002.

[13] M.-J. Chen, M.-C. Chu, and C.-W. Pan, "Efficient motion-estimation algorithm for reduced frame-rate video transcoder," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 4, pp. 269–275, Apr. 2002.

[16] R. Dugad and N. Ahuja, "A fast scheme for image size change in the compressed domain," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 4, pp. 461–474, Apr. 2001.

[30] *ITU-T Recommendation H.263 - "Video Coding for Low Bitrate Communication"*, ITU-T, Feb. 1998.

[41] U.-V. Koc and K. J. R. Liu, "DCT-based motion estimation," *IEEE Transactions on Image Processing*, vol. 7, no. 7, pp. 948–965, Jul. 1998.

[48] Y.-R. Lee, C.-W. Lin, S.-H. Yeh, and Y.-C. Chen, "Low-complexity DCT-domain video transcoders for arbitrary-size downscaling," in *Proceedings of the IEEE International Workshop on Multimedia Signal Processing (MMSP)*, Sep. 2004, pp. 31–34.

[49] C.-H. Li, H. Lin, C.-N. Wang, and T. Chiang, "A fast H.264-based picture-in-picture (PIP) transcoder," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, vol. 3. IEEE, Jun. 2004, pp. 1691–1694.

[50] C.-H. Li, C.-N. Wang, and T. Chiang, "A low complexity picture-in-picture transcoder for video-on-demand," in *IEEE International Conference on Wireless Networks, Communications and Mobile Computing*, vol. 2, Jun. 2005, pp. 1382–1387.

[52] Y. Liang, L.-P. Chau, and Y.-P. Tan, "Arbitrary downsizing video transcoding using fast motion vector re-estimation," *IEEE Signal Processing Letters*, vol. 9, no. 11, pp. 352–355, Nov. 2002.

[65] *MPEG-4: Video Verification Model - version 18.0 - ISO/MPEG N3908*, MPEG, Jan. 2001.

[67] Y. Noguchi, D. G. Messerschmitt, and S.-F. Chang, "MPEG video compositing in the compressed domain," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 2. IEEE, May 1996, pp. 596–599.

[75] P. Pirsch, N. Demassieux, and W. Gehrke, "VLSI architectures for video compression - a survey," *Proceedings of the IEEE*, vol. 83, no. 2, pp. 220–246, Feb. 1995.

[84] R. Reeves and K. Kubik, "Compressed domain image matching using symmetric convolution," in *Proceedings of IEEE Region 10 Annual Conference on*

Speech and Image Technologies for Computing and Telecommunications - TENCON'97. Brisbane, Queensland: Queensland QUT Publications, Dec. 1997.

[86] N. Roma and L. Sousa, "Insertion of irregular-shaped logos in the compressed DCT domain," in *Proceedings of the IEEE International Conference on Digital Signal Processing (DSP)*, vol. 1. Santorini, Greece: IEEE, Jul. 2002, pp. 125–128.

[87] N. Roma and L. Sousa, "Transcoding architectures for object insertion in compressed video," INESC-ID – Lisboa, Portugal, Tech. Rep. RT/006/2002, Oct. 2002.

[88] N. Roma and L. Sousa, "Fast transcoding architectures for insertion of non-regular shaped objects in the compressed DCT-domain," *Signal Processing: Image Communication*, vol. 18, no. 8, pp. 659–683, Sep. 2003.

[89] N. Roma and L. Sousa, "Least squares motion estimation algorithm in the compressed DCT domain for H.26x/MPEG-x video sequences," in *Proceedings of the IEEE International Conference on Advanced Video and Signal-Based Surveillance (AVSS)*. Como - Italy: IEEE, Sep. 2005, pp. 576–581.

[90] N. Roma and L. Sousa, "Efficient hybrid DCT-domain algorithm for any arbitrary integer re-size video downscaling," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, no. 57291, pp. 1–16, Sep. 2007.

[91] N. Roma and L. Sousa, "Fully compressed-domain transcoder for PIP/PAP video composition," in *Proceedings of the Picture Coding Symposium (PCS)*, Lisbon - Portugal, Nov. 2007, pp. CD–ROM.

[108] Y.-P. Tan, Y. Liang, and H. Sun, "On the methods and performances of rational downsizing video transcoding," *Signal Processing: Image Communication*, vol. 19, pp. 47–65, 2004.

[113] J. Xin, C.-W. Lin, and M.-T. Sun, "Digital video transcoding," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 84–97, Jan. 2005.

[114] P. Yin, A. Vetro, B. Liu, and H. Sun, "Drift compensation for reduced spatial resolution transcoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 11, pp. 1009–1020, Nov. 2002.

[115] J. Youn, M.-T. Sun, and C.-W. Lin, "Motion vector refinement for high performance transcoding," *IEEE Transactions on Multimedia*, vol. 1, no. 1, pp. 30–40, Mar. 1999.

7

Conclusions and Future Research Directions

Contents

This final chapter concludes this thesis and presents an overview of the main achievements that resulted from the described research work. A brief discussion of the most important accomplishments will be presented, as well as some relevant open issues and possible future research directions.

7.1 Conclusions

From the research work that was conducted in the scope of this thesis, it resulted the proposal of a set of efficient algorithms and processing architectures to implement video composition transcoding operations of precoded video sequences, directly in the DCT-domain. Two particular video compositing operations were specially considered by this research: *static video composition* and *dynamic video composition*.

The most significant results and conclusions of this thesis are summarized bellow.

- **Efficient transcoding algorithms and architectures for static video composition**

  A comprehensive class of static video composition structures for the insertion of stationary or quasi-stationary non-regular shaped objects, such as visible logos or subtitles, was presented. Such structures are based on a DCT-domain compositing technique and make use of a symmetric convolution operator of the even type-II DCT.

  Based on these techniques, new and efficient processing algorithms were presented. Such algorithms not only directly operate with the decoded DCT-coefficients received from the incoming compressed video stream, but they also restrict their processing to the set of pixels corresponding to the objects that are intended to be inserted in the video scene. Moreover, by using the proposed techniques, the presence of undesired semi-transparent rectangular regions around the inserted irregular-shaped objects was circumvented.

  From the developed research work, it was proposed a class of fast transcoding architectures to insert non-regular shaped objects in compressed video signals. The proposed transcoders incorporate the developed logo insertion module on their structure, to manipulate the DCT coefficient blocks of the decoded video streams.

  A detailed analysis of the main characteristics of the considered transcoding architectures has shown that DCT-domain transcoders, when compared with their corresponding pixel-domain counterparts, may provide significant advantages, both in terms of the subjective video quality and of the resulting com-

putational cost. Depending on the video sequence under processing and on the corresponding amount of movement that it contains, compressed DCT-domain insertion algorithms may require as few as 50% of the number of operations required by the traditional pixel-domain insertion algorithms. Moreover, the distinctive features presented by the proposed architectures provide the means to adapt the insertion scheme to the particular requirements of the target application.

- **Efficient transcoding algorithms and architectures for *dynamic* video composition**
  Contrasting with the static video compositing schemes, the proposed algorithms for dynamic video composition manipulate most of the data structures that are processed by the several modules of the video encoder.
  An efficient transcoding architecture to perform the composition of several precoded video sequences was presented. The proposed architecture provides the composition operation of one or more "foreground" video sequences over the displaying area corresponding to the "background" video sequence. By directly operating with the partially decoded DCT coefficients of the involved video sequences, the proposed approach provides significant advantages in what concerns the output video quality performance. Such enhancement is mainly due to the absence of both the IDCT and DCT processing blocks, which makes it less prone to round-off and fixed-precision arithmetic errors. When compared with a pixel-domain approach, such quality gains can be as high as 2 dB. Moreover, when compared with other previous proposals, the presented approach significantly improves the temporal prediction mechanism of the output video sequence, by incorporating the developed DCT-domain motion re-estimation algorithm. Contrary to the refinement schemes that were proposed by other authors, this re-estimation procedure may consider any dimension for the search area, leading to an inherent improvement of the output video quality and to a significant reduction of the required bit rate. When the proposed video composition architecture is compared with a simpler transform-domain approach, without re-estimation of the MVs, such reduction may lead to bandwidth gains as high as 35%. Furthermore, the presented DCT-domain approach does not impose any restriction on the composition setup, allowing each "foreground" video sequence to be placed over any location of the "background" video scene. Such important feature offers a flexible and easy implementation mechanism for most current video composition setups, such as PIP, PAP and POP.

- **Improved DCT-domain video processing operations**
  Several common video processing operations, required by the proposed transcoding algorithms for video composition, had to be adapted and transposed into the DCT-domain. In certain cases, some important changes and improvements were actually proposed and introduced. Such improvements were very important not only to optimize the processing of the video objects, by directly using the DCT coefficients of the decoded video objects, but also to better adapt and suit their implementations to the data structures adopted by the considered video standards.
  A special emphasis was devoted to the improvement of the following two important operations:

  - *DCT-domain video space-scaling*
    An innovative and efficient hybrid transcoding algorithm for video downscaling in the transform domain, by any arbitrary integer scaling factor, was proposed. Due to its particular good adaptation to the block-based data structures of most video standards, this algorithm offers a considerable advantage in what concerns the computational cost, by taking advantage of the scaling mechanism and by only performing the operations that are really needed to compute the desired output values. Although such computational cost greatly depends on the considered scaling factor, the conducted experimental procedures revealed that the number of operations required by the proposed DCT-domain algorithm may be as few as 30% of those required by an equivalent pixel-domain implementation. On the other hand, when compared with a DCT decimation procedure, the proposed algorithm presented particularly significant advantages for scaling factors other than integer powers of 2. In such cases, the proposed scheme only needs about 20% of the operations required by the DCT decimation approach.
    In order to meet a wide range of computational restrictions that may be imposed by the adopted implementation system, an optional combination of the presented algorithm with high frequency DCT coefficients discarding techniques was also proposed. It provides a flexible complexity scalability feature and it gives rise to an adaptable trade-off between the involved scalable computational cost and the resulting video quality and bit rate. By applying a pre-filter mechanism to implement such trade-off and to balance the computational cost of these two considered DCT-domain approaches, it was observed that the proposed algorithm is

able to process more decoded DCT coefficients than the DCT decimation approach (leading to video quality advantages as high as 7 dB). Moreover, for certain considered setups, such improvement was also obtained with a reduction of the output bit rate.

- *DCT-domain motion estimation*
  A new compressed-domain ME algorithm was proposed. This algorithm is based on an iterative scheme that estimates the new MVs by applying a LSE technique. This algorithm makes use of the DCT coefficients directly obtained from the input video stream and provides an efficient alternative to refine, or even to re-compute, the set of MVs that are required to implement the motion compensated prediction mechanism of the output video sequence. By comparing the obtained experimental results with those that resulted from simply re-using the MVs decoded from the input video stream, it was observed that the refined MVs (obtained with the proposed algorithm) provide a significant improvement of the temporal prediction mechanism of the processed video sequences, leading to a consequent reduction of the output bit rate as high as 20%. Such significant results evidence that the computed MVs may be regarded as good approximations of the optimal MVs, estimated with a pixel-domain exhaustive search. Nevertheless, contrasting with a pixel-domain full-search block-matching approach, the proposed algorithm offers the possibility to scale the involved computational cost according to the restrictions imposed by the adopted processing system. Whenever the number of required operations has to be restricted, the proposed ME algorithm may consider only an arbitrary subset of non-null DCT coefficients.

The set of improvements resulting from the proposal of these two algorithms was particularly important to the development of the previously described compressed-domain video compositing algorithms. While the proposed down-scaling algorithm significantly contributes to provide this compositing architecture with particularly good video quality performances, the new compressed domain ME algorithm actively contributes to a reduction of the resulting bit rate, by improving the temporal prediction of the processed video sequences.

- **Optimization of the considered transcoding algorithms and architectures**
  The proposal of the several transcoding algorithms that were presented in this thesis was complemented with the development of several DCT-domain

transcoding architectures that implement the considered video processing operations. Nevertheless, a tight relation between the transcoding algorithms development stage and the architectures design phase was considered. This relation is highly important, in order to cope with the several restrictions that are usually imposed both by the data structures associated to the considered video standards, and by the computational and memory access requirements and characteristics of the target implementation platforms.

From the obtained results, it was shown that DCT-domain transcoding architectures may offer significant advantages not only in terms of image quality, but also in terms of the computational cost. Moreover, the presented study devoted a particular attention to certain trade-offs that are inherent to most of the considered transcoding structures. Such trade-offs focused the following key factors, that significantly affect the actual feasibility of the transcoders in current video processing systems: *i)* the involved computational cost, *ii)* the output video quality, and *iii)* the resulting bit rate. Consequently, many of the algorithms and architectures that were proposed in this thesis offer the possibility to properly adapt the computational cost required by the several involved operations with both the resulting bit rate and the real-time and video quality requirements of the intended transcoding system. In order to attain such objective, a flexible scaling of the overall complexity level that is involved in each operation, as well as an accurate assessment of the resulting influence on both the obtained video quality and bit rate, were carefully analyzed. By adopting such approach, it was possible to confer the transcoder designer the possibility to trade-off the involved computational cost with the resulting output video quality performance and the obtained bit rate, in order to easily adapt the computational requirements of the processing architecture to the particular characteristics of the target application.

7.2 Future research directions

From the author's point of view, the work herein presented raises a series of interesting problems and lines of research that are important enough to require further investigation in the near future. Some of the most interesting open directions for further developments and improvements of this work include the following issues:

1. Integration, in a single framework, of the proposed transcoding algorithms and architectures for both static and dynamic video composition. To accomplish such integrated and generalized video composition transcoding framework, the

previously defined concepts of static and dynamic composition operations of video objects will have to be revised. In particular, the adaptation of some of the developed techniques (such as spatial downscaling and motion estimation in the DCT-domain), in order to support the processing of Non-Regular Shaped Objects will have to be considered. In the same trend, the adaptation of the insertion techniques presented in chapter 4, in order to consider the implementation, in a particular efficient way, of the composition operation of video objects characterized by non-constant texture components will have to be studied and further investigated.

2. Adaptation of the proposed transcoding algorithms and architectures to the coding techniques adopted by some of the most recent video standards, such as the H.264/AVC [31]. Contrasting with previously proposed standards based on the even type-II DCT, such as the H.261 [29], H.263 [30], MPEG-1 Video [26], MPEG-2 Video [27] and MPEG-4 Visual [28], this standard makes use of the Integer Discrete Cosine Transform (IntDCT) to circumvent the accuracy mismatch in the computation of the transform at the encoder and decoder ends of the video transmission system.
   However, despite the orthogonal nature and the computational simplicity that is offered by this alternative transform, the extension of some of the algorithms and architectures that were proposed in this thesis to these video standard requires the analysis and the implementation of some important mathematical relations in the scope of this alternative transform-domain, such as the *multiplication-convolution property* and the *derivative operator*. These relations are required for the implementation of the static video composition transcoding techniques for the insertion of visible video objects, defined in section 4.2, and to allow the application of the LSE technique in the motion re-estimation procedure that was proposed in section 5.3.

3. Adaptation and extension of the proposed transcoding architectures to the processing structures of the several video standards that have been adopted by most current multimedia applications. Some particular examples of these adaptation needs are the following:

   - Support for both interlaced and non-interlaced frame encoding mechanisms: considering that the luminance blocks structure is different for frame DCT encoding and for field DCT encoding, the proposed compositing transcoding techniques will have to be properly adapted in order to

support video standards that make use of this mechanism (e.g.: MPEG-2 Video [27], etc.).

- Support for enhanced temporal prediction mechanisms based on the usage of multiple reference frames to predict both the P-frames and the B-frames.
- Support for temporal prediction mechanisms using variable block size in the motion compensation procedure, such as those used in the H.264/AVC [31] video standard.
- Support for spatial prediction mechanisms of INTRA type frames: this technique is adopted by the most recently proposed video standards (e.g.: H.264/AVC [31]), where each INTRA-coded region may be predicted by considering the previously encoded areas as references.

4. Development and design of optimized arithmetic units targeting the requirements of the proposed transcoding algorithms and architectures for video composition. In addition, specially efficient frame memory modules and the corresponding access mechanisms should also be investigated, in order to allow an highly efficient concurrent execution of the several blocks that integrate a video transcoding system.

5. Development and design of dedicated computational structures to implement the proposed transcoding algorithms and architectures, by adopting concurrent and parallel execution models of the several computational units. In particular, the following computational paradigms deserve a special attention:

   - System-On-Chip (SOC) *versus* System-On-Board (SOB) alternatives to implement high performance embedded systems for video transcoding, as well as a detailed study and a comparison analysis of the specific requirements and performance potentials of each of these alternatives;
   - Multi-core parallel systems, where some particularly interesting research aspects concerning the adopted computational architectures, the inter-core communication techniques, as well as the subjacent memory access mechanisms deserve further research. Some examples of these systems are the Cell architecture [68], from IBM, and the several recently proposed stream-based computing models, based on Graphics Processing Units (GPUs) [70].

These lines of research represent an interesting set of promising directions to continue and further develop the investigation and the set of contributions presented in this

thesis about high performance transform domain video transcoding systems for video composition.

References

[26] *MPEG-1: ISO/IEC JTC1 CD 11172 - "Coding of moving pictures and associated audio for digital storage media up to 1.5 Mbit/s – Part 2: Video"*, ISO, 1992.

[27] *MPEG-2: ISO/IEC JTC1 CD 13818 - "Generic coding of moving pictures and associated audio – Part 2: Video"*, ISO, 1994.

[28] *MPEG-4: ISO/IEC 14496-2:2004. Information technology – Coding of audio-visual objects – Part 2: Visual*, ISO, 2004.

[29] *ITU-T Recommendation H.261 - "Video Codec for Audiovisual Services at* $p\times64$ *Kbit/s"*, ITU-T, Mar. 1993.

[30] *ITU-T Recommendation H.263 - "Video Coding for Low Bitrate Communication"*, ITU-T, Feb. 1998.

[31] *ITU-T Recommendation H.264, "Advanced Video Coding for Generic Audiovisual Services"*, ITU-T, May 2003.

[68] K. O'Brien, A. Eichenberger, K. O'Brien, M. Gschwind, and P. Wu, "Architecture and compilation techniques for CELL (tutorial)," in *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques (PACT)*, Sep. 2005.

[70] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. E. Lefohn, and T. J. Purcell, "A survey of general-purpose computation on graphics hardware," *Computer Graphics Forum*, vol. 26, no. 1, pp. 80–113, 2007.

Appendices

A

Application of the pseudo-phases shift estimation to 2-D signals

The pseudo-phases technique for extracting shift values from the DCT pseudo-phases, described in subsection D of section 3.3.5, (page 113), can be extended to the 2-D case and be directly applied to the problem of motion estimation. To clarify the following description, it will be assumed a translation motion model in which a given object moves translationally by m_x in the horizontal direction and by m_y in the vertical direction, between the time instances corresponding to two consecutive frames, taken at instants $t-1$ and t, respectively. The formulation of such model was illustrated in fig. 3.30 and is also reproduced in fig. A.1, to ease the following description.

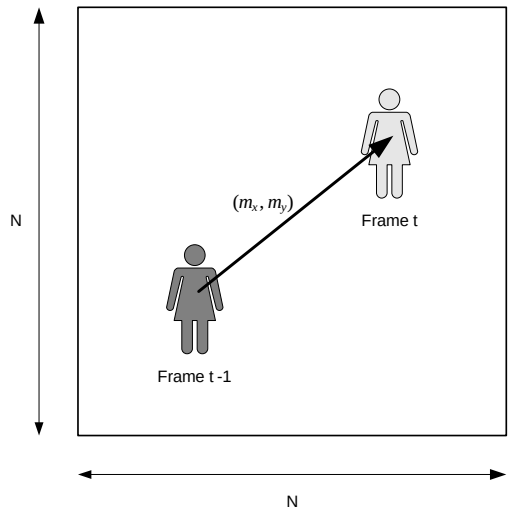


Figure A.1: Adopted translation motion model.

By following a procedure entirely similar to the 1-D case, the extension of the DCT pseudo-phases technique is accomplished by computing a set of required trigonometric transforms. Hence, the current frame (x_t) should be fed into 2-D DCT/DST cores to calculate the following transforms: DCCT-IIe, DCST-IIe, DSCT-IIe and DSST-IIe. Each of these transforms is defined as a 2-D separable function, computed with 1-D DCT-IIe/DST-IIe kernels.

$$X_t^{cc}(k,l) = \frac{4}{N^2}\,\xi(k)\,\xi(l)\sum_{m,n=0}^{N-1} x_t(m,n)\cos\left(\frac{k\left(m+\frac{1}{2}\right)\pi}{N}\right)\cos\left(\frac{l\left(n+\frac{1}{2}\right)\pi}{N}\right),$$
$$\text{with } k,l \in \{0,\ldots,N-1\} \tag{A.1}$$

$$X_t^{cs}(k,l) = \frac{4}{N^2}\,\xi(k)\,\xi(l)\sum_{m,n=0}^{N-1} x_t(m,n)\cos\left(\frac{k\left(m+\frac{1}{2}\right)\pi}{N}\right)\sin\left(\frac{l\left(n+\frac{1}{2}\right)\pi}{N}\right),$$
$$\text{with } k \in \{0,\ldots,N-1\} \text{ and } l \in \{1,\ldots,N\} \tag{A.2}$$

$$X_t^{sc}(k,l) = \frac{4}{N^2}\,\xi(k)\,\xi(l)\sum_{m,n=0}^{N-1} x_t(m,n)\sin\left(\frac{k\left(m+\frac{1}{2}\right)\pi}{N}\right)\cos\left(\frac{l\left(n+\frac{1}{2}\right)\pi}{N}\right),$$
$$\text{with } k \in \{1,\ldots,N\} \text{ and } l \in \{0,\ldots,N-1\} \tag{A.3}$$

$$X_t^{ss}(k,l) = \frac{4}{N^2}\,\xi(k)\,\xi(l)\sum_{m,n=0}^{N-1} x_t(m,n)\sin\left(\frac{k\left(m+\frac{1}{2}\right)\pi}{N}\right)\sin\left(\frac{l\left(n+\frac{1}{2}\right)\pi}{N}\right),$$
$$\text{with } k,l \in \{1,\ldots,N\} \tag{A.4}$$

or symbolically,

$$\mathbf{X}_t^{cc} = \text{DCCT-IIe}(\mathbf{x}_t)$$
$$\mathbf{X}_t^{cs} = \text{DCST-IIe}(\mathbf{x}_t)$$
$$\mathbf{X}_t^{sc} = \text{DSCT-IIe}(\mathbf{x}_t)$$
$$\mathbf{X}_t^{ss} = \text{DSST-IIe}(\mathbf{x}_t)$$

Just like the 2-D DCT/DST cores of the first kind (DCCT-Ie, DCST-Ie, DSCT-Ie and DSST-Ie) of the previous frame ($\mathbf{x}_{t-1}$) should be computed with 1-D DCT-Ie/DST-Ie kernels:

$$Z_{t-1}^{cc}(k,l) = \frac{4}{N^2}\,\xi(k)\,\xi(l)\sum_{m,n=0}^{N-1} x_{t-1}(m,n)\cos\left(\frac{km\pi}{N}\right)\cos\left(\frac{ln\pi}{N}\right),$$
$$\text{with } k,l \in \{0,\ldots,N\} \tag{A.5}$$

$$Z_{t-1}^{cs}(k,l) = \frac{4}{N^2}\,\xi(k)\,\xi(l)\sum_{m,n=0}^{N-1} x_{t-1}(m,n)\cos\left(\frac{km\pi}{N}\right)\sin\left(\frac{ln\pi}{N}\right),$$
$$\text{with } k \in \{0,\ldots,N\} \text{ and } l \in \{1,\ldots,N-1\} \tag{A.6}$$

$$Z_{t-1}^{sc}(k,l) = \frac{4}{N^2}\,\xi(k)\,\xi(l)\sum_{m,n=0}^{N-1} x_{t-1}(m,n)\sin\left(\frac{km\pi}{N}\right)\cos\left(\frac{ln\pi}{N}\right),$$
$$\text{with } k \in \{1,\ldots,N-1\} \text{ and } l \in \{0,\ldots,N\} \tag{A.7}$$

$$Z_{t-1}^{ss}(k,l) = \frac{4}{N^2}\,\xi(k)\,\xi(l)\sum_{m,n=0}^{N-1} x_{t-1}(m,n)\sin\left(\frac{km\pi}{N}\right)\sin\left(\frac{ln\pi}{N}\right),$$
$$\text{with } k,l \in \{1,\ldots,N-1\} \tag{A.8}$$

or symbolically,

$$\begin{aligned}\mathbf{Z}_{t-1}^{cc} &= \text{DCCT-Ie}(\mathbf{x}_{t-1})\\ \mathbf{Z}_{t-1}^{cs} &= \text{DCST-Ie}(\mathbf{x}_{t-1})\\ \mathbf{Z}_{t-1}^{sc} &= \text{DSCT-Ie}(\mathbf{x}_{t-1})\\ \mathbf{Z}_{t-1}^{ss} &= \text{DSST-Ie}(\mathbf{x}_{t-1})\end{aligned}$$

Similarly to the 1-D case, it is possible to derive a set of equations to relate the DCT/DST coefficients of $x_{t-1}(m,n)$ with those of $x_t(m,n)$:

$$Z_{t-1}(k,l)\cdot\theta(k,l) = x_t(k,l), \quad \text{for } k,l \in \{1,\ldots,N-1\} \tag{A.9}$$

where:

$$Z_{t-1}(k,l) = \begin{bmatrix} +Z_{t-1}^{cc}(k,l) & -Z_{t-1}^{cs}(k,l) & -Z_{t-1}^{sc}(k,l) & +Z_{t-1}^{ss}(k,l)\\ +Z_{t-1}^{cs}(k,l) & +Z_{t-1}^{cc}(k,l) & -Z_{t-1}^{ss}(k,l) & -Z_{t-1}^{sc}(k,l)\\ +Z_{t-1}^{sc}(k,l) & -Z_{t-1}^{ss}(k,l) & +Z_{t-1}^{cc}(k,l) & -Z_{t-1}^{cs}(k,l)\\ +Z_{t-1}^{ss}(k,l) & +Z_{t-1}^{sc}(k,l) & +Z_{t-1}^{cs}(k,l) & +Z_{t-1}^{cc}(k,l)\end{bmatrix}, \tag{A.10}$$

$$\theta(k,l) = \begin{bmatrix} g_{m_xm_y}^{cc}(k,l)\\ g_{m_xm_y}^{cs}(k,l)\\ g_{m_xm_y}^{sc}(k,l)\\ g_{m_xm_y}^{ss}(k,l)\end{bmatrix} = \begin{bmatrix} \cos\left[\frac{k\pi}{n}\left(m_x+\frac{1}{2}\right)\right]\cos\left[\frac{l\pi}{n}\left(m_y+\frac{1}{2}\right)\right]\\ \cos\left[\frac{k\pi}{n}\left(m_x+\frac{1}{2}\right)\right]\sin\left[\frac{l\pi}{n}\left(m_y+\frac{1}{2}\right)\right]\\ \sin\left[\frac{k\pi}{n}\left(m_x+\frac{1}{2}\right)\right]\cos\left[\frac{l\pi}{n}\left(m_y+\frac{1}{2}\right)\right]\\ \sin\left[\frac{k\pi}{n}\left(m_x+\frac{1}{2}\right)\right]\sin\left[\frac{l\pi}{n}\left(m_y+\frac{1}{2}\right)\right]\end{bmatrix} \tag{A.11}$$

and

$$x_t(k,l) = \begin{bmatrix} X_t^{cc}(k,l)\\ X_t^{cs}(k,l)\\ X_t^{sc}(k,l)\\ X_t^{ss}(k,l)\end{bmatrix}. \tag{A.12}$$

The matrix $Z_{t-1}(k,l) \in \mathbb{R}^{4\times4}$ is denoted as the *system matrix* at (k,l). At the boundaries of each block, the DCT coefficients of $x_{t-1}(m,n)$ and $x_t(m,n)$ have an 1-D relationship, as given by:

$$\begin{bmatrix} +Z_{t-1}^{cc}(k,l) & -Z_{t-1}^{cs}(k,l)\\ +Z_{t-1}^{cs}(k,l) & +Z_{t-1}^{cc}(k,l)\end{bmatrix}\begin{bmatrix}\cos\left[\frac{l\pi}{N}\left(m_y+\frac{1}{2}\right)\right]\\ \sin\left[\frac{l\pi}{N}\left(m_y+\frac{1}{2}\right)\right]\end{bmatrix} = \begin{bmatrix} X_t^{cc}(k,l)\\ X_t^{cs}(k,l)\end{bmatrix}$$
$$\text{for } k=0,\ l\in\{1,\ldots,N-1\}, \tag{A.13}$$

$$\begin{bmatrix} +Z_{t-1}^{cc}(k,l) & -Z_{t-1}^{sc}(k,l) \\ +Z_{t-1}^{sc}(k,l) & +Z_{t-1}^{cc}(k,l) \end{bmatrix} \begin{bmatrix} \cos\left[\frac{k\pi}{N}\left(m_x+\frac{1}{2}\right)\right] \\ \sin\left[\frac{k\pi}{N}\left(m_x+\frac{1}{2}\right)\right] \end{bmatrix} = \begin{bmatrix} X_t^{cc}(k,l) \\ X_t^{sc}(k,l) \end{bmatrix}$$
$$\text{for } l=0,\ k\in\{1,\ldots,N-1\}, \quad \text{(A.14)}$$

$$\begin{bmatrix} +Z_{t-1}^{cc}(k,l) & -Z_{t-1}^{cs}(k,l) \\ +Z_{t-1}^{cs}(k,l) & +Z_{t-1}^{cc}(k,l) \end{bmatrix} \begin{bmatrix} \cos\left[\frac{l\pi}{N}\left(m_y+\frac{1}{2}\right)\right] \\ \sin\left[\frac{l\pi}{N}\left(m_y+\frac{1}{2}\right)\right] \end{bmatrix} = (-1)^{m_x} \begin{bmatrix} X_t^{sc}(k,l) \\ X_t^{ss}(k,l) \end{bmatrix}$$
$$\text{for } k=N,\ l\in\{1,\ldots,N-1\}, \quad \text{(A.15)}$$

$$\begin{bmatrix} +Z_{t-1}^{cc}(k,l) & -Z_{t-1}^{sc}(k,l) \\ +Z_{t-1}^{sc}(k,l) & +Z_{t-1}^{cc}(k,l) \end{bmatrix} \begin{bmatrix} \cos\left[\frac{k\pi}{N}\left(m_x+\frac{1}{2}\right)\right] \\ \sin\left[\frac{k\pi}{N}\left(m_x+\frac{1}{2}\right)\right] \end{bmatrix} = (-1)^{m_y} \begin{bmatrix} X_t^{cs}(k,l) \\ X_t^{ss}(k,l) \end{bmatrix}$$
$$\text{for } l=N,\ k\in\{1,\ldots,N-1\}, \quad \text{(A.16)}$$

$$(-1)^{m_y}\, Z_{t-1}^{cc}(k,l) = X_t^{cs}(k,l)$$
$$\text{for } k=0,\ l=N, \quad \text{(A.17)}$$

$$(-1)^{m_x}\, Z_{t-1}^{cc}(k,l) = X_t^{sc}(k,l)$$
$$\text{for } k=N,\ l=0. \quad \text{(A.18)}$$

Four different directions may be considered for an object displacement in a 2-D space: NE ($m_x > 0,\ m_y > 0$), NW ($m_x < 0,\ m_y > 0$), SE ($m_x > 0,\ m_y < 0$), SW ($m_x < 0,\ m_y < 0$). As it was seen for the 1-D case, the orthogonal equation for the DST-IIe kernel (see eq. 3.140) can be applied to the pseudo-phase $g_m^s(k)$ to determine m and to obtain the direction and magnitude of the shift. On the other hand, to characterize de displacement (m_x, m_y) in the 2-D case, it is necessary to compute the pseudo-phases $g_{m_x m_y}^{sc}$ and $g_{m_x m_y}^{cs}$. By taking the previously defined block boundary equations into consideration (see eq. A.13-A.18), two pseudo-phases functions can be defined as follows:

$$f_{m_x m_y}(k,l) = \begin{cases} g_{m_x m_y}^{cs}(k,l), & \text{for } k,l\in\{1,\ldots,N-1\}, \\ \frac{Z_{t-1}^{cc}(k,l)X_t^{cs}(k,l) - Z_{t-1}^{cs}(k,l)X_t^{cc}(k,l)}{\left[Z_{t-1}^{cc}(k,l)\right]^2+\left[Z_{t-1}^{cs}(k,l)\right]^2}, & \text{for } k=0, l\in\{1,\ldots,N-1\}, \\ \frac{Z_{t-1}^{cc}(k,l)X_t^{cs}(k,l) + Z_{t-1}^{sc}(k,l)X_t^{ss}(k,l)}{\left[Z_{t-1}^{cc}(k,l)\right]^2+\left[Z_{t-1}^{sc}(k,l)\right]^2}, & \text{for } l=N, k\in\{1,\ldots,N-1\}, \\ \frac{X_t^{cs}(k,l)}{Z_{t-1}^{cc}(k,l)}, & \text{for } k=0, l=N, \end{cases}$$
$$\text{(A.19)}$$

$$g_{m_x m_y}(k,l) = \begin{cases} g^{sc}_{m_x m_y}(k,l), & \text{for } k,l \in \{1,\ldots,N-1\}, \\ \frac{Z^{cc}_{t-1}(k,l)X^{sc}_t(k,l) - Z^{sc}_{t-1}(k,l)X^{cc}_t(k,l)}{\left[Z^{cc}_{t-1}(k,l)\right]^2 + \left[Z^{sc}_{t-1}(k,l)\right]^2}, & \text{for } l = 0, k \in \{1,\ldots,N-1\}, \\ \frac{Z^{cc}_{t-1}(k,l)X^{sc}_t(k,l) + Z^{cs}_{t-1}(k,l)X^{ss}_t(k,l)}{\left[Z^{cc}_{t-1}(k,l)\right]^2 + \left[Z^{cs}_{t-1}(k,l)\right]^2}, & \text{for } k = N, l \in \{1,\ldots,N-1\}, \\ \frac{X^{sc}_t(k,l)}{Z^{cc}_{t-1}(k,l)}, & \text{for } k = N, l = 0. \end{cases} \tag{A.20}$$

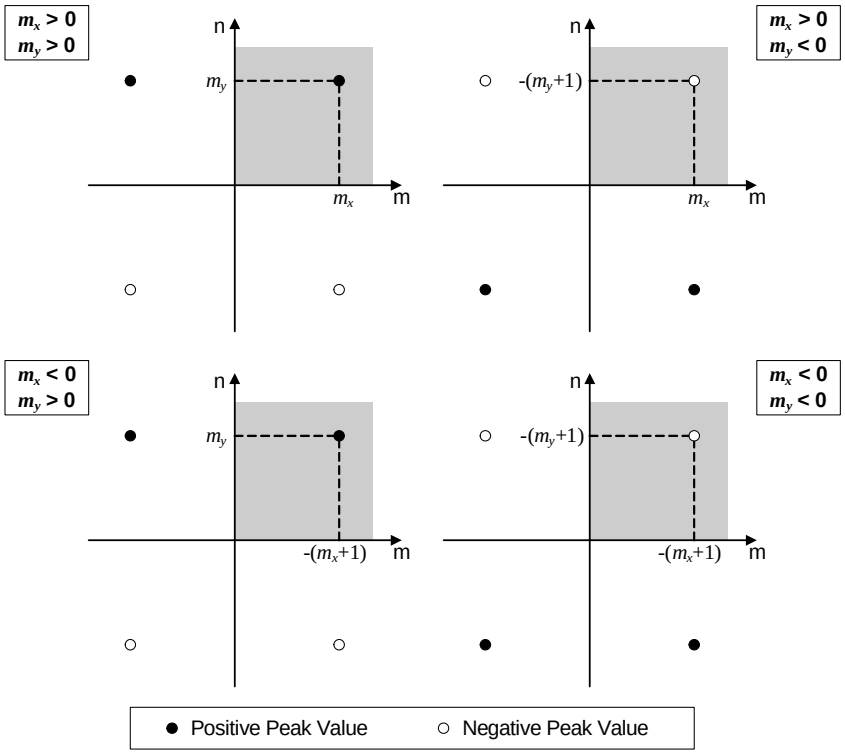
By computing the inverse transform functions (IDCST-IIe and IDSCT-IIe) of these two pseudo-phases, and by taking into account the orthogonal properties of the even type-II cosine and sine transforms (see eqs. 3.140 and 3.141), it is possible to obtain:

$$\begin{aligned} \mathrm{DCS}(m,n) &= \text{IDCST-IIe}\left(\xi(k)\xi(l)f_{m_x m_y}(k,l)\right) \\ &= \frac{4}{N^2}\sum_{k=0}^{N-1}\sum_{l=1}^{N}\xi^2(k)\xi^2(l)f_{m_x m_y}(k,l)\cos\left[\frac{k\pi}{N}\left(m+\frac{1}{2}\right)\right]\sin\left[\frac{l\pi}{N}\left(n+\frac{1}{2}\right)\right] \\ &= \left[\delta(m-m_x)+\delta(m+m_x+1)\right]\cdot\left[\delta(m-m_y)-\delta(m+m_y+1)\right], \end{aligned} \tag{A.21}$$

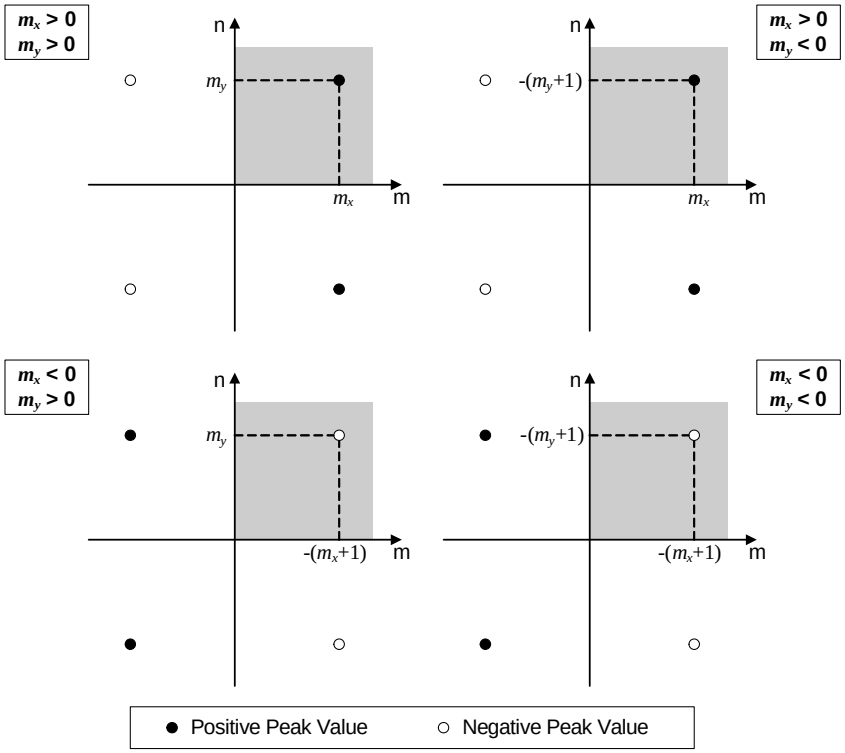
$$\begin{aligned} \mathrm{DSC}(m,n) &= \text{IDSCT-IIe}\left(\xi(k)\xi(l)g_{m_x m_y}(k,l)\right) \\ &= \frac{4}{N^2}\sum_{k=1}^{N}\sum_{l=0}^{N-1}\xi^2(k)\xi^2(l)g_{m_x m_y}(k,l)\sin\left[\frac{k\pi}{N}\left(m+\frac{1}{2}\right)\right]\cos\left[\frac{l\pi}{N}\left(n+\frac{1}{2}\right)\right] \\ &= \left[\delta(m-m_x)-\delta(m+m_x+1)\right]\cdot\left[\delta(m-m_y)+\delta(m+m_y+1)\right]. \end{aligned} \tag{A.22}$$

Similarly to the 1-D case, the above 2-D IDCT-IIe and IDST-IIe inverse transforms limit the observable index space of the DCS and DSC functions, defined in eqs. A.21 and A.22, to the first quadrant of the entire index space, as it is shown in fig. A.2 (represented as gray shaded areas). If m_x is positive, the observable peak value of $\mathrm{DSC}(m,n)$ will be positive, independently of the sign of m_y, since $\mathrm{DSC}(m,n) = \delta(m-m_x)\cdot[\delta(n-m_y)+\delta(n+m_y+1)]$ in the observable index space. Similarly, if m_x is negative, the observable peak value of $\mathrm{DSC}(m,n)$ will be negative, since $\mathrm{DSC}(m,n) = -\delta(m+m_x+1)\cdot[\delta(n-m_y)+\delta(n+m_y+1)]$ in the observable index space. Hence, the sign of the observable peak value of $\mathrm{DSC}(m,n)$ determines the sign of m_x. The same observation can be formulated by applying $\mathrm{DCS}(m,n)$ to the evaluation of the sign of m_y.

Consequently, the displacement $d = (m_x, m_y)$ can be obtained by locating the peaks of $\mathrm{DSC}(m,n)$ and $\mathrm{DCS}(m,n)$ in the observable region $\mathbf{\Phi} = \{0,\ldots,N-1\}^2$.



(a) From DCS.



(b) From DSC.

Figure A.2: Application of the sinusoidal orthogonal principle to DCS and DSC pseudo-phases to evaluate the direction and magnitude of the shift between two 2-D signals.

Table A.1: Methodology to determine the direction of the displacement (m_x, m_y) from the signs of DSC(m,n) and DCS(m,n) in the observable region $\mathbf{\Phi} = \{0, \ldots, N-1\}^2$.

Sign of DSC Peak	Sign of DCS Peak	Peak Index	Motion Direction
+	+	(m_x, m_y)	NE
+	-	$(m_x, -(m_y+1))$	SE
-	+	$(-(m_x+1), m_y)$	NW
-	-	$(-(m_x+1), -(m_y+1))$	SW

In table A.1 it is summarized the methodology to determine the direction of the displacement (m_x, m_y) from the signs of DSC(m,n) and DCS(m,n) functions. Once the direction is found, the displacement magnitude d can be obtained as:

$$m_x = \begin{cases} i_{\mathrm{DSC}} = i_{\mathrm{DCS}}, & \text{if } \mathrm{DSC}(i_{\mathrm{DSC}}, j_{\mathrm{DSC}}) > 0, \\ -(i_{\mathrm{DSC}}+1) = -(i_{\mathrm{DCS}}+1), & \text{if } \mathrm{DSC}(i_{\mathrm{DSC}}, j_{\mathrm{DSC}}) < 0 \end{cases} \tag{A.23}$$

$$m_y = \begin{cases} j_{\mathrm{DCS}} = j_{\mathrm{DSC}}, & \text{if } \mathrm{DCS}(i_{\mathrm{DCS}}, j_{\mathrm{DCS}}) > 0, \\ -(j_{\mathrm{DCS}}+1) = -(j_{\mathrm{DSC}}+1), & \text{if } \mathrm{DCS}(i_{\mathrm{DCS}}, j_{\mathrm{DCS}}) < 0 \end{cases} \tag{A.24}$$

where:

$$(i_{\mathrm{DCS}}, j_{\mathrm{DCS}}) = \arg \max_{m,n \in \mathbf{\Phi}} |\mathrm{DCS}(m,n)|\,, \tag{A.25}$$

$$(i_{\mathrm{DSC}}, j_{\mathrm{DSC}}) = \arg \max_{m,n \in \mathbf{\Phi}} |\mathrm{DSC}(m,n)|\,. \tag{A.26}$$

Koc and Liu empirically formulated a peak selection rule to choose the best (i_D, j_D) index of these peaks in situations where the presence of noise makes them inconsistent. Such proposed rule was stated in terms of a minimum Non-Peak-to-Peak Ratio (NPR). This measure is defined as the ratio of the average of all absolute non-peak values to the absolute peak value, giving rise to values in the interval $0 \leq \mathrm{NPR} \leq 1$, and presenting the null value ($\mathrm{NPR} = 0$) when applied to an impulse function $\delta(m,n)$. This arbitration rule is stated as follows:

$$(i_D, j_D) = \begin{cases} (i_{\mathrm{DSC}}, j_{\mathrm{DSC}})\,, & \text{if } \mathrm{NPR(DSC)} < \mathrm{NPR(DCS)}, \\ (i_{\mathrm{DCS}}, j_{\mathrm{DCS}})\,, & \text{if } \mathrm{NPR(DSC)} > \mathrm{NPR(DCS)}. \end{cases} \tag{A.27}$$

Furthermore, in situations where slow motion is preferred, Koc and Liu suggested looking for the peak value in a zig-zag search pattern along the several coefficients of the block, as usually used in run-length coding algorithms adopted in standard video coders. By starting from the index $(0,0)$, the DCS or DSC peaks are scanned

according to a zig-zag search pattern, marking each point as the new peak index if the value at that point (i, j) is larger than the current peak value by more than a considered threshold. By adopting this procedure, large spurious spikes at the higher index points will not affect the performance and also improve the noise immunity.

B

Computational cost efficiency of the NRSO insertion transcoders

This appendix presents complementary experimental results concerning the computational load required by the proposed NRSO insertion transcoding architectures. The experimental setup is entirely similar to the evaluation procedure that was described in section 6.2.3 and was applied to insert the logo and subtitle presented in figs. B.1(a) and B.1(b) in the *Silent* and *Carphone* video sequences, illustrated in figs. B.2(a) and B.2(b), respectively. Similarly to what was done in section 6.2.3 two quantization setups were considered: $Q = 4$ and $Q = 15$.

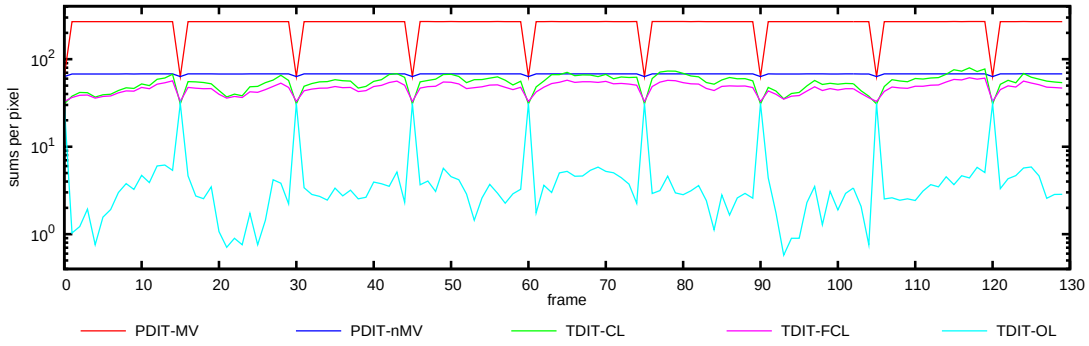
(a) Logo. **(b)** Subtitle.

Figure B.1: Considered set of NRSOs ($C_T = 0$).

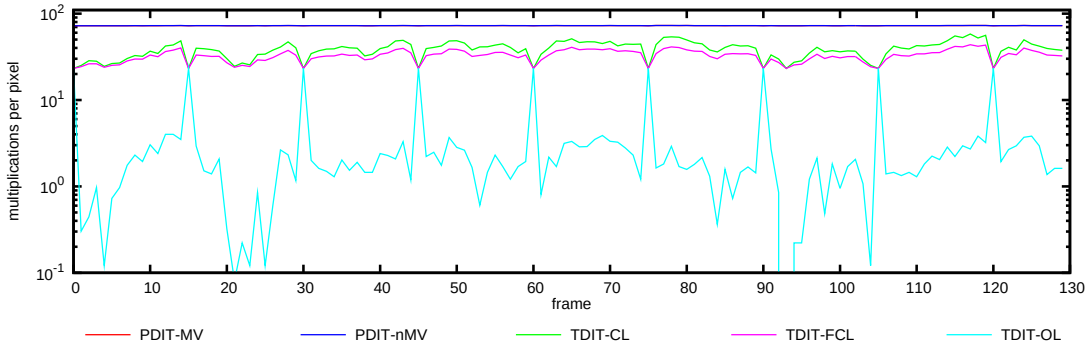
(a) *Silent-Voice*. **(b)** *Carphone*.

Figure B.2: *Silent-Voice* and *Carphone* test video sequences.

In figs. B.3 through B.6 it is presented the observed variations of the number of required additions and multiplications for the considered video sequences. To accommodate the high dynamic range of the number of performed operations presented in all these figures, these charts were represented using a logarithmic scale in the y axis.

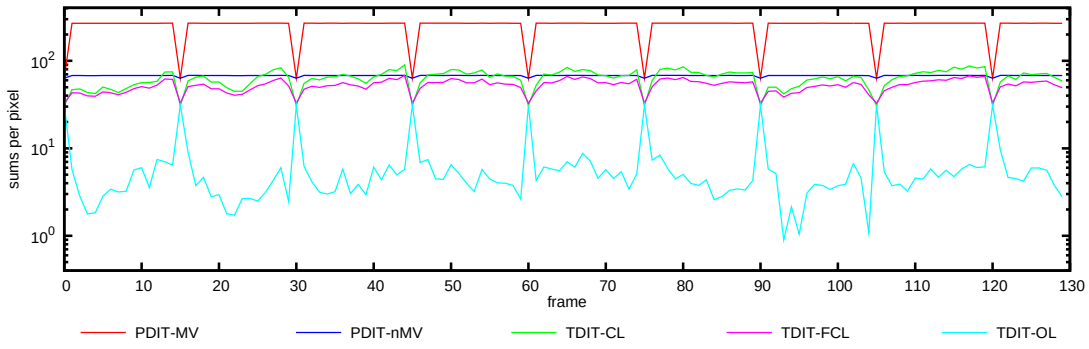


(a) Additions.

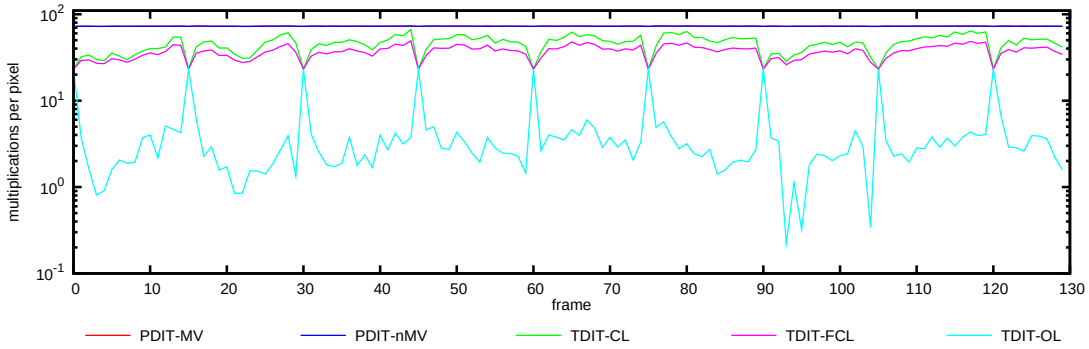


(b) Multiplications.

Figure B.3: Number of operations required to insert the considered NRSOs in the *Silent-Voice* video sequence with $Q = 4$.

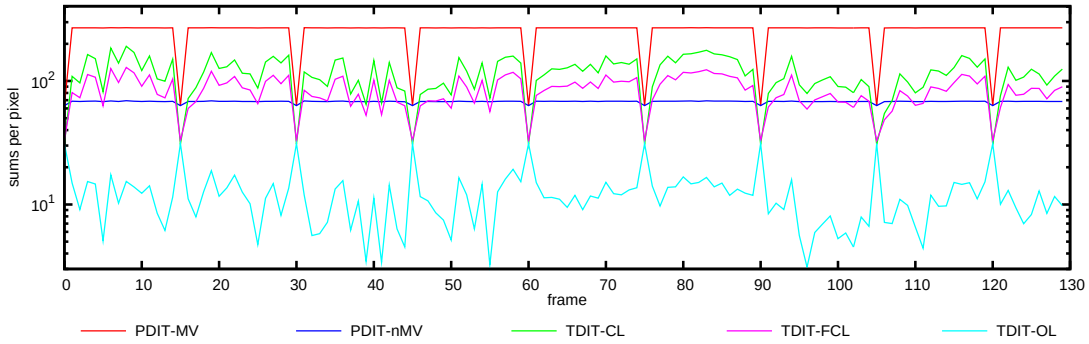


(a) Additions.

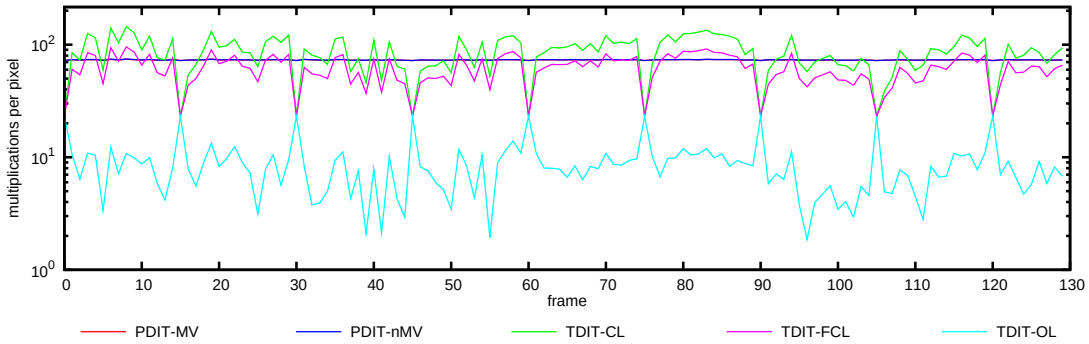


(b) Multiplications.

Figure B.4: Number of operations required to insert the considered NRSOs in the *Silent-Voice* video sequence with $Q = 15$.

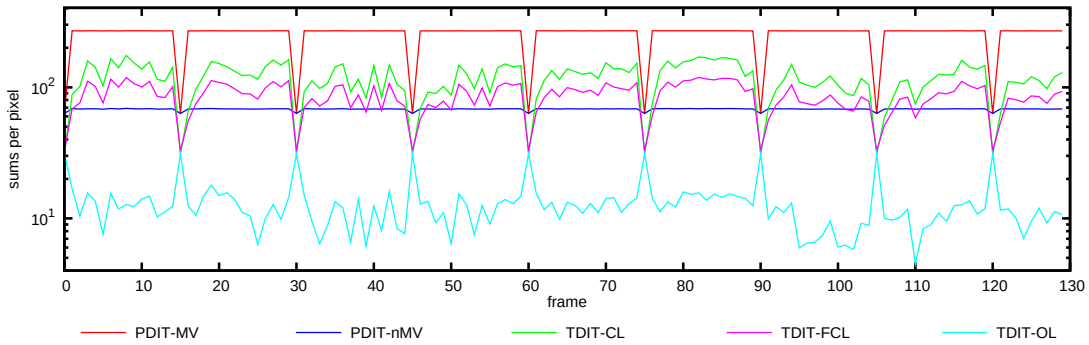


(a) Additions.

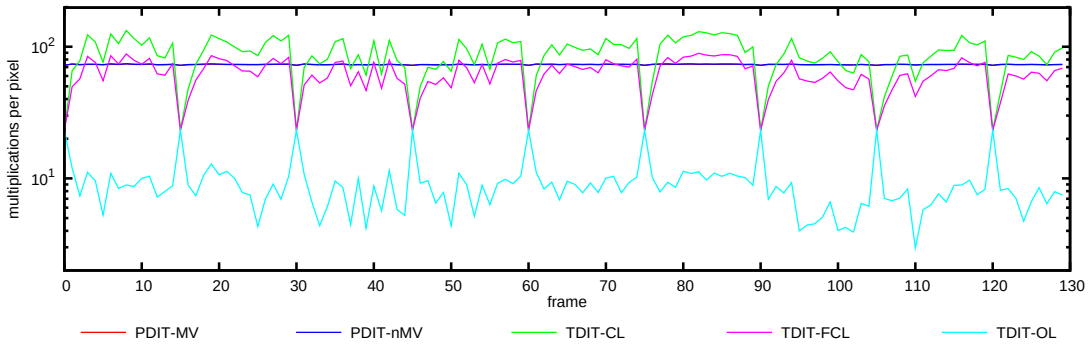


(b) Multiplications.

Figure B.5: Number of operations required to insert the considered NRSOs in the *Carphone* video sequence with $Q = 4$.



(a) Additions.



(b) Multiplications.

Figure B.6: Number of operations required to insert the considered NRSOs in the *Carphone* video sequence with $Q = 15$.

Bibliography

[1] I. Ahmad, X. Wei, Y. Sun, and Y.-Q. Zhang, "Video transcoding: An overview of various techniques and research issues," *IEEE Transactions on Multimedia*, vol. 7, no. 5, pp. 793–804, Oct. 2005.

[2] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Transactions on Computers*, vol. C-23, no. 1, pp. 90–93, 1974.

[3] Y. Arai, T. Agui, and M. Nakajima, "A fast DCT-SQ scheme for images," *Transactions of the Institute of Electronics, Information and Communication Engineers (IEICE)*, vol. E71, no. 11, pp. 1095–1097, 1988.

[4] P. Assunção and M. Ghanbari, "Buffer analysis and control in CBR video transcoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, no. 1, pp. 83–92, Feb. 2000.

[5] P. Assunção and M. Ghanbari, "Post-processing of MPEG2 coded video for transmission at lower bit rates," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Gorgia USA, May 1996, pp. 1998–2001.

[6] P. Assunção and M. Ghanbari, "A frequency-domain video transcoder for dynamic bit-rate reduction of MPEG-2 bitstreams," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 8, pp. 953–967, Dec. 1998.

[7] V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards: Algorithms and Architectures*, 2nd ed. Kluwer Academic Publishers, Jun. 1997.

[8] W.-K. Cham, "Family of order-4 four-level orthogonal transforms," *IEE Electronic Letters*, vol. 19, no. 21, pp. 869–871, Oct. 1983.

[9] S.-F. Chang and D. G. Messerschmitt, "Compositing motion-compensated video within the network," in *Proceedings of the International Workshop on Multimedia Communications (MULTIMEDIA)*. IEEE, Apr. 1992, pp. 40–56.

[10] S.-F. Chang and D. G. Messerschmitt, "A new approach to decoding and compositing motion-compensated DCT-based images," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 5. IEEE, apr 1993, pp. 421–424.

[11] S.-F. Chang and D. G. Messerschmitt, "Manipulation and compositing of MC-DCT compressed video," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 1, pp. 1–11, Jan. 1995.

[12] J. Chen, U.-V. Koc, and K. J. R. Liu, *Design of Digital Video Coding Systems - A Complete Compressed Domain Approach*. Marcel Dekker, 2002.

[13] M.-J. Chen, M.-C. Chu, and C.-W. Pan, "Efficient motion-estimation algorithm for reduced frame-rate video transcoder," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 4, pp. 269–275, Apr. 2002.

[14] W.-H. Chen, C. H. Smith, and S. C. Fralick, "A fast computational algorithm for the discrete cosine transform," *IEEE Transactions on Communications*, vol. COM-25, pp. 1004–1009, Sep. 1977.

[15] "Condor webpage," http://www.cs.wisc.edu/condor, 2008.

[16] R. Dugad and N. Ahuja, "A fast scheme for image size change in the compressed domain," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 4, pp. 461–474, Apr. 2001.

[17] A. Eleftheriadis and D. Anastassiou, "Constrained and general dynamic rate shaping of compressed digital video," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, Arlington, Virginia, U.S.A., Oct. 1995.

[18] I. Foster, *The Grid: Blueprint for a New Computing Infrastructure*, 2nd ed. Morgan Kaufmann, 2004.

[19] I. Foster, "Globus toolkit version 4: Software for service-oriented systems," in *Proceedings of the IFIP International Conference on Network and Parallel Computing (NPC)*, vol. LNCS 3779. Springer-Verlag, 2006, pp. 2–13.

[20] K.-T. Fung, Y.-L. Chan, and W.-C. Siu, "Dynamic frame skipping for high-performance transcoding," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, Oct. 2001, pp. 425–428.

[21] "Globus webpage," http://www.globus.org, 2008.

[22] A. Haar, "Zur theorie der orthogonalen funktionen-systeme [on the theory of orthogonal function systems]," *Mathematische Annalen*, no. 69, pp. 331–371, 1910.

[23] H. Hedberg and P. Nilsson, "A survey of various discrete transforms used in digital image compression algorithms," in *Proceedings of the Swedish System-On-Chip Conference*, Bastad - Sweden, Apr. 2004.

[24] Q. Hu and S. Panchanathan, "Image/video spatial scalability in compressed domain," *IEEE Transactions on Industrial Electronics*, vol. 45, no. 1, pp. 23–31, Feb. 1998.

[25] J.-N. Hwang, T.-D. Wu, and C.-W. Lin, "Dynamic frame-skipping in video transcoding," in *Proceedings of the IEEE International Workshop on Multimedia Signal Processing (MMSP)*, 1998, pp. 616–621.

[26] *MPEG-1: ISO/IEC JTC1 CD 11172 - "Coding of moving pictures and associated audio for digital storage media up to 1.5 Mbit/s – Part 2: Video"*, ISO, 1992.

[27] *MPEG-2: ISO/IEC JTC1 CD 13818 - "Generic coding of moving pictures and associated audio – Part 2: Video"*, ISO, 1994.

[28] *MPEG-4: ISO/IEC 14496-2:2004. Information technology – Coding of audio-visual objects – Part 2: Visual*, ISO, 2004.

[29] *ITU-T Recommendation H.261 - "Video Codec for Audiovisual Services at* $p \times 64$ *Kbit/s"*, ITU-T, Mar. 1993.

[30] *ITU-T Recommendation H.263 - "Video Coding for Low Bitrate Communication"*, ITU-T, Feb. 1998.

[31] *ITU-T Recommendation H.264, "Advanced Video Coding for Generic Audiovisual Services"*, ITU-T, May 2003.

[32] *JPEG: ITU-T Recommendation T.81 - "Digital compression and coding of continuous-tone still images"*, ITU-T, 1993.

[33] *ITU-T/SG16/VCEG (Q.6), H.26L Test Model Long-Term Number 8 (TML-8)*, ITU-T, Video Coding Experts Group (VCEG), Sep. 2001.

[34] A. K. Jain, *Fundamentals of Digital Image Processing.* Prentice Hall, 1989.

[35] J. R. Jain and A. K. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Transactions on Communications*, vol. COM-29, no. 12, pp. 1799–1808, Dec. 1981.

[36] G. Keesman, R. Hellinghuizen, F. Hoeksema, and G. Heideman, "Transcoding of MPEG bitstreams," *Signal Processing: Image Communication*, vol. 8, pp. 481–500, 1996.

[37] B. Kernighan and D. Ritchie, *The C Programming Language*, 2nd ed. Prentice Hall Software, 1988.

[38] U.-V. Koc and K. J. R. Liu, "Low-complexity motion estimation scheme utilizing sinusoidal orthogonal principle," in *Proceedings of the IEEE International Workshop on Visual Signal Processing and Communications (VSPC)*, New Brunswick, NJ, Sep. 1994, pp. 57–62.

[39] U.-V. Koc and K. J. R. Liu, "Discrete-cosine/sine-transform based motion estimation," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, vol. 3, Austin, Texas, Nov. 1994, pp. 771–775.

[40] U.-V. Koc and K. J. R. Liu, "Adaptive overlapping approach for DCT-based motion estimation," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, Washington, DC, 1995.

[41] U.-V. Koc and K. J. R. Liu, "DCT-based motion estimation," *IEEE Transactions on Image Processing*, vol. 7, no. 7, pp. 948–965, Jul. 1998.

[42] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion-compensated interframe coding for video conferencing," in *Proceedings of the National Telecommunications Conference*, New Orleans, LA, Nov. 1981, pp. G5.3.1–G5.3.5.

[43] W. Kou and T. Fjällbrant, "A direct computation of DCT coefficients for a signal block taken from two adjacent blocks," *IEEE Transactions on Signal Processing*, vol. 39, no. 7, pp. 1692–1695, Jul. 1991.

[44] R. Kresch and N. Merhav, "Fast DCT domain filtering using the DCT and the DST," *IEEE Transactions on Image Processing*, vol. 8, no. 6, pp. 821–833, Jun. 1999.

[45] A. Y. Lan and J.-N. Hwang, "Scene context dependent reference frame placement for MPEG videocoding," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 4, Munich - Germany, Apr. 1997, pp. 2997–3000.

[46] Y.-R. Lee and C.-W. Lin, "DCT-domain spatial transcoding using generalized DCT decimation," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, Genoa - Italy, Sep. 2005.

[47] Y.-R. Lee, C.-W. Lin, and C.-C. Kao, "A DCT-domain video transcoder for spatial resolution downconversion," in *Proceedings of the International Conference on Recent Advances in Visual Information Systems*, Mar. 2002, pp. 207–218.

[48] Y.-R. Lee, C.-W. Lin, S.-H. Yeh, and Y.-C. Chen, "Low-complexity DCT-domain video transcoders for arbitrary-size downscaling," in *Proceedings of the IEEE International Workshop on Multimedia Signal Processing (MMSP)*, Sep. 2004, pp. 31–34.

[49] C.-H. Li, H. Lin, C.-N. Wang, and T. Chiang, "A fast H.264-based picture-in-picture (PIP) transcoder," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, vol. 3. IEEE, Jun. 2004, pp. 1691–1694.

[50] C.-H. Li, C.-N. Wang, and T. Chiang, "A low complexity picture-in-picture transcoder for video-on-demand," in *IEEE International Conference on Wireless Networks, Communications and Mobile Computing*, vol. 2, Jun. 2005, pp. 1382–1387.

[51] H. Li and H. Shi, "A fast algorithm for reconstructing motion compensated blocks in compressed domain," *Journal of Visual Languages and Computing*, vol. 10, no. 6, pp. 607–623, Dec. 1999.

[52] Y. Liang, L.-P. Chau, and Y.-P. Tan, "Arbitrary downsizing video transcoding using fast motion vector re-estimation," *IEEE Signal Processing Letters*, vol. 9, no. 11, pp. 352–355, Nov. 2002.

[53] J. S. Lim, *Two-Dimensional Signal and Image Processing.* Prentice-Hall, 1990.

[54] C.-W. Lin and Y.-R. Lee, "Fast algorithms for DCT-domain video transcoding," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, Thessaloniki - Greece, Oct. 2001, pp. 421–424.

[55] B. Liu and A. Zaccarin, "New fast algorithms for the estimation of block motion vectors," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 3, no. 2, pp. 148–157, Apr. 1993.

[56] S. Liu and A. C. Bovik, "Local bandwidth constrained fast inverse motion compensation for DCT domain video transcoding," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Salt Lake City, UT, May 2001.

[57] S. Liu and A. C. Bovik, "Local bandwidth constrained fast inverse motion compensation for DCT-domain video transcoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 5, pp. 309–319, May 2002.

[58] S. A. Martucci, "Symmetric convolution and discrete sine and cosine transforms," *IEEE Transactions on Signal Processing*, vol. SP-42, no. 5, pp. 1038–1051, May 1994.

[59] S. A. Martucci, "Image resizing in the discrete cosine transform domain," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, vol. 2, Washington D.C. - USA, Oct. 1995, pp. 244–247.

[60] "Matlab webpage," http://www.mathworks.com/products/matlab, 2008.

[61] J. Meng and S.-F. Chang, "Embedding visible video watermarks in the compressed domain," *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, vol. 1, pp. 474–477, 1998.

[62] N. Merhav and V. Bhaskaran, "A fast algorithm for DCT-domain inverse motion compensation," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 4, Atlanta, GA, USA, May 1996, pp. 2307–2310.

[63] N. Merhav and V. Bhaskaran, "Fast algorithms for DCT-domain image downsampling and for inverse motion compensation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, no. 3, pp. 468–476, Jun. 1997.

[64] J. L. Mitchell, W. B. Pennebaker, C. E. Fogg, and D. J. Legall, *MPEG video compression standard*. Chapman & Hall, 1996.

[65] *MPEG-4: Video Verification Model - version 18.0 - ISO/MPEG N3908*, MPEG, Jan. 2001.

[66] B. K. Natarajan and B. Vasudev, "A fast approximate algorithm for scaling down digital images in the DCT domain," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, vol. 2, Washington D.C. - USA, Oct. 1995, pp. 241–243.

[67] Y. Noguchi, D. G. Messerschmitt, and S.-F. Chang, "MPEG video compositing in the compressed domain," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 2. IEEE, May 1996, pp. 596–599.

[68] K. O'Brien, A. Eichenberger, K. O'Brien, M. Gschwind, and P. Wu, "Architecture and compilation techniques for CELL (tutorial)," in *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques (PACT)*, Sep. 2005.

[69] "Octave webpage," http://www.octave.org, 2008.

[70] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. E. Lefohn, and T. J. Purcell, "A survey of general-purpose computation on graphics hardware," *Computer Graphics Forum*, vol. 26, no. 1, pp. 80–113, 2007.

[71] K. Panusopone, X. Chen, and F. Ling, "Logo insertion in MPEG transcoder," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Salt Lake City - USA, May 2001.

[72] Y. S. Park and H. W. Park, "Arbitrary-ratio image resizing using fast DCT of composite length for DCT-based transcoder," *IEEE Transactions on Image Processing*, vol. 15, no. 2, pp. 494–500, Feb. 2006.

[73] V. Patil, R. Kumar, and J. Mukherjee, "A fast arbitrary factor video resizing algorithm," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 9, pp. 1164–1171, Sep. 2006.

[74] F. Pereira and T. Ebrahimi, Eds., *The MPEG-4 Book*. Prentice Hall PTR, 2002.

[75] P. Pirsch, N. Demassieux, and W. Gehrke, "VLSI architectures for video compression - a survey," *Proceedings of the IEEE*, vol. 83, no. 2, pp. 220–246, Feb. 1995.

[76] B. Porat, *A Course in Digital Signal Processing*. John Wiley & Sons, Inc., 1997.

[77] T. Porter and T. Duff, "Compositing digital images," *Proceedings of the ACM International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, vol. 18, no. 3, pp. 253–259, Jul. 1984.

[78] W. K. Pratt, *Digital Image Processing*. John Wiley & Sons, Inc., 1978.

[79] W. K. Pratt, J. Kane, and H. C. Andrews, "Hadamard transform image coding," *Proceedings of the IEEE*, vol. 57, no. 1, pp. 58–68, Jan. 1969.

[80] W. K. Pratt, W.-H. Chen, and L. R. Welch, "Slant transform image coding," *IEEE Transactions on Communications*, vol. 22, no. 8, pp. 1075–1093, Aug. 1974.

[81] M. Püschel and J. M. F. Moura, "The algebraic approach to the discrete cosine and sine transforms and their fast algorithms," *Society for Industrial and Applied Mathematics Journal on Computing*, vol. 32, no. 5, pp. 1280–1316, 2003.

[82] K. R. Rao and P. Yip, *Discrete Cosine Transform: algorithms, advantages and applications*. Academic Press, Inc., 1990.

[83] R. Reeves, "Image matching in the compressed domain," Ph.D. dissertation, Queensland University of Technology, Australia, 1999.

[84] R. Reeves and K. Kubik, "Compressed domain image matching using symmetric convolution," in *Proceedings of IEEE Region 10 Annual Conference on Speech and Image Technologies for Computing and Telecommunications - TENCON'97*. Brisbane, Queensland: Queensland QUT Publications, Dec. 1997.

[85] J. Ridge, "Efficient transform-domain size and resolution reduction of images," *Signal Processing: Image Communication*, vol. 18, no. 8, pp. 621–639, Sep. 2003.

[86] N. Roma and L. Sousa, "Insertion of irregular-shaped logos in the compressed DCT domain," in *Proceedings of the IEEE International Conference on Digital Signal Processing (DSP)*, vol. 1. Santorini, Greece: IEEE, Jul. 2002, pp. 125–128.

[87] N. Roma and L. Sousa, "Transcoding architectures for object insertion in compressed video," INESC-ID – Lisboa, Portugal, Tech. Rep. RT/006/2002, Oct. 2002.

[88] N. Roma and L. Sousa, "Fast transcoding architectures for insertion of non-regular shaped objects in the compressed DCT-domain," *Signal Processing: Image Communication*, vol. 18, no. 8, pp. 659–683, Sep. 2003.

[89] N. Roma and L. Sousa, "Least squares motion estimation algorithm in the compressed DCT domain for H.26x/MPEG-x video sequences," in *Proceedings of the IEEE International Conference on Advanced Video and Signal-Based Surveillance (AVSS)*. Como - Italy: IEEE, Sep. 2005, pp. 576–581.

[90] N. Roma and L. Sousa, "Efficient hybrid DCT-domain algorithm for any arbitrary integer re-size video downscaling," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, no. 57291, pp. 1–16, Sep. 2007.

[91] N. Roma and L. Sousa, "Fully compressed-domain transcoder for PIP/PAP video composition," in *Proceedings of the Picture Coding Symposium (PCS)*, Lisbon - Portugal, Nov. 2007, pp. CD–ROM.

[92] C. L. Salazar and T. D. Tran, "On resizing images in the DCT domain," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, vol. 4, Oct. 2004, pp. 2797–2800.

[93] K.-D. Seo and J.-K. Kim, "Motion vector refinement for video downsampling in the DCT domain," *IEEE Signal Processing Letters*, vol. 9, no. 11, pp. 356–359, Nov. 2002.

[94] K.-D. Seo and J.-K. Kim, "Fast motion vector re-estimation for transcoding MPEG-1 into MPEG-4 with lower spatial resolution in DCT-domain," *Signal Processing: Image Communication*, vol. 19, no. 4, pp. 299–312, Apr. 2004.

[95] T. Shanableh and M. Ghanbari, "Heterogeneous video transcoding to lower spatio-temporal resolution and different encoding formats," *IEEE Transactions on Multimedia*, vol. 2, no. 2, pp. 101–110, Jun. 2000.

[96] T. Shanableh and M. Ghanbari, "Transcoding of video into different encoding formats," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 4, Jun. 2000, pp. 1927–1930.

[97] T. Shanableh and M. Ghanbari, "Transcoding architectures for DCT-domain heterogeneous video transcoding," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, Thessaloniki - Greece, Oct. 2001.

[98] T. Shanableh and M. Ghanbari, "Hybrid DCT/pixel domain architecture for heterogeneous video transcoding," *Signal Processing: Image Communication*, vol. 18, no. 8, pp. 601–620, Sep. 2003.

[99] B. Shen and I. K. Sethi, "Block-based manipulations of transformed-compressed images and videos," *ACM Multimedia System Journal*, vol. 6, no. 2, pp. 113–124, Mar. 1998.

[100] B. Shen, I. K. Sethi, and V. Bhaskaran, "DCT convolution and its application in compressed domain," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 8, pp. 947–952, Dec. 1998.

[101] B. Shen, I. Sethi, and B. Vasudev, "Adaptive motion-vector resampling for compressed video downscaling," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 6, pp. 929–936, Sep. 1999.

[102] H. Shu and L.-P. Chau, "An efficient arbitrary downsizing algorithm for video transcoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 6, pp. 887–891, Jun. 2004.

[103] H. Shu and L.-P. Chau, "A resizing algorithm with two-stage realization for DCT-based transcoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 2, pp. 248–253, Feb. 2007.

[104] B. C. Smith and L. A. Rowe, "Algorithms for manipulating compressed images," *IEEE Computer Graphics and Applications*, pp. 34–42, Sep. 1993.

[105] G. Strang, "The discrete cosine transform," *Society for Industrial and Applied Mathematics Review*, vol. 41, no. 1, pp. 135–147, 1999.

[106] H. Sun, W. Kwok, and J. Zdepski, "Architectures for MPEG compressed bitstream scaling," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 2, pp. 191–199, Apr. 1996.

[107] H. Sun, X. Chen, and T. Chiang, *Digital Video Transcoding for Transmission and Storage*. CRC Press, 2004.

[108] Y.-P. Tan, Y. Liang, and H. Sun, "On the methods and performances of rational downsizing video transcoding," *Signal Processing: Image Communication*, vol. 19, pp. 47–65, 2004.

[109] A. Vetro, P. Yin, B. Liu, and H. Sun, "Reduced spatio-temporal transcoding using an INTRA refreshing technique," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, 2002, pp. IV723–IV726.

[110] Z. Wang, "Fast algorithms for the discrete W transform and for the discrete Fourier transform," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 4, pp. 803–816, Aug. 1984.

[111] S. Wee and B. Vasudev, "Splicing MPEG video streams in the compressed domain," in *Proc. IEEE Workshop on Multimedia Signal Processing*, Princeton, Jun. 1997.

[112] J. W. C. Wong and O. C. Au, "Modified predictive motion estimation for reduced-resolution video from high-resolution compressed video," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 4, 1999, pp. 524–527.

[113] J. Xin, C.-W. Lin, and M.-T. Sun, "Digital video transcoding," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 84–97, Jan. 2005.

[114] P. Yin, A. Vetro, B. Liu, and H. Sun, "Drift compensation for reduced spatial resolution transcoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 11, pp. 1009–1020, Nov. 2002.

[115] J. Youn, M.-T. Sun, and C.-W. Lin, "Motion vector refinement for high performance transcoding," *IEEE Transactions on Multimedia*, vol. 1, no. 1, pp. 30–40, Mar. 1999.

[116] W. Zhu, K. H. Yang, and M. J. Beacken, "CIF-to-QCIF video bitstream down-conversion in the DCT domain," *Bell Labs Technical Journal*, vol. 3, no. 3, pp. 21–29, Jul. 1998.