Departamento de Engenharia Electrotécnica e de Computadores

Faculdade de Ciências e Tecnologia

Universidade de Coimbra

Multiscale Recurrent Pattern Matching Algorithms for Image and Video Coding

Nuno Miguel Morais Rodrigues

Coimbra

October 2008

Multiscale Recurrent Pattern Matching Algorithms for Image and Video Coding

A thesis submited for the degree of

Doctor of Philosophy

at the

Departamento de Engenharia Electrotécnica e de Computadores

Faculdade de Ciências e Tecnologia

Universidade de Coimbra

by

Nuno Miguel Morais Rodrigues

October 2008

...a coisa mais divina

Que há no mundo

É viver cada segundo

Como nunca mais...

From *Tomara*, by Vinicius de Moraes

The greatest thing you'll ever learn;

Is just to love and be loved in return.

From *Nature Boy*, by Eden Ahbez

To my wife, Catarina
and to Miguel, my nature boy.

Abstract

In this thesis we investigate new and efficient algorithms for image and video compression, based on a recently proposed generic lossy data compression method that uses the approximate pattern matching paradigm, called multidimensional multidimensional parser (MMP). MMP presents an excellent performance level when compared with traditional pattern matching algorithms for lossy image compression applications. When compared with the state-of-the-art transform-based methods, MMP is able to achieve significant gains for the compression of non-smooth images, like text and compound images, but has a compromising performance deficit for the compression of smooth, natural images.

The techniques developed in this thesis overcome this limitation, improving the performance of MMP-based smooth image coding up to a state-of-the-art level, while maintaining its relevant performance gains for non-smooth image compression. The performance gains resulted from the use of adaptive predictive schemes, together with new dictionary design strategies. Experimental results show consistent PSNR performance gains over the original MMP algorithm, for all image types and compression ratios. When compared with the state-of-the-art, transform-based encoders, the proposed methods achieve relevant gains (up to 6 dB) for non-smooth images and a comparable performance for smooth images.

A new MMP-based video compression algorithms was also investigated. The proposed method, referred to as MMP-Video, combines the hybrid video coding model with optimised multiscale adaptive pattern matching algorithms, in order to exploit the particular features of video signals. Experimental results show a general performance above the levels achieved by the current state-of-the-art H.264/AVC standard.

These results show that, in spite of its higher computational complexity, the MMP paradigm can be regarded as a viable alternative to the traditional transform-quantisation-entropy coding-based methods.

Resumo

O tema principal desta tese é o estudo de algoritmos eficientes de compressão de imagens e vídeo digitais baseados no paradigma da correspondência aproximada de padrões. Os métodos desenvolvidos são baseados num esquema de compressão recentemente proposto, denominado MMP (do original *multidimensional multidimensional parser*). Quando aplicado à compressão com perdas de imagens, o algoritmo MMP tem um desempenho acima do dos melhores codificadores baseados em aproximação de padrões. Quando comparado com os esquemas baseados em transformadas, o MMP consegue ganhos consideráveis para a compressão de imagens não suaves, mas apresenta perdas relevantes na compressão de imagens naturais.

As técnicas propostas eliminam estas perdas, elevando o desempenho do algoritmo MMP para a compressão de imagens suaves para um nível semelhante ao dos melhores codificadores baseados em transformadas, ao mesmo tempo que mantêm, ou melhoram, os ganhos observados para as imagens não suaves. Este ganhos resultam da utilização de métodos de codificação preditiva, utilizados em conjunção com novos esquemas associados ao processo de actualização do dicionário. Testes experimentais demonstram ganhos consideráveis na qualidade objectiva das imagens, quando comparamos as técnicas propostas com o algoritmo MMP original.

Esquemas eficientes de compressão de sinais vídeo baseados no algoritmo MMP foram também investigados. O método proposto, denominado MMP-Video, combina a arquitectura híbrida de codificação de vídeo com um codificador baseado em MMP, optimizado de modo a explorar de uma forma eficiente as características dos sinais de vídeo digital. Testes experimentais revelam valores de desempenho, em termos de medidas de qualidade *vs.* taxa de compressão, acima dos atingidos pela mais eficiente norma de compressão de vídeo, a norma H.264/AVC.

Os resultados apresentados demonstram a viabilidade do paradigma proposto, como alternativa aos esquemas tradicionais de compressão de imagens e vídeo, apesar do aumento da complexidade computacional verificado.

Keywords:

pattern matching, image coding, video coding, recurrent patterns, dictionary based methods.

Palavras chave:

correspondência de padrões, compressão de imagens, compressão de sinais de vídeo, padrões recorrentes, métodos de compressão baseados em dicionários.

Acknowledgements

I would like to express my gratitude to Dr. Vitor Silva, Dr. Sérgio de Faria and Dr. Eduardo da Silva, whose expertise, support and enthusiasm added considerably to my experience as a PhD student. For their wise guidance and fruitful discussions, that lightened the tunnel throughout this journey; for their encouragement and trust, that never failed me when I most needed them; for their friendship, that I will always treasure, my most sincere thanks. I would also like to acknowledge Dr. Murilo de Carvalho, for his comments and fruitful discussions, that contributed for this work.

This work would not have been possible without the support of Escola Superior de Tecnologia e Gestão of Instituto Politécnico de Leiria, Portugal, and the effort of all my colleagues, that enabled me to focus on this task. I also wish to thank Instituto de Telecomunicações, for the means it has provided me during my research activities, and Laboratório de Processamento de Sinais of Universidade Federal do Rio de Janeiro, for the excellent conditions provided during my work in Brazil. Finally, I also express my gratitude to Fundação para a Ciência e Tecnologia, for the financial support during this period.

I thank my colleagues in IT and ESTG, in Leiria, for the friendship and the wonderful working environment. To my colleagues and dear friends in LPS, for receiving me as one of their own and providing me with a home away from home, my eternal gratitude.

To Eddie, José Fernando, Nelson and Danillo, for sharing the enthusiasm and for the privilege of our joint work.

To my family, to whom I owe all that I have become; to my dear friends and their wonderful children, that help to fill my life with joy; to my wife Catarina, for amazing me every day with her love and support and to Miguel, that touches my life in the most wonderful way... all my devotion.

Contents

List of Figures

List of Tables

List of Abbreviations

ABS	Adaptive block size
B	(slice/macroblock) A slice that uses bi-directional motion compensation
BAC	Binary arithmetic coding
bpp	bits-per-pixel
CABAC	Context-adaptive binary arithmetic coding
CAVLC	Context-adaptive variable length coding
CBP	The coded block pattern parameter of H.264/AVC standard
CIF	Common Intermediate Format - 352×288 pixels image or video frame size
dB	Decibel
DC	Direct Current (refers to the constant component of a waveform)
DCT	Discrete cosine transform
DVB	Digital Video Broadcasting
DWT	Discrete wavelet transform
FBS	Fixed block size
FIR	Finite impulse response
FRExt	Fidelity range extensions
GG	Generalised Gaussian

H.264/AVC	Video compression standard, also known as H.264, MPEG-4 Part 10 or MPEG-4 AVC
H.264/AVC NO MPE	An algorithm based on the H.264/AVC encoder, that dues not use the compression of the motion-compensated prediction error
I	(slice/macroblock) A slice that uses only macroblocks coded with intra prediction
IEC	International Electrotechnical Commission
IR	Impulse response
ISO	International Organisation for Standardisation
IT	Integer transform
ITU	International Telecommunication Union
ITU-T	ITU Telecommunication Standardisation Sector
JBIG	The Joint Bi-level Image Experts Group and the bi-level image compression standard, ITU-T Recommendation T.82
JBIG2	The successor of the JBIG compression standard, for bi-level images
MB	An image or video frame macroblock
MC	Motion compensation
ME	Motion estimation
MFV	Most frequent value
MMP	Multidimensional multiscale parser
MMP-I	An algorithm that combines MMP with Intra predictive coding
MMP-II	An algorithm that combines MMP-I with Improved dictionary adaptation techniques
MMP-Video	An algorithm that uses the MMP paradigm for video compression

MMR	The Modified Modified READ (MMR) bi-level image coding technique, ITU-T Recommendation T.6
MP	Matching pursuit
MPEG	Moving Picture Experts Group
MV	Motion vector
P	(slice/macroblock) A slice that uses simple motion compensation
PSNR	Peak signal-to-noise ratio
QCIF	Quarter CIF - 176 $\times$ 144 pixels image or video frame size
QP	Quantisation parameter
RD	Rate-distortion
SAD	Sum of absolute differences
SATD	Sum of absolute transformed differences
SIF	Source Input Format - 352 $\times$ 240 pixels image or video frame size
SSD	Sum of squared differences
VBR	Variable bit-rate
VCEG	Video Coding Experts Group of ITU-T
VLC	Variable length code

Chapter 1

Introduction

In this chapter we present an introduction to this thesis and the associated research work. After a brief overview of the open research topics that served as motivation for this work, the main objectives of this thesis are presented. The major contributions of this research are then listed and an outline of this document is provided.

1.1 Motivation

In spite of the ever growing available bandwidth and network speeds, the need for efficient image and video compression algorithms is as important now as it has ever been. Recent advances in consumer electronics, associated with digital cameras as well as scanning and display devices, have caused a rapid proliferation in the use of digital image and video signals. Obvious examples can be found in the generalised replacement of the traditional film and analog video cameras by digital devices, as well as on the current migration to digital video broadcasting (DVB), that is expected to replace the existing analog services within a few years time. The availability of these systems to the average consumer boomed the amount of digital visual information that needs to be handled. The massive storage requirements, together with the recent focus on transmission, for instant sharing of visual contents, justifies the constant pursuit of ever more efficient compression schemes.

In spite of the well known intrinsic limitations of the transform-quantisation-based encoding methods, this class of algorithms has been dominant for the last few decades both in image and video compression. The efficiency of these methods results from the energy compaction properties of the used transforms, either the traditional discrete cosine

transform (DCT) and discrete wavelet transform (DWT), or the new integer transform (IT) proposed by recent encoding standards for video compression. Prediction schemes may also be used, in order to reduce the spatial (and temporal) correlation of the input signal. The resulting transform coefficients are then compressed by using a quantisation process, followed by some type of lossless entropy coding. The use of convenient quantisation procedures minimises the perceptual impact on the compressed signal, while entropy coding reduces the statistical correlation and improves the overall compression efficiency.

The transform-quantisation-entropy encoding paradigm is particularly efficient for smooth, low-pass images, where most of the transform coefficients representing the highest frequencies tend to be negligible or of little importance. This allows the use of a coarse quantisation, achieving a high compression of these components, without compromising the quality of the reconstructed images. Nevertheless, this suitability for low-pass images cannot be generalised to other image types. In fact, for images of a different nature, like text, computer generated, compound (text and graphics) and texture, among others, the use of transform-based compression schemes often results in a poor rate-distortion (RD) performance and highly disturbing visual artifacts, like ringing and blocking. Unfortunately, encoding methods that are adequate to handle these types of images tend to have serious limitations when used to encode smooth images.

This fact justifies the on-going research on alternative compression paradigms for image and video signals. The investigation work described in this thesis explores such a paradigm. It is built upon a generic lossy data compression algorithm, called multidimensional multiscale parser (MMP) [1, 2]. Unlike the major image and video compression standards, MMP in based on the approximate pattern matching coding paradigm, but includes several important new features. MMP uses an adaptive dictionary of vectors, formed by recursively parsing an original block of data, in order to recurrently approximate variable-length blocks of the input image. Additionally, geometric transformations are used to scale each element of the dictionary into the dimensions of the block segment that is being considered.

The adaptive dictionary update procedure uses blocks obtained by concatenations and scale transformations (expansions and contractions) of previously occurred patterns in the original image, which are utilised to approximate future image blocks. Although MMP incorporates encoding strategies from the traditional pattern matching strategies, namely from the work of Lempel and Ziv [3, 4] and from standard vector quantisation (VQ) [5],

it introduces a new and important principle: approximate pattern matching with scales. MMP thus uses a multiscale recurrent pattern encoding paradigm, that, unlike the case of traditional transform-based schemes, makes no assumption about the nature of the input signal.

MMP was initially proposed as a generic data compression algorithm, but was immediately used in image coding applications [1, 2]. Experimental tests showed that the performance of the original MMP algorithm tops the one of other approximate pattern matching-based image compression schemes, such as lossy extensions of the Lempel-Ziv algorithm [6, 7] and adaptive VQ [8, 9]. When compared with the state-of-the-art image compression methods, the relative performance of MMP depends on the input image type. For non-smooth images, like text, compound (text and graphics) and texture images, MMP performs well, outperforming the transform-quantisation-based image coders like SPIHT [10], JPEG [11] and JPEG2000 [12]. Nevertheless, for smooth grayscale images, although performing better than JPEG, MMP presents a performance deficit, when compared to the DWT-based methods like SPIHT and JPEG2000 and the H.264/AVC intra frame encoding algorithm [13, 14]. This disadvantage compromises the applicability of MMP image compression.

Besides the tests on image compression, MMP has demonstrated its suitability for coding a large set of very different data sources. In fact, MMP has been successfully applied to the compression of various signal types, like electrocardiograms [15, 16, 17] and stereoscopic images [18]. These observations encouraged the investigation of MMP for the compression of other signals, namely digital video.

1.2 Main objectives

The discussion presented in the previous section identified some open research issues, regarding the problems of image and video compression using pattern matching-based compression schemes in general, and the MMP algorithm in particular. Among the pattern matching-based algorithms used for image compression, MMP is able to achieve an excellent performance level, due to its particular features. Nevertheless, in spite of its excellent results for some image classes, the MMP performance issues for smooth image coding still compromise its applicability as a general image compression scheme, especially when one

compares it with the current state-of-the-art, transform-based image encoders.

Additionally, the investigation of a related problem, namely the MMP efficiency for the compression of video sequences, has not been addressed in previous works.

The main research topics of this thesis were thus focused on two major goals:

- **Improving the efficiency of MMP for image coding.** The focus will be on designing new and efficient algorithms that improve the MMP efficiency for image compression. A special attention will be given to methods that improve the performance of MMP specifically for the compression of smooth, natural like images. However, the proposed techniques should maintain, or even improve, the good results of the original algorithm for non-smooth images, like text and compound images.

  In order to access their efficiency, the results of the proposed schemes should be compared with the compression performance of the original MMP algorithm, and also with the top state-of-the-art encoders. The objective is to develop the performance of MMP-based image encoding up to a point were it becomes a viable alternative to the transform-based image compression schemes.

- **Investigating the efficiency of the MMP paradigm for video coding applications.** Based on the good results achieved by MMP for other signal types, and on the knowledge gathered from all the tests in MMP image coding, the use of the MMP paradigm for video compression will be investigated. Adequate encoding architectures should be studied, as well as some specific optimisations of the MMP encoder, that allow the exploitation of the particular features of video signals. The experimental results will be assessed by comparison with the current state-of-the-art video compression algorithm: the H.264/AVC high profile video encoder.

1.3 Summary of original contributions

In this section we present a summary of the most relevant original contributions that resulted from the research work developed during this thesis. Most of these contributions are related specifically with the MMP algorithm, both for image and video compression, since these are the main topics of this thesis. Nevertheless, many of the proposed methods are associated with some general aspects of pattern matching-based compression, and may be used by other algorithms.

As the investigation progressed, spacial care was given to the validation of the work within the scientific community. As a result, most of the contributions have either been published in international journals or in the proceedings of international conferences.

The most relevant contributions of this thesis are related with the objectives described in the previous sections, namely:

- **The MMP-I algorithm: improving MMP performance with predictive techniques**

  An investigation on the performance of MMP for the compression of signals with different probability distributions showed that the efficiency of the dictionary adaptation increases for sources with narrower probability distributions. This observation motivated the development of a new compression scheme that combines the use of MMP with predictive coding techniques, referred to as MMP-I. The use of adaptive image prediction methods transforms the input images into predicted residue signals with highly concentrated probability distributions, that were modelled using generalised Gaussian functions. By using adaptive prediction techniques, MMP-I is able to generate a prediction residue signal with a probability distribution that is exploited by the dictionary adaptation process in a much more efficient way, when compared with the original image's patterns. Experimental results show a significant improvement on the compression performance of MMP-I, when compared with MMP, especially for smooth images. For non-smooth images, MMP-I is able to maintain the same levels of performance of MMP. Papers J1, C9 and C11 (see Appendix D) resulted from this investigation.

- **The MMP-II algorithm: efficient dictionary adaptation for multiscale recurrent pattern algorithms**

  Several new dictionary adaptation techniques for multiscale recurrent pattern matching image coding were proposed. The use of these methods in MMP-I resulted in an enhanced coding scheme that we refer to as MMP-II. The developed methods explore several features of MMP-based encoding, in order to improve its efficiency. The proposed methods were related with: redundancy reduction techniques, to avoid the insertion of disadvantageous code-vectors; context conditioning methods, that increase the performance of the adaptive arithmetic encoder for MMP symbols; an

L^1-norm equalisation procedure, that adapts the new dictionary patterns according to a probability distribution criterion; enhanced dictionary updating strategies, that create additional code-vectors in order to increase the dictionary's approximation power; scale-adaptive methods, to eliminate unnecessary dictionary vectors, allowing a significant reduction in processing time with negligible performance losses. In spite of their MMP-related genesis, most of the proposed dictionary adaptation schemes are generic enough to be easily used with other adaptive pattern matching algorithms.

The proposed schemes allow MMP-II to achieve quality gains over both MMP and MMP-I for every tested image and all compression ratios. For non-smooth images, these gains further increase the advantage of MMP-based algorithms over the state-of-the-art, transform-based encoders. For smooth images, MMP-II reaches a performance level that is similar to that of the JPEG2000 algorithm and very close to that of H.264/AVC high profile intra frame encoder. Papers J1, C4 and C8 (see Appendix D) resulted from this investigation.

- **Study of multiscale recurrent pattern using a signal decomposition analogy**

  An analysis of the MMP coding process revealed some interesting analogies with a decomposition of the original signal into a set of orthogonal, non overlapping, bases functions. In the original method, the initial dictionary uses a set of pulses with different amplitudes and scales, that can be related with the scaling function of the Haar's wavelet. The approximation of the input signal can be regarded has a combination of these pulses, using appropriate amplitudes, shifts and scales, that are determined by the MMP's rate-distortion controlled coding (or analysis) algorithm. The decoding (or synthesis) process uses the same functions to reconstruct the approximated signal. Paper C5 (see Appendix D) resulted from this investigation.

- **Study of new prediction prediction methods for multiscale recurrent pattern encoders**

  The original MMP-I prediction process uses eight directional prediction modes plus a new mode, based on the most frequent value of the prediction pixels. An investigation was conducted in order to improve the performance of the original prediction process. This study revealed that the use of intra-frame prediction for blocks down to 2×2 pixels allows a better performance than the use of the original 4×4 block size limit,

inherited from the H.264/AVC encoder. Also, the use of a new prediction mode, that is not based on directional prediction but on template matching, proved to have favourable results, while the elimination of some of the original prediction modes has no impact on the final encoding performance. Paper C2 (see Appendix D) resulted from this investigation.

- **Improving multiscale recurrent pattern image coding with post-processing deblocking filtering**

  The investigation of a deblocking method for MMP-II was motivated by the observation of some blocking artifacts, especially for high compression ratios. The use of two deblocking methods originally proposed for MMP was studied for the MMP-II case, revealing several compromising inefficiencies. A new deblocking method was then investigated. The MMP-II deblocking algorithm uses an adaptive FIR filter to process each image block. The filter's shape is adapted according to the image region that is being processed. Two filtering parameters are determined by the encoder, in order to maximise the deblocking performance for smooth images and eliminate the severe PSNR losses for non-smooth ones. The parameters are transmitted to the decoder using a negligible overhead. Experimental results show that, for smooth images, the new deblocking method improves both objective and subjective quality levels. Furthermore, the proposed scheme does not affect the quality of non-smooth images, making it suitable for any image type. Papers B1 and C7 (see Appendix D) resulted from this investigation.

- **Developing efficient recurrent pattern matching-based video encoding schemes**

  One of the main objectives of this thesis is the investigation of MMP-based video compression algorithms. As a result of this research work, a new method, referred to as MMP-Video, was proposed. This algorithm uses MMP in a hybrid video coder framework, as the compression scheme for the motion-compensated prediction residue signal. The use of the multiscale recurrent pattern paradigm for video compression was optimised, by taking into consideration some of the techniques that resulted from the investigation on MMP as a digital image encoder, but also by developing new methods, especially designed to exploit the particular features of digital video

encoding.

As part of the study on the current video compression algorithms, an H.264/AVC-based video compression scheme that does not use the transmission of the motion-compensated residue, was also investigated. An analysis of this method reveals similarities with pattern matching-based video compression, namely using Lempel-Ziv and vector-quantisation algorithms. Experimental tests show that the proposed encoder achieves a performance advantage over H.264/AVC, when the motion-compensated error is not encoded for B slices, for low to medium rates.

Experimental results with the MMP-Video encoder demonstrate advantageous results over H.264/AVC high profile, especially for medium to high bit-rates and for B-slice data, where the coding gains range up to 2dB. These results validate the use of the multiscale recurrent pattern matching paradigm also for video compression.

Papers B2, C1, C3, C6 and C10 (see Appendix D) resulted from this investigation.

At this stage of our research work, a secondary role as been given to the computational complexity issues associated with the MMP-based algorithms. This thesis will focus mainly on developing the most efficient methods in terms of encoding performance, in order to prove that the MMP paradigm may be regarded as a viable alternative to the current algorithms. Nevertheless, some computationally efficient implementation techniques were also investigated, that achieve a better compromise between coding efficiency and computational complexity.

1.4 Outline of the thesis

This thesis is organised into nine chapters and four appendices. The current chapter introduces the research work described in this document.

Chapter 2 presents a brief description of the most important pattern matching compression algorithms, as well as a state-of-the-art of pattern matching-based methods for image and video compression.

Chapter 3 is entirely dedicated to the multidimensional multiscale parser algorithm. A detailed description of MMP for image coding is given, together with an evaluation of the corresponding experimental results. These results are compared with the ones of the state-of-the-art transform-based image encoders. An experimental evaluation of some functional

aspects of MMP is also presented: the MMP bit stream as well as the MMP's dictionary updating process are investigated. A discussion on these topics reveals some interesting research lines that will be further exploited in other sections of this work.

Chapter 4 analyses the MMP paradigm, providing new insights into the studied algorithm. A comparison of MMP with the traditional pattern matching coding methods, like vector quantisation and Lempel-Ziv, is presented. Furthermore, other less likely analogies are described, namely by relating MMP with a signal decomposition algorithm and with a non-uniform sampling scheme. A discussion on possible ways to exploit these relations is also presented in this chapter.

Chapter 5 investigates the joint use of MMP with predictive schemes, and describes a new coding algorithm, referred to as MMP-I. The proposal of this method was motivated by a study on the efficiency of MMP on sources with different probability distributions, described by generalised Gaussian functions. Experimental results demonstrate that MMP-I allows for a consistent improvement on the performance of MMP for smooth images, while maintaining its excellent performance for other image types. An analysis of the probability distribution of the MMP-I prediction residue signal demonstrates the accuracy of the considered model.

Chapter 6 describes several dictionary adaptation techniques, that increase the efficiency of the MMP-I encoder by improving the code-vectors' use and dictionary updating. The combination of MMP-I with the new techniques defines a new encoding method, referred to as MMP-II (MMP-I with Improved dictionary adaptation). The results for MMP-II are presented and compared with those of the benchmark methods.

Chapter 7 presents a study on other aspects of MMP-II, that are not directly related with dictionary adaptation. The prediction process originally used by MMP-I and MMP-II is revisited: an evaluation of the performance and usefulness of the original prediction modes is presented and new prediction modes, based on different paradigms, are tested. A post-processing deblocking method for MMP-II, that increases both the perceptual and the objective quality for smooth images, is also presented in this chapter.

Chapter 8 investigates the use of the pattern matching and MMP paradigms for video compression. Following the good performance of MMP for image coding, namely in the compression of the intra-predicted error signal, observed in the previous chapters of this thesis, two pattern matching-based approaches to video compression are proposed in this

chapter. First, we investigate a version of the H.264/AVC video compression standard for which the prediction residue error compression stage is disabled. A formal analysis of this process relates it pattern matching paradigms, not usually associated with efficient video compression schemes. Nevertheless, experimental results demonstrate performance improvements over the original H.264/AVC, for Bi-predictive (B) slices, especially at low to medium bit rates. An MMP-based video encoding algorithm, named MMP-Video, is also presented. It uses MMP to encode the motion compensated residue, in a hybrid video coding scheme. The adaptation of the MMP-based algorithms for video compression is described, together with an investigation on new procedures developed specifically for MMP-video. Experimental results show that the proposed methods are often advantageous over the H.264/AVC standard, validating the proposed paradigm.

Appendix A presents an overview of the test images used throughout this work. Several test images of different types were considered. A summary of the video sequences used in the experimental tests is also presented. As for the images, several sequences with particular features were used, varying from slow motion, head-and-shoulder sequences to video signals with high, complex motion.

Appendix B presents an algorithmic description of some of the most relevant methods proposed in the previous chapters of this document. In order to provide a clearer presentation of the proposed techniques, some specific details, that are important mainly from an implementation point of view, are described only in this appendix. The main text includes references to the location of the relevant sections of this appendix.

Appendix C has a set of additional experimental results, that complement the ones presented in the previous chapters of the thesis. As for the previous appendix, this was done in order to improve the clarity of the document. The data presented in this appendix is meant as a complement of the results presented in the main text.

Appendix D has a summary of the published papers, describing contributions of the research work of this thesis.

Chapter 2

Pattern matching image coding

The methods investigated in this thesis belong to a general family of algorithms that are usually referred to as pattern matching compression schemes, but are also know as dictionary-based or string matching algorithms. This chapter presents a discussion on pattern matching coding. Section 2.1 has a general description of some of the features of these methods. Section 2.2 briefly describes some of the most popular and well known pattern matching algorithms, namely the Lempel-Ziv (LZ) based encoders (also called string matching algorithms) and vector quantisation (VQ) methods. Besides these well known algorithms, other methods have been proposed, that also use this coding paradigm. Some of these algorithms are discussed in Section 2.3.

The pattern matching image coding algorithm that we investigate in this thesis, named Multiscale Recurrent Pattern (MMP) [1, 2], is an extension of traditional approximate pattern matching schemes, that uses an adaptive dictionary with variable block size vectors to approximate the original image. A detailed presentation of MMP will be done in the following chapter.

2.1 The pattern matching image coding paradigm

The general class of pattern matching algorithms can be regarded as a unifying umbrella, under which a large set of image coding techniques may be classified. In this section we present a brief discussion on the pattern matching coding paradigm and describe some common features of these methods.

Pattern matching methods usually partition the input message into segments or blocks

of samples. Each block is approximated by one pattern, that is chosen among a set of available vectors. Some pattern matching schemes are also called *dictionary-based* algorithms, because they store the available approximation patterns (code-vectors) in an indexed list, called a *dictionary* or *codebook*, that is kept by both the encoder and decoder applications. For these algorithms, the vector that is chosen to encode each of the input signal's segments is identified by its index. The indexes values are then sequentially encoded and transmitted, thus allowing for the reconstruction of the signal by the decoder. This is the coding paradigm used by the LZ78-based (see Section 2.2.1) and vector quantisation (see Section 2.2.2) algorithms. One other approach uses previously encoded patterns of the original signal to approximate the message elements that have not yet been encoded. In these cases the use of an explicit dictionary is avoided, since the relevant patterns are identified by their length and relative position to the message segment that is being encoded. This is the case of LZ77-based algorithms, also described in Section 2.2.1.

Pattern matching algorithms, as other image coding methods, may be classified as lossy or lossless. Lossless pattern matching algorithms avoid a mismatch that would introduce an error in the encoded signal by using code-vectors that are exact matches of the original message blocks. For dictionary-based methods, this is possible only if the dictionary contains a set of patterns that allows the reconstruction of all possible input signal segments. If this is not the case, a higher rate encoding mechanism is used to compress the signal patterns that do not exactly correspond to an existing code-vector. This is the case of the original Lempel-Ziv family of string matching methods (see Section 2.2.1). In these methods the new patterns that are transmitted are also inserted in the dictionary. The methods that have the ability to update the pattern database during the encoding process as also known as *adaptive* pattern matching schemes. The methods that do not possess this ability are usually called *non-adaptive* or *fixed codebook* methods.

In lossy encoding schemes one accepts some degree of distortion in the reconstructed signal, as a trade-off for a more efficient compression. Such methods are commonly used in image and audio compression systems, because it is possible to greatly increase the compression ratio of the encoded signal by introducing a distortion level that is negligible for the human perceptual system. Lossy pattern matching methods chose a codeword that provides an accurate (but not necessarily exact) representation of the original message. This is generally the case of vector quantisation based methods (see Section 2.2.2). As will

be discussed later, the use of LZ-based lossy encoding algorithms has not yet been able to achieve a good efficiency.

2.2 Recurrent pattern matching algorithms

Compression methods that are based on the Lempel-Ziv string-matching encoders have become ubiquitous in computational applications for lossless compression of generic data files. Examples of compression tools based on these schemes, that are familiar to all personal computer (PC) users, are the Zip, Unix Compress, Gzip and Arj applications. Among the LZ-based image compression schemes which have achieved a great popularity are the GIF and PNG image formats. LZ methods are usually divided into two types: LZ77 and LZ78, that are described in Section 2.2.1. Vector quantisation compression schemes have also become popular among pattern matching methods for signal compression. The most relevant aspects of VQ methods are briefly described in Section 2.2.2.

2.2.1 The Lempel-Ziv string matching methods

One important family of recurrent pattern matching algorithms emerged from the work of Abraham Lempel and Jacob Ziv [3, 4]. Lempel-Ziv encoders parse the input sequence into non-overlapping blocks of different lengths, that are encoded by using previously transmitted segments of the message. LZ methods use two main paradigms for data compression:

- In a paper from 1977 [3], Lempel and Ziv describe a method that uses pointers to identify the longest match for the current block within a search buffer, composed by the most recently encoded data elements. This algorithm is the reference for a family of encoding methods, usually referred to as LZ77.

- Another paradigm developed by Lempel and Ziv was presented in a paper from 1978 and is thus referred to as LZ78. Unlike LZ77, the LZ78 methods use an explicit dictionary, composed by an indexed list of previously used portions of the message.

Both LZ77 and LZ78 families of methods became very successful in several data compression applications. A brief description of these algorithms and of some of their most important variations is presented in the following sections.

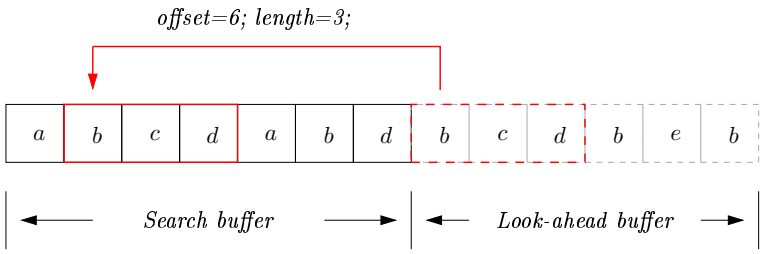


Figure 2.1: An example of block matching for LZ77 algorithms.

The LZ77 algorithms

LZ77 algorithms may be regarded as dictionary-based methods, where the dictionary corresponds to the most recently encoded portion of the message, called *search buffer*. A second buffer, called *look-ahead buffer*, corresponds to the message segment that is going to be encoded next. Figure 2.1 represents these buffers for the compression of a string of characters. The sequence in the look-ahead buffer is encoded by searching the search-buffer until a match is found for the first symbol to be transmitted. When this happens, a *length* and an *offset* parameters are determined. These values signal that the next *length* symbols of the message should be copied from the search-buffer, starting at position *offset*. The matching stage always returns the longest match among the data that is present in the sliding window corresponding to the search-buffer. For each match, a set of three symbols are encoded using a fixed-length code: the offset, the length and the first symbol that occurs in the look-ahead buffer, that was not represented by the match. This allows the representation of a symbol of the look-ahead buffer that does not exist in the search buffer. In this case, both the offset and the length values are set to zero and the new symbol value is transmitted.

One way to improve the performance of LZ77-based algorithms is to avoid the transmission of the first character that does not belong to the match. In order to do this, the LZSS [19] algorithm uses a one bit flag to signal whether a *(offset, length)* pair is being transmitted or the code for a new character is being inserted in the bit-stream. Other proposals describe the use of more efficient strategies for LZ data compression. One of these schemes, that is used in the PNG digital image file format, is called the Deflate algorithm [20]. It allies LZ77 with Huffman coding [21]. Other methods that combine LZ77 with variable-length encoders are the well known file compression schemes Zip, Gzip and Arj. The combination of LZSS with variable-length encoders was also proposed: LZB [22] is a

combination of LZSS with variable sized pointers, while LZH [23] combines LZSS with a Huffman encoder. Some methods that use variable-sized look-ahead and search windows, instead of the fixed buffers defined by the original method, were also proposed. One example is the LZR [24] algorithm, that uses LZ77 with variable-size pointers that can index a sub-string anywhere in the previously encoded data.

The LZ78 algorithms

All LZ77 algorithms assume a locality property for the matching patterns, *i.e.* the repeating segments of the message are located within the length of the search window. LZ78-based algorithms do not make this assumption. LZ78 uses an adaptive dictionary to store the patterns for the match, instead of using the elements of a sliding window of the compressed data. Moreover, the same block substitution paradigm is used, but the match is made using one element of the adaptive dictionary. The adaptive dictionary is updated by the LZ78 encoder and decoder using the same procedure, meaning that the same code-vectors are available at each moment. Similarly to LZ77, the algorithm searches for the longest match between the input string and any dictionary vector. If a match exists, the encoder outputs the index of the dictionary element that corresponds to the longest match, followed by the first symbol of the input block that was not encoded in this step. The string composed by the concatenation of the dictionary word and the first unmatched character is then inserted in the adaptive dictionary and the search procedure is repeated. If the search for a match fails, index 0 is transmitted, followed by the symbol that was not found in the dictionary.

The LZW algorithm [25] is an improved version of LZ78, that eliminates the transmission of the first unmatched symbol in the look-ahead buffer. In order to do this, LZW initialises the LZ78 dictionary with all possible source blocks of length one. LZW is used in Unix Compress and in the GIF digital image file format. Both these methods use a variable dictionary size: the initial codebook uses 2^n words (n bit indexes); when the full capacity is reached, the dictionary size is doubled and $n+1$ bit indexes are used; whenever the number of elements reaches the maximum size, the dictionary length is updated, as well as the number of bits used by the indexes. Other LZ78-based methods introduce some adaptations to the original method: LZJ [26] is similar to LZW, but restricts the dictionary to the N least recently used codewords. LZT [27] is similar to LZJ, but rearranges the dictionary entries according to their recent use. Other methods combine the ideas of both

LZ77 and LZ78, like LZFG [28], which uses a dictionary to store particular codewords (like LZ78) to complement the blocks encoded using a sliding window, like the one of LZ77.

String matching algorithms for image coding

In spite of the great success achieved by LZ-based encoders in the lossless compression of one-dimensional data, there have been rare proposals to adapt these methods to lossy two-dimensional data (images) and video coding. The lossless nature of the original LZ methods is often maintained in the LZ image compression algorithms, as most of the proposals refer to lossless coding methods. One example is proposed in [29], where a lossless two-dimensional pattern-matching image compression scheme is presented. Each image block is encoded using a previously transmitted pattern, searched in a window of previously encoded pixels. The dimensions of the blocks used in the matching stage are adaptive, as in the original LZ methods. Whenever the pattern matching fails, the block is encoded using a prediction scheme plus a lossless variable-length coding of the residue. The authors of this scheme claim a compression efficiency superior to all other dictionary-based methods and comparable to that of traditional state-of-the-art lossless methods, like JPEG2000 and JPEG-LS.

Despite the good compression performance described for some lossless algorithms, lossy LZ compression methods have generally failed to achieve competitive results with transform-based image encoders. In [6, 7, 30, 31] several methods that use the LZ paradigm in lossy compression algorithms, with applications in image coding are described. In [31], a simple LZ77 compression scheme that allows for approximate pattern matching is used, resulting in a lossy image encoder. An error threshold is used to control the rate-distortion point for the encoded image. In [6] the image pixels are approximated by vectors, meaning that this scheme can be considered an extension of the 1D LZ encoder to the compression of 2D image data. It uses the LZ77 paradigm, as it searches for the longest match of the uncompressed segment of the image, that *approximately* occurs in a previously encoded part of the image. The method also introduces some extensions to the basic LZ approach, namely related with the adaptation of the patterns of the search window prior to the matching stage. In [30] a two-dimensional extension of the previous method is presented. The authors apply this algorithm to both image and video coding. An LZ78-like dictionary stores a set of patterns that may be used to approximate each image block. Alternatively,

the image patterns may be approximated by a planar surface, whose parameters are determined using a least-square optimisation. The coordinates of an anchor point of the encoded block, the block dimensions and the used approximation method are transmitted for all image blocks, using a variable-length encoder.

Other schemes propose the use of LZ-based encoders to compress the transform coefficients in transform-based image encoding. One example is [7], that applies LZ schemes in sub-band coding, namely by compressing the image components at the output of a quadrature mirror filter (QMF) [32].

In spite of the interesting approaches described in these papers, they report compression efficiencies for the described methods that are more or less equivalent to the one of the JPEG algorithm. This places lossy LZ image compression algorithms well below the current state-of-the-art transform-based encoders, like JPEG2000, in terms of coding performance.

2.2.2 Vector quantisation

Vector quantisation [5] is another well known paradigm for dictionary-based algorithms. In these algorithms, the input signal is partitioned into blocks or vectors. For a one-dimensional signal, blocks of N consecutive samples are used. For the case of image compression, the input image is divided into blocks of pixels (usually rectangular $M \times N$ blocks), where each pixel may be regarded as a component of an $(M.N)$-dimensional vector. Each block of the input message is approximated by one of the vectors of the *dictionary* or *codebook*, $\mathcal{D}$. The dictionary holds a set of code-vectors, $\mathbf{s}_i$, that are representative of the message patterns. Each dictionary vector is identified by a unique *index*. Synchronised copies of the dictionary are maintained by both the VQ encoder and decoder. For each input block, the VQ encoder searches the dictionary for the code-vector that minimises the distortion for a given input block. The index of the chosen block, i, is then stored or transmitted to the decoder. The decoder determines the index value and then fetches block i from its local copy of the dictionary. It then uses the code-vector $\mathbf{s}_i$ to approximate the original block, as can be seen in Figure 2.2.

For lossy methods, the set of blocks that compose the dictionary are a quantised representation of the input signal space. The use of VQ may be regarded as a joint quantisation of all samples of each input vector, instead of independently using a scalar quantiser on each of them. Because VQ jointly considers several input samples, it is able to better ex-

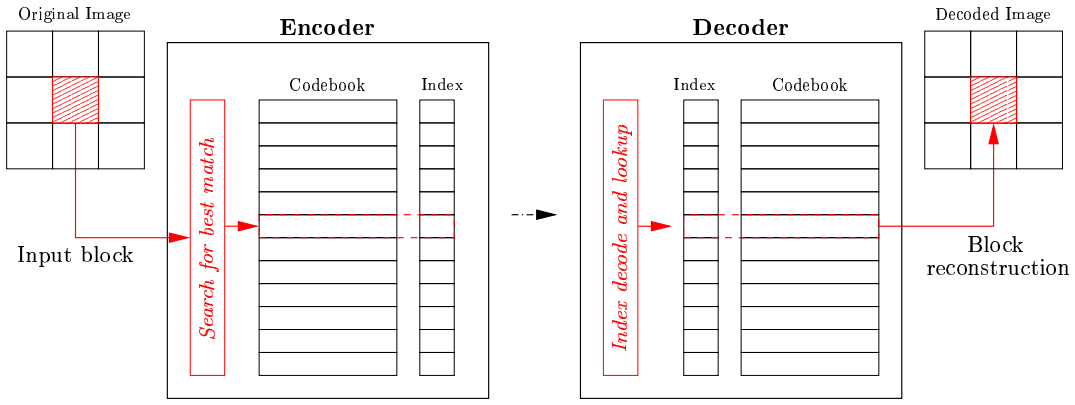


Figure 2.2: A vector quantisation compression architecture.

ploit any structure (*e.g.* correlation between the vector elements) that exists in the input signal. It is also possible to incorporate the knowledge about the input signal features into the VQ design, improving the efficiency of the coding process when compared with scalar quantisation. Nevertheless, this efficiency also depends on several other factors. One of them is the size of the codebook, *i.e.* the number of code-vectors: larger codebooks generally achieve a more accurate approximation of the input signal, but at the cost of a higher average bit count for each index representation. Another important factor is the dictionary design, *i.e.* the process of choosing the code-vectors. A commonly used method for VQ dictionary design is the LBG (Linde-Buzo-Gray) [33] algorithm, that iteratively refines a set of initial points until a minimum distortion is achieved for the vectors of a training set. The vectors chosen to initialise the algorithm and compose the training set are very important for the efficiency of LBG dictionary.

As for other dictionary-based methods, several strategies may be used regarding VQ codebook usage. An adaptive scheme may be used for the VQ dictionary, instead of the fixed dictionary model. As for the previously presented string-matching algorithms, this adaptation is usually performed when the VQ encoder is not able to find a satisfactory match for a given input block. In this case, a new pattern is transmitted to the decoder, by using an alternative (higher rate) encoding scheme. This pattern is then used to approximate the original block and is also inserted in the dictionary. If a fixed sized dictionary is used, the insertion of a new code-vector is done by replacing one of the existing elements. Several strategies may be used to determine the element that should be replaced. For both adaptive and non-adaptive VQ methods, the definition of the initial codebook may be im-

plicit or information about the used dictionary may be transmitted prior to the encoding process. If a fixed, predefined dictionary is used, then both the encoder and the decoder know all the code-vectors *a priori*. If this is not the case, namely for those methods that use explicit dictionary design for each input signal, then the initial dictionary patterns must be encoded and transmitted to the decoder. This causes an additional overhead, that may be favourably compensated by the added efficiency of the encoding process using the adaptive dictionary. For a more complete discussion on these and other aspects of VQ, that are out of the scope of this overview, refer to [5].

Several VQ methods have been proposed for image and video coding applications. In [34] a survey of some of these methods is presented. An adaptive VQ scheme that uses approximate string matching with a rate-distortion criterion, called RDLZ, is presented in [35]. An *a priori* constructed codebook is used (it may be known by the decoder or transmitted before the encoding). During the image coding process, indexes of the codebook are transmitted in the bit-stream for the cases where a successful match is achieved. When the best match is unfavourable, based on a RD evaluation, the encoder transmits a new pattern, that is used to update the dictionary. RDLZ performs comparably, sometimes favourably, to static codebook VQ trained on the same sources or images. In [36] an adaptive VQ method is proposed, that uses information about the previous use of code-vectors to adaptively maintain two codebooks. Results presented in the paper show an advantageous performance over several other VQ image coding methods. Several adaptive VQ schemes have also been proposed for video coding applications. Generally, these methods report a better performance than that of non-adaptive VQ [37] but worse than that of traditional transform-quantisation-based encoders [38].

2.3 Other pattern matching methods

In this section we briefly describe the JBIG and matching pursuits algorithms. In spite not being as popular as the LZ and VQ algorithms, these methods introduce interesting new approaches to the pattern matching coding paradigm.

2.3.1 JBIG image coding

JBIG2 [39] is the successor of the JBIG (ITU-T Recommendation T.82) compression standard [40], for the compression of bi-level images, like facsimile and digital documents. Both standards were developed by the Joint Bi-level Image Experts Group. Unlike JBIG, JBIG2 supports not only lossless and progressive lossy-to-lossless but also lossy compression. For lossless compression it is able to achieve gains of up to 4 times, when compared with the previous standards. The lossy mode enables higher compression ratios with a "perceptually lossless" quality, *i.e.* with almost unnoticeable image degradation [41]. In spite of their specific field of application, the JBIG family of algorithms has the merit of being one of the rare cases of pattern matching-based algorithms that effectively "made it" to the standards.

Both JBIG standards combine the use of two main techniques in the compression of binary images: a lossless arithmetic coding algorithm is used to transmit the image patterns, while a progressive transmission algorithm is used for the generation of lower resolution images. These lower resolution representations of the image may be sufficient for some applications. JBIG specifies an algorithm for resolution reduction, that considers a pixel neighbourhood in order to determine the value of the low resolution sample [40]. If the user requests a more detailed version of the image, the low resolution image may be used as reference for the high resolution representation.

JBIG uses a context adaptive variation of an arithmetic encoder, known as Q-coder [42]. In a typical text image there are regions where white pixels occur with a probability close to one, while in other image areas black pixels occur with a high probability. JBIG's context adaptive arithmetic coding explores this fact. It uses the previously encoded neighbouring pixels to determine the probability context for a given pixel of the image. Neighbourhood regions with 10 pixels are used. The pixels values are grouped into a binary word that serves as an index to the probability context.

In lower resolution representations of the images, the position of the neighbourhood template is chosen in order to maximise the performance of the encoder. This ability to move the reference neighbourhood is useful to capture particular image structures, that occur frequently in text and halftone images. For higher resolution representations, JBIG uses the pixels of the low resolution image as a part of the context that is used by the arithmetic encoder.

Like its predecessor, JBIG2 is optimised for bi-level image coding, namely text or

halftone images, but it also supports regions that are not classified into any of the previous two categories, and thus are of a "generic" type. The encoder partitions the input page according to data type (text, halftone and generic regions) and encodes each segment using the compression technique with the best performance. The partitioning information is transmitted to the decoder and each type of data is grouped into a specific data segment.

Image areas that are not classified as text or halftone regions are compressed by using either the same context adaptive arithmetic encoding scheme used on JBIG or the Modified READ (MMR) technique used for fax standards groups 3 and 4 [43]. The text and halftone regions of the image are compressed using dictionary-based encoding procedures. For text regions, the original image is first scanned and a set of symbols is constructed and used to build a dictionary that is then transmitted to the decoder. The dictionary elements are encoded using the techniques defined for the generic image areas. For each character instance, the dictionary symbol and the position of the character (usually relative to another previously encoded symbol) are encoded, using either Huffman or arithmetic coding. For lossless or lossy-to-lossless (progressive) compression, a refinement pattern is transmitted for each symbol, after the first approximation with a dictionary element. The higher efficiency of lossy compression is achieved by neglecting the very small differences, that may occur among several instanced of similar characters, which do not affect the perceptual experience of the human observer. The halftone regions may be compressed by using a dictionary-based encoder, similar to the one described for the text regions. The halftone dictionary patterns are transmitted in a halftone dictionary segment, that is independent from the one used to transmit the text dictionary. Additionally, halftone regions may be encoded by converting them into grayscale levels, that are then compressed and transmitted to the decoder, where they are converted back into the corresponding halftone patterns.

JBIG2 uses a codebook determined *a priori* by the encoder, that is transmitted to the decoder. This codebook is then updated as the encoding progresses, by using new patterns that are compressed using one of the previously described methods. The new patterns are generated whenever a satisfactory match between the current image symbol and a dictionary element is not achieved. For some encoding modes, a different dictionary can be used in conjunction with the original codebook, in order to store refinement patterns that minimise the final distortion for each match. Dictionary design for JBIG2 is somewhat

different from the process used in generic grayscale images, due to the bi-level features of the input signal and the coding methods used by the standard. [44] presents a discussion of some of the used dictionary design methods. The JBIG2 compression algorithm has also been adopted by third party encoders, like the DjVu algorithm[1] [45, 46], as the algorithm for compression of the text regions of compound documents.

2.3.2 Matching pursuit

Matching pursuit (MP) is a coding paradigm that combines the concepts of transform-based and pattern matching compression. It has been proposed in [47] as an efficient way to encode the expansion of a signal using an overcomplete set of functions, stored as the code-vectors of a dictionary. In a traditional transform-based encoder, a signal $x(t)$ is decomposed into a linear combination of atom functions $f_i(t)$, weighted by the coefficients α_i:

$$x(t) = \sum_i \alpha_i f_i(t). \tag{2.1}$$

The atom functions depend on the used transform and span the signal space [32]. If $f_i(t)$ are not linearly independent (but span the signal space) we have a redundant or over-complete representation. Traditional transform-based encoders determine the set of transform coefficients for all atom functions and encode them, namely by using a quantisation procedure followed by a variable length encoding step. In MP, the redundant atom functions are stored in a dictionary and each input block, $x(t)$, is represented by a linear combination of *dictionary elements*, $g_i(t)$ (traditionally a family of 2D Gabor oriented wavelets). This means that each successive MP approximation step encodes both a dictionary index, representing one of the dictionary's atom functions, and the corresponding quantised coefficient.

Matching pursuits have been the focus of several research works for the last two decades. Several schemes have been proposed for both image [48, 49] and video coding [50], as well as for optimising the MP encoding algorithms, namely aspects like dictionary design, quantisation and rate-distortion optimisation [51, 52, 53].

[1] The DjVu (pronounced *déjà vu*) file format is a commercial image encoder used mainly for digital (scanned) document compression.

2.4 Conclusions

In this chapter we have presented a brief summary on some of the most important pattern matching data compression algorithms. The Lempel-Ziv and the vector quantisation families of methods were given greater relevance in this presentation. Two reasons account for their importance for our work: first, these methods are the most commonly associated with the pattern matching coding paradigm; and second, they are the genesis of a recently proposed algorithm that will be studied throughout this thesis.

The Multidimensional Multiscale Parser (MMP) algorithm was initially proposed as a generic lossy data compression method, but was promptly applied to image coding [1, 2]. MMP can be regarded as an extension of traditional VQ methods, that approximates data segments (in this case image blocks), using words from a multiscale adaptive dictionary $\mathcal{D}^l$. Also, it can be related with LZ methods, because it uses segments of the encoded signal as a source for new code-vectors. Nevertheless, the dictionary adaptation process and the scale adaptability of the pattern matching step are two unique features of this method, that account for its performance gains, when compared with traditional pattern matching algorithms.

A detailed explanation of the MMP algorithm is presented in the following two chapters. Chapter 3 presents a description of the algorithm, followed by an experimental evaluation of several aspects of MMP. The results of these tests revealed some weaknesses of the original method. The discussion of these results proved to be very influential in the work presented in this thesis. Chapter 4 presents a formal analysis of the MMP algorithm, that relates it to the traditional pattern matching algorithms. Also, unexpected affinities between MMP and transform-based decomposition schemes are revealed, that also motivated some of the work presented in this document.

Chapter 3

Multiscale recurrent patterns: the MMP algorithm

In the previous chapter we have described some of the most important pattern matching data compression algorithms. This chapter is dedicated to yet another pattern matching coding method: the multidimensional multiscale parser, or MMP, algorithm. After a detailed presentation of MMP for image coding, in Section 3.1, the remaining sections of this chapter present an experimental study of this method. In Section 3.2 we evaluate the experimental results of MMP for image coding and compare them with the performance of current state-of-the-art image encoders. These results establish some benchmark performances for our work: first, the performance of the initial MMP will serve as a comparison for all the new techniques that will be proposed in this text; second, the performance of the state-of-the-art encoders will serve as a reference for the results of the transform-quantisation-entropy encoding methods, that are currently accepted as being the state-of-the-art methods for image compression.

In the remaining sections of this chapter we present an experimental evaluation of some functional aspects of MMP. In Section 3.3 we analyse the bit stream generated by the MMP encoding process. In Section 3.4, MMP's dictionary updating is investigated. The importance of this procedure as well as some possible ways to improve its efficiency are revealed by this analysis. A discussion of these experimental studies is presented in Section 3.5.

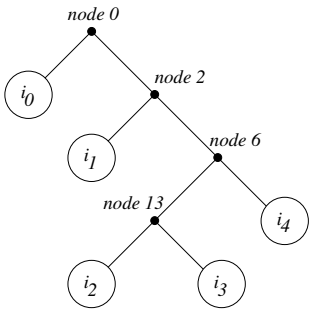


Figure 3.1: Binary segmentation tree of an MMP encoded block.

3.1 The MMP algorithm

This section describes the original Multidimensional Multiscale Parser (MMP) algorithm and its use for the compression of digital images. A discussion of its most important features is presented, as well as a brief summary of some successful applications of MMP for the compression of other signals.

MMP is a block-based coding algorithm, that divides the original image into adjacent blocks of uniform dimensions, that are compressed independently, using a standard raster scan order. Each image block is partitioned into variable size texture regions, using dyadic segmentations. These regions are encoded using vectors from a multi-scale adaptive dictionary, $\mathcal{D}$. The segmentation structure of each original block, $\mathbf{X}$, is represented by a binary segmentation tree, $\mathcal{T}$, as shown in Figure 3.1. Figure 3.2 represents the block segmentation resulting from the example of Figure 3.1.

Each sub-block (segmentation tree leaf) is represented by a dictionary vector, $\mathbf{S}_i^l$, identified by its index i and its scale l. Alternatively, the approximation of $\mathbf{X}^l$ may be obtained by using the concatenation of smaller words of the dictionary. In this case, a node is inserted in the segmentation tree. The decision to perform the bisection of a sub-block $\mathbf{X}^l$ is based on a rate-distortion criteria. The choice of partitioning each block is repeated recursively (see Section 3.1.1). Figure 3.2 shows an example where the used indexes for each of the sub-blocks are represented by the dictionary indexes i_0 to i_4. These indexes are associated with the corresponding leafs of the binary tree, $\mathcal{T}$, represented in Figure 3.1. The binary tree may thus be regarded as the MMP representation of the encoded block, because it holds all the information generated during the encoding process.

The root-node of $\mathcal{T}$ corresponds to the input image block. Each node represents the

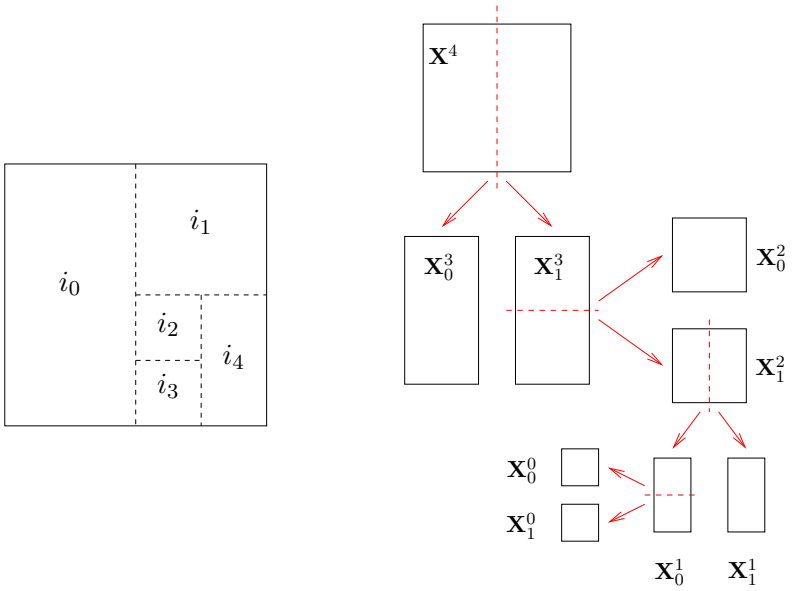


Figure 3.2: Example of the MMP segmentation decisions for a 4×4 input block (level 4).

decision to divide the corresponding block, while each tree leaf represents an non-segmented rectangular sub-block. There is a direct correspondence between the dimension of each sub-block and the level of its representation in $\mathcal{T}$. This is expressed by the use of superscript l, that identifies both the scale of $\mathbf{X}^l$ and the level of the segmentation tree that it belongs to. The segmentation of a block of scale l creates two blocks of scale $l-1$ with half the pixels of $\mathbf{X}^l$. Level 0 is the deepest level and corresponds to blocks of dimension 1×1. Square blocks, corresponding to even levels, are segmented into two vertical rectangles, meaning that a block of level l has dimensions

$$2^{\lfloor \frac{l+1}{2} \rfloor} \times 2^{\lfloor \frac{l}{2} \rfloor}. \tag{3.1}$$

where $\lfloor \quad \rfloor$ represents rounding towards the smallest integer [1]. As an example, a block size of 16×16 corresponds to blocks of scale 8, which means that the root of $\mathcal{T}$ is at level 8. The example of Figures 3.1 and 3.2 represents the segmentation of a block of scale 4 (4×4 pixels). During the coding process, the maximum considered scale corresponds to dimensions of the macroblocks processed by MMP, typically 16×16 (scale 8).

[1] Alternatively, the algorithm may partition the square blocks into two horizontal rectangles (see Section B.1).

3.1.1 Optimisation of the segmentation tree

An important process in MMP is the choice of the best segmentation tree for a given image block. Initial versions of MMP used the sum of squared differences (SSD) (see equation (B.4)) as the distortion measure, $D(\mathbf{X}^l, \mathbf{S}_i^l)$, and a fixed threshold, d^*, to make these decisions: if $d^* < D(\mathbf{X}^l, \mathbf{S}_i^l)$ then the block should be segmented, otherwise it should be approximated as a whole. This has the disadvantage of being a local decision, unable to globally optimise the segmentation process. An RD optimisation scheme based on a *Lagrangian multiplier*, λ [54, 55], was then developed. This allows the algorithm to weight both the minimum distortion achieved by the block approximation and the rate required to encode it.

The problem of minimising the rate $R(\mathcal{T})$ for a given maximum distortion D (or alternatively, determining the segmentation tree that allows a minimum distortion for a given maximum rate), is solved by minimising the value of a cost function $J(\mathcal{T})$, given by:

$$J(\mathcal{T}) = D(\mathcal{T}) + \lambda R(\mathcal{T}), \tag{3.2}$$

where $D(\mathcal{T})$ is the distortion of the approximation represented by $\mathcal{T}$ and $R(\mathcal{T})$ is the rate associated with it. Further details on the computation of this cost function are presented in Section B.1 of Appendix B.

The value of the Lagrangian multiplier, λ, is an input parameter of the encoder and is kept unchanged during the coding process. By setting the value of λ, it is possible to vary the compression ratio and the final distortion of the compressed image. Each λ value results in one (R, D) point for the compressed signal. High values of λ correspond to high compression ratios, due to an increased relevance of the rate factor in equation (3.2). On the other hand, low values of λ correspond to lower distortion and lower compression ratios.

Figure 3.3 shows the decision process for each node of the segmentation tree. For each block, the cost of encoding it as a leaf, $J(\mathcal{T})$, is compared wit the cost of segmenting its approximation into two blocks of level $l-1$. The option with the smaller Lagrangian cost is chosen. This procedure is recursively repeated for each node, starting at the bottom levels of $\mathcal{T}$ and progressing upwards, while testing all possible segmentation options. This results in an optimum segmentation tree, in an RD sense, for each block approximation. A more detailed description of the MMP algorithm is presented in Appendix B.

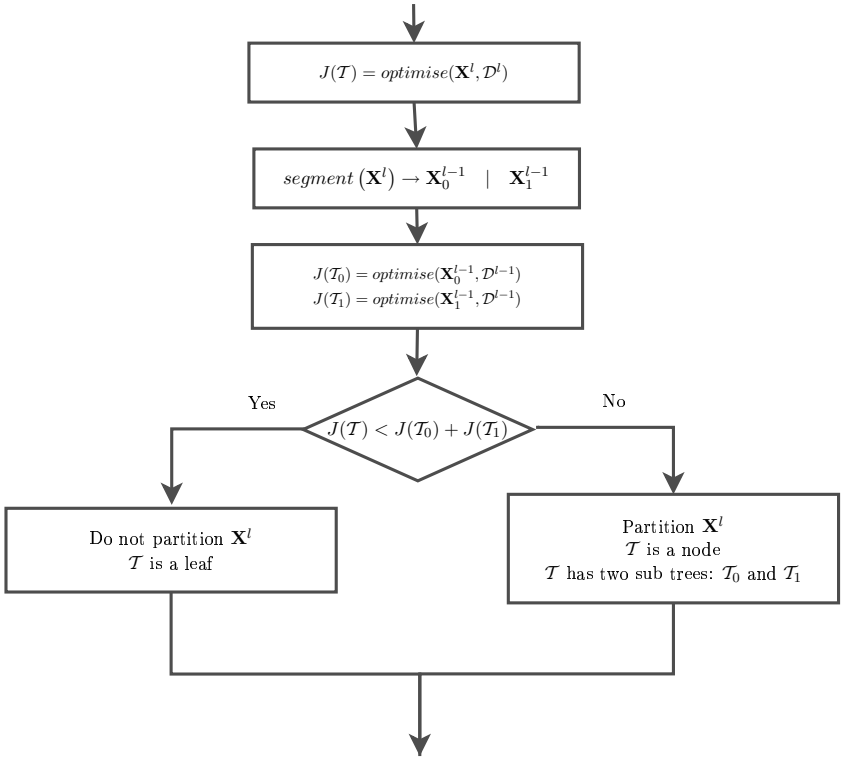


Figure 3.3: Flowchart for the MMP algorithm.

3.1.2 Encoding the MMP data

In order to transmit the MMP data, the binary tree representation is converted into an ordered string of symbols, where:

- a binary segmentation flag '0' is used to represent a block segmentation;
- a binary segmentation flag '1' indicates that the current sub-block should not be segmented;
- each flag '1' is always followed by one index symbol, used to encode the corresponding sub-block.

Since the blocks of level 0 cannot be further divided, no segmentation flag is used at this level, and only the dictionary index is encoded.

The conversion of the binary segmentation tree into a string of symbols is performed using a standard top-down order approach. For each node, the sub-tree that corresponds to the left branch is first encoded, followed by the right branch sub-tree. In the final bit-stream, each leaf flag is followed by an index, that identifies the word of the dictionary that

should be used to approximate the corresponding sub-block. In the example of Figures 3.1 and 3.2, $i_0 \ldots i_4$ are the indexes that were chosen to encode each of the sub-blocks, and so this block would be encoded using the following string of symbols:

$$0 \quad 1 \quad i_0 \quad 0 \quad 1 \quad i_1 \quad 0 \quad 0 \quad i_2 \quad i_3 \quad 1 \quad i_4.$$

In order to increase the compression efficiency of the encoding process, the symbols of the segmentation tree are encoded using an adaptive arithmetic encoder [56], with probability distributions that depend on the level and symbol type (flags or indexes).

Unlike conventional vector quantisation algorithms, MMP uses *approximate block matching with scales* and an *adaptive dictionary*. These particular features of MMP are responsible for its good relative performance and will be discussed in the following sections.

3.1.3 Scale adaptive pattern matching

Block matching with scales is an extension of ordinary approximate pattern matching, in the sense that it allows the matching of vectors with different dimensions. In order to approximate an original vector $\mathbf{X}^l$, using one code-vector $\mathbf{S}^k$, MMP uses a 2D scale transformation, T_k^l, to convert block $\mathbf{S}^k$, of dimensions $2^{\lfloor \frac{k+1}{2} \rfloor} \times 2^{\lfloor \frac{k}{2} \rfloor}$, into a scaled version, $\mathbf{S}^l$, of dimensions $2^{\lfloor \frac{l+1}{2} \rfloor} \times 2^{\lfloor \frac{l}{2} \rfloor}$.

The scale transformation, T, is implemented using traditional sampling rate change methods [32], that are easily extendable to blocks of any dimension, when separable transformations are used [57]. Detailed information about the transformations used in the original implementation of the MMP algorithm can be found in [2].

In [1, 2] a comparative study on the performance of standard VQ methods with that of VQ methods whose dictionary is built using scaled versions of input blocks is presented, for the case of Gaussian sources. The shown results demonstrate the advantage of the use of dictionaries with scales for these sources.

3.1.4 Adaptive dictionary

MMP uses an adaptive dictionary that is updated while data is encoded, with blocks that were created to approximate the original image data. When a block, $\mathbf{X}^l$, is segmented, instead of using a single word of the dictionary this block is approximated by the concatenation of the dictionary words used to approximate each of its halves ($\mathbf{S}_{i_{left}}^{l-1}$ and $\mathbf{S}_{i_{right}}^{l-1}$,

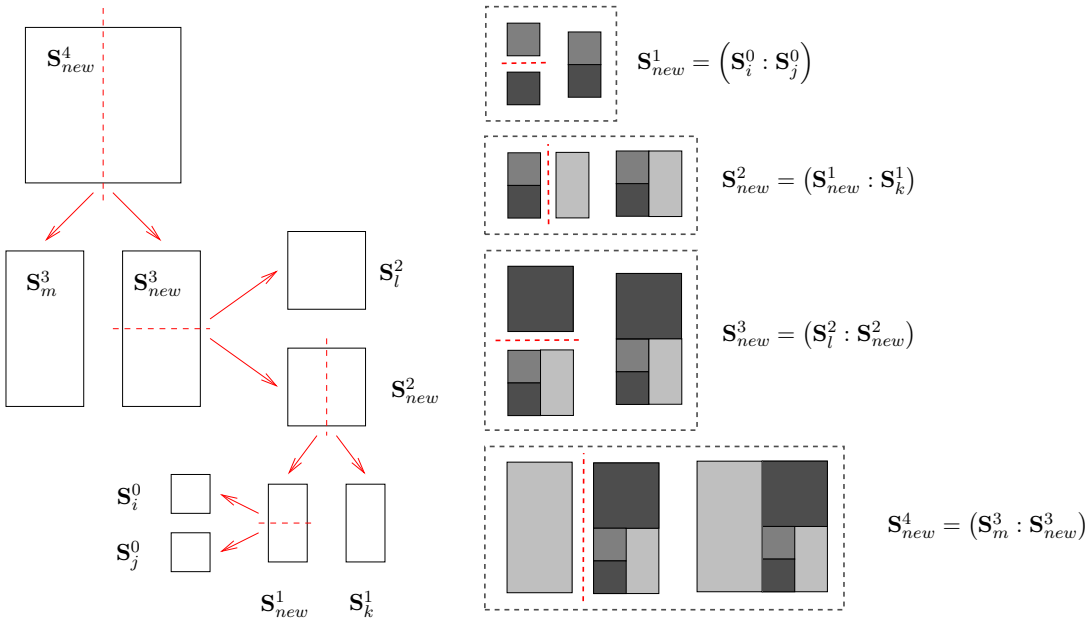


Figure 3.4: Dictionary update for the MMP algorithm.

respectively). In this case, the approximation of $\mathbf{X}^l$ is a new block $\hat{\mathbf{X}}^l$, that is defined by:

$$\hat{\mathbf{X}}^l = \left(\mathbf{S}^{l-1}_{i_{left}} : \mathbf{S}^{l-1}_{i_{right}}\right), \tag{3.3}$$

where (:) represents block concatenation. $\hat{\mathbf{X}}^l$ is thus generated adaptively by the encoding process, in order to approximate an image area that is different from the code-vectors available at the time. This new pattern uses a combination of existing blocks and is not only used to encode $\mathbf{X}^l$, but also to update the dictionary. The new block, $\hat{\mathbf{X}}^l$, thus becomes available to approximate future image areas. The scale adaptation procedure assures that this new pattern may be used to encode all future blocks of the image, irrespective of their size.

The dictionary update procedure for the example of Figures 3.1 and 3.2 is represented in Figure 3.4. In this example, the concatenation of two blocks of level 0, $\mathbf{S}^0_i$ and $\mathbf{S}^0_j$, was used in order to create a new pattern of scale 1, $\mathbf{S}^1_{new}$. This new vector is used to update the dictionary at all scales. In this example, vectors $\mathbf{S}^2_{new}$, $\mathbf{S}^3_{new}$ and $\mathbf{S}^4_{new}$ are also created from the concatenation of previously encoded patterns and thus cause new updates of the dictionary. Note that the dictionary updating procedure is performed as soon as the corresponding node of the segmentation tree is processed. This means that, in the blocks represented in Figure 3.4, pattern $\mathbf{S}^1_{new}$ is inserted first, while the top block ($\mathbf{S}^4_{new}$) is the last one to be inserted in the dictionary.

The MMP encoder performs the dictionary updating during the transmission of the binary tree data. In the decoder, the dictionary updating procedure uses only information that can be exclusively inferred from the encoded segmentation flags and dictionary indexes. This means that the decoder is able to keep an exact copy of the dictionary used by the encoder, using no side information.

Instead of using a single dictionary $\mathcal{D}$, that holds a unique version of each new block at its original scale, independent copies of the dictionary $\mathcal{D}^l$ are kept for each scale l. In order to do this, whenever a new pattern, $\mathbf{X}_{new}^{l_{orig}}$, of a given scale, l_{orig}, is created, the scale transformation and dictionary updating procedures are performed for all scales, *i.e.* for every available scale l, $\mathcal{D}^l$ is updated with:

$$\mathbf{X}_{new}^{l} = T_{l_{orig}}^{l}(\mathbf{X}_{new}^{l_{orig}}). \tag{3.4}$$

This is done in order to improve the performance of the block matching search. The use of $\mathcal{D}$ would force the search step to repeatedly perform scale transformations in order to adapt the dimensions of each code-vector to the size of the input block. With the use of several dictionary copies $\mathcal{D}^l$, the scale transformation is performed only once, during the insertion of the new vector. While searching for the best match, only the dictionary of the corresponding scale is used, avoiding the repetition of the scale transformation procedure.

In the updating process, a test condition is used to avoid inserting blocks that are already present in each dictionary $\mathcal{D}^l$. This has the advantage of allowing the dictionaries in different scales to grow independently. In general, although the initial dictionary has the same number of blocks for every scale, during the coding process the dictionaries for smaller scales tend to have less elements than those of larger scales. This happens because the scale transform, especially the down sampling case, is a many-to-one operation. Therefore, at smaller scales it is more likely that a vector $\hat{\mathbf{X}}^l$ will match an already existing vector.

MMP uses a very simple initial dictionary. In our implementations, it is composed by uniform blocks with 64 different intensity values, evenly distributed in the range $[0, 255]$, at all available scales. This initial dictionary, common for all images, has small approximation power, but the use of the dictionary updating procedure allows the algorithm to "learn" the patterns present in a particular image. This process not only enriches the dictionary, but it does so by adapting its patterns to the data that is being processed, enabling a good coding performance for a wide range of input data.

Due to the dictionary adaptation process and the use of adaptive arithmetic encoding, the node costs that are used in the encoding optimisation (see Sections 3.1.1 and B.1) are not completely independent. However, an optimum approach, that exhaustively considers all possible dictionary updates has a prohibitive computational cost. A search method that uses an intermediate dictionary to track the changes caused by each decision in the original dictionary during the approximation of each image block was proposed in [58]. This method is called MMP-RDI and produces very similar results to the full search RD optimisation, using a much simpler algorithm. Nevertheless, experimental tests demonstrated that the dependency effects introduced by dictionary updating during block optimisation are often negligible, especially when smaller macroblock sizes are used (8×8, or smaller). In spite of this, the RDI procedure will be used in this work.

3.2 Experimental results for MMP

In the previous section we have described the use of MMP for image coding. In this section we present an evaluation of its coding performance. The MMP results are compared with the ones of two well known state-of-the-art image encoders, namely JPEG2000 [12] and H.264/AVC high profile intra frame encoder [13, 59], for several image types.

3.2.1 Introduction and experimental setup

Several evaluations of the performance of the proposed algorithms for image coding will be presented throughout this text. For this purpose we use a set of test images comprised not only of natural smooth images, but also text and compound (text and grayscale) images. Six images were chosen because they are good representatives of these image types, allowing for a fair assessment of the proposed encoder for different image models. A small version of these images is displayed in Figure 3.5, while a larger resolution version is presented in Appendix A. Experimental results for some or all of these images will be presented along the main sections of this thesis. Occasionally, some results will not be included, in order to avoid a tedious repetition of similar relative performances. In some cases, additional results are presented in an appendix's section. In these cases a reference to their location is inserted in the text.

Four well known grayscale natural images were chosen: Lena, Goldhill, Peppers and

Figure 3.5: Some of the used test images. From left to right, top to bottom: Lena, Goldhill, PP1205, PP1209, Cameraman and Peppers.

Cameraman. The first three images have 512×512 pixels, while image Cameraman has dimensions 256×256. These images were chosen because of their extensive use in the image processing/compression literature. Besides these smooth, natural images, the performance of the tested algorithms will also be evaluated for other image types, namely text and compound (text and graphics) images. Two images were mainly used for this purpose: text image, PP1205, and compound image, PP1209, of dimension 512×512 pixels, were scanned, respectively, from pages 1205 and 1209 of the *IEEE Transactions on Image Processing*, volume 9, number 7, July 2000. These images were used because they are representative of the image types they belong to and also because of their use in previous MMP-related publications. All plots shown for these two images are limited to compression ratio intervals for which the visual distortion of the compressed images do not make them unusable, *i.e.* do not severely compromise the clarity of the text regions.

The current state-of-the-art image encoding methods that were chosen as a reference for this thesis are the JPEG2000 [12] and the H.264/AVC *high profile Intra frame* encoder [59]. These algorithms were chosen as performance benchmarks for all encoding methods described throughout this document. JPEG2000 has a representative performance for wavelet-based image encoders, that makes it a commonly used reference in image compression literature. In spite of having been developed for digital video compression applications,

the efficiency of the H.264/AVC's Intra frame encoding methods for image coding applications is well documented [60]. The performance of this method is generally accepted as being superior to that of wavelet-based schemes, namely JPEG2000. These observations were confirmed in this work, hence justifying the use of H.264/AVC in the comparisons. The *FRExt* high profile coding configuration was chosen because it achieves the best coding performance for the H.264/AVC encoder [60, 61].

3.2.2 Performance evaluation

In this thesis we adopted the ubiquitous convention for measuring the rate-distortion performance that represents the peak signal-to-noise ratio (PSNR) as a function of the compression ratio, measured in bits-per-pixel (bpp). The PSNR is the most commonly used measure of distortion in image compression systems. It is usually measured in decibels (dB) and defined as

$$PSNR = 10log_{10}\frac{255^2}{\frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\Big(I(m,n)-\hat{I}(m,n)\Big)^2} = 10log_{10}\frac{255^2}{MSE(I,\hat{I})}, \qquad (3.5)$$

where M and N are the dimensions (in pixels) of the original image, I, and of its noisy approximation, $\hat{I}$. MSE is the mean squared error between the two signals. Bits-per-pixel is a measure of the average number of bits that are used to represent each pixel of the encoded image. An uncompressed grayscale image with 256 levels uses eight bits to represent each pixel. This means that a value of 1 bpp is equivalent to a compression ratio of 8:1.

Figure 3.6 shows the rate-distortion performance of the MMP algorithm for smooth images Lena, Goldhill and Cameraman[2]. For image Cameraman, the MMP results are very close to the ones of JPEG2000, especially for high to medium compression ratios. Nevertheless, for the other smooth images, the PSNR performance of the MMP encoder is about 2 dB inferior to that of the transform-quantisation-based encoders. Figure 3.7 shows the RD performance charts for text image PP1205 and compound image PP1209. In these cases one may notice an inversion of the previously observed comparative results. For text image PP1205, the MMP encoder presents a PSNR advantage of about 3 dB over H.264/AVC and of about 4 dB over JEPG2000. For compound image PP1209, the MMP

[2]The results for test images Goldhill and Peppers are presented in Figures C.1 and C.2 of Appendix C.

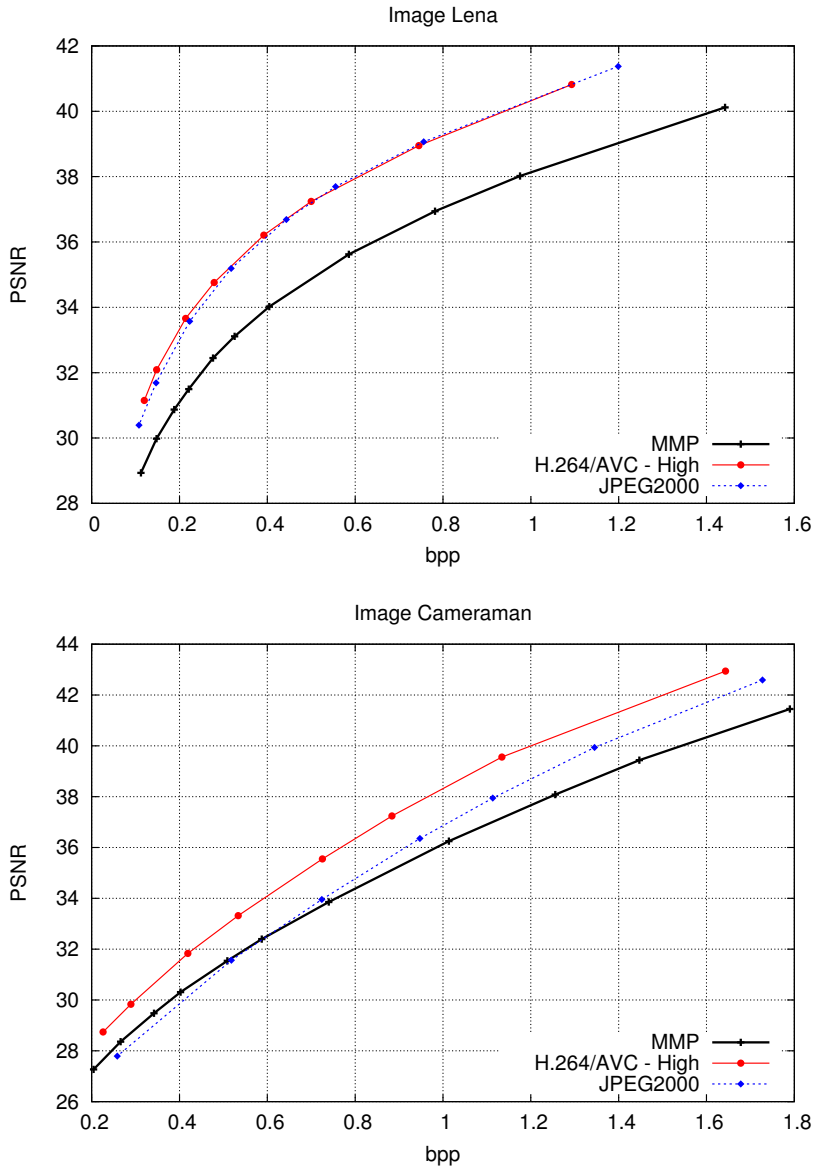


Figure 3.6: Experimental results of MMP for smooth test images Lena and Cameraman (256×256).

results are equivalent to those of the best state-of-the-art encoder (H.264/AVC) and about 1 dB better than those of JPEG2000.

From these results we notice a consistent performance advantage of the H.264/AVC encoder when compared with the JPEG2000 standard. This advantage is observed for all image types at all compression ratios. Concerning the MMP algorithm, we observe a good relative performance for non-smooth images, when compared with transform-quantisation-based algorithms. Nevertheless, for smooth images, the state-of-the-art methods have an

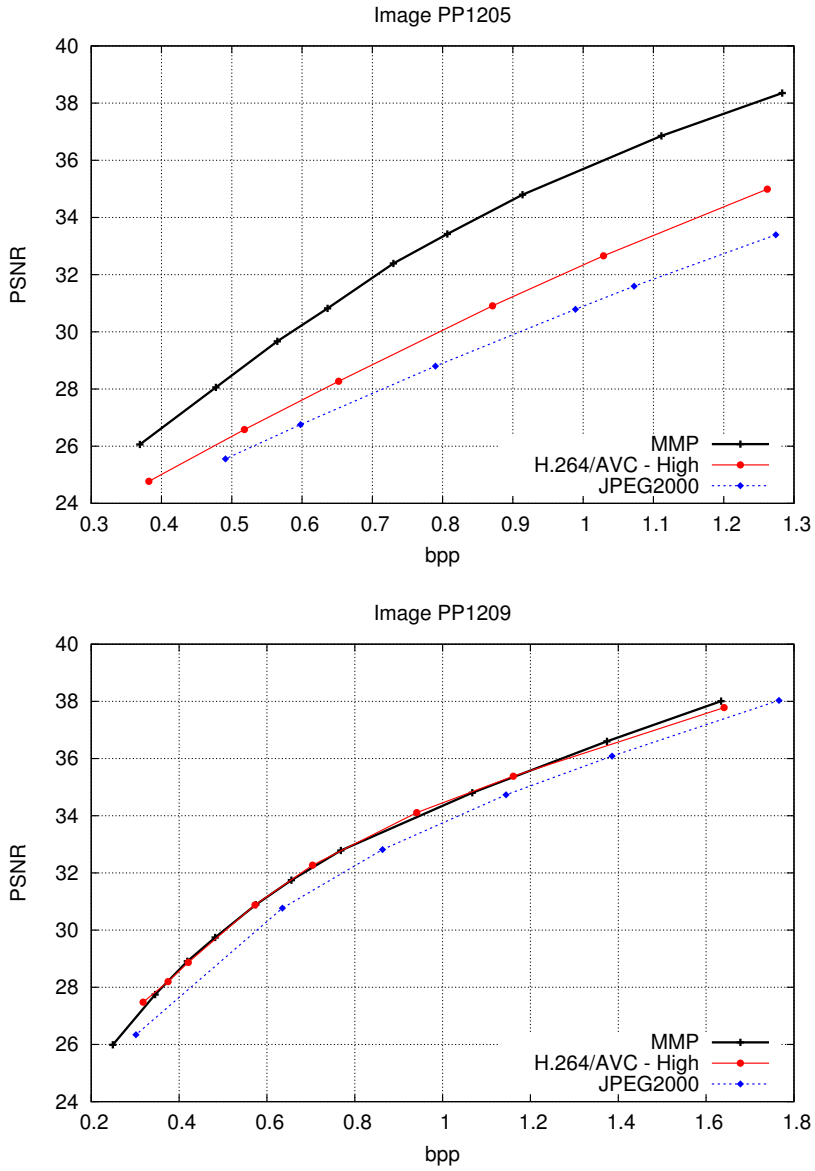


Figure 3.7: Experimental results of MMP for test images PP1205 and PP1209.

advantage over MMP. These observations justify one of the major objectives of this thesis, namely the investigation of new techniques that allow for the improvement of the MMP algorithm, especially for smooth images. The remaining sections of this chapter present experimental studies of some important aspects of the MMP algorithm. These new insights on MMP compression will be useful in the definition of new encoding techniques, that are presented along the following chapters of this thesis.

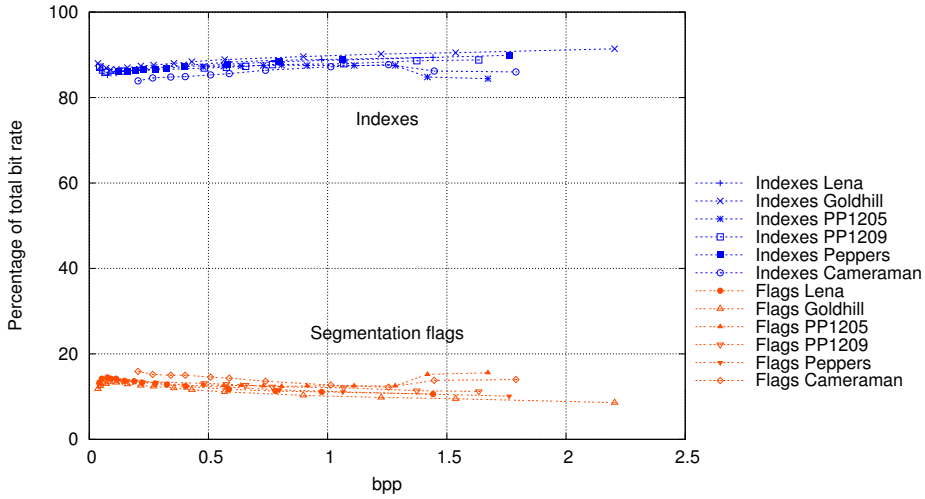


Figure 3.8: Bit rate usage for the indexes and segmentation flags, for MMP.

3.3 Analysis of the MMP bit stream

MMP encodes the input data by generating a string composed of dictionary indexes (i^l) and binary segmentation flags (f^l). Each symbol is encoded using an adaptive arithmetic encoder, that keeps independent probability contexts for each data type (index or segmentation flag) and block scale, l. Therefore, each symbol uses an adaptive probability model: $P(i^l|\mathcal{D}^l)$, for the indexes, or $P(f^l|l)$, for the flags, allowing for the exploitation of different per scale distributions of each symbol. This is important because the probability distributions of the symbols across different scales present different behaviours, as will be shown shortly.

Figure 3.8 shows the percentage of MMP's bit rate used by the indexes and segmentation flags symbols, for a wide range of compression ratios and different test images. This figure allows the observation of an important feature of MMP's bit stream: the indexes' symbols consistently use between 85 and 90% of the total bit rate, *independently of the input image and the final compression ratio.*

The higher rate associated with the dictionary's indexes' symbols is an expected consequence of the greater entropy of these symbols. This results from the very large cardinality of the adaptive dictionaries, when compared with only two possible segmentation flags, which have a maximum entropy of one bit. Considering the RD optimisation algorithm used by MMP, one would expect a larger number of block segmentations at higher bit rates, corresponding to lower values of the Lagrangian parameter λ. For decreasing values

of λ, the relative importance of the distortion factor in the final Lagrangian cost becomes progressively higher (see equation (3.2)). In this case the optimisation process will pursuit lower approximation errors at the cost of higher rates, resulting in the use of smaller blocks in the encoding process. This also means that for smaller compression ratios, the cardinality of the adaptive dictionary should increase (as will be observed in the next section). In terms of the effects on the bit stream, on one hand, the higher cardinality increases the entropy of the indexes' symbols, but on the other hand, the use of smaller blocks increases the number of segmentation flags and the importance of the bit rate associated with these symbols. Figure 3.8 shows that both bit rate components increase proportionally, when the compression ratio decreases.

The large amount of data associated with the MMP indexes means that the study of more efficient ways to encode these symbols has good potential to increase the overall performance of the method.

3.4 Analysis of the dictionary usage and adaptation process

In this section we analyse the adaptive updating of the MMP dictionary and the index usage in the encoding process. The adaptive dictionary is responsible for the good versatility of the algorithm, allowing it to learn a set of pixel patterns that efficiently represent the image blocks. The growth process of the MMP dictionary has a twofold influence in the method's performance: on one hand, large dictionaries tend to decrease the block distortion; on the other hand, the larger number of elements tends to increase the entropy of the index symbols. MMP benefits from the better approximation power, but suffers from higher average rates associated with a larger number of dictionary elements. In this section we evaluate the effects of these phenomena and present a set of experimental observations related with the updating process of the dictionary.

Figure 3.9 shows the evolution of the final cardinality for the MMP dictionary of largest scale ($\mathcal{D}^8$) for several test images. An approximately linear increase of the final dictionary size with the bit rate may be observed, for all input images. Also, the growth rate is similar for all tested images with dimensions 512×512. For image Cameraman (256×256) the growth rate is also constant, at about one quarter of that of other images; note that, since it has smaller dimensions, this is the same proportion for the number of image blocks.

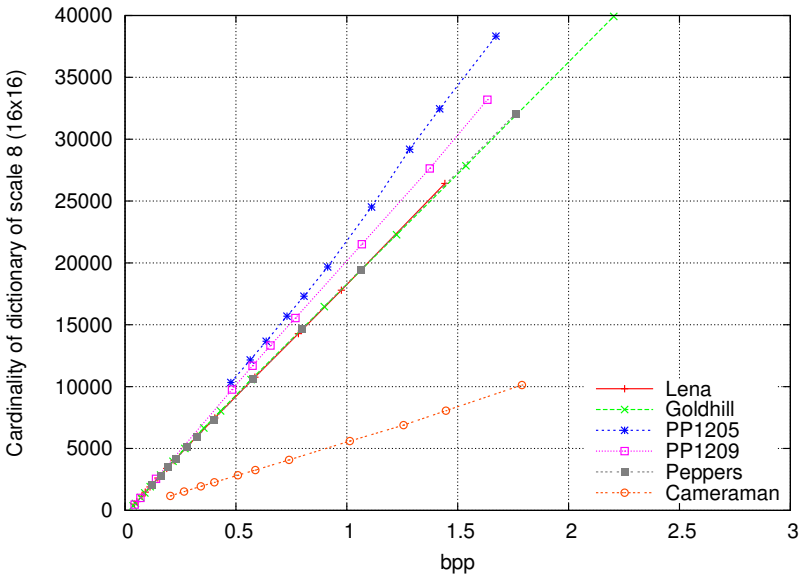


Figure 3.9: Final number of elements in the dictionary of scale 8 (16×16 blocks) for MMP.

This means that for a given target bit rate, the number of new dictionary elements that is created for each encoded pixel of the input image is approximately the same for every tested image and only depends on the final compression ratio, which can be directly related to the Lagrangian parameter λ used in MMP encoding.

In order to analyse this dependency, Figure 3.10 shows the final number of elements for every scale of the MMP dictionary, for images Lena and PP1205, using different compression ratios[3]. In these figures we observe the effect of the test condition mentioned in Section 3.1.4, that avoids the inclusion of similar blocks in a given scale of the dictionary, causing the dictionaries for each scale to grow independently. Because of the property of many-to-one mapping of the scale reduction process, we observe that dictionaries for smaller scales tend to have less elements than for larger scales. The results presented for image Lena are representative of what generally happens for smooth images. In this case we observe a significant reduction in the dictionary size only for $\mathcal{D}^0$ and $\mathcal{D}^1$. For non-smooth images (like PP1205) we also notice some reduction on the size of $\mathcal{D}^2$, due to the particular features of these images, namely the preponderance of a small number of grey levels.

Figure 3.11 presents the increase in the cardinality of $\mathcal{D}^8$ as the coding progresses

[3]The corresponding results for other test images are presented in Figure C.3 of Appendix C.

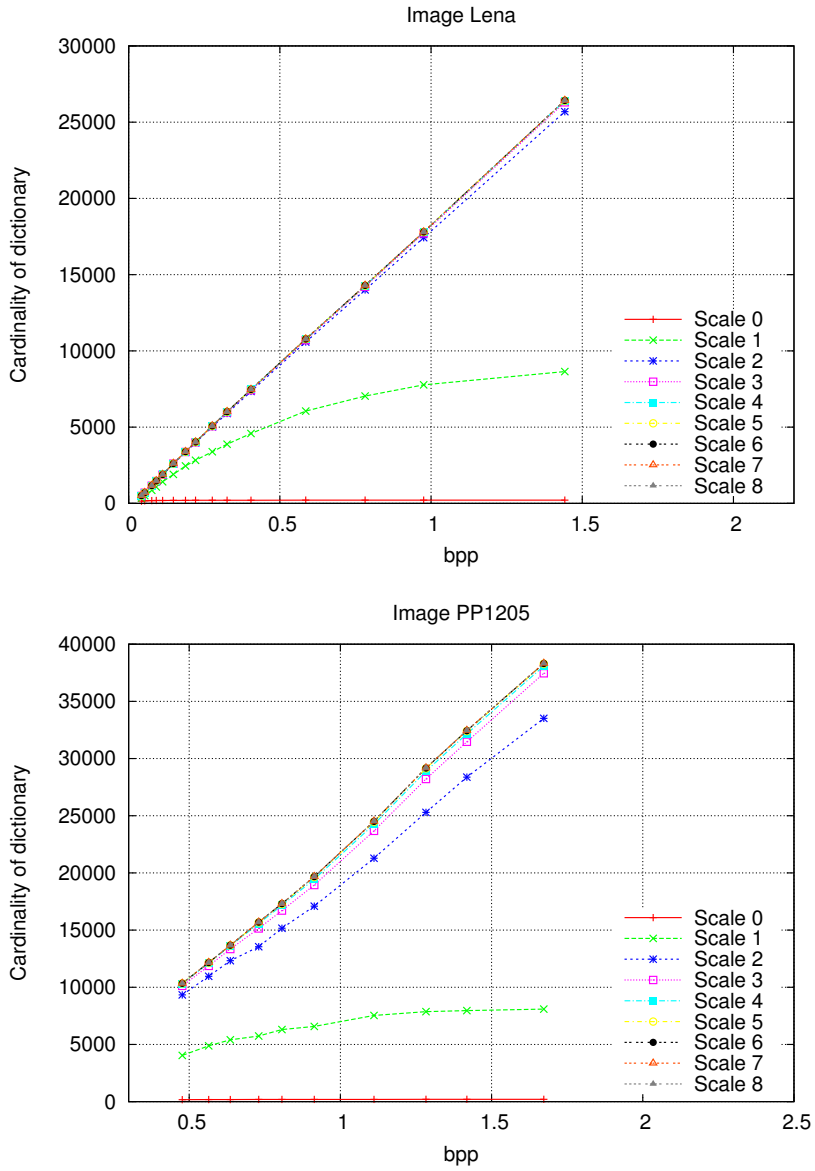


Figure 3.10: Final number of elements for all scales of the dictionary for MMP, for images Lena and PP1205.

(represented in terms of the number of processed macroblocks, scanned using a standard raster order) for images Lena and PP1205 at various compression ratios. We observe that the growth rate of the dictionary is approximately constant for these images, but depends, to a small extent, on the area of the image that is being encoded. Smoother areas of the image tend to be approximated by larger blocks, which slows the growth rate of the dictionary (as can be seen for the initial blocks of image PP1205, that correspond to a smooth area). Areas of the image with more detail tend to cause more block segmentations,

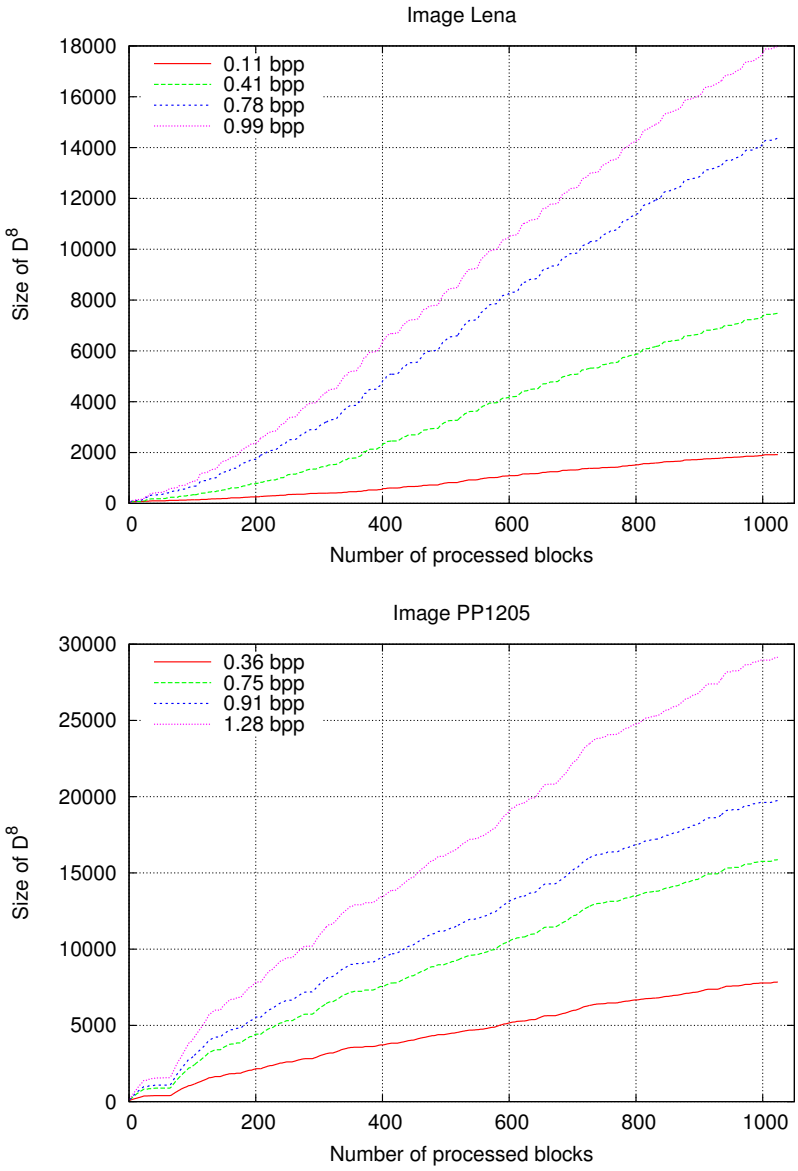


Figure 3.11: Evolution of the number of elements for scale 8 of the dictionary for MMP, for images Lena and PP1205.

increasing the number of dictionary updates. Similar observations were made for other test images. Figure 3.11 shows that the growth rate of the dictionary tends to slow down as the encoding progresses. This is a result of the learning process of new image blocks, that allows the method to use already existing vectors of the dictionary, instead of having to segment the input blocks, originating new indexes. Figure 3.12 shows the growth rate for dictionaries of all scales, for image Lena encoded using approximately 1 bpp. In this figure we clearly notice the learning effect for the dictionaries of scales 0 and 1. Nevertheless, for

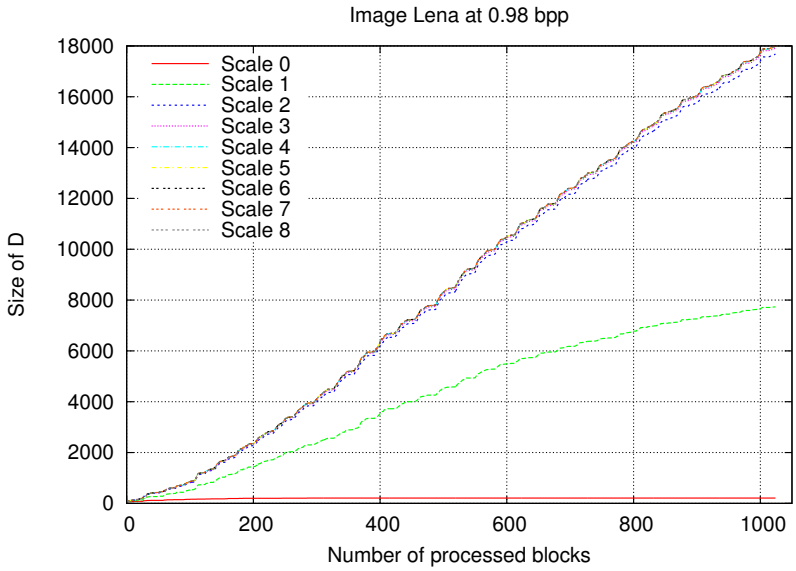


Figure 3.12: Evolution of the number of elements for all scales of the dictionary for MMP, for image Lena encoded at 0.98 bpp.

the dictionaries of higher scales the learning process has a very small effect in slowing the growth rate of the indexes.

Figure 3.13 shows the final number of elements for each scale of the dictionary and compares these values with the number of blocks that were encoded by MMP at each scale, for image Lena. It clearly demonstrates that, as the target compression ratio decreases, MMP tends to progressively use smaller vectors of the dictionary (*i.e.* a larger number of segmentations) to encode the input blocks. We may observe that for 0.11 bpp, the input blocks are approximated using mostly large blocks, *i.e.* vectors from the higher scales of the dictionary. For larger bit rates we notice a tendency to use smaller blocks in the MMP approximations. This has the effect of increasing the final number of dictionary elements for all scales, due to the MMP updating scheme.

Another relevant phenomenon observed in these plots is the fact that the final number of elements for each scale l of the dictionary is much larger than the actual number of blocks that were approximated at that scale, for all target compression ratios. This means that there are a lot of dictionary elements that are never actually used in the approximation of the original image patterns. This effect, shown here for image Lena, was also observed for all other tested images. The unused blocks have the unfavourable effect of increasing the entropy of the dictionary's indexes, without effectively improving the dictionary approxi-

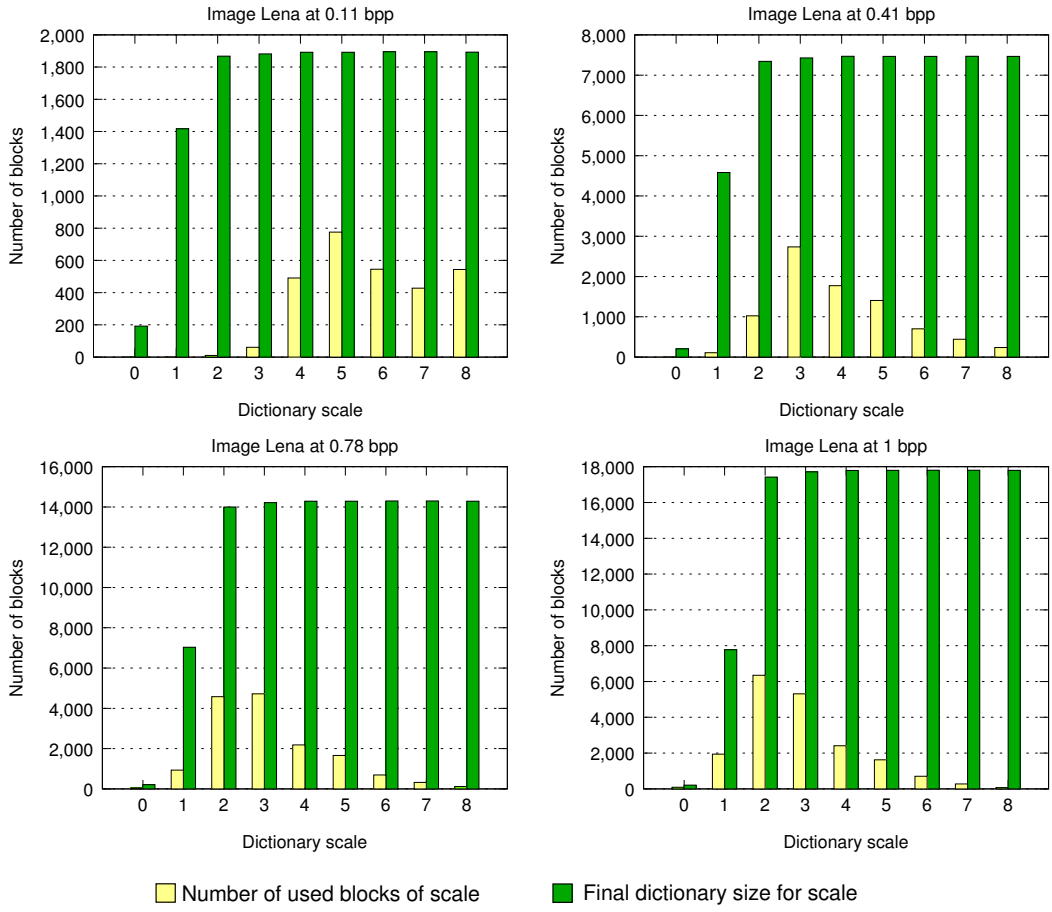


Figure 3.13: Final number of elements at each dictionary scale and the actual number of blocks used by MMP at each scale, for image Lena..

mation power, *i.e.* the ability to achieve a low distortion when using a dictionary vector to approximate an image block. This suggests that further developments in MMP dictionary adaptation process may be worth investigating.

3.4.1 First tests on dictionary adaptation

In order to investigate the dictionary adaptation process, we have implemented some techniques that aim to increase the efficiency of dictionary usage by eliminating some of the unused blocks. An obvious technique for controlling the growth of the MMP dictionary is to impose a limit to the number of elements that are inserted in each $\mathcal{D}^l$. All previous versions of MMP, from its initial proposal to the several implementations that have been developed, do not use such a limit. The reason for this is that, historically, informal tests revealed that these type of techniques consistently lead to performance losses. Neverthe-

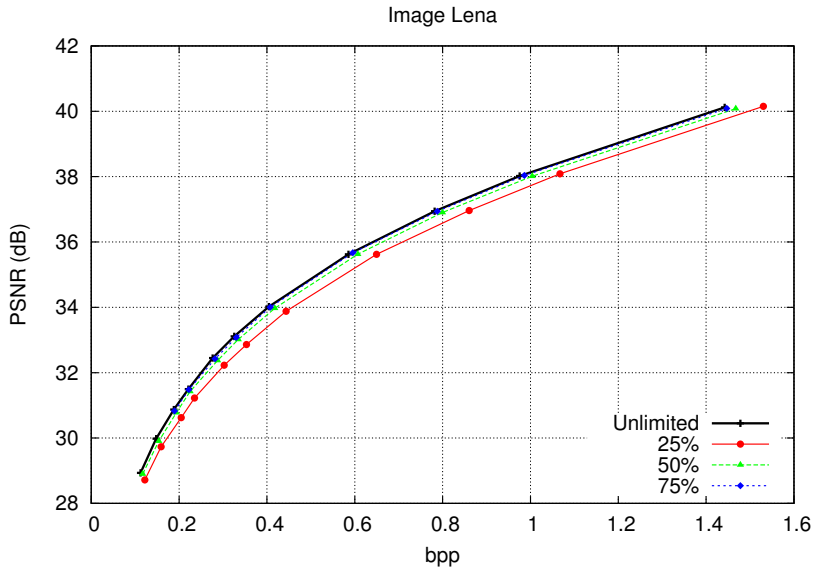


Figure 3.14: Results for MMP when dictionary updating is interrupted at a percentage of the total number of encoded blocks, for image Lena.

less, a thorough study of the effects of limiting the dictionary growth was performed, in order to clarify this issue.

Setting a maximum size for the dictionary may be performed by using one of two basic schemes:

- The dictionary is allowed to grow freely up to the maximum size, where the updating process is terminated;
- Alternatively, once the maximum size has been reached, the inclusion of a new element in the dictionary is done by replacing an existing vector.

These two methods were studied and their experimental results are presented and analysed in what follows. Interrupting dictionary adaptation was tested using several thresholds corresponding to the point at which the learning process was suspended. Figures 3.14 and 3.15 present the results of these tests for images Lena, PP1205 and PP1209[4]. The plot key shows the point at which the updating process was terminated, in terms of the percentage of the total number of blocks (these test images have 1024 16×16 blocks). All of the tests led to a decrease in the RD performance of the algorithm. An equivalent performance to that of the original method was only reached for some cases, when almost

[4]The corresponding results for other test images are presented in Figure C.4 of Appendix C.

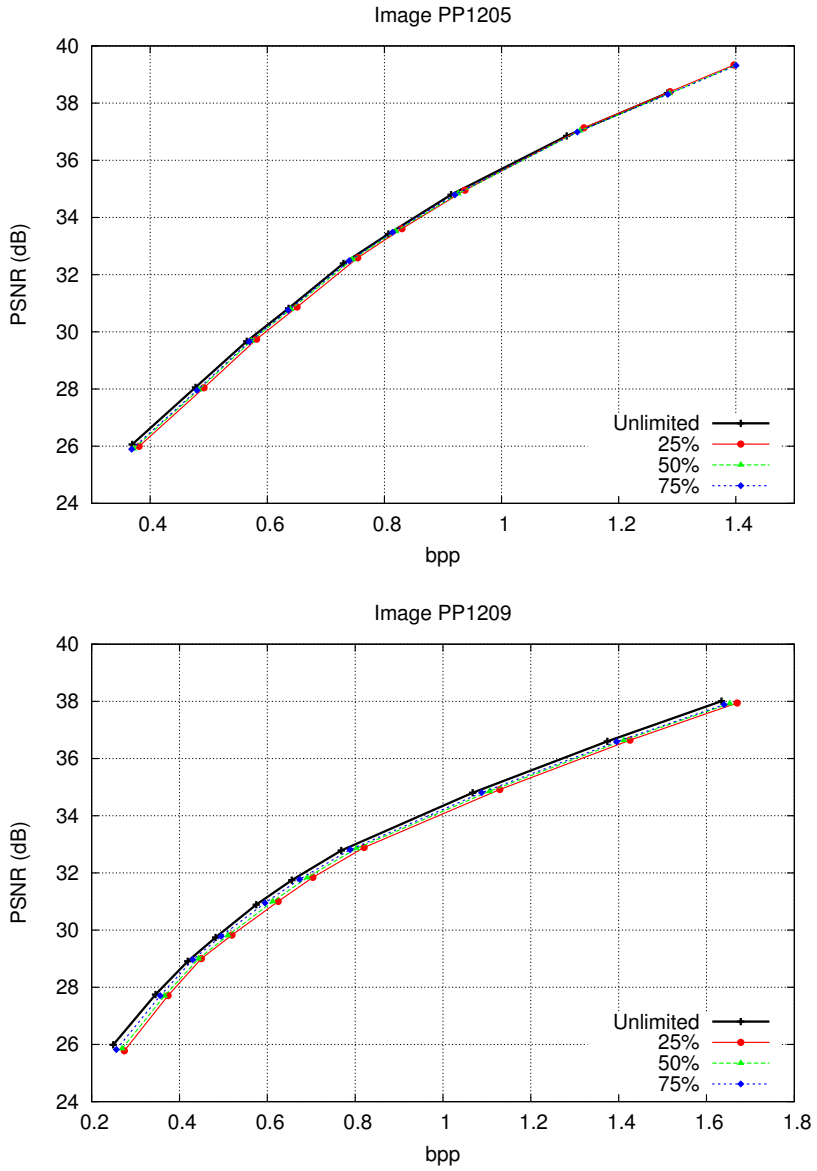


Figure 3.15: Results for MMP when dictionary updating is interrupted at a percentage of the total number of encoded blocks, for images PP1205 and PP1209.

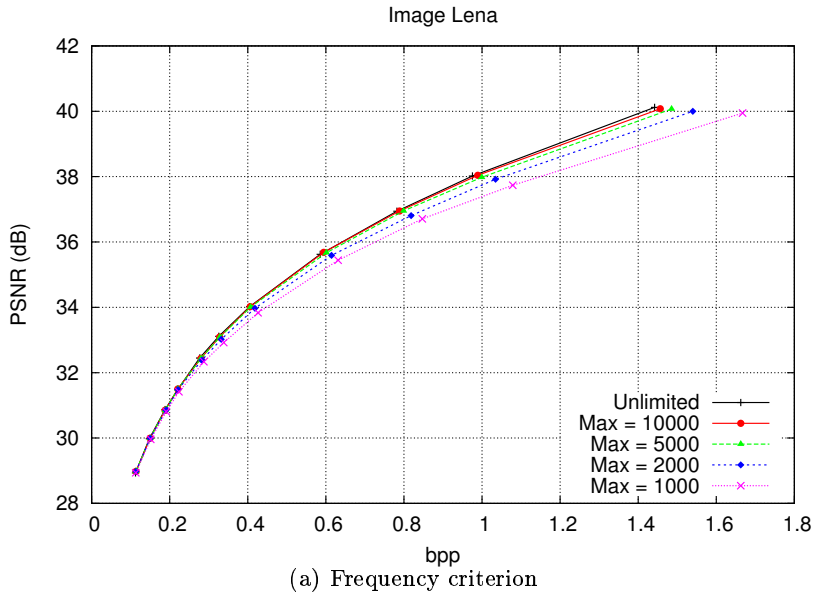
every block of the image (more than 75%) was used in the learning process. It is also possible to observe that the quality losses depend on the input image: for text image PP1205 we observe a small decrease in the performance, while for smooth images the effects of interrupting the updating process are more noticeable. Two factors explain this: first, the text image patterns consist of combinations of black and white pixels, that can be more easily approximated when only a small number of blocks was used for dictionary adaptation. This is not the case for the smooth images, whose patterns' characteristics vary

considerably with the position of the block in the image. The second factor that explains the reduced effect of these techniques for text images is that, once the dictionary updating process is concluded, the MMP encoder compensates for the poorer set of dictionary vectors by using smaller blocks in the approximations. This causes an increase in the bit rate, that explains the reduction in the performance. Because MMP originally uses much more segmentations for text and compound images than for the smoother ones, this effect has a much greater impact for natural images.

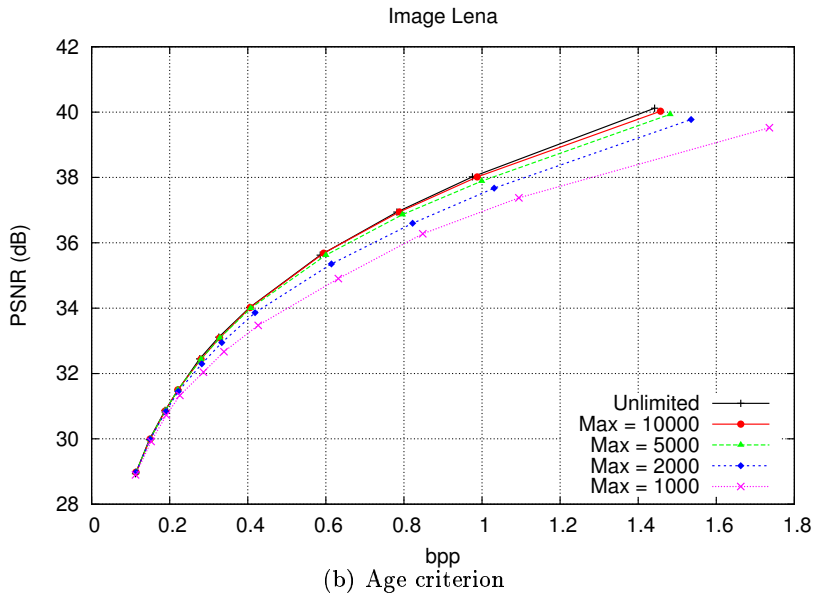
The second option to limit the dictionary growth is to set a maximum size for the dictionary. Once the dictionary reaches a threshold size, each new insertion is only performed once an existing vector is removed. In our tests we have also investigated which dictionary vector should be eliminated. Two criteria were used: a *frequency criterion*, where the least used block is chosen and an *age criterion*, that eliminates the block which has not been used for a longer time. If there are blocks with a similar age, one should choose the one that has been used less times. Similarly, the frequency criterion uses an age measure to choose among vectors that were used the same number of times. We can see the results for these tests in Figures 3.16 and 3.17, for images Lena and PP1205, respectively[5]. From these figures we observe that the frequency criterion is consistently better than the age test. However, even this criterion most often results in performance losses, when compared with the case where the dictionary is allowed to grow freely. An exception to this observation is the text image, for which the restriction of dictionary size combined with the frequency criterion introduces small gains for higher rates. Again, the gains for this particular case may be explained by the arguments presented earlier.

In our tests the maximum dictionary size was fixed for all scales and rates. One would intuitively accept that the optimum value for this threshold would depend on the input image and on the target rate for the encoding process: larger compression ratios should allow for smaller dictionaries, while higher rates should impose larger dictionary sizes. Experimental observations led to the conclusion that the optimum maximum size was consistently very close to the number of elements that the original dictionary would have if no restriction was used. In another test we have used a scale adaptive threshold, that sets the maximum values for the dictionary size depending on the current scale. This test was justified by the observation that the number of blocks for each scale that were actually

[5]The corresponding results for other test images are presented in Figure C.5 and C.6, of Appendix C.



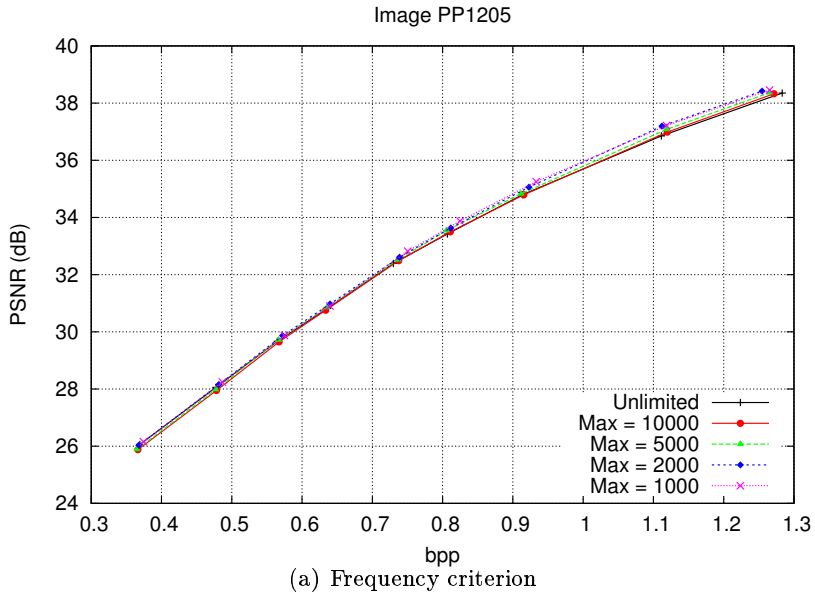
(a) Frequency criterion



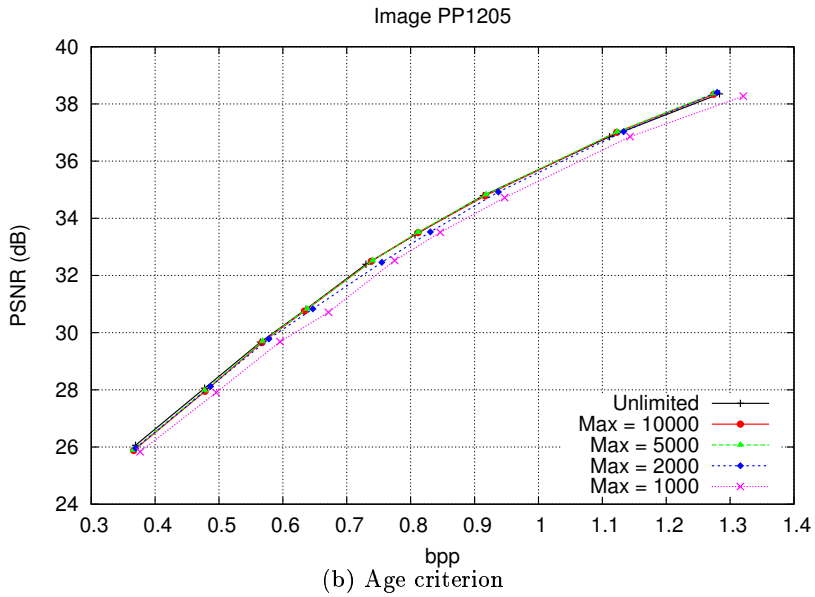
(b) Age criterion

Figure 3.16: Results for MMP using a maximum dictionary size and the two tested elimination criteria, for image Lena.

used in the encoding process varied significantly. Several tests were performed using this framework: more elements were allowed for the most used scales, the maximum number of elements was set to grow with the current scale (larger scales use larger dictionaries), etc. A related test established the maximum size of the dictionary of each scale to the actual number of elements that would be used if no restriction was imposed. The results for this test revealed severe losses in the encoding performance.



(a) Frequency criterion



(b) Age criterion

Figure 3.17: Results for MMP using a maximum dictionary size and the two tested elimination criteria, for image PP1205.

One concludes that these attempts to restrict the dictionary size generally compromise the coding performance. This is because the used criteria to eliminate blocks from the dictionary do not take into consideration the future impact of that elimination: one block may have been useless up to the present moment, but one has no way of knowing if this will be the case for future image blocks. Also, optimising the maximum dictionary size threshold would be a hard task, since this would depend on the input image and encoding

parameters. Nevertheless, these techniques may be useful in a future framework, were one would be willing to accept a trade-off between image quality and encoding complexity. That is not, however, the main focus of this work, that concentrates on developing new techniques that enable rate-distortion performance gains of the MMP methods, mainly for smooth images.

3.5 Concluding remarks

In this chapter we have presented the multidimensional multiscale parser algorithm, or MMP. This method is the main focus of the investigation described in this thesis, which accounts for the detailed explanation of this algorithm, that was presented in Section 3.1. Section 3.2 showed an experimental evaluation of the image coding performance of MMP. The results presented in this section revealed the adaptability of MMP. A wide range of images types are efficiently encoded by using an initial dictionary built from a very sparse set of uniform blocks and adapting it to the patterns that are being encoded. A comparative analysis of the coding performance was also done, using two state-of-the-art transform-based image encoders, JPEG2000 and H.264/AVC high profile Intra-frame encoder. This comparison revealed excellent results for non-smooth images, for which MMP is able to achieve a better performance than JPEG2000 and H.264/AVC. However, this is not the case for smooth images, where MMP shows an inferior performance.

In order to pursue one of the major goals of this thesis, *i.e.* to develop new coding schemes that improve the performance of MMP, especially for smooth images, an experimental analysis of some of its most important aspects was carried out in the remaining sections of this chapter. The discussion of the presented results clearly demonstrate the usefulness of more efficient ways of improving the learning process of the MMP dictionary. This study raised some interesting questions that motivated many investigation lines, that are described in the next chapters.

Section 3.3 demonstrated that the indexes encoding account for the majority of the generated bits of the MMP file. New ways to increase the efficiency of the indexes' entropy encoder, that use context adaptive arithmetic encoding, are investigated in Chapter 6.

Section 3.4 revealed some inefficiency issues related with the MMP dictionary updating process. Also, preliminary tests, presented in Section 3.4.1, showed that the use of trivial

solutions to limit the dictionary growth has a negative effect on the overall rate-distortion performance of the method.

This motivated the investigation of other efficient dictionary adaptation methods, that are presented later, in Chapters 5 and 6. In the following chapter we present some new insights on MMP. Using different points-of-view, we are able to relate MMP not only to traditional pattern matching algorithms, as LZ and VQ-based methods, but also to transform-based encoders and non-uniform sampling schemes.

Chapter 4

New insights into MMP

In spite of its unique features, the Multidimensional Multiscale Parser algorithm has a lot in common with other pattern matching algorithms, like string matching (Lempel-Ziv) [3, 4] and vector quantisation [5], previously described in Section 2.2. The first two sections of this chapter present an analysis of the MMP algorithm considering both the LZ and VQ viewpoints. The main similarities and differences between MMP and these methods are highlighted.

Other perspectives on MMP are also presented in this chapter. Section 4.3 investigates some interesting relations between MMP and a signal decomposition technique. Section 4.4 develops another somewhat unexpected insight into MMP, by presenting it as a non-uniform sampling scheme. This is followed by a study on possible interpolation schemes, that reconstruct the input signal from the MMP-generated samples, as an alternative to the standard MMP decoding process.

4.1 A string matching point of view

Being a dictionary-based method, MMP shares some common features with the LZ family of string matching encoders. The explicit use of a dictionary relates MMP with an LZW-based method (see Section 2.2.1), since no individual symbol is transmitted with each code-vector. As in the LZ78 and LZW cases, the initial dictionary stores only a small set of basic symbols. Because MMP does not have the ability to transmit new patterns without the use of code-vectors, one condition for MMP lossless coding is the insertion of all possible symbols in the initial dictionary. As in LZ encoders, the definition of new

code-vectors is based on the concatenation of previously encoded parts of the message with recently encoded patterns. In LZ, this concatenation uses the last encoded element of the message, that is transmitted independently. MMP uses a much more flexible dictionary updating process, in the sense that it may use the concatenation of any two existing code-vectors. Also, it does not have to explicitly transmit any message block nor any other side information for the dictionary adaptation process. As in LZ-based methods, both the encoder and decoder use similar dictionary updating procedures, that guarantee the use of synchronised copies of the dictionary at all times.

One important feature of MMP is the implicit use of blocks of different scales. LZ77 methods use adaptive block sizes in the matching process by explicitly transmitting the length of the used pattern. In LZ78 and LZW this is avoided by performing the match with a code-vector which has implicit dimensions. Nevertheless, LZ-based methods use each dictionary pattern in its original dimensions while MMP has the ability of using the patterns in the dictionary at different scales. This means that MMP is able to approximate the new image segments not only by using previously transmitted image patterns, but also by using one of the patterns created by the scale transforms. This increases the flexibility of the MMP dictionary updating procedure, that is able to adapt to the image patterns much faster. This is important for the coding efficiency of the method, because as for the case of LZ-based algorithms, the initial dictionary usually contains only a reduced set of simple patterns.

4.2 A vector quantisation point of view

The main affinities between MMP and VQ-based algorithms lie on the use of a pattern dictionary. In fact, if we disregard the scale adaptive encoding procedure on MMP we would get a compression algorithm that could be classified as an adaptive VQ method. Nevertheless, even this fixed scale version of MMP would have innovative features for a VQ method, like the dictionary updating procedure (note that this procedure would have to use a scale transformation on the concatenation of two blocks of a fixed scale, l, in order to produce a pattern of scale l). Thus, the MMP dictionary updating procedure, that was shown to be one of the key factors for its coding performance, is also one of the major distinctions between MMP and the traditional VQ methods.

The second major difference is the use of adaptive block sizes for the matching procedure. By adaptively choosing blocks of different dimensions, MMP is able to achieve an arbitrary rate-distortion trade-off for the signal representation. From a VQ point of view, the MMP scale transformation procedure may be regarded as an implicit adaptation of the quantisation vectors that are available to approximate the signal. For a given scale l, the quantisation vectors of a standard VQ method would correspond to the blocks stored in level l of the MMP dictionary, $\mathcal{D}^l$. In traditional adaptive VQ schemes, if the matching step fails with the vectors of $\mathcal{D}^l$, one must encode the new pattern $\mathbf{S}_i^l$ as side information [5]. This pattern is then inserted in the codebook and a new quantisation vector becomes available to the encoder. In MMP, the available quantisation vectors correspond not only to the blocks of $\mathcal{D}^l$, but also to all possible concatenations of blocks from the smaller scales dictionaries, $\mathcal{D}^k$, with $k = 0, .., l-1$. If $\mathcal{D}^0$ stores blocks of 1×1 pixels with all values within the dynamic range of the image, it is possible to cover all signal space with the combination of these patterns, resulting in a lossless encoder. This capability of arbitrarily concatenating the blocks of smaller scales provides a large number of approximation choices (quantisation vectors); this lends MMP a good deal of versatility. Also, due to the scale transformations, each concatenation of two blocks of smaller scales results in new code-vectors for all scales. This increases the available quantisation vectors exponentially, without requiring the transmission of additional data for the dictionary updating procedure.

4.3 A signal decomposition point of view

Transform-based coding is probably the most popular paradigm for image compression. Most of the state-of-the-art methods use this paradigm, combining it with quantisation and entropy coding, in order to efficiently compress the images' spectral data [62]. The transform step compacts the input signal's information into a sparse set of transform coefficients, that are encoded according to their individual features. The remaining values, that hold a negligible amount of information, are then either discarded or coarsely quantised, achieving high compression levels with low distortion values.

Transforms are a special case of signal decomposition. In this section we present an analogy between MMP and a signal decomposition algorithm, that uses redundant atom

functions. The initial MMP dictionary uses a set of discrete rectangular pulses with different amplitudes and scales. The approximation of the input signal can be regarded has a combination of these pulses, using appropriate amplitudes, shifts and scales, that are determined by the MMP's rate-distortion optimised coding algorithm. In our analogy we thus relate MMP coding to a signal analysis algorithm. The decoding (or synthesis) process uses the same basis functions to reconstruct the approximated signal.

4.3.1 Signal decompositions

We start by reviewing some basic notions about signal decompositions using the simpler 1-D case. A transform decomposes the input signal, $x(t)$, into a linear combination of functions, $f_i(t)$, weighted by the coefficients, c_i:

$$x(t) = \sum_i c_i f_i(t). \tag{4.1}$$

Functions $f_i(t)$ are called *synthesis* functions, as they correspond to the "building blocks" of the reconstructed signal. The transform coefficients, c_i, can be determined by evaluating the projection of an input signal into a set of *analysis* functions, $g_i(t)$. This is done by determining the inner product between $x(t)$ and $g_i(t)$:

$$c_i =< g_i(t), x(t) > . \tag{4.2}$$

Some commonly used transforms are the Fourier transform (FT) and the cosine transform (CT) as well as their discrete counterparts, DFT and DCT. The use of the discrete cosine transform for digital image compression is very common, due to the popular JPEG image coding standard [11].

For the discrete case, if any input signal, $x(n)$, of length N, can decomposed into a set of N linearly independent sequences $f_i(n)$, then f_i, $i = 1, ...N$, define a *basis* of the signal space [63]. In this case we have

$$x(n) = \sum_{i=1}^{N} c_i f_i(n). \tag{4.3}$$

Any signal may thus be represented by the set of coefficients $\{c_1, c_2, ..., c_N\}$, which implicitly corresponds to a linear combination the basis sequences

$$x(n) = c_1 f_1(n) + c_2 f_2(n) + ... + c_N f_N(n). \tag{4.4}$$

Another important case is the use of redundant (*i.e.* linearly dependent) functions for signal decomposition. In this case, a signal $x(n)$ is decomposed into the sum of a set of M *atoms*, $f_m(n)$:

$$x(n) = \sum_{m \in I_M} c_m f_m(n), \tag{4.5}$$

with $m \in I_M$ and $I_M \subset \{1, ..., k\}$. The k atom sequences, $\{f_1, ..., f_k\}$, define a $frame$ for the N-dimensional signal space and one typically has that $k \gg M$ [63]. The previous discussions can be expanded to the two dimensional case, either by using original 2-D functions or by computing them with a separable combination of the 1-D functions (in which case we have a so-called separable transform) [64].

4.3.2 MMP signal decomposition

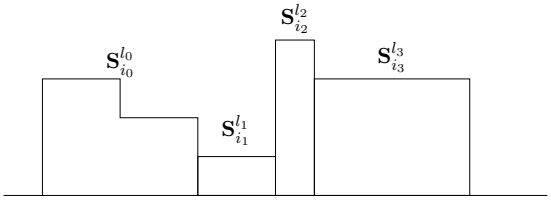
In this section we describe some aspects of MMP compression that can be related to a signal decomposition process. The intention is not to formally perform a theoretical analysis of MMP as a signal decomposition, but to point out some analogies between these two different paradigms. In the following analysis we consider the particular case of an initial dictionary composed only by impulses and rectangular pulses; the scale transformations consist of zero order interpolators (for the expansions) and simple decimators (for the contractions). This ensures that the concatenation of any number of blocks composed by rectangular pulses is still a concatenation of rectangular pulses at any scale. Figure 4.1 has a representation of this case[1]. Note that this is not the case for the original MMP, since its scale transforms use an interpolation (low pass) filtering. Some comments on the implications of using scale transformations with a filtering effect are presented later in this section.

In MMP encoding, the approximation of each input block, $\mathbf{X}^l$, can be regarded as a combination of several dictionary vectors, $\mathbf{S}_i^{l_i}$, at different scales $l_k \in [0, l_{Max}]$:

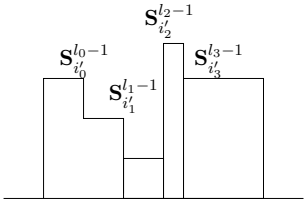
$$\hat{\mathbf{X}}^l = [\mathbf{S}_{i_0}^{l_{i_0}} \quad \mathbf{S}_{i_1}^{l_{i_1}} \quad \mathbf{S}_{i_2}^{l_{i_2}} \quad ... \quad \mathbf{S}_{i_k}^{l_{i_k}}], \tag{4.6}$$

where l_{Max} is the largest scale used by MMP.

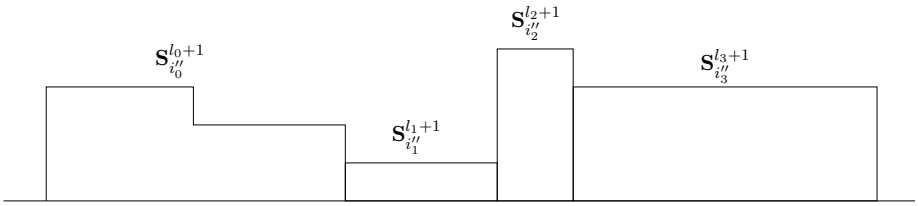
[1] We consider again the case of an one-dimensional (and continuous) time signal, for the sake of clear graphical representation. The conversion for the discrete case would be performed using a simple uniform sampling procedure at $t = n$, $n \in \mathbb{N}$.



(a) Original block



(b) Block contraction



(c) Block expansion

Figure 4.1: MMP code-vectors and scale transforms.

Each segment, $\mathbf{S}_{i_k}^{l_k}$, may be one of the vectors that were used to build the initial dictionary, or a more elaborate pattern that was created by the dictionary update procedure. In the first case, $\mathbf{S}_{i_k}^{l_k}$ is a pulse (or a 2D pixel block) of constant amplitude, α_{i_k}, and width 2^{l_k} (or dimensions $2^{\lfloor \frac{l_k+1}{2} \rfloor} \times 2^{\lfloor \frac{l_k}{2} \rfloor}$ for the 2D case).

MMP may thus be regarded as a signal decomposition using a frame [63], that corresponds to the redundant functions $\mathbf{S}_i^l$ (of all available scales) that are stored in the dictionary at a given time. The MMP coding algorithm chooses a set of atoms and combines them, in order to approximate the input image, according to equation (4.6). One interesting feature about MMP is that the used frame is both adaptive and dependent on the input image. It is adaptive because new atoms are created by the dictionary updating procedure. On the other hand, for each input image (and coding parameters), the algorithm uses a different set of code-vectors.

Since the dictionary update procedure builds new patterns using the concatenation of scaled versions of code-vectors, each MMP atom function can ultimately be represented

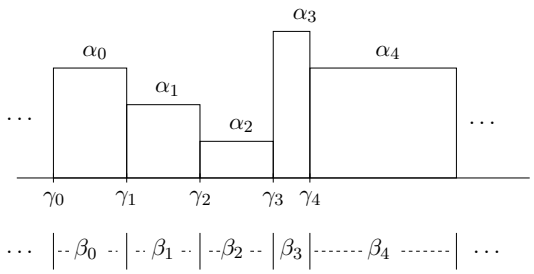


Figure 4.2: Decomposition of a signal into a set of pulses with amplitude α_k, scale β_k and position γ_k.

by a combination of basic pulses, $\phi(t)$, defined by:

$$\phi(t) = \begin{cases} 1 & 0 \leq t < 1 \\ 0 & otherwise \end{cases} . \tag{4.7}$$

Each code-vector used in the initial dictionary corresponds to a single pulse. As the updating procedure progresses, new blocks are created by the concatenation of a set of pulses characterised by their amplitude, width and position, *i.e.*:

$$s_i^l(t) = \sum_{k=0}^{K'-1} \alpha_k \phi\left(\frac{t-\gamma_k}{\beta_k}\right). \tag{4.8}$$

The parameter α_k corresponds to the amplitude of each pulse; β_k is the support of each rectangular pulse, $\beta_k \in \{2^{l_k}, l_k \in [0, l_{Max}]\}$; γ_k represents each pulse's position, in our case, a positive integer.

The code-vectors defined by equations 4.6 and 4.8 correspond to the atoms of the MMP frame. Since the decoded signal, $\hat{x}(t)$, is built from concatenated atoms, it can also be represented in terms of a combination of these modulated, scaled and shifted pulses:

$$\hat{x}(t) = \sum_{k=0}^{K-1} \alpha_k \phi\left(\frac{t-\gamma_k}{\beta_k}\right), \tag{4.9}$$

A graphical representation of this pulse combination is given in Figure 4.2.

From Figure 4.2 we observe that each β_k, relating to the MMP scale of each pulse, may be represented as a function of the positions of two adjacent pulses:

$$\beta_k = \gamma_{k+1} - \gamma_k. \tag{4.10}$$

The reconstructed signal may therefore be represented by an ordered set of pairs

$$\xi_k = (\alpha_k, \gamma_k), \tag{4.11}$$

that characterise each atom function that was used to compose $\hat{x}(t)$. Therefore, every encoded block, dictionary vector and ultimately the whole input signal, can be represented by

$$\hat{x} = \{\xi_0, \quad \xi_1, \quad \ldots, \quad \xi_{K-1}\}, \tag{4.12}$$

that correspond to

$$\hat{x}(t) = \sum_{k=0}^{K-1} \alpha_k \phi\left(\frac{t-\gamma_k}{\gamma_{k+1}-\gamma_k}\right). \tag{4.13}$$

From the previous discussion we can establish an analogy between the MMP encoding procedure and a signal "analysis" algorithm, that determines the best representation of the input signal $x(t)$ according to the signal decomposition described by equation (4.13). This analysis determines the best atoms (code-vectors) for the signal decomposition.

Furthermore, the code-vectors that result from the dictionary adaptation process correspond themselves to a set of simpler atoms, that were concatenated and scaled. Each MMP dictionary index can therefore be regarded as an efficient joint encoding of a set of pairs that describe the dictionary pattern (see equation (4.12)), and become available for future approximations. Another interesting observation is that, while the dictionary is composed by just rectangular pulses, the MMP compression scheme implicitly encodes the β_k and the γ_k parameters with the segmentation tree, while the corresponding α_k values are encoded with the dictionary index. Nevertheless, as more complex code-vectors are created, all parameters of each ξ_k are jointly encoded by MMP's dictionary indexes and segmentation flags.

In the previous analysis of MMP, one notices two important features of the MMP signal decomposition process: first, the atoms used by MMP to approximate the image are orthogonal, since they have non-intersecting supports; and second, the chosen code-vectors can be expressed by a concatenation of other atoms, represented at a finer scale. This reveals an interesting relation between MMP and a multiresolution analysis of the input signal, that can be expressed as follows. Be V_l the set of functions that can be generated by MMP at a given scale, l. Then one has that:

$$V_l \subset V_{l-1} \subset \cdots \subset V_0. \tag{4.14}$$

For scale 0 (blocks with 1×1 pixels), MMP is able to achieve lossless compression of the input image if all possible pixel values are included in $\mathcal{D}^0$. In practice, this is not the case,

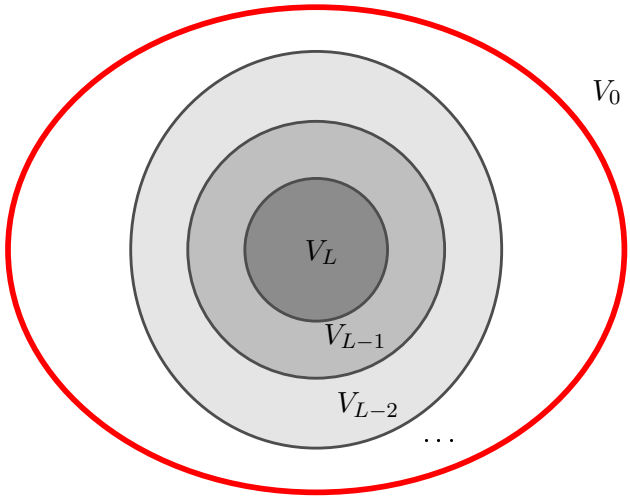


Figure 4.3: Function spaces generated by MMP at each approximation scale, where $V_L \subset V_{L-1} \subset V_{L-2} \subset \cdots \subset V_0$.

due to the quantisation used in the definition of the initial blocks' amplitudes. This may be regarded as a quantisation of the coefficients c_m used in the signal decomposition, described in equation (4.5). Nevertheless, any atom function from $V_1, \ldots, V_L$ can be represented as a composition of functions from V_0. Since the MMP encoder is able to simultaneously use all subspaces V_n, MMP compression may be regarded as the optimal choice of some orthogonal signals, from a larger set of redundant atoms, that span the generated approximation of the input image. A representation of this process may be found in Figure 4.3.

In spite of the this multiresolution feature, the previous interpretation of MMP has important differences when compared with the theory of multiresolution analysis [63]. In this case, a function in V_{j+1} can be decomposed into a function in V_j (generated by a linear combination of scaling functions $\phi_j(t)$, at scale j), plus a detail signal, composed by a linear combination of translates of a corresponding *wavelet* function, $\psi(t)$, at the same scale:

$$f_{(j+1)}(t) = \sum_k c_{j,k}\phi_{j,k}(t) + \sum_k d_{j,k}\psi_{j,k}(t). \tag{4.15}$$

If W_j is the set of functions generated by $\psi_j(t)$, then we obtain:

$$V_{j+1} = V_j \oplus W_j. \tag{4.16}$$

From equations (4.15) and (4.16) one has that $V_j = V_k \oplus W_k \oplus W_{k+1} \oplus \cdots \oplus W_{j-1}$, for $k < j$. Thus, any function in V_{j+1} can be decomposed into a sum of functions that start with a low

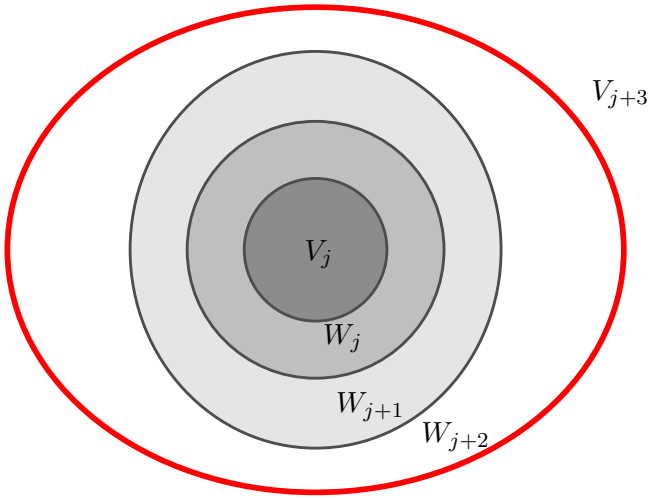


Figure 4.4: Relation between the function spaces generated by the several scales of a multiresolution decomposition.

resolution scaling function representation, combined with a set of wavelet representations of the residue details. This may be represented by the function spaces of Figure 4.4. Note that the residues (detail spaces) are not used by MMP.

One important feature of MMP as a signal decomposition scheme is its adaptability. In general it is not feasible to construct a transform for a specific signal, since nonstationary signals often demand computations of the used transform and the overhead required to transmit the transform update may compromise compression gains. To a certain degree, MMP encoding solves this problem by means of its highly adaptive multiscale representation. By using several redundant approximation subspaces, MMP optimises signal decomposition by using larger blocks in the more regular regions (lower rate) and smaller blocks in detailed areas (lower distortion). This process, implemented by the RD optimisation scheme, assures that, for each image segment, it is possible to choose the best representation for the original pattern in an RD sense, given an initial dictionary and its updating rules.

The use of orthogonal, non-overlapping functions has an important role in MMP coding. It permits an independent optimisation of the approximation of each block, *i.e.* the approximation of a particular block will not be influenced by the functions used to approximate its neighbours (either past or future). This allows for a computationally efficient optimisation of the MMP approximation. If this was not the case, the Lagrangian cost

of the approximation of a given block would have to consider not only the distortion that results from the use of a given dictionary pattern for the current block, but also the effects of the approximation of all neighbouring blocks. Nevertheless, it is important to keep in mind that maintaining this useful feature of the method restricts the atom functions $\phi(n)$ that may be used in the MMP analysis process.

Another feature of MMP is that the atoms used for the analysis and synthesis (see equations (4.7) and (4.13)) are the same. The MMP decoder uses the same shifted pulses to reconstruct the signal, that were used at the encoder to optimise the signal decomposition. The use of rectangular pulses has the disadvantage of originating some blocking artifacts on the decoded MMP image, which arise from the independent optimisation of each image block. This may cause the appearance of some discontinuities in the neighbouring pixels of the blocks. One should notice that the restriction of the atom functions to be non-overlapping does not apply to the functions used by the decoder. In fact, the decoder simply has to combine the vectors defined in the bit stream, in order to compose the approximated image. In [1, 2, 65] some tests were performed using the usual encoder but with a decoder that performs a post-filtering operation on the reconstructed blocks. This procedure, discussed in the following section, may be related with replacing the rectangular synthesis pulses by different signals, namely triangular and Gaussian functions, with overlapping supports.

4.3.3 MMP synthesis with deblocking filtering

In [1, 2, 65], a method related with MMP synthesis using overlapping functions was implemented as a way to control the blocking artifacts in the reconstructed image. For this purpose, a running two-dimensional FIR filter is applied to the reconstructed image. The filter's kernel dimensions are successively adapted to the scale of the code-vector that was used to approximate the corresponding image area. MMP tends to use, in smooth image areas, blocks $\mathbf{S}_i^l$ of large scales, which have larger support regions. In image areas with higher activity it tends to use blocks with small values of l. This can be exploited by adapting the support of the smoothing FIR filter according to the dimensions of each image segment $\mathbf{X}_i^l$ that is being considered.

Figure 4.5 depicts an one-dimensional representation of a reconstructed portion of a signal, that was approximated by the concatenation of three blocks, $\begin{pmatrix} \mathbf{x}_{i_0}^{l_0} & \mathbf{x}_{i_1}^{l_1} & \mathbf{x}_{i_2}^{l_2} \end{pmatrix}$, with

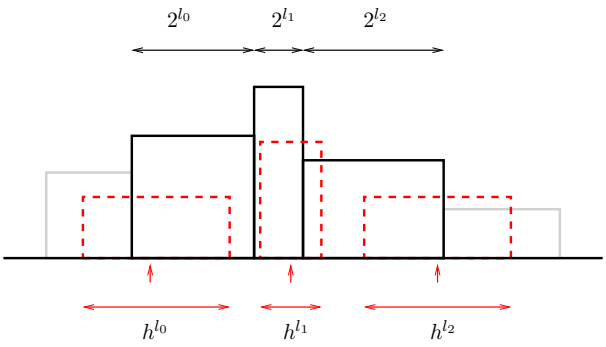


Figure 4.5: Use of FIR filters with adaptive support according to the MMP block scale.

different scales: l_0, l_1 and l_2. At each filtered pixel, represented in the figure by a red arrow, the kernel support of the deblocking filter has been set according to the scale l_k of the block $\mathbf{x}_{i_k}^{l_k}$ that it belongs to.

In [1], two models were used for the FIR filter h: a running average (rectangular) filter and a Gaussian filter. For both cases, a support of $2^{l_k}+1$ samples was used. The two-dimensional filtering is separable. It is performed first in one direction and then on the other [64]. The rectangular filter has a highly smoothing effect, which could compromise the quality of the reconstructed image, but the support adaptation process controls its deblocking strength according to the detail level of the region that is being processed. This reduces the blurring artifacts that are usually caused by the use of exaggerated smoothing. Nevertheless, the original filtering process still results in a reduction in objective quality for smooth images. On the other hand, the use of a Gaussian kernel with support $2^{l_k}+1$ samples, resulted in PSNR gains for smooth images [1, 2]. Figure 4.6 shows the results for the MMP post-processing deblocking filter for a detail of Lena image coded at 0.29 bpp[2]. It is possible to see that both filtering kernels are able to reduce the blocking artifacts and increase the perceptual quality of the decoded image. It can also be noticed that the rectangular kernel has a higher smoothing effect than the Gaussian, introducing some blurring artifacts in the decoded image. In terms of the RD results one has that the use of the rectangular kernel consistently decreases the objective quality of the reconstructed image. Note, however, that its use increases the subjective quality, when compared with the original decoded image. For the shown compression ratio, this technique causes a

[2]Refer to Figures C.8 and C.9 of Appendix C for a representation of the entire images.

(a) Original (32.55 dB) (b) Rectangular (32.05 dB) (c) Gaussian (32.91 dB)

Figure 4.6: Detail of the MMP post-processing deblocking filters for image Lena coded at 0.29 bpp: a) MMP decoded image b) deblocking with rectangular kernel and c) deblocking with a Gaussian kernel.

PSNR loss of about 0.5 dB. On the other hand, the Gaussian deblocking kernel, besides increasing the perceptual quality, also achieves gains of more than 0.3 dB in PSNR for this compression ratio. Generally, the use of a Gaussian kernel achieves PSNR gains across all compression ratios, while the use of a rectangular kernel decreases the PSNR for all rates[3].

The adaptive post-filtering of the original pulses may be regarded as a modification of the shape of the vectors used in MMP reconstruction. The use of rectangular filters can be roughly regarded as the replacement of the original MMP synthesis functions by approximations of triangular pulses, that correspond to the use of two rectangular blocks, for the MMP reconstruction and the filtering kernel[4]. This shows that the use of overlapping functions in MMP synthesis is possible and even advantageous in some cases. Nevertheless, the previous process is not as much of a signal reconstruction method as it is a post-processing one. The careful control of the filtering kernel allows the decoder to reduce the destructive effects caused by the tails of the overlapping atom functions. For smooth images this results in an efficient deblocking scheme, but this is not the case for highly detailed images, like text and compound images. Due to the smoothing effect introduced by the low-pass filtering, the deblocking process disrupts the highly detailed

[3]Figure C.7 of Appendix C presents a plot of the RD performance for image Lena.

[4]Note that, in spite of its similitude, the used process is not a normal convolution: only a part of the convolution is performed and the impulse response of the filter changes according to the location and the MMP scale of blocks.

text regions, resulting in a severe decrease in the final values of PSNR[5]. This fact limits the applicability of the deblocking schemes of [1, 2, 65], because there is no practical way of avoiding the blurring of images that do not need deblocking. In Section 7.2 we further develop this deblocking scheme, in order to solve these problems and improve its efficiency also for low pass images.

4.4 A non-uniform sampling point of view

In the previous section we have related MMP with a signal decomposition method that uses the Haar scaling function as a building block for the signal reconstruction. Each block from the reconstructed signal may be regarded as the concatenation of several rectangular pulses, corresponding to the synthesis functions (see Figure 4.2). This means that the decoded signal may be represented by a set of non-uniformly spaced samples, with amplitude α_k, located at the points γ_k, that are independently convolved with a rectangular pulse with support β_k. This is the equivalent to the use of a traditional *zero order hold* interpolation filter, with time varying supports due to the use of non-uniformly spaced samples.

This description of the MMP synthesis relates MMP encoding with a non-uniform sampling process. The signal samples are represented by the pairs $\xi_k = (\alpha_k, \gamma_k)$, since the value of β_k is implicitly stored in the MMP segmentation tree. MMP decoding can thus be interpreted as the interpolation of a set of samples, determined by the encoder, using a space variant interpolation filter, consisting of a rectangular pulse with adaptive support. From this point of view one may also interpret the tests described in the previous sections as an interpolation process, that uses interpolation functions with overlapping supports.

In this section we use a modification of the MMP decoder in order to explicitly recover the encoded signal from the sample points defined by the MMP encoder. Using this sparse array on non-uniformly spaced samples, we evaluate the use of interpolation techniques for MMP synthesis. Once more we started by studying a version of MMP that does not use any filtering in the scale transforms. This is done in order to maintain the direct correspondence between the rectangular pulses that exist in the reconstructed image and the basic dictionary atoms. A discussion on the effects of using the low pass scale transforms will be presented later in this section.

[5] The results of the deblocking process for these images are presented in Figures C.10 and C.11 (PP1205) and C.12 and C.13 (PP1209).

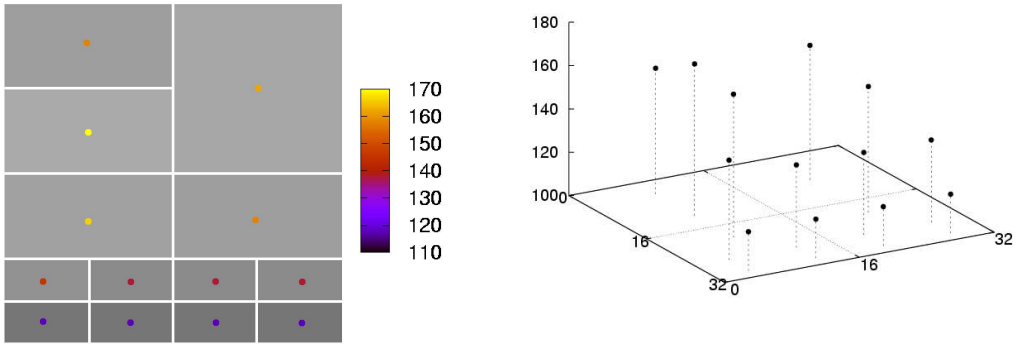
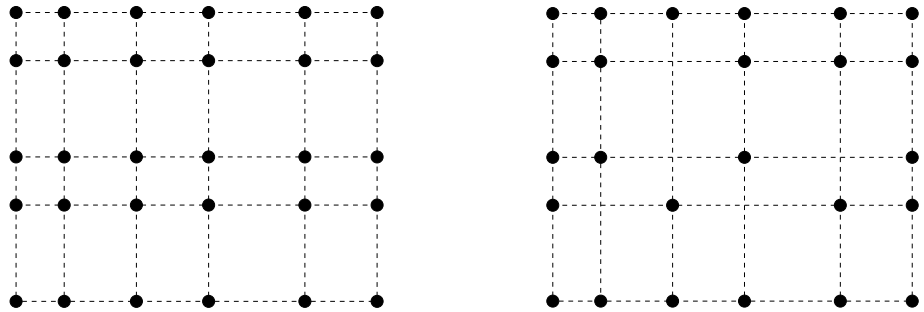


Figure 4.7: Detail of the sampling points defined by MMP for a 32×32 pixel area of image Lena.

Figure 4.7 shows two representations of the set of 2D sampling points determined by the MMP encoding process for a 2D example, in this case a region of 32×32 pixes of image Lena. These samples are extracted by identifying the rectangular pulses of the reconstructed image. A discrete impulse is placed in the centre of the region that corresponds to each rectangular area, using the corresponding amplitude. From these representations we may see the different amplitudes and the non-uniform distribution of the sampling points that will be passed to the interpolation process.

Interpolating with *thin-plate splines*

As a way to demonstrate the previous idea we implemented a simple test were we tried to reconstruct the decoded image using the MMP sampling points and an interpolation procedure. Polynomial interpolation methods (*e.g.* Lagrange interpolation) were considered as interpolation techniques for this study. Nevertheless, these methods suffer from severe oscillation artifacts (Runge's phenomenon), introduced by the high order polynomials used in the interpolation of a large number of points (n sample points use interpolating polynomials of order $n + 1$). Because of this, polynomial interpolation is generally not suited to be used with a large number of sampling points. One common alternative has been the use of *piecewise polynomials* as interpolating functions. Among these methods, *spline* interpolation [66] has gained great relevance. Splines are piecewise polynomial functions with segments that are smoothly connected together. A spline with degree n connects each point (or knot) by using a polynomial of degree n, subjected to continuity constraints



(a) (b)

Figure 4.8: Examples of two non-uniform sampling grids: a) allows for a separable processing of the samples; b) only allows for non-separable processing of the samples.

of the spline and its derivatives of order below n. The main advantage of this method is that, because interpolation is optimised for each interval, the error is kept small, even for low degree spline polynomials. This results in a reduction of Runge's phenomenon compared with higher degree polynomials. One interesting result characterises splines as an expansion of *B-spline* functions of a given order. A B-spline of order 0 corresponds to the familiar Haar scaling function:

$$\beta^0(x) = \begin{cases} 1, & |x| < \frac{1}{2} \\ 0, & otherwise \end{cases} . \tag{4.17}$$

A B-spline of order n is constructed by using a $(n+1)$-fold convolution of the rectangular β^0 pulse. This means that the original MMP synthesis process may also be related with a signal reconstruction using zero order B-splines.

Traditional spline interpolation of two-dimensional signals uses B-spline functions in a separable way, meaning we first interpolate the sample rows (columns) and then the columns (rows) of the intermediate signal. Our problem is one of non-separable, non-uniform sampling. This means that the sample points are not distributed according to a "complete" pattern, in which there is a constant number of points per row (column), uniformly aligned across columns (rows). Figure 4.8 shows two examples of non-uniformly spaced sample points: in the first case (a), a distribution that allows for a separable processing is shown; Figure 4.8 b) shows a non-separable distribution. The non-separable distribution implies the use of a two-dimensional interpolation process that is able to process samples in arbitrary positions. This problem is solved by using the so called *thin-*

plate splines (TPS) [67].

The name thin-plate comes from the physical equivalent to the mathematical process, that tries to simulate how a thin metal plate would behave if it was forced to pass through the control points, that correspond to the signal samples. In some cases, for instance when the control point coordinates are noisy, the interpolation requirements are relaxed so that the resulting surface does not have to go exactly through the control points. In this case we have a *smoothing* TPS. A smoothing TPS determines the interpolating function f that minimises the weighted sum

$$E(f) + \lambda R(f), \tag{4.18}$$

where $E(f)$ is the squared error between the interpolated signal f and the samples at their original points:

$$E(f) = \sum_{i=1}^{k} ||y_i - f(x_i)||, \tag{4.19}$$

and $R(f)$ is a roughness measure, based on the space integral of the square of the second order derivatives of the mapping function, given by:

$$R(f) = \iint \left(f_{xx}^2 + 2f_{xy}^2 + f_{yy}^2\right) dxdy. \tag{4.20}$$

By varying the value of parameter λ it is possible to chose between a higher interpolation accuracy and a smoother interpolation function.

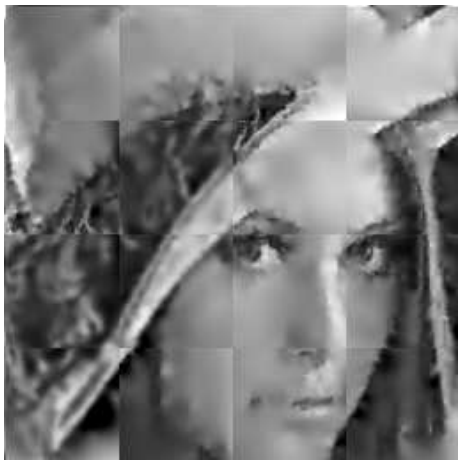
Unlike B-splines, the matrices used for least square minimisation in TPS problems are non-sparse. Moreover, the TPS matrices grow very fast with the number of sampling points, which limits the number of points that may be processed in one single iteration of the method. This means that the used processing with TPS[6] is not able to handle the sampling of an entire image. Because of this we processed the image blocks independently and concatenated the result of the interpolation. A trivial solution for this problem originates severe blocking artifacts, since the optimisation of each region is made independently, with no control of the function value at the borders of the blocks (see Figures 4.9 a) and b) for a representation of these effects for a detail of image Lena, encoded at 0.12 bpp[7]).

In order to solve this issue we performed the TPS optimisation of each block considering both the sampling points of the current block and its neighbouring blocks. After the optimisation, the area of the interpolated signal that corresponds to the original block is

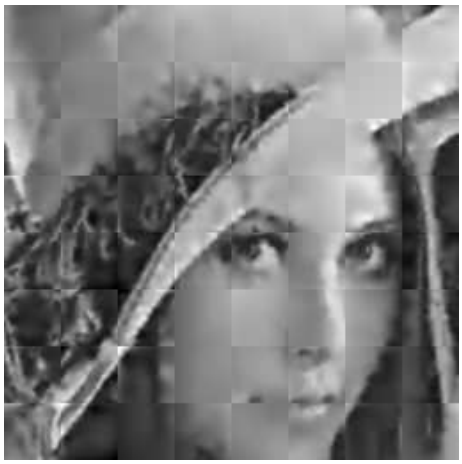
[6]The *spline toolbox* of Matlab© [68] was used for this purpose.

[7]Refer to Figure C.14 of Appendix C for a representation of the entire images.

(a) Original MMP block



(b) Independent 64×64



(c) Independent 32×32

(d) Overlapping 64×64

(e) Overlapping 32×32

Figure 4.9: Results of MMP reconstruction (a) compared with thin-plate spline interpolation, using *independent* optimisation of image blocks (64×64 (b) and 32×32 (c)) and optimisation of *overlapping* image blocks (64×64 (d) and 32×32 (e)).

MMP	p=0.1	p=0.15	p=0.25	p=0.35	p=0.5	p=1
28.787 dB	27.874 dB	27.960 dB	27.974 dB	27.928 dB	27.840 dB	27.564 dB

Table 4.1: PSNR values for image Lena, for the original MMP image and for the interpolated versions with TPS, using different values of parameter p.

isolated and concatenated in order to reconstruct the decoded image. The points of the neighbouring blocks serve as border conditions to the optimisation, allowing a considerable reduction of the blocking artifacts in the interpolated image, as can be observed in Figures 4.9 c) and d)[8]. In these figures we can also observe that the interpolation of large image regions (64×64 and 32×32) results in a noticeable smoothing effect in the reconstructed signal. Because of this we have used 16×16 pixel blocks in the interpolation step (this is the minimum block size that guarantees a minimum of 4 control points required by the Matlab©'s implementation of TPS), which produced noticeable improvements in the interpolation result. In our case the smoothing TPS is determined by minimising the function

$$pE(f) + (1-p)R(f), \tag{4.21}$$

where parameter $p \in [0,1]$ replaces parameter λ of equation (4.18), as a compromise measure between interpolation accuracy and smoothness. Several values were tested in order to determine the best working point for the interpolation. Figure 4.10 compares the result of the interpolation process with the original MMP decoded image, for a detail of image Lena encoded at 0.12 bpp[9]. Parameter p was set to 0.25, because this value returned the highest value of PSNR for the interpolated signal. Table 4.1 compares the values of PSNR for the original MMP image and for the interpolated versions, using the represented values of p.

In Figure 4.10 we can observe the blocking effect introduced by MMP for a high compression ratios. Regarding the interpolated signal one may observe a clear reduction of the blocking artifacts, but also some smoothing and ringing artifacts introduced by the TPS interpolation. This was also the case for lower compression ratios, where a drop in PSNR performance was also observed. Figure 4.11 presents the results of the original MMP syn-

[8]Refer to Figure C.15 of Appendix C for a representation of the entire images.
[9]Refer to Figure C.16 of Appendix C for a representation of the entire images.

(a) MMP (28.787 dB) (b) TPS (27.974 dB)

Figure 4.10: Image Lena compressed at 0.12 bpp: a) original MMP synthesis; b) Thin-plate spline interpolation using 16×16 overlapping blocks.

(a) MMP (33.855 dB) (b) TPS (31.496 dB)

Figure 4.11: Detail of image Lena compressed at 0.45 bpp: a) original MMP synthesis; b) *Thin-plate spline* interpolation using 16×16 overlapping blocks.

thesis and of TPS interpolation for a detail of image Lena encoded at 0.45 bpp[10]. Once more we may observe the smoothing effect of the interpolation and its negative effect on the perceptual quality of the reconstructed image. In this case, the TPS interpolation dropped the original PSNR value of 33.855 dB, achieved by the MMP decoding process, to 31.496 dB.

[10]Refer to Figure C.17 of Appendix C for a representation of the entire images.

Compression ratio	MMP synthesis	TPS interpolation (p=0.25)
0.11 bpp	29.057 dB	27.957 dB
0.4 bpp	34.077 dB	31.383 dB

Table 4.2: Results for the TPS interpolation in MMP with low-pass filtering in the scale transforms, for image Lena.

When the original scale transforms are used, incorporating a smoothing filtering, the synthesis atoms for the reconstructed image no longer have a direct correspondence with rectangular functions. These rectangular pulses, that exist in the initial dictionary, are changed into smooth functions by the successive scale transformations used in the dictionary updating process. This makes it difficult to determine the sampling point that corresponds to each block used by MMP, since a direct relation no longer exists between the patterns used in the reconstruction and the basic synthesis functions. Nevertheless, even in this case one may consider that, ultimately, any discrete signal may be characterised by the concatenation of a number of rectangular pulses, that may have unitary supports (*i.e.* they become discrete impulses). In this case we may maintain the previous discussion and relate the constant regions in the image with the basic blocks ξ_k used in the synthesis, instead of using the information about the scale of the used blocks. Thus, a greater difficulty in the determination of the interpolating function may be expected, since the sampling points would be in a larger number and located very close to each other. Experimental tests confirmed this. The results for MMP using the original scale transforms, with low-pass filtering, as well as the results for the TPS interpolation are summarised in table 4.2, for two tests performed using image Lena. This table, as well as the visual inspection of the resulting images (not shown here), show results that are very similar to those presented for MMP with no low pass filtering.

Some comments on MMP as a non-uniform sampling process

The previous discussion and experimental results relate MMP with a non-uniform sampling-based image compression scheme. From this point of view, a *thin-plate spine* interpolation algorithm was implemented as a way to reconstruct the signal from the samples that were determined based on MMP encoding. It is clear from the previous sections that the interpo-

lation process is not advantageous over the original MMP synthesis. This may be explained by the fact that the encoding procedure does not optimise the signal representation for the TPS interpolation process. Such an algorithm would be computationally infeasible, since it would imply a joint optimisation of all the sampling points. On the other hand, the TPS interpolation process itself has some limitations: the least square optimisation of the interpolation generally results in severe smoothing effects for the reconstructed signal. When we try to compensate for these effects by privileging interpolation accuracy over function smoothness, ringing artifacts appear. Both options result in a noticeable loss of objective quality. TPS interpolation may also be regarded as a simple post processing deblocking algorithm for MMP. In this case, for large compression ratios, an effective reduction of the blocking artifacts is in fact achieved. However, the efficiency of TPS interpolation as a deblocking algorithm is inferior to that of the method discussed in Section 4.3.3.

The presented results validate the formal relation between MMP and a non-uniform sampling scheme. From this point of view, other non-uniform two-dimensional interpolation schemes could be investigated, in order to improve performance of MMP signal reconstruction; this may be an interesting future research subject.

The next chapters describe some alternative optimisation schemes that relate also with the encoding process, instead of tackling only the problem of signal reconstruction.

Chapter 5

MMP with predictive coding: the MMP-I Algorithm

Following the discussion presented in Chapter 3, we investigate new ways to improve the efficiency of the dictionary adaptation process used by MMP. This chapter presents the joint use of MMP with predictive schemes, as a way to modify the probability distribution of the source, adapting it to a new model that can be better exploited by MMP dictionary adaptation. In Section 5.1 we discuss the efficiency of MMP on sources with different probability distributions and demonstrate the advantage of using input signals with narrow probability distributions, that we represent by generalised Gaussian functions. This study motivated the development of an image encoding method, that is proposed in Section 5.2. We refer to the proposed algorithm as MMP-I (MMP with Intra predictive coding), as it uses a pool of predictive methods in order to generate a residue signal that is then encoded with MMP. The experimental results, presented in Section 5.3, demonstrate the efficiency of the described techniques. An evaluation of the predicted residues, performed in Section 5.4, confirms the expected effects of the prediction step in adapting the probability distribution of the encoded signal.

5.1 Improving dictionary adaptation with modified signal distributions

The dictionary updating strategies of MMP exploit the spacial redundancies of the image, by inserting new vectors in the dictionary and using these patterns to approximate blocks of the input signal. The gains achieved by the use of the new code-vectors depend on the existence of image areas that can be efficiently approximated by these patterns. Therefore, the efficiency of the dictionary updating, and ultimately of the entire encoding process, depends on the similarity between image blocks, and thus on the statistical properties of the encoded data. Intuitively, one has that smoother images, with higher spatial redundancies, tend to favour the efficiency of the dictionary adaptation process. Ultimately, a uniform image could be approximated by MMP using one single block of the initial dictionary, maximising the coding efficiency of the process. On the other hand, high activity images (like text and compound images) have blocks that tend to generate a less regular set of patterns, that are harder to be learnt by the MMP updating processes. In fact, even natural smooth images have a large variety of patterns that have to be approximated by the MMP encoder.

The previous discussion, as well as the results presented in Chapter 3, suggest that the use of a more "well behaved" signal, *i.e.* a signal composed by a more regular set of patterns, could improve the efficiency of the MMP dictionary adaptation process. The use of predictive coding techniques has the well know property of altering the signal's distribution, generating a set of residues with a few higher valued samples located near image edges, and reducing the variance of the prediction error when compared with that of the input source. In fact, predictive coding generates residue samples that have highly peaked probability distributions, centred around zero. The concentration of these distributions may vary according to the input image and the used prediction techniques, so they are best represented by generalised Gaussian (GG) distributions (also called exponential power distributions) [69]. Initial work on MMP [1, 2] includes a theoretical study that demonstrates the advantage of using vector matching with scales over standard VQ methods, for Gaussian sources. This result suggests that when signals with narrower distributions are employed, MMP is expected to top the performance of VQ methods, that are also known to benefit from the use of prediction [5]. Nevertheless, this study is not conclusive about

the relative efficiency of MMP when used to compress signals with different probability distributions.

In this section we first characterise the predicted error signal probability distribution. We then compare the performance of MMP for signals with highly peaked distributions to the performance achieved for signals with wider, more uniform distributions. The goal of this analysis is to evaluate possible compression gains of using MMP to encode the predicted residue, instead of the input image data.

5.1.1 The generalised Gaussian model

Image predicted residues are usually modelled as having Laplacian distributions, but a more general representation of this signal can be achieved by using a generalised Gaussian distribution [69, 70]. This class of distributions is described according to:

$$p(x) = \left[\frac{\alpha\eta(\alpha,\beta)}{2\Gamma(1/\alpha)}\right] e^{-(\eta(\alpha,\beta)|x|)^{\alpha}}, \tag{5.1}$$

where

$$\eta(\alpha,\beta) = \beta^{-1}\left[\frac{\Gamma(3/\alpha)}{\Gamma(1/\alpha)}\right]^{1/2} \tag{5.2}$$

and $\Gamma(.)$ is the gamma function. The generalised Gaussian distribution has two positive parameters: α is a shape parameter, that describes the exponential rate of decay and β is the standard deviation of the distribution, meaning that the variance of the random variable x is given by $\sigma^2{=}\beta^2$. For $\alpha = 2$ we have the Gaussian distribution

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}, \tag{5.3}$$

while for $\alpha = 1$ we have the Laplacian distribution

$$p(x) = \frac{1}{\sqrt{2}\sigma} e^{-\sqrt{2}|\frac{x}{\sigma}|}. \tag{5.4}$$

The uniform distribution can also be represented as a special case of a generalised Gaussian distribution, when $\alpha \to \infty$.

The behaviour of the generalised Gaussian functions of unit variance is illustrated in Figure 5.1, for several values of the shape parameter α. The generalised Gaussian distribution has been used for modelling the probability density function of transform and

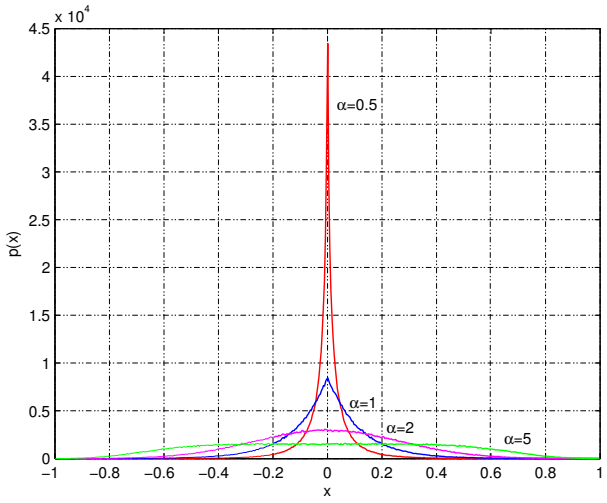


Figure 5.1: Generalised Gaussian distributions for selected values of the shape parameter α.

subband coefficients in image coding [71, 72], as well as for image's prediction error. The shape adaptive features of the generalised Gaussian also make this a versatile model for sources with unstable probability distributions [73].

5.1.2 Coding of generalised Gaussian sources with MMP

Several random grayscale images were generated using different probability distributions. A generalised Gaussian model with varying shape parameter was used to generate the test images. The MMP performance for these signals was assessed.

In order to make a fair evaluation of the relative performance for the different signals, we generated six sets of four grayscale test images, using GG probability distributions. Four images of each type were used in order validate the results. Six different probability models were used in order to generate different types of test signals, according to the parameters defined in Table 5.1. Every image using a generalised Gaussian distribution, with dimensions of 256×256 pixels, was generated using the following procedure:

1. generate a set of 66198 samples using the chosen probability distribution;

2. discard a values that are further from the signal average (in our case 1% of the samples) in order to eliminate samples with very high values (and small probability), that would compromise the normalisation;

Test Set	Distribution	Q_P value	Entropy range
1	Gen. Gaussian, $\alpha = 0.5$	1	[6.28 : 6.35]
2	Gen. Gaussian, $\alpha = 0.6$	1.2	[6.31 : 6.35]
3	Gen. Gaussian, $\alpha = 0.75$	1.45	[6.31 : 6.35]
4	Laplacian (Gen. Gaussian $\alpha = 1$)	1.75	[6.33 : 6.35]
5	Normal (Gen. Gaussian, $\alpha = 2$)	2.45	[6.31 : 6.32]
6	Uniform	3.2	[6.32 : 6.33]

Table 5.1: Parameter values for the probability distributions of the test signals.

3. normalise the resulting 256^2 samples to the interval [0:255] by:

$$x(i) = 127.5\frac{x(i)}{m_x} + 127.5,$$

where m_x is the sample with the highest absolute value;

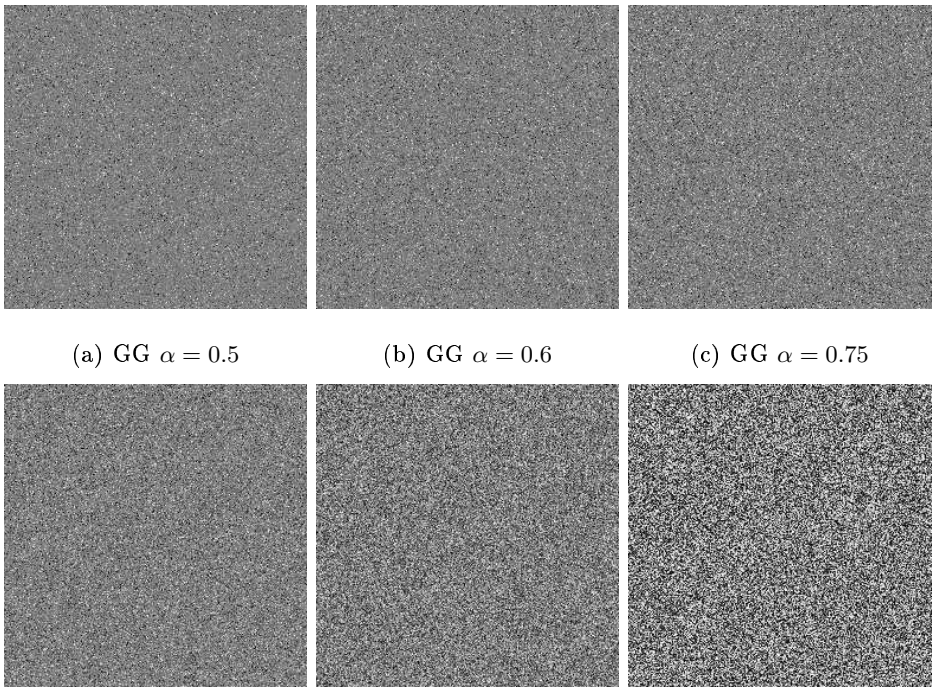
4. quantise $x(i)$ using an appropriate quantisation step Q_P (see Table 5.1), in order to adjust the entropy of the resulting signal:

$$x_q(i) = \left\lfloor \frac{x(i)}{Q_P} \right\rfloor Q_P.$$

The quantisation procedure was used to guarantee that all used test images have the same entropy values. The used value of the quantisation parameter Q_P for each distribution, as well as the minimum and maximum entropy values for each test set, are shown in Table 5.1. Figure 5.2 shows the first image of each probability distribution group. We may observe the different pixel activities for the images with varying probability distributions. Figure 5.3 shows the histograms for these images. From these plots one may observe the expected differences in the probability distributions of the test signals.

Figure 5.4 shows the RD plots (PSNR *vs.* compression ratio) of MMP coding for the described test images. Results are shown for all images of the same group using a single line type. We may observe that the RD efficiency increases as the probability distribution of the input signal becomes narrower, *i.e.* the value of the GG distribution's shape parameter becomes smaller.

Another interesting property of these signals is the fact that for high rate and dimension, the GG vectors will tend to cluster on a "shell" of constant $L^\alpha - norm$ [74]. In spite of



(a) GG $\alpha = 0.5$ (b) GG $\alpha = 0.6$ (c) GG $\alpha = 0.75$

(d) Laplacian (GG $\alpha = 1$) (e) Normal (GG $\alpha = 2$) (f) Uniform

Figure 5.2: Test images with known probability distributions (256×256 pixels).

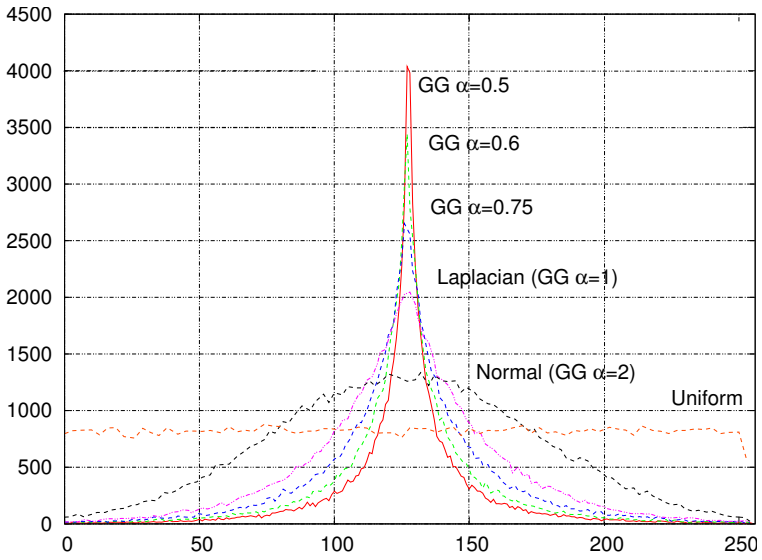


Figure 5.3: Histograms of test images for each probability distribution.

being an asymptotic result, the effects of this property are valid for large vector sizes, depending on the value of the shape parameter of the GG distribution. This means that both the image vectors and the dictionary blocks will be concentrated around a given area

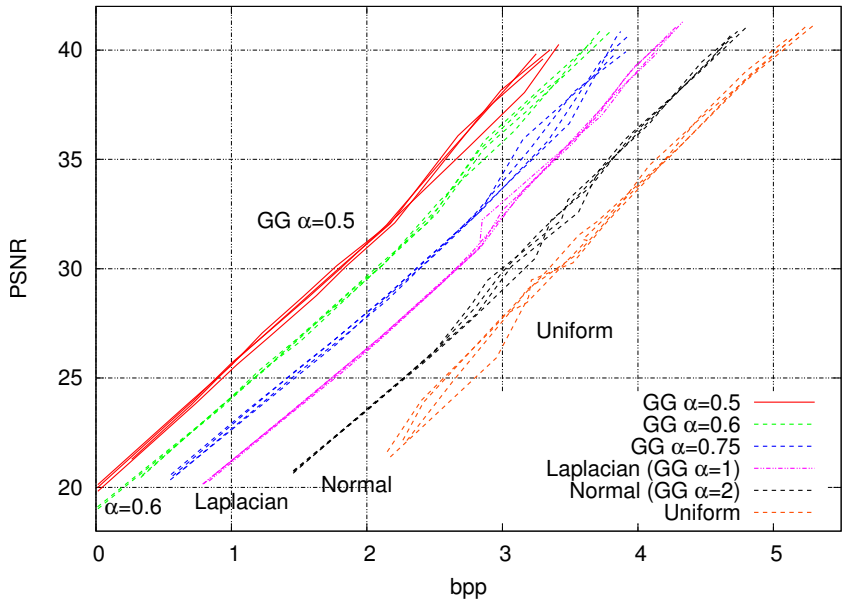


Figure 5.4: RD performance of MMP for the compression of test images with known probability distributions.

of the image space, instead of being widely scattered. This fact increases the efficiency of the dictionary adaptation for signals with narrow GG distributions.

Figure 5.5 analyses the dictionary adaptation efficiency for each test. Figure 5.5 a) shows the final number of elements in the largest scale of the dictionary (16×16) for each image, while 5.5 b) plots the final PSNR value as a function of the final size of $\mathcal{D}^8$. From the first plot we observe that the growth rate of the MMP dictionary is very similar for all test signals, while in Figure 5.5 b) we observe that, for the same number of dictionary elements, MMP is able to achieve a much better approximation quality for the signals with narrower distributions. This demonstrates that the efficiency of the dictionary usage in MMP is greater for input signals with concentrated luminance values. Also, the smaller size of the MMP dictionary relates to a higher concentration of the vectors in the image space, that results from the already mentioned property of the GG sources.

5.2 The MMP-I algorithm

The previous section showed the advantage of adapting the input signal distribution before compressing it with MMP. Following these results, a new image encoding method, that combines the use of predictive coding with multiscale multidimensional parsing was devel-

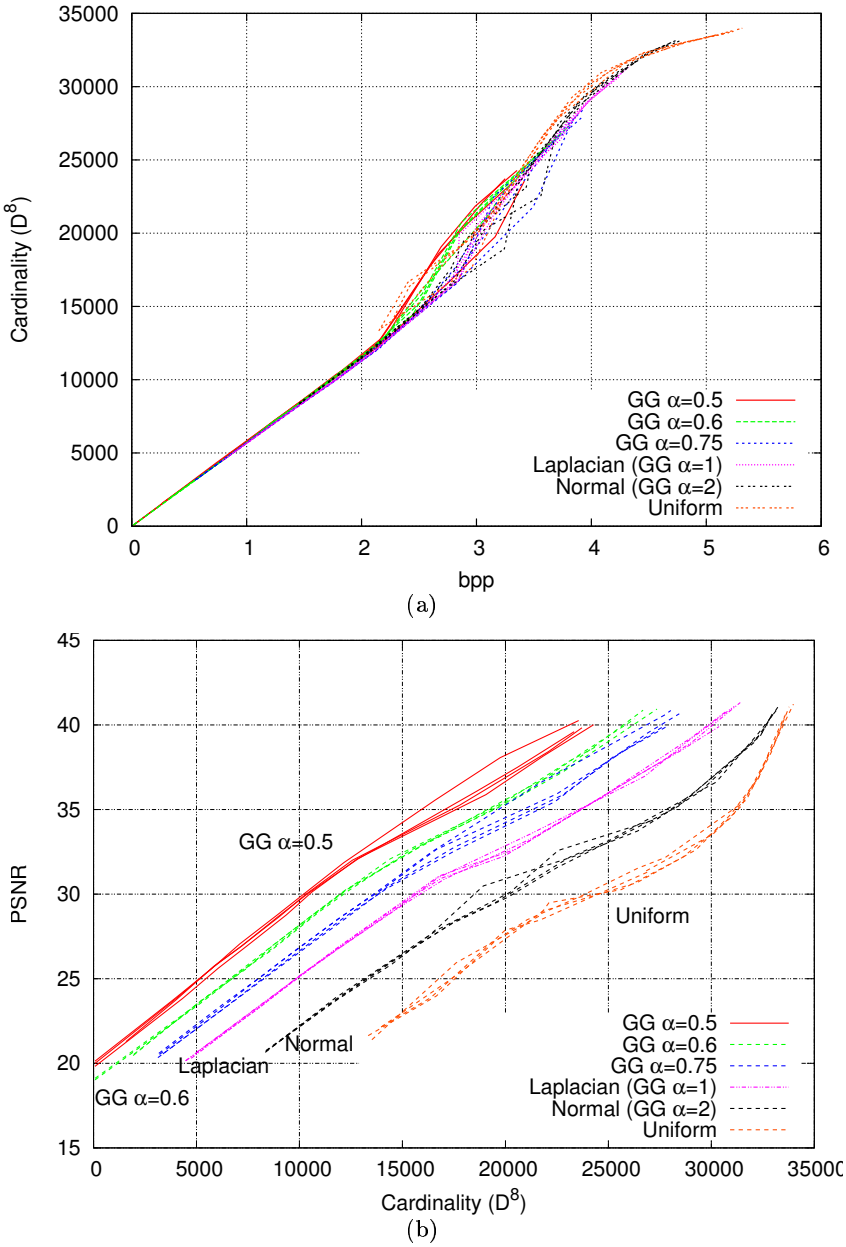


Figure 5.5: MMP coding of test images with known probability distributions: a) final cardinality of $\mathcal{D}^8$ *vs.* compression ratio; b) PSNR *vs.* final cardinality of $\mathcal{D}^8$.

oped. This new algorithm, that we refer to as MMP-I (MMP with Intra-frame predictive coding), uses predictive methods in order to generate a residue signal that is then encoded with MMP. This signal improves the efficiency of the MMP's dictionary adaptation process, resulting in increased performance, evidenced by the experimental results presented in Section 5.3. In Section 5.4 we show an analysis of the transformation of the input sig-

nals's probability distribution, caused by the prediction process. We also show that the new probability distribution is a highly-peaked function, that is conveniently represented by a generalised Gaussian model.

5.2.1 Overview

MMP-I uses the neighbouring samples of previously coded blocks, which are to the left and/or above the current block, to determine the prediction error signal, $\mathbf{R}_M^l$:

$$\mathbf{R}_M^l = \mathbf{X}^l - \mathbf{P}_M^l, \tag{5.5}$$

where M refers to the used intra prediction mode, $\mathbf{X}^l$ is the input image block and $\mathbf{P}_M^l$ is the prediction block associated with mode M.

When encoding an image block, MMP-I chooses among a set of available prediction modes, the one that minimises a rate-distortion Lagrangian cost. The information about the chosen prediction mode for each image block, M, is included in the bitstream. Because the prediction procedure is defined based only on previously encoded image blocks, once the MMP-I decoder retrieves the prediction mode information, it is able to determine the corresponding prediction block, $\mathbf{P}_M^l$, based on the same neighbouring pixel blocks. After this, it decodes the approximation of the residue block, $\hat{\mathbf{R}}_M^l$, and determines the decoded image block by adding the decoded residue block with the prediction pixels, *i.e.*

$$\hat{\mathbf{X}}^l = \mathbf{P}_M^l + \hat{\mathbf{R}}_M^l. \tag{5.6}$$

Because both the encoder and the decoder use the same scanning order for the image blocks and generate the same MMP trees (see Section 3.1), they keep a similar reconstructed version of the neighbour output blocks, allowing for the determination of the prediction block $\mathbf{P}_M^l$.

Predictive coding in MMP-I initially used a simple fixed-size block prediction method, in which the prediction step is always performed for the blocks of the highest level of the segmentation tree. This method is referred to as MMP-I FBS (Fixed Block Size). In this case, for each block of level l_{top} of the image, MMP-I FBS first determines what prediction modes may be used, based on the available prediction pixels. Each available prediction mode is then tested in order to determine the best mode for the current block, M. MMP-I FBS then encodes and transmits the prediction mode flag associated with mode M, followed

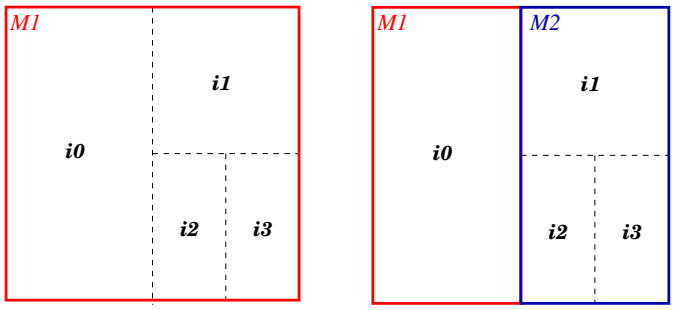


Figure 5.6: Segmentation of the prediction step and MMP coding step for a) MMP-I FBS and b) MMP-I using adaptive block size.

by the MMP information used to encode the residue block (see Chapter 2). As for MMP, experimental tests demonstrated that the use of 16×16 blocks is the best compromise between coding performance and computational complexity: smaller blocks generally lead to performance losses, while larger blocks increase the computational complexity of the process, with no gains in coding efficiency.

Nevertheless, it has been verified that it is advantageous to use more elaborated schemes of predictive coding for MMP-I, in which prediction does not use a fixed block size but is applied adaptively, by using variable-block sizes. In this case the prediction step may be used several times for each block of level l_{top}, meaning that different areas of $\mathbf{X}^{l_{top}}$ may use different prediction modes. Prediction using larger blocks (corresponding to higher levels of the binary segmentation tree) is tested first; then smaller blocks are hierarchically tested, in order to decide which is the most efficient way to perform the block prediction. This creates a new segmentation pattern that is associated only with the prediction process. Thus, the segmentation tree used for prediction can be either identical to the one used when a block is encoded by MMP or can be a pruned version of it.

In the example of Figure 5.6, the prediction of an input block, $\hat{\mathbf{X}}^{l}_{P_M}$, is performed in both halves, $\mathbf{R}^{l-1}_{P_{M_1}}$ and $\mathbf{R}^{l-1}_{P_{M_2}}$, using independent prediction modes M_1 and M_2. In this case, the left residue block is determined using prediction mode M_1 and the corresponding residue block, $\mathbf{R}^{l-1}_{M_1}$, is approximated using a single code-vector, with index i_0. The decoded version of the first half of this block, given by:

$$\hat{\mathbf{X}}^{l-1}_{left} = \mathbf{P}^{l-1}_{M_1} + \hat{\mathbf{R}}^{l-1}_{M_1}, \tag{5.7}$$

is used to determine the prediction block for the right half of the block, $\mathbf{P}_{M_1}^{l-1}$. The right half uses only one prediction step (mode M_2), but the corresponding residue block, $\mathbf{R}_{M_2}^{l-1}$, is encoded by MMP using the concatenation of three dictionary vectors:

$$\hat{\mathbf{R}}_{M_2}^{l-1} = \left(\hat{\mathbf{S}}_{i_1}^{l-2} \ : \ \hat{\mathbf{S}}_{i_2}^{l-3} \ : \ \hat{\mathbf{S}}_{i_3}^{l-3}\right). \tag{5.8}$$

The second half of the original block can be reconstructed by using:

$$\hat{\mathbf{X}}_{right}^{l-1} = \mathbf{P}_{M_2}^{l-1} + \hat{\mathbf{R}}_{M_2}^{l-1}, \tag{5.9}$$

and finally, the approximation of the original block may be determined by the concatenation of these two blocks, *i.e.*:

$$\hat{\mathbf{X}}^{l} = \left(\hat{\mathbf{X}}_{left}^{l-1} \ : \ \hat{\mathbf{X}}_{right}^{l-1}\right). \tag{5.10}$$

The use of adaptive block sizes for prediction purposes means that information about the dimensions of the prediction blocks must also be encoded, along with the prediction mode, so that the decoder is able to replicate the prediction procedure. Instead of independently encoding the binary segmentation trees, that represent the sub-blocks used for prediction and the MMP coding data, this information is jointly encoded, using a single segmentation tree. For this purpose, three segmentation flags are used, instead of the two original flags described in Chapter 2. Considering the example in Figure 5.6, the string of symbols generated by the MMP-I encoder for this block is:

$$0 \quad 1 \quad M_1 \quad i_0 \quad 2 \quad M_2 \quad 1 \quad i_1 \quad 0 \quad 1 \quad i_2 \quad 1 \quad i_3,$$

where M_1 and M_2 are the prediction mode flags and $i_0, ..., i_3$ are the indexes used by MMP-I to approximate each of the sub-blocks. The new segmentation flags are:

- flag '0' indicates a tree node (block segmentation) for which no prediction is necessary. This flag is used when no prediction mode has been set and both the prediction block and MMP block must be segmented (like in the first flag used for the previous example) or when the prediction step has already been performed, but the MMP block should be segmented (as in the case of the second '0' flag of the example);

- flag '1' indicates a tree leaf. If no prediction mode has been previously set for the pixels of this block, this flag is followed by a prediction mode flag, that describes the prediction process which should be used at this scale (used in the first half of

the block represented in the example of Figure 5.6, followed by the prediction mode flag M_1). If prediction data has already been defined for this block, flag '1' signals a simple tree leaf and is followed by the dictionary index of the vector used for the approximation (as in the case of the sub-blocks that use indexes i_2 and i_3 in the example);

- flag '2' also corresponds to a tree node (MMP segmentation), but at a scale where prediction should be performed. It is always followed by the corresponding prediction mode flag (like for the second half of the block represented in the example of Figure 5.6). From this point on no further prediction should be used and all flags represent coding decisions related to the MMP process.

As in MMP, all symbols are encoded using an adaptive arithmetic encoder. The previously presented joint encoding procedure was chosen because it is more efficient than transmitting two independent segmentation trees, one for block prediction and another for MMP data. In the above example, 13 symbols are encoded by using a single segmentation tree. An obvious alternative would be to encode and transmit an independent tree with the block prediction data, plus a separate tree with the MMP approximation. For the presented example this would be more inefficient than the used approach, because it would require a total of 16 symbols: 5 symbols for the prediction tree plus 11 symbols for the MMP data.

5.2.2 Rate-distortion optimisation

Unlike the algorithm described in Section 3.1.1, that only considers two cases for each tree node (each block can either be segmented or approximated by using a single dictionary vector), the MMP-I encoder must also optimise the block prediction. In order to do this, MMP-I uses an adapted version of the original RD optimisation algorithm. The distortion and rate values associated with the residue block at each binary tree node are estimated, and the values for the corresponding cost functions are determined. During optimisation, the Lagrangian costs for the prediction/coding options are compared, and the most favourable solution is chosen.

MMP-I with a fixed prediction block size (MMP-I FBS) uses a straightforward prediction optimisation (see Figure 5.7): for each available prediction mode, M, it first determines

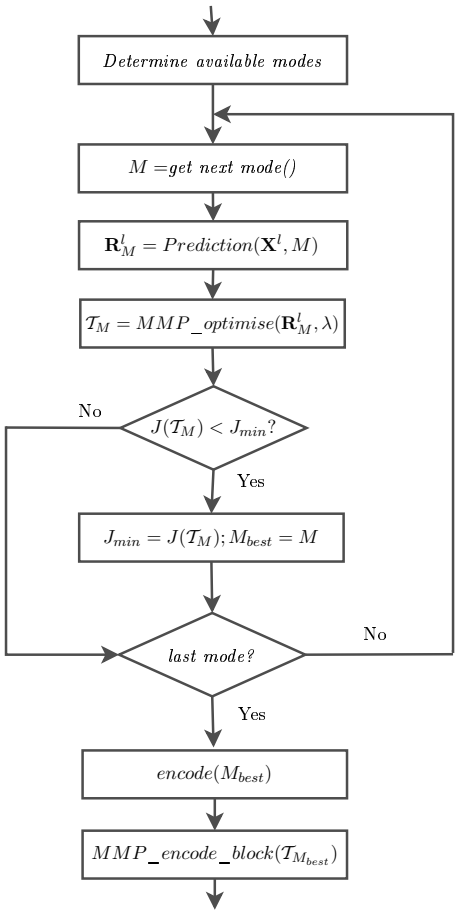


Figure 5.7: Flowchart for the MMP-I FBS algorithm.

the prediction error block of the original scale, $\mathbf{R}_M^l$, and then uses the MMP optimisation algorithm to determine the MMP approximation of this block, given by $\hat{\mathbf{R}}_M^l$. The MMP optimisation routine also returns the value of the cost function associated with prediction mode M: $J(\mathcal{T}_M)$. After repeating this procedure for every available prediction mode, the process simply chooses the option that minimises the value of the RD optimisation cost function and uses it to encode the input block. A formal description of the MMP-I FBS algorithm is presented in Section B.3 of Appendix B.

MMP-I with adaptive block size prediction tests all possible combinations of prediction block size and prediction mode, in order to determine the most efficient option. In order to do this, the MMP-I FBS procedure is used recursively, performing the optimisation decisions for a given image block $\mathbf{X}^l$, of fixed size. Besides the optimisation of the MMP compression of the current node, this function also has to decide if the current block should use a single prediction mode, or be predicted in two or more separate steps. For

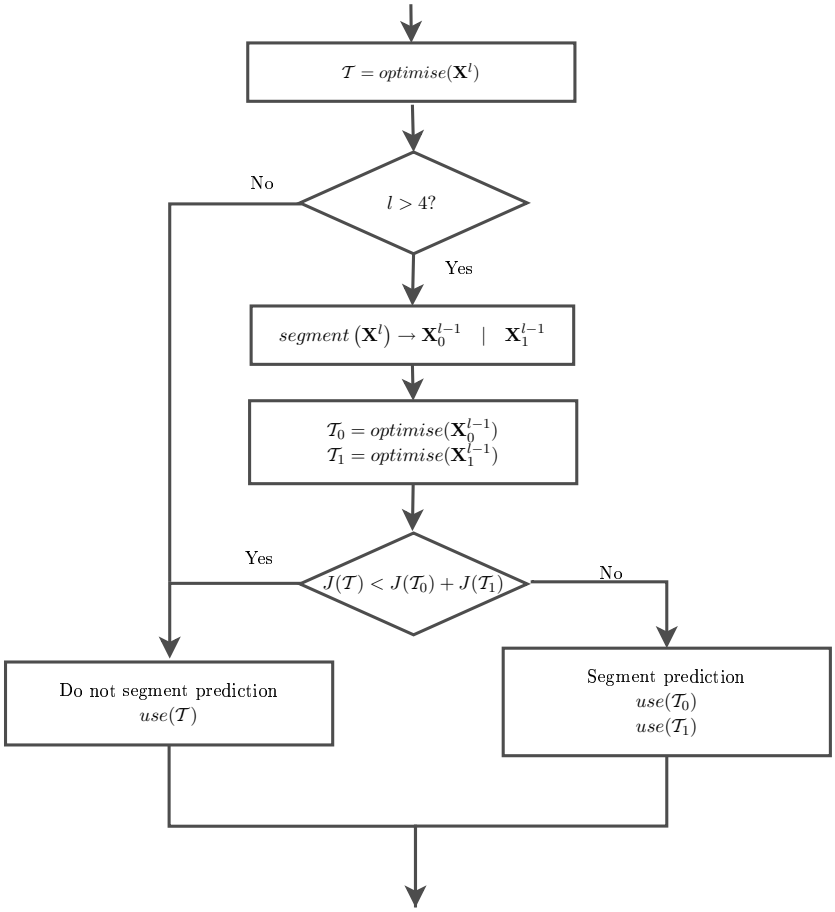
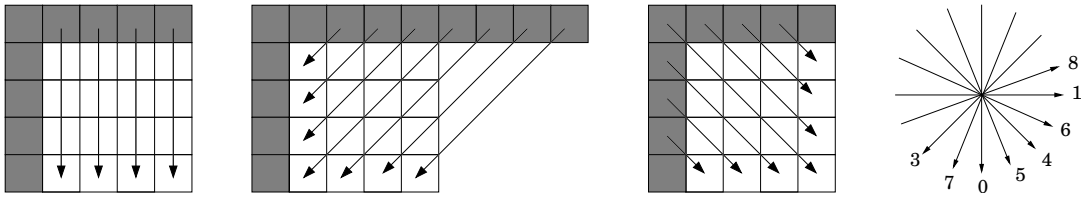


Figure 5.8: Flowchart for the MMP-I algorithm.

this purpose, the algorithm partitions the image block into two separate sub-blocks and recursively optimises the MMP-I compression for each half. After evaluating the RD optimisation cost function associated with these two coding options, the method determines the best prediction pattern for the current block and the corresponding MMP encoding data. A simple flowchart for this process is presented in Figure 5.8. An algorithmic description of MMP-I is presented in Section B.4.

The MMP-I optimisation process reflects a compromise between two factors: first, the use of small prediction block sizes favours prediction accuracy, but has the disadvantage of increasing the overhead associated with the prediction mode flags; second, the use of larger prediction blocks saves prediction bits, but tends to generate predicted residue blocks with more activity, that usually require more bits to be compressed. Thus, the use of a hierarchical prediction scheme, combined with the described RD optimisation techniques, allows the encoder to determine the best trade-off between the prediction performance and



(a) Mode 0 - Vertical

(b) Mode 3 - Diagonal Down/Left

(c) Mode 4 - Diagonal Down/Right

(d) Used directions

Figure 5.9: Three of the directional prediction modes and the eight prediction directions used in MMP-I.

the allocated rate.

5.2.3 The prediction process

In order to enhance the performance of the prediction process, which aims to produce a low energy residue pattern for every processed block, several prediction models were incorporated into MMP-I. These modes were inspired by those used by H.264/AVC [13], plus a set of other modes, that determine the prediction block based on non-directional prediction schemes. Moreover, some adaptations were also made on the original H.264/AVC prediction process. The DC mode, that is also used by H.264/AVC, was initially considered, but was replaced by a new prediction mode that uses the most frequent value (MFV) among the neighbouring pixels, instead of their average. This change, allowed MMP-I to achieve good performances even for non-smooth images. The MFV prediction mode will be explained later in this section, after a brief discussion of the directional prediction modes used by MMP-I.

Directional prediction modes

Directional prediction modes are efficient at predicting areas of the image with structured texture. Several directions are used in order to cover texture orientations ranging from vertical and horizontal patterns to a set of diagonal modes, that correspond to the possible prediction directions that use causal reference blocks. The image areas used as reference for prediction correspond to the neighbouring pixels to the left and above the current block, as represented in Figure 5.9. Depending on the location of the image block that is being

predicted, not all prediction modes may be available. For example, if the left reference pixels are not available (like in the case of image blocks located at the left border of the picture), only vertical (0) and "left" (3 and 7) modes may be used. In this case, all other modes are marked as unavailable and are not considered in the encoding process.

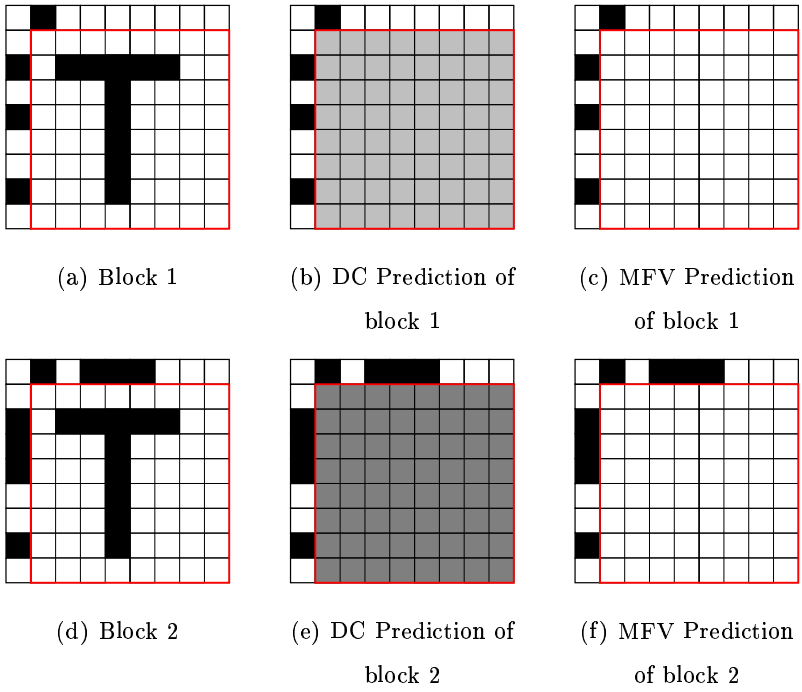
The implementation of the directional modes used an adaptation of the process described in the H.264/AVC standard [13]. Because MMP-I uses prediction blocks with adaptive dimensions, unlike H.264/AVC that only defines the prediction blocks for the 16×16, 8×8 and 4×4 cases, the original prediction process was adapted to the new prediction block sizes, using a straightforward expansion of the H.264/AVC prediction process. A detailed description of the original prediction process can be found in [13], while references [75, 76] present a very useful overview of these methods.

The directional prediction modes introduce visible artifacts in the prediction signal, that are unfavourable to an efficient compression, especially for block sizes larger than 4×4 [77]. To reduce the artifacts, the pixels used for prediction are filtered using a low-pass filter with impulse response:

$$h(n) = \left\{ \frac{1}{4}, \quad \frac{2}{4}, \quad \frac{1}{4} \right\}.$$

As in H.264/AVC, this filtering step increases the efficiency of the prediction process.

The efficiency of the directional prediction modes is greater for smaller block sizes, that have more regular texture orientations. Larger block sizes often show no uniform texture or several areas with distinct textures and different orientations. Therefore, for larger block sizes some prediction modes are deactivated, as the prediction process tends to be assured by other prediction schemes. This is also the case of H.264/AVC encoder (where only vertical and horizontal direction are used for 16×16 blocks). MMP-I uses only the vertical and horizontal directions for blocks with scales greater than 6 (8×8 blocks). In these cases, another prediction scheme is used, called Plane prediction, that uses an interpolation method to determine a plane function that best fits the upper and left reference prediction samples. This method is more efficient for larger blocks, due to the higher number of available reference pixels, and has a good performance in image areas with smooth variations of luminance.



(a) Block 1 (b) DC Prediction of block 1 (c) MFV Prediction of block 1

(d) Block 2 (e) DC Prediction of block 2 (f) MFV Prediction of block 2

Figure 5.10: Effects of DC and MFV prediction modes on two text image blocks.

The MFV prediction mode

In MMP-I, the DC prediction mode, traditionally used in several well known predictive coding methods (and in H.264/AVC), was replaced by a new prediction mode that uses the most frequent values (MFV) among the reference samples of the neighbouring blocks.

The DC prediction mode creates a uniform prediction pattern. The value of each pixel of this uniform block is the average of all considered reference samples. Unlike H.264/AVC, that uses transform coding to compress the prediction error block, MMP-I encodes this signal using blocks from a dictionary. Experiments showed that, for some image areas with high activity, the use of DC value for prediction is not as beneficial for MMP-I as for the case of transform coding. This is the case for text and compound images.

Figure 5.10 represents an example of the prediction of two text regions. These regions are characterised by a near white background with a few very dark pixels as foreground. In this case, DC prediction tends to generate error blocks with a uniform grey value, that can have severe variations, depending on the number of white and black pixels of the reference areas. This causes the DC values of the several predicted residue blocks to also have very different amplitudes, as can be seen in the examples of Figures 5.10 b) and e).

When using a transform-quantisation-based method this fact only corresponds to a shift in the DC coefficient of the transformed prediction residue block and poses no problem. Nevertheless, MMP has to use two code-vectors with the same shape (AC component), but with different average values. In this case, the use of DC prediction introduces more vectors in the dictionary, compromising the efficiency of the learning process of the dictionary and impairing the encoding performance. In these cases, the use of the most frequent value for the prediction has the advantage of creating prediction error blocks with samples that are more consistently clustered around zero (see Figures 5.10 c) and f)). This leads to a dictionary with fewer words, thus enhancing the overall coding efficiency. In addition, experimental tests have shown that, in the case of smooth images, the use of the MFV instead of DC prediction has no effect on the coding performance.

5.2.4 The MMP-I dictionary

MMP-I shares the adaptive pattern matching paradigm with the original MMP algorithm. It also uses the same dictionary updating strategies and scale transformations, that provide good adaptability for a wide range of input signals. Nevertheless, MMP-I encodes prediction residues instead of the image patterns processed by MMP, meaning that the range of sample values of the dictionary's vectors is now $[-255, 255]$. The dictionary initialisation was thus modified, to take into consideration the characteristics of the predicted residue signal.

Like in MMP, a set of uniform blocks is used for the initial MMP-I dictionary, but the DC values of these blocks are not uniformly spaced along the dynamic range of the vector samples. As prediction error samples tend to be clustered around zero, blocks with small values tend to be more frequent than those with larger sample values. Due to this fact, an initial dictionary with intensity values with higher density near zero is used. Experimental tests confirmed the advantage of this initialisation procedure, when compared with the use of uniformly spaced samples. Considering the initial dictionary vectors, $\mathbf{S_0}_i^l$, sorted in increasing amplitude order, the difference q between the amplitudes of two consecutive

vectors is set to:

$$\begin{cases} q = 2, & \text{if} \quad |I(\mathbf{S_0}_i^l)| \leq 10; \\ q = 4, & \text{if} \quad 10 < |I(\mathbf{S_0}_i^l)| \leq 22; \\ q = 8, & \text{if} \quad 22 < |I(\mathbf{S_0}_i^l)| \leq 86; \\ q = 13, & \text{if} \quad |I(\mathbf{S_0}_i^l)| > 86 \end{cases} \tag{5.11}$$

Despite the coding inefficiency caused by the use of this highly sparse initial dictionary, the MMP's dictionary updating procedure quickly adapts the dictionary to the typical patterns that are being encoded. In order to favour this adaptation process, dictionary $\mathcal{D}^0$, containing blocks with dimension 1×1, is an exception to the presented rule and is initialised with all the integers in the range $[-255, 255]$. This ensures that the algorithm is able to accurately approximate any input block at the highest precision (blocks being partitioned to 1×1), allowing every pixel to be independently approximated by any possible value, in spite of the fact that some amplitude values are not present in the upper scales of the dictionary. It also allows for lossless coding when $\lambda = 0$ in equations (3.2) and (3.3), that leads to a null distortion for every approximated block.

The gains introduced by the use of equation 5.11 in the creation of the initial dictionary are only marginal. In fact, experimental tests demonstrated the high adaptation capability of the MMP-I process, independently of the method used to create the initial dictionary.

In order to avoid the insertion of repeated blocks in the dictionary, a test condition is used in the updating stage. Like in MMP, independent copies of the dictionary are kept for each scale. As a result, the smaller scales of the dictionary tend to have less elements than the ones for the larger scales, because at smaller scales it is more likely that a new vector $\hat{\mathbf{X}}^l$ will match an existing dictionary block.

5.3 Algorithm's evaluation and experimental results

In this section we evaluate the experimental results for MMP-I. Section 5.3.1 compares the RD performance of MMP-I with that of MMP, in order to assess the quality gains introduced by the use of predictive coding. We also present some considerations about the MMP-I coding process, as well as an evaluation of the effects of using FBS for prediction and of using the MFV prediction mode, instead of the original DC mode. In Section 5.3.2 we compare the performance of MMP-I against that of the current state-of-the-art image

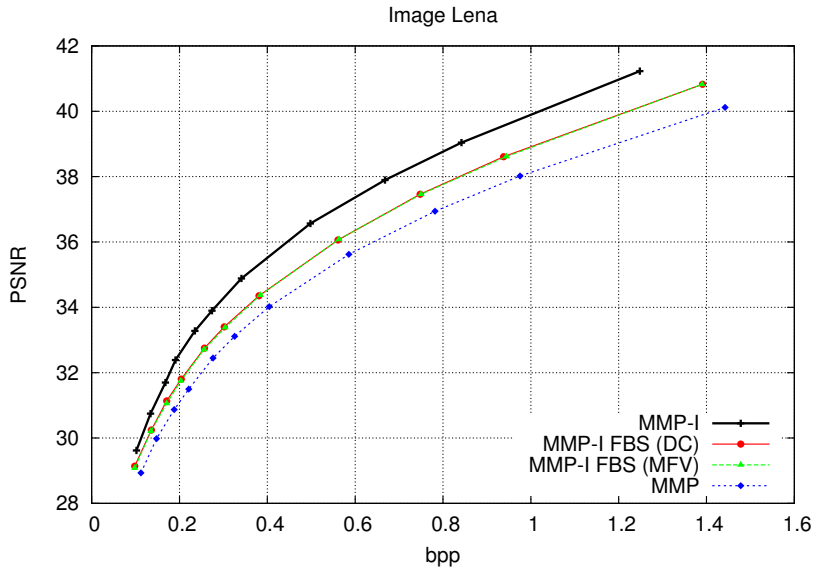


Figure 5.11: Experimental results for MMP-I for smooth image Lena.

compression schemes, namely JPEG2000 [12] and H.264/AVC's *FREext* high profile Intra encoder [13].

5.3.1 First measurements and comparison with MMP

Figures 5.11 and 5.12 compare the rate-distortion performance of MMP-I against that of the original MMP algorithm, for natural image Lena, text image PP1205 and compound image PP1209[1]. The final results of MMP-I with fixed prediction block size (FBS) may be observed, for the cases where the H.264/AVC's DC prediction mode is used ("MMP-I FBS (DC)") and when DC is replaced by the MFV prediction mode ("MMP-I FBS (MFV)"). The results for MMP-I with ABS (MMP-I) are also presented. From Figure 5.11 one may observe the performance gains introduced by all versions of MMP-I when compared with the original MMP algorithm, for smooth images (namely image Lena). These consistent gains, that go up to more than 2 dB, demonstrate the usefulness of the signal distribution adaptation paradigm, implemented by the predictive step of the MMP-I algorithm. In fact, for smooth images, even the FBS version of MMP-I is able to increase the performance of the MMP algorithm by almost 1 dB. Additionally, the advantage of using the more efficient adaptive block size prediction algorithm is clear from this plot. Another interesting

[1]Results for other test images are presented in Figure C.18 of Appendix C.

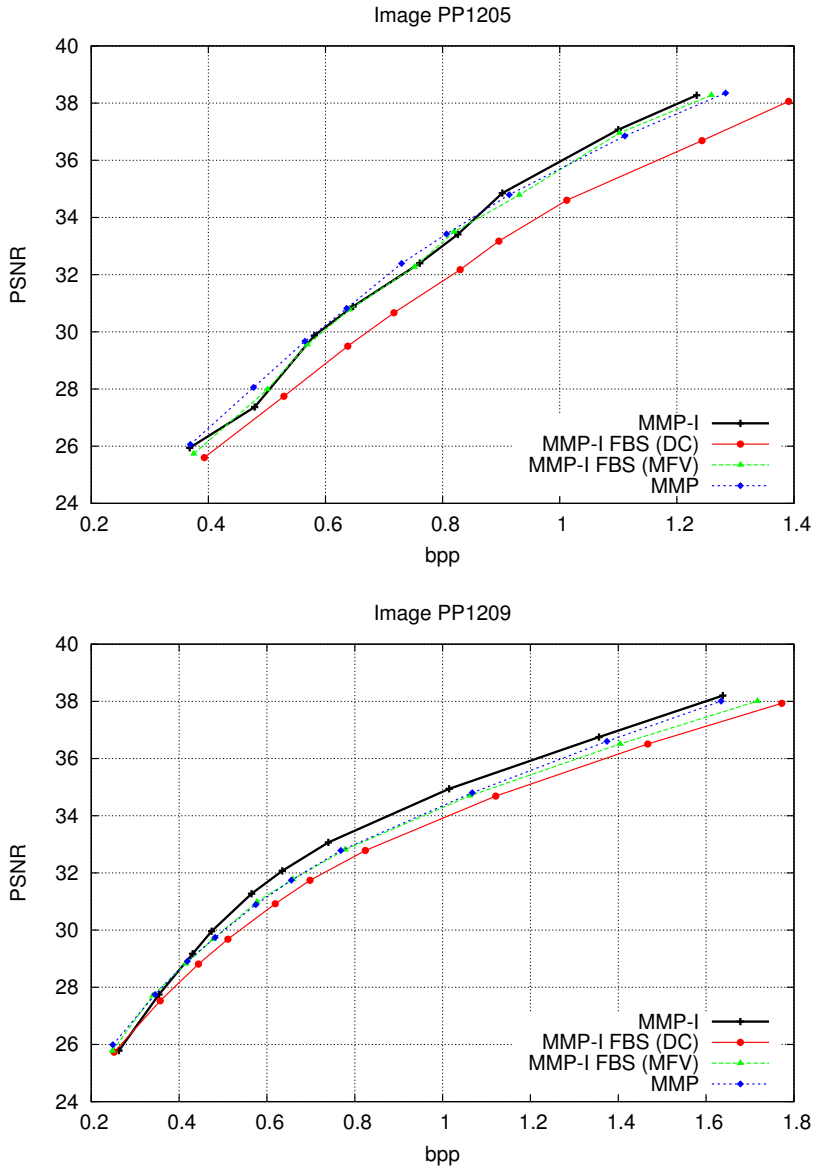


Figure 5.12: Experimental results for MMP-I for text image PP1205 and compound image PP1209.

observation can be made regarding the use of the MFV prediction scheme in replacement of the original DC mode: for smooth images, this change has no noticeable effects in the performance of the algorithm, as was previously discussed in Section 5.2.3.

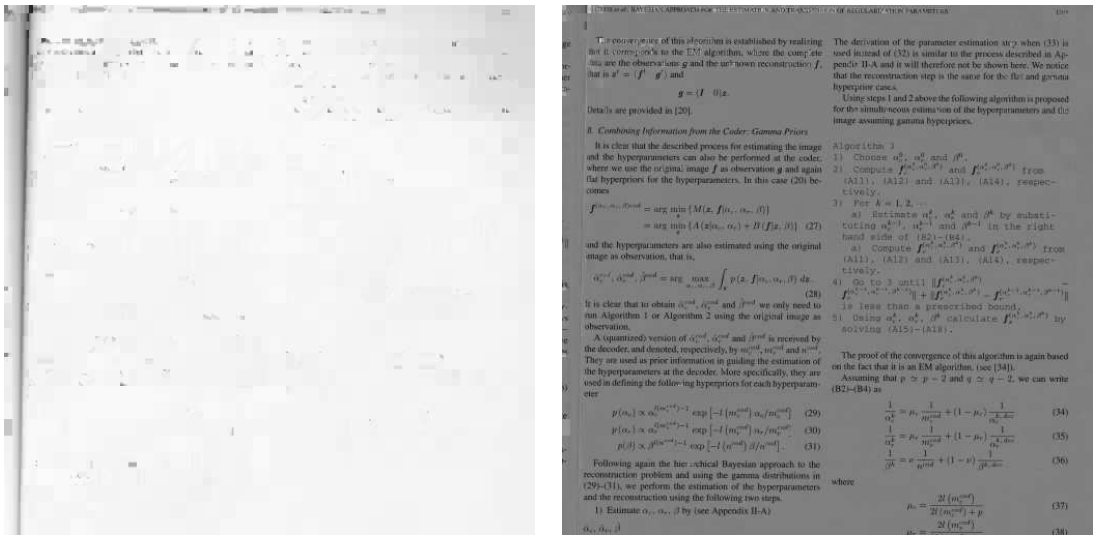
Figure 5.12 presents the same comparisons, for text image PP1205. For this image, as for other text, compound or non-smooth images in general, the efficiency of the prediction process is limited, due to the high spatial activity of the images' patterns. Because of this, the prediction residue does not have the same regular features as the ones observed

(a) (b)

Figure 5.13: Predicted image (a) and prediction error image (b) for image Lena encoded with MMP-I at approximately 0.35 bpp.

for the case of smooth images. This is also reflected in the probability distribution of the prediction residue, that is not altered in a way that favours the dictionary adaptation process. Because of this, the use of predictive techniques by MMP-I is not able to introduce any RD performance gains in relation with the original MMP method. Notice, however, that the prediction data generates an additional overhead, that is compensated by a higher efficiency in the residue encoding stage of MMP-I. Finally, one has that the use of the original DC prediction mode severely compromises the performance of the MMP-I method for text images. In this case there is a performance loss of almost 2 dB, due to the dictionary scattering effects of the use of DC predicted blocks, that was previously discussed in Section 5.2.3.

Figure 5.12 also shows the RD plots for the compound image PP1209. Like for smooth images, MMP-I is able to achieve a consistent gain over the original MMP algorithm. Even though the quality gains are not as large as for the case of smooth images, MMP-I is able to achieve quality improvements in the range of 1 dB. From the conclusions drawn from the previously analysed images, it is possible to infer that such gains over MMP come mainly from coding the smooth areas of the image. For the text regions, MMP-I maintains the coding performance of MMP.



(a) (b)

Figure 5.14: Predicted image (a) and prediction error image (b) for image PP1205 encoded with MMP-I at approximately 0.8 bpp.

Figures 5.13 and 5.14 show the prediction signals and the prediction error images generated by MMP-I for images Lena and PP1205, respectively. In the prediction error images the neutral tones of grey are used to represent the zero, while progressively darker and brighter regions represent prediction error pixels with larger absolute values. For the smooth image Lena (Figure 5.13) it is possible to observe the accuracy of the prediction process, that is able to conveniently estimate the original image patterns. This results in a very regular prediction error image, that allows for a more efficient adaptation of the MMP dictionary, resulting in a much more efficient compression.

Figure 5.14 presents the prediction and residue signals for text image PP1205. From these images it is clear that the prediction process is most often unable to conveniently estimate the highly detailed patterns of the input image. Because of this, the original image patterns are mostly repeated in the prediction error image. As a result, the use of MMP-I ends up having approximately the same compression efficiency as the original MMP algorithm, for this type of patterns.

Figure 5.15 shows the number of MMP-I blocks that used each of the available prediction modes, at each of the block scales used for prediction. The plots compare these results for images Lena and PP1205. For the smooth image case (Figure 5.15 a)) one may observe

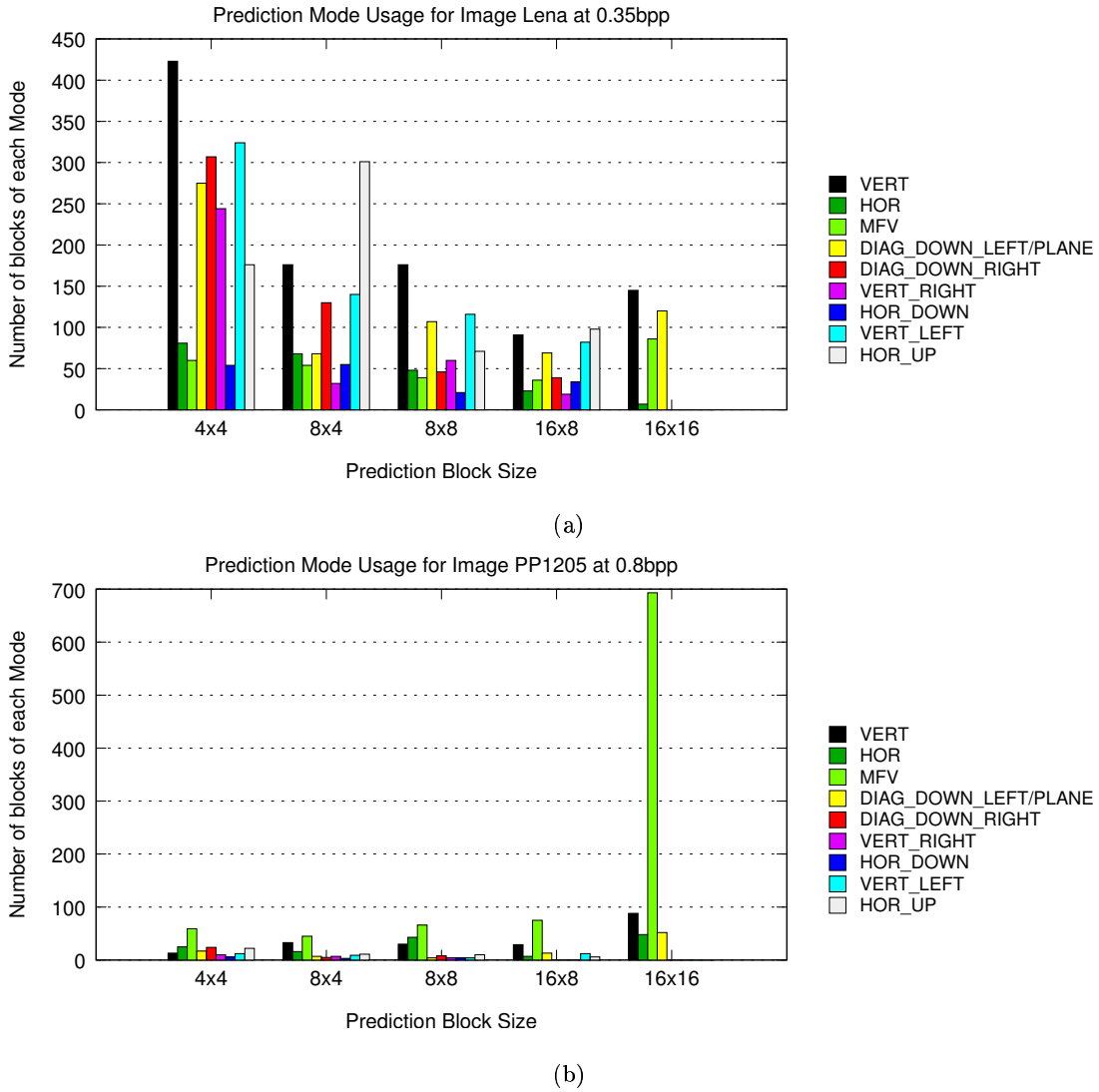


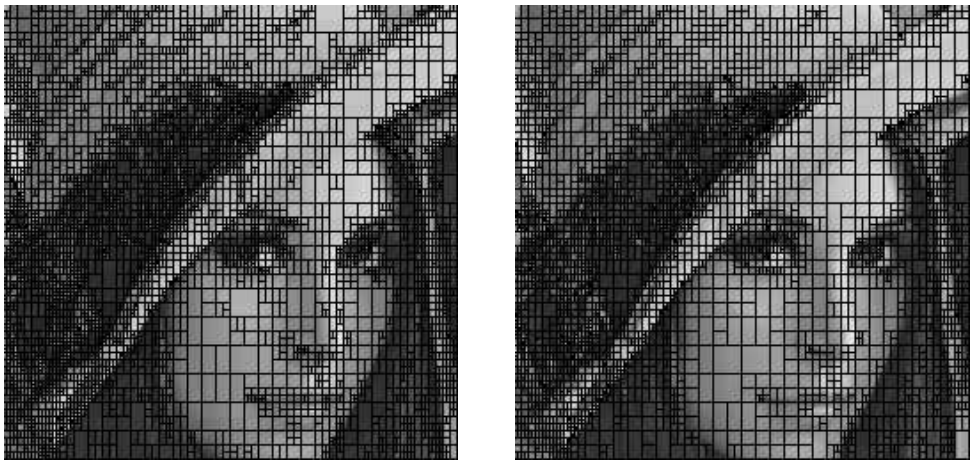
Figure 5.15: Number of blocks that used each of the prediction modes, for each scale, for images a) Lena and b) PP1205.

a distribution of all prediction modes across all prediction block sizes. Smaller prediction blocks result in a more efficient estimation of the image patterns and allow MMP-I to save bits used by the MMP compression of the prediction error. Nevertheless, this increases the overhead associated with the prediction data. Each pair of prediction mode and prediction block size is chosen by the MMP-I RD optimisation algorithm. For text image PP1205 (Figure 5.15 b)), one may notice the predominant use of very large block sizes for the prediction step. Also, the frequency of the MFV prediction mode is clearly higher than that of the other modes, for all prediction block sizes. This happens because of the

Figure 5.16: MMP-I segmentation for image Lena coded at 0.5 bpp. The prediction blocks are represented by the black lines and the MMP segmentation is represented by the white lines.

difficulty in performing an efficient prediction of the text patterns, that generally do not fit into any particular directional texture. Also, because the prediction process tends not to be advantageous, the RD optimisation chooses to use mostly large prediction blocks, in order to save bits associated with the image prediction process.

Figure 5.16 represents the block segmentation used by MMP-I when it is applied to image Lena encoded at 0.5 bpp. One may observe the adaptiveness of the MMP-I encoder both regarding the prediction block size (represented by the black lines) and the MMP partition of the prediction residues (shown in white). The image regions with higher detail levels generally correspond to the use of smaller blocks, both for the prediction and the MMP encoding steps. Nevertheless, for some image areas one may notice the use of large prediction blocks associated with the segmentation of the predicted error patterns. Figure 5.17 compares the block segmentation for the MMP and the MMP-I encoders, for a detail of image Lena encoded at 0.5 bpp. In this case both segmentations used by the MMP-I encoder are combined, in order to show the real size of the used block segments. It is possible to observe that, generally, the prediction techniques allow the MMP-I encoder to



(a) (b)

Figure 5.17: Final segmentation for image Lena coded at 0.5 bpp, for a) MMP and b) MMP-I.

use much larger partitions when compressing the input image. This is particularly evident in the more detailed regions of the image, like the nose and lips of Lena, as well as along the image edges. From the analysis of the performance of MMP for GG signals, shown in the early sections of this chapter, one would expect the more regular prediction error patterns, that are mostly concentrated around a specific shell of the image space, to favour the dictionary adaptation. This result shows that, besides this, the prediction process also favours the use of blocks of larger scales in the approximation process, which are more efficient than the smaller blocks.

5.3.2 Comparison with state-of-the-art methods

In this section we evaluate the relative RD performance of the MMP-I algorithm against that of the current state-of-the-art image encoding methods JPEG2000 [12] and H.264/AVC high profile intra frame encoder [59]. Note that the H.264/AVC's high profile also uses 8×8 blocks for intra prediction and transform coding, in addition to the 16×16 and 4×4 blocks defined in the baseline and main profiles. This yields a good approximation to the hierarchical prediction performed by MMP-I, in a DCT coding framework.

Experimental tests were performed using several types of images (see Section 3.2.1). Figures 5.18 and 5.19 show the RD performance for natural (smooth-like) images Lena and Peppers and Cameraman. These figures show once again the significant improvement

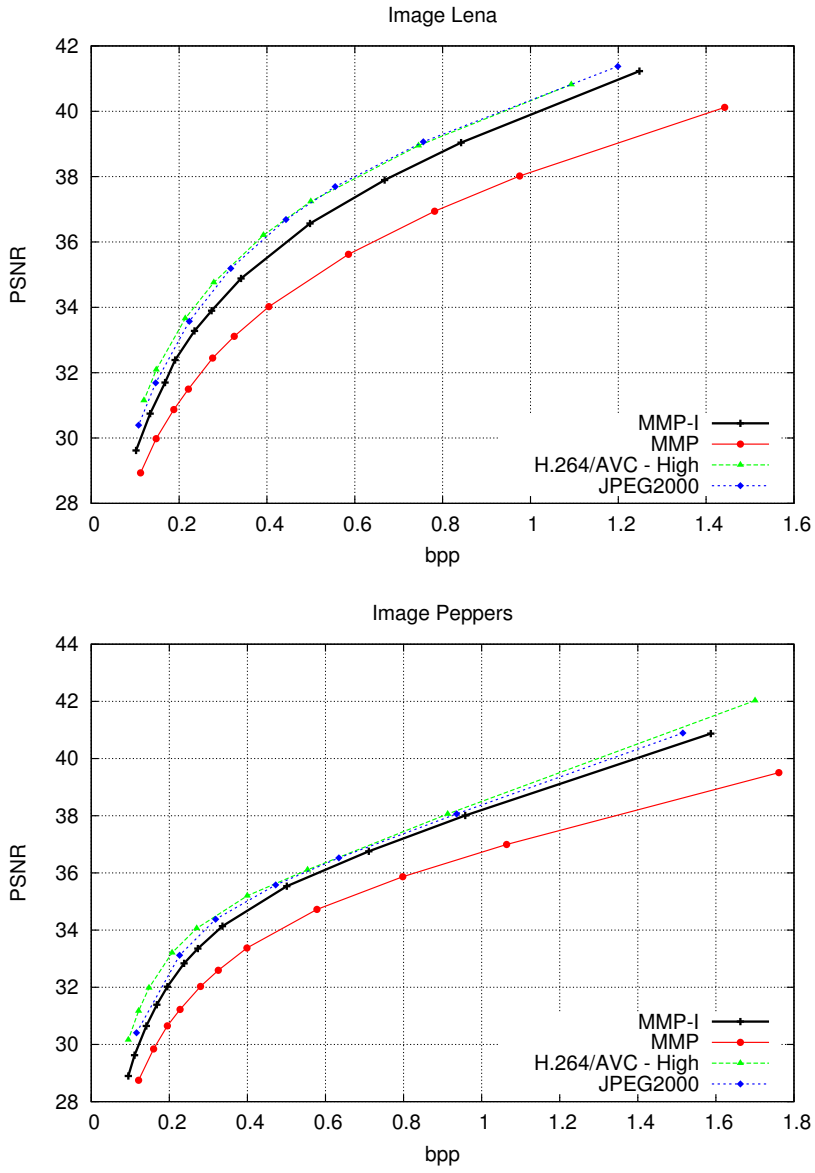


Figure 5.18: Experimental results of MMP-I for grayscale smooth test images Lena and Peppers.

in objective quality introduced by MMP-I for the compression of smooth grayscale images (up to almost 2 dB, when compared with MMP). The gains in RD performance of the transform-based methods over MMP-I are now small, compared with the performance gap over MMP. These gains strongly depend on the test image, but one may observe small PSNR differences when MMP-I is compared with JPEG2000. For image Cameraman, MMP-I is even better than the DWT-based encoder (see Figure 5.19), while for images Lena and Peppers, the differences in objective quality are inferior to 0.7 dB and 0.5 dB,

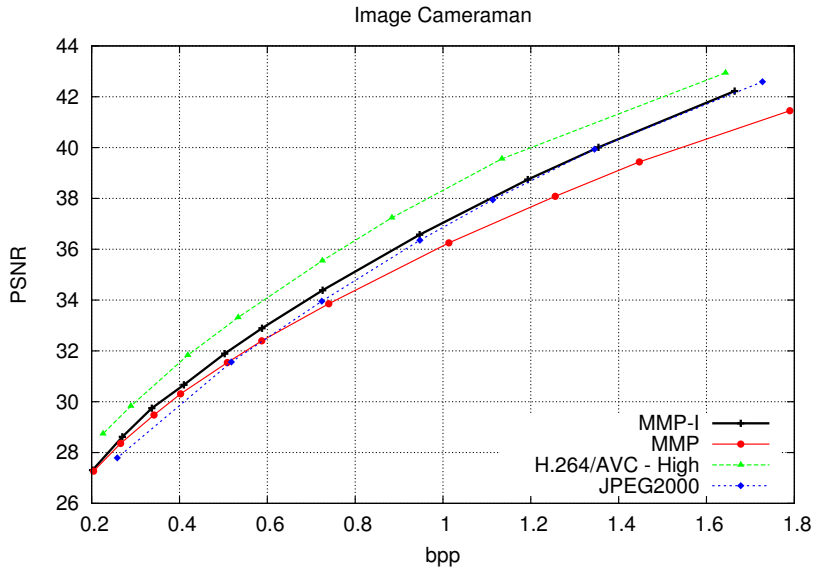


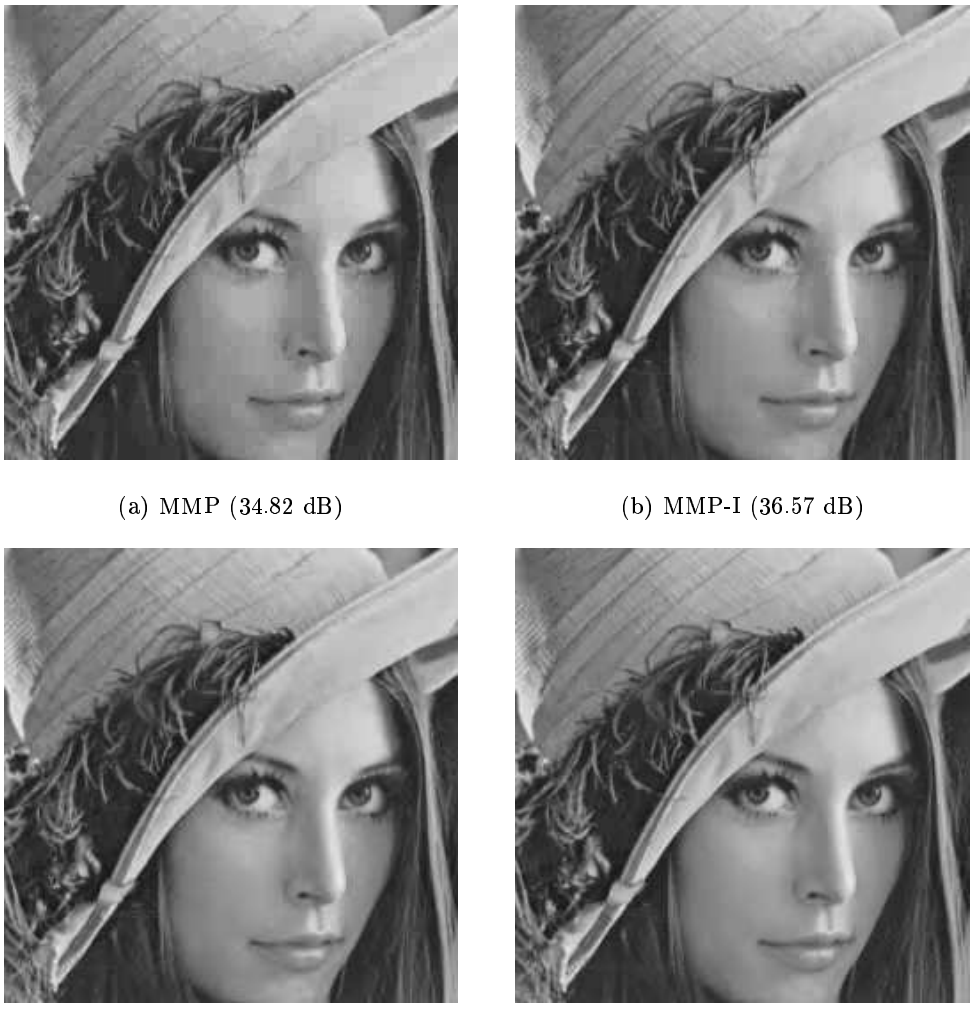
Figure 5.19: Experimental results of MMP-I for grayscale smooth test image Cameraman (256×256).

respectively (see Figure 5.18). In spite of this, the traditional transform-quantisation-based methods, that are highly optimised for low-pass images, are still more efficient than MMP-I for this type of images. One may also observe the relative performance of H.264/AVC encoder and JPEG2000, for the tested images. For images Lena and Peppers these encoders present approximately the same RD performance. Image Cameraman is an exception to this rule, since H.264/AVC is able to achieve quality gains of more that 1.5 dB over JPEG2000 and 1 dB above that of MMP-I.

Figure 5.20 shows a detail for image Lena compressed at approximately 0.5 bpp using MMP, MMP-I, JPEG2000 and H.264/AVC[2]. From these images one may notice the improvement in perceptual quality that is introduced by MMP-I when compared with MMP for this image. Analysing the perceptual results of transform-based image encoders, we notice a smoother representation of the image when compared with MMP-I, that presents some blocking artifacts. This partially explains the PSNR gains achieved by these algorithms.

Figure 5.21 compares the performance of MMP-I for text (PP1205) and compound (PP1209) images, with that of MMP, JPEG2000 and H.264/AVC. These plots show the advantage of using the multiscale recurrent pattern-based encoders for these types of im-

[2]Refer to Figures C.29 to C.31 of Appendix C for a representation of the entire images.



(a) MMP (34.82 dB)

(b) MMP-I (36.57 dB)

(c) JPEG2000 (37.15 dB)

(d) H.264/AVC (37.24 dB)

Figure 5.20: Detail for image Lena compressed at approximately 0.5 bpp using a) MMP; b) MMP-I; c) JPEG2000; and d) H.264/AVC.

ages. It is also possible to observe a consistent performance advantage of H.264/AVC over JPEG2000, that amounts to quality gains of about 1 dB for all tested rates. In these cases, the highly adaptive compression scheme used by H.264/AVC, with its various prediction modes and variable block size blocks for transform coding and block prediction, allows for a greater efficiency than that of the DWT-based encoder. Nevertheless, neither of these transform-based methods is able to efficiently cope with the high activity patterns that are typical of text images. This happens because these patterns violate the low-pass assumption for the frequency spectrum of the input image. Both H.264/AVC and JPEG2000, as generally all other transform-based image compression algorithms, are highly optimised in

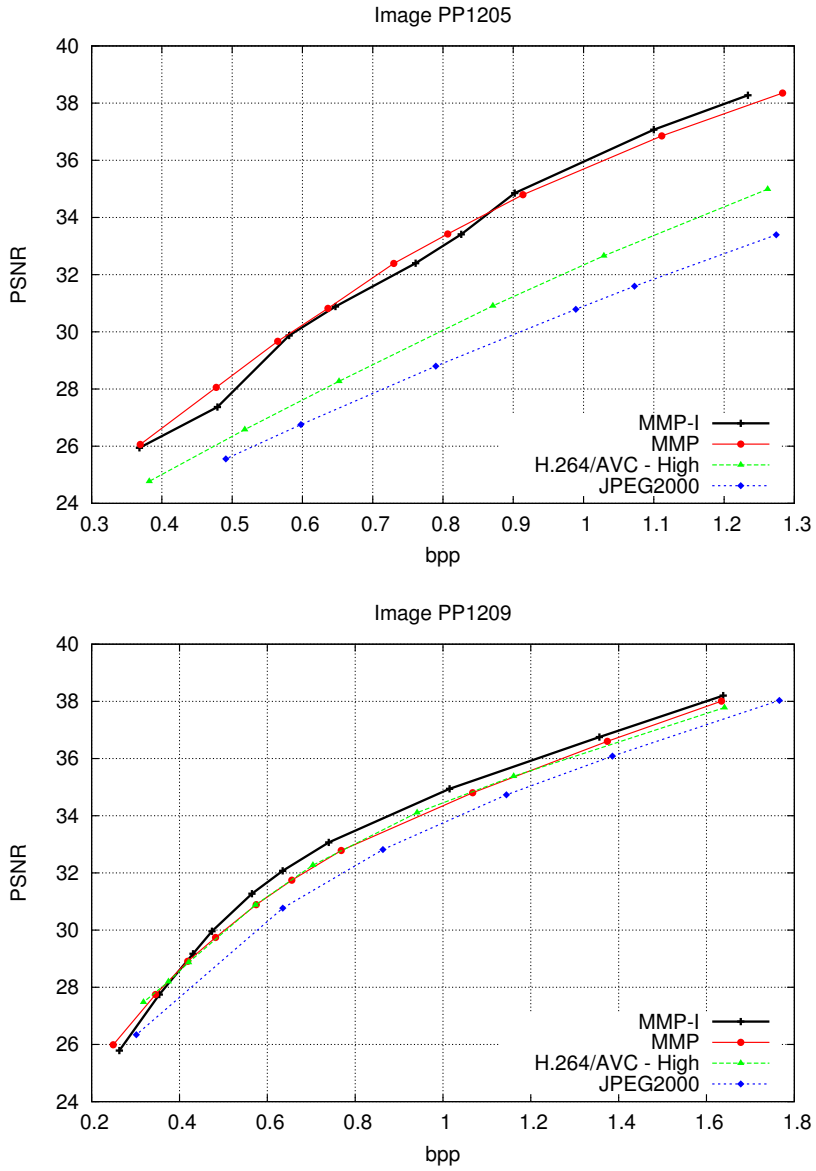
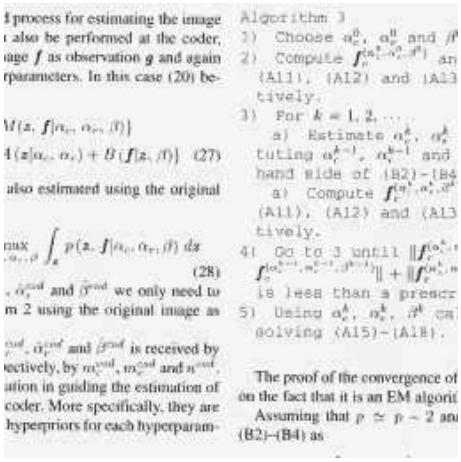
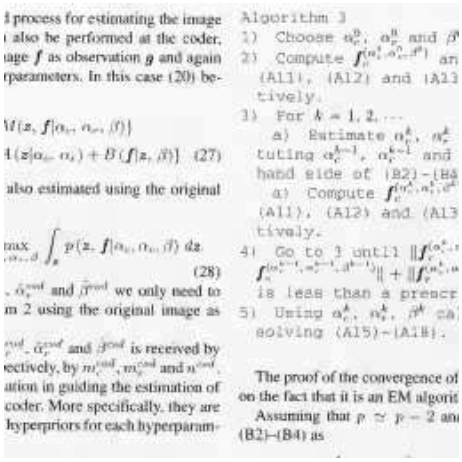


Figure 5.21: Experimental results of MMP-I for text image PP1205 and compound image PP1209.

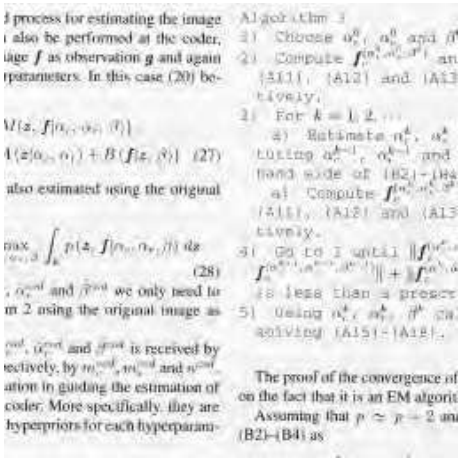
exploiting this feature of most natural images. Nevertheless, their performance degrades severely when this is not the case. Because the MMP-based algorithms do use any *a priori* assumption about the input signal, this quality degradation is not observed for the cases of MMP and MMP-I. As a result, for text image PP1205, where MMP and MMP-I have similar performances, the PSNR gains of these algorithms are about 4 dB, when H.264/AVC is considered, and about 5 dB in comparison with JPEG2000. The perceptual results for



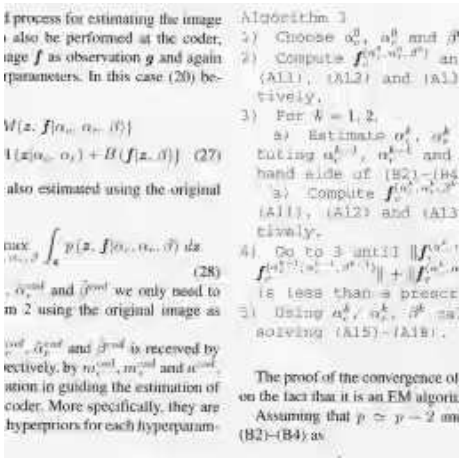
(a) MMP (30.82 dB)



(b) MMP-I (30.89 dB)



(c) JPEG2000 (27.21 dB)



(d) H.264/AVC (28.27 dB)

Figure 5.22: Detail for image PP1205 compressed at approximately 0.65 bpp using a) JPEG2000 and b) H.264/AVC.

this image, encoded approximately at 0.65 bpp, may be observed in Figure 5.22[3]. From these images one may clearly notice the quality advantage of both MMP-based encoders, when compared with JPEG2000 and H.264/AVC.

For compound image PP1209 (see Figure 5.21 b)), MMP-I outperforms all other tested methods. When compared to JPEG2000, MMP-I is able to improve the objective quality by about 1 dB. The H.264/AVC consistently achieves better results than JPEG2000 also for this type of images. Its RD performance is equivalent to that of MMP but inferior to that of MMP-I, by about 0.5 dB.

[3]Refer to Figures C.32 to C.34 of Appendix C for a representation of the entire images.

These experimental results show that MMP-I is able to significantly improve the quality of MMP for smooth images, while maintaining, or improving, the excellent results for text and compound images, when compared with state-of-the-art encoders. Unlike the traditional transform-quantisation-entropy coding-based methods, that have poor coding efficiency for non-smooth images, and original MMP, that has a large performance deficit for smooth images, MMP-I has a good performance for all image types. This good all-round performance of MMP-I results from its ability to efficiently exploit the features of the prediction error signal, that allow for a more efficient MMP dictionary adaptation. The result is a highly adaptive encoding method with a useful universal feature.

The presented results motivated the investigation of improved dictionary adaptation techniques for MMP-I. One important goal of this investigation is to further increase the performance of this method for smooth images. The aim to achieve an equivalent performance to the state-of-the-art transform-based methods for smooth images should not, however, compromise the excellent performance of MMP-I for the text and compound images. The results of this investigation are presented in the following chapters. Prior to this discussion, the next section presents an analysis of the prediction error probability distribution, in order to determine the accuracy of the assumed GG model and determine new, useful insights on the features of the prediction error signal.

5.4 On the probability distribution of the MMP-I prediction error

The effects of using intra prediction in MMP were first observed in Section 5.3.1, where Figures 5.13 and 5.14 allowed for a visual analysis of both the prediction and the prediction error images. In this section we further investigate the MMP-I prediction error signal. We show that this signal consistently presents highly peaked probability distributions, that we can accurately model using generalised Gaussian functions. We also estimate the values for the shape parameter and standard deviation of these models. These experimental results show that the prediction error signal has probability distributions that are consistently narrower than the Laplacian one.

Figure 5.23 shows the histogram for the original image Lena and the histogram of the prediction error signal generated by MMP-I, for the represented compression ratios. We

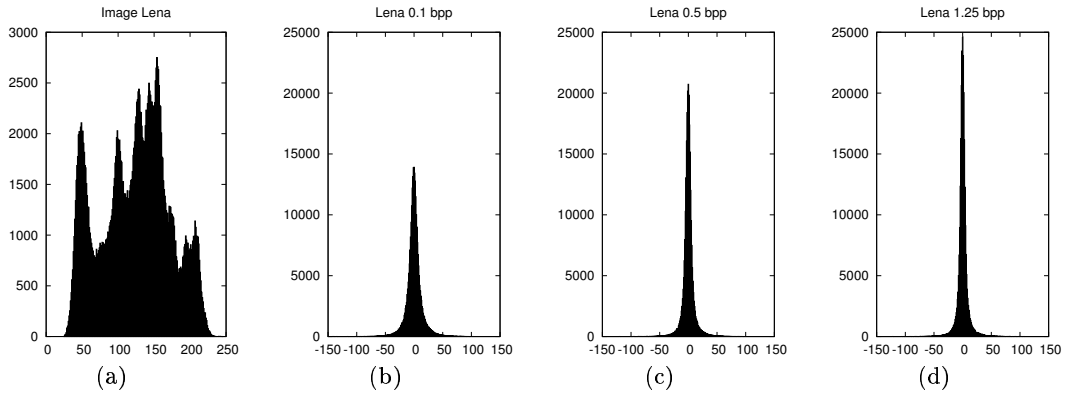


Figure 5.23: Histograms for a) original image Lena and MMP-I prediction residues at compression ratios: b) 0.1 bpp; c) 0.5 bpp and d) 1.25 bpp.

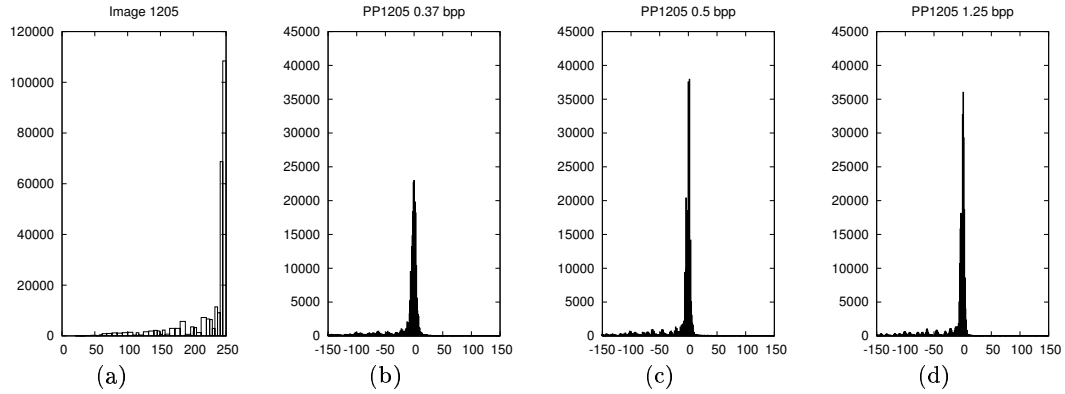


Figure 5.24: Histograms for a) original text image PP1205 and MMP-I prediction residues at compression ratios: b) 0.37 bpp; c) 0.5 bpp and d) 1.25 bpp.

observe the transformation of the probability distribution function, that is now concentrated around a reduced set of low luminance values. This corresponds to the probabilistic counterpart of the regularity features, that were inferred by the visual inspection of the prediction error signal, performed in Figure 5.13. Figure 5.23 shows that, as the compression ratio decreases, the distribution becomes narrower, meaning that the prediction process for these higher rates is more accurate. Two factors explain this fact: first, for higher rates MMP-I is able to use smaller prediction block sizes, increasing the prediction accuracy; and second, for smaller compression ratios the distortion of the encoded pixels that are used in the prediction step decreases, which also improves prediction efficiency.

The prediction process for all other smooth test images has the same features as the

ones described for image Lena. Nevertheless, non-smooth images, like text and compound images, present very different features, that also affect the prediction process. Figure 5.24 represents the original histogram of text image PP1205 and the histograms of the predicted residues for some selected compression ratios. We observe the existence of a few luminance values with extremely high probability in the input image histogram, that correspond to the near white values of the background pixels. Also, the probability distribution of the predicted residues has two very important differences when compared with that of smooth images: first, it is asymmetrical; and second, there are many prediction samples with a high absolute value, which have a significant probability. These facts are caused by the prediction process itself: because most of the prediction is made using the MFV mode, the prediction pixels tend to be centred around 255. As a result, the predicted residues are mainly negative and their probability depends on the original histogram. These facts demonstrate the conclusions that were taken in Section 5.3.1 from the visual inspection of the prediction error images (see Figure 5.14).

5.4.1 Estimation of the probability model of MMP-I predicted residues

From Figures 5.23 and 5.24 one may observe the highly-peaked shape of the the MMP-I's predicted residues' probability distribution, especially for smooth images. In this section we further analyse this probability distribution function and relate it to a generalised Gaussian distribution. In order to do this, the values for the shape parameter, α, and of the standard deviation, β, were estimated [69, 70]. The actual samples of the prediction error were used, for each test image and for different target compression ratios. Table 5.2 shows a summary of the estimated values for parameters α and β, for the mentioned test images and compression ratios.

The analysis of Table 5.2 allows for some interesting conclusions. For natural grayscale images we observe probability density functions consistently narrower than the Laplacian one. Also, both the value for the shape parameter and for the standard deviation decrease with the decreasing compression ratio. This means a growing concentration of the predicted error distribution with the decreasing average distortion of the encoded image. For all observed images, the estimated value for the shape parameter is mostly concentrated between 0.5 and 0.75, except for image Cameraman, that tends to have a narrower distribution, and image Goldhill, for which the estimated value of α goes up to 0.95 at higher

Lena			Cameraman			Goldhill			PP1205			PP1209		
bpp	α	β	bpp	α	β	bpp	α	β	bpp	α	β	bpp	α	β
0.11	0.74	13.8	0.20	0.36	29.0	0.11	0.95	15.4	0.37	0.64	35.2	0.26	0.51	26.0
0.34	0.61	11.1	0.34	0.34	29.1	0.32	0.86	12.5	0.52	0.60	34.3	0.35	0.50	28.1
0.50	0.59	10.3	0.51	0.34	29.2	0.50	0.81	11.8	0.75	0.63	35.0	0.56	0.49	28.0
0.67	0.57	10.2	0.72	0.33	27.8	0.79	0.77	11.3	0.90	0.64	35.3	0.75	0.46	29.4
0.83	0.56	10.0	0.98	0.31	28.2	1.08	0.75	11.0	1.25	0.65	35.4	1.00	0.49	28.2
1.25	0.54	9.6	1.36	0.31	24.4	1.36	0.73	10.8	1.41	0.63	35.2	1.64	0.50	26.7

Table 5.2: Estimated values of the generalised Gaussian parameters for some test images.

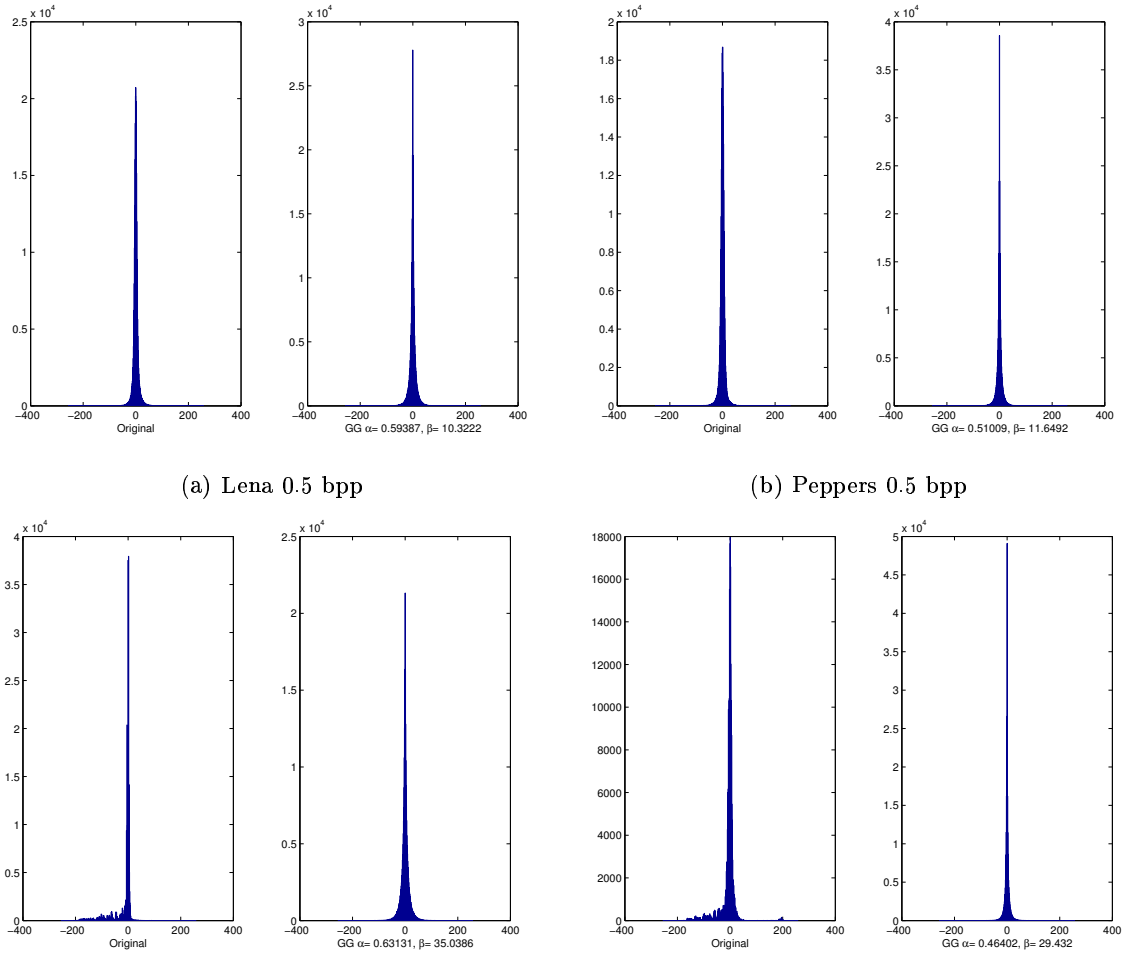
compression ratios.

Figure 5.25 compares the actual distribution of the prediction error samples with the generalised Gaussian distribution, having the α and β parameters equal to the estimated parameters. The accuracy of the predicted model depends on the features of the test image, but generally we may observe a good match between the original signal distributions and the estimated generalised Gaussian. This demonstrates the adequacy of this model for representing the prediction error.

5.5 Conclusions

In this chapter we have presented MMP-I, a new compression scheme that combines the use of MMP with predictive coding techniques. This investigation was motivated by a study on the performance of MMP for the compression of signals characterised by different probability distributions. In Section 5.1 we have shown that the efficiency of the dictionary adaptation increases for input signals with narrower probability distributions, which results in a better encoding performance for MMP. By using adaptive image prediction processes, like those of H.264/AVC [13], we are able to produce predicted residue signals with highly concentrated probability distributions, that correspond to a generalised Gaussian model. MMP-I achieves a greater efficiency of the dictionary adaptation process, by compressing the predicted error signal instead of the input image data (see Section 5.2).

Experimental results, shown in Section 5.3, demonstrate a significant improvement on the compression performance of MMP-I, when compared with MMP, especially for smooth



(a) Lena 0.5 bpp

(b) Peppers 0.5 bpp

(c) PP1205 0.75 bpp

(d) PP1209 0.75 bpp

Figure 5.25: Original distribution of the predicted errors compared with a generalised Gaussian distribution with the estimated parameters.

images.

In Section 5.4 we were also able to demonstrate the suitability of the used GG model for representing the prediction error signal, especially for the case of natural images. This result validates one of the initial assumptions that were considered for the development of the MMP-I algorithms. Additionally, a GG probability distribution framework for the encoded signal allows for a better understanding of the distribution of the code-vectors throughout the dictionary space. Previous works demonstrated the tendency for these vectors to cluster around a thin shell of constant L^{α} norm [74]. This brings an intuitive understanding of why the MMP dictionary adaptation process is more successful at "learning" these clustered patterns, than the scattered blocks that exist in the original image.

Also, this points to some interesting future research, based on previous studies related to computationally efficient ways to encode such signals [78]. Finally, new, more efficient ways to explore the dictionary adaptation may be found by using the results of this study. The development of such techniques is the main subject of the next chapter.

Chapter 6

Efficient dictionary adaptation

The performance gains of MMP-I result from the transformation of the input image samples into a prediction error signal. The probability distribution of this signal was shown to favour the performance of the MMP algorithm, because it leads to a more efficient use of the dictionary patterns. In this sense, the use of predictive coding in MMP-I may be looked upon as a dictionary adaptation process.

Nevertheless, the use of the predicted error does not correct the vulnerabilities of the dictionary updating process. Because MMP-I inherits the same procedures used by MMP, it suffers from the same inefficiency problems that were previously described in Chapter 3: on one hand, experimental observations revealed an over populated dictionary, with a lot of unnecessary blocks that increase the entropy of the index symbols; on the other hand, the introduction of new symbols in the dictionary provides a richer set of patterns and increases the probability of successful block matches, that enhance the coding performance. In this chapter we tackle these important issues, in order to explore new ways to improve the efficiency of the MMP-I encoder. Our investigation resulted in new techniques that improve several aspects of MMP-I dictionary adaptation and usage, namely:

- Section 6.1 describes the use of context conditioning to improve the efficiency of arithmetic encoding of the dictionaries' indexes;
- Section 6.2 presents a discussion on redundancy control for the dictionary's elements;
- Section 6.3 studies ways to increase the approximation power of the dictionary, by using additional patterns;

- Section 6.4 presents two further methods that limit the inclusion of unnecessary elements in the dictionary;

- Section 6.5 presents the use of norm equalisation to adjust the scaled patterns to the signals' distribution;

We generally refer to the combination of MMP-I with the new dictionary adaptation techniques as MMP-II, from MMP-I with *Improved dictionary adaptation.*

After discussing each of the previous techniques, Section 6.6 presents a brief discussion on the computational complexity of MMP-based methods. While experimental results are presented individually for each of the proposed techniques in the corresponding sections, Section 6.7 presents the results of their combined use. These results reveal consistent improvements in the encoder's performance.

6.1 Dictionary partitioning

As for MMP, the analysis of the MMP-I bit stream revealed that the rate used to compress the segmentation flags is generally close to 15% of the total bit rate. For MMP-I, one has also to consider the additional use of prediction mode flags. Figure 6.1 shows the relative percentage of the total bit rate that is used by each component of the MMP-I bit stream. Figure 6.1 a) compares the percentage of bit rate used by segmentation flags with that used for the remaining data, composed by the dictionary indexes plus the prediction mode flags. Comparing these data with Figure 3.8 it is possible to observe the similarities with the MMP case. Figure 6.1 b) independently represents the percentage of bit rate allocated to the dictionary indexes and compares it to the rate used by the prediction mode flags. Two interesting observations result from these plots:

- The relative importance of the joint bit rates for the indexes and prediction mode flags is approximately constant for all tested images at all target compression ratios;

- The percentage of bit rate used for the prediction data is larger for very high compression ratios, meaning that for these cases the MMP-I encoder tends to rely more on the prediction process to approximate the input image. However, as the compression ratio decreases, MMP-I progressively uses more dictionary blocks, in order to achieve a smaller distortion in the representation of the encoded image.

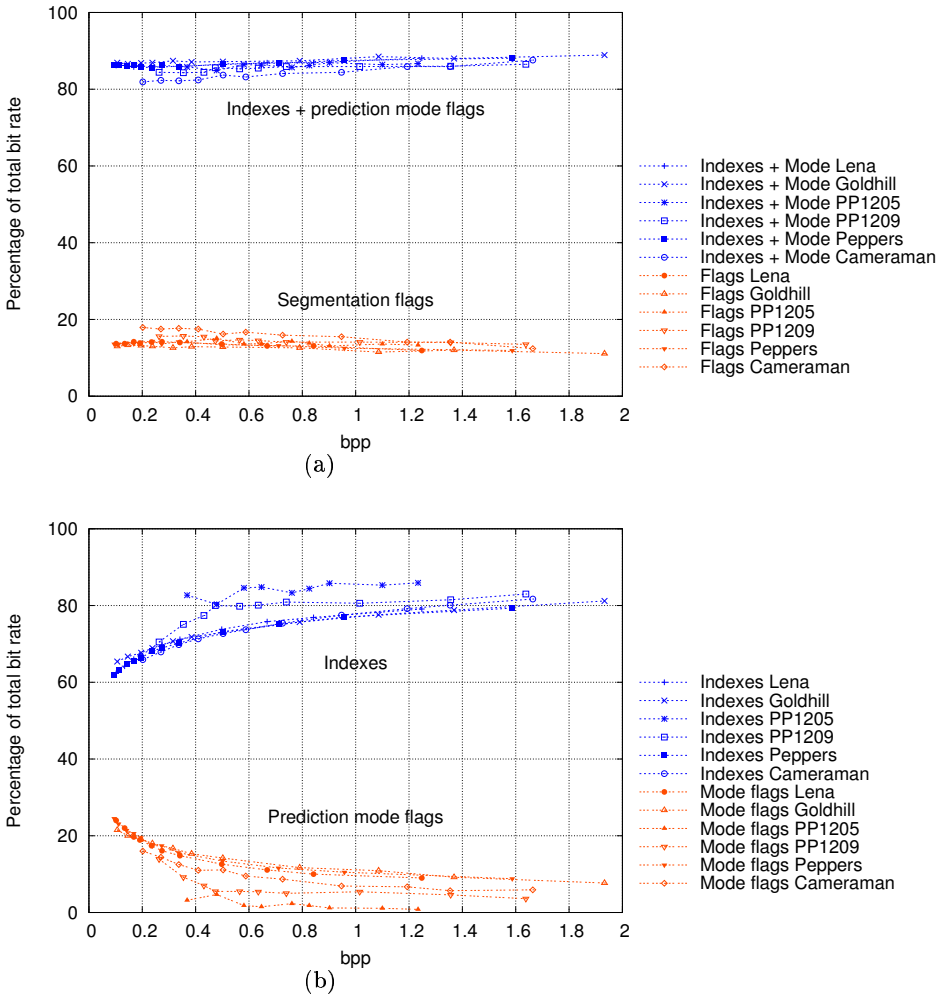


Figure 6.1: Bit rate usage for MMP-I: a) percentage of bit rate used by the indexes combined with prediction mode flags *vs* segmentation flags; b) percentage of bit rate used by the indexes *vs* prediction mode flags.

Figure 6.1 reveals that a significant part of the total bit rate is allocated to the transmission of the dictionary indexes, that use up to 80% of the total rate. This observation led to the study of more efficient ways to encode these indexes. Originally, a context adaptive arithmetic encoder was employed, that used the symbol's scale as a context, to losslessly compress the code-vectors' indexes, as well as the segmentation and prediction mode symbols. This context definition is implicit for the decoder, so no side information is required.

The use of a context conditioning scheme for the arithmetic encoder of the dictionary indexes was first proposed for MMP in [79]. This concept explores the statistical distribu-

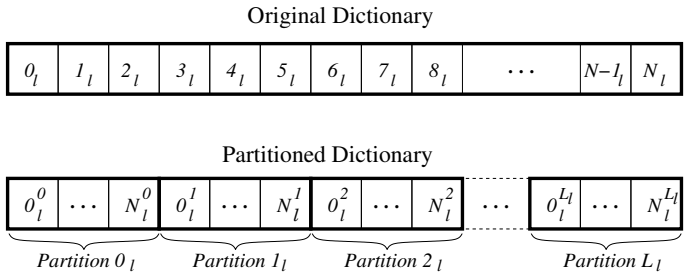


Figure 6.2: Division of dictionary $\mathcal{D}^l$ into $L_l + 1$ partitions.

tion of the indexes' symbols for each dictionary $\mathcal{D}^l$, by assigning each pattern to a given partition, according to the original scale of the block. Independent probability contexts for arithmetic encoding are associated to each dictionary partition. This means that, instead of assigning sequential indexes to all available patterns of $\mathcal{D}^l$, each block is classified according to the dictionary partition it belongs to, c_l, and an identifier that points to the relevant element inside the partition, $i_j^{c_l}$. This process, represented in Figure 6.2, requires the transmission of two symbols per block, instead of just one. Although this may at first seem to introduce an additional overhead, it may lead to gains through the exploitation of the statistical distributions of these symbols. This is because, depending on the criteria used to partition the dictionaries, the entropy of the symbol used to identify the context may be much lower than $\lceil \log_2(N_P) \rceil$ (where N_P is the number of partitions). Thus, by using an adaptive arithmetic coder to encode the symbols c_l and $i_j^{c_l}$, a decrease in the overall rate can be achieved. One may argue that the entropy of a dictionary word is the same irrespective of the number of steps in which it is encoded. However, if the criteria used to partition the dictionaries are conveniently chosen, then an adaptive arithmetic coder could reach the entropy of the sources at a much faster rate.

When testing this method with MMP-II, two context criteria were investigated, namely: the prediction mode that is being used for the current block and the scale at which the block was created. In the first case, each dictionary partition contains the vectors that were added to $\mathcal{D}^l$ when a specific prediction mode was used. In the second case, each dictionary partition contains the vectors that were originally created at a given scale l_o (by the concatenation of two blocks of scale $l_o - 1$). Both criteria favour the frequent use of a given set of blocks, when a particular coding condition is verified (either a particular prediction mode or current block scale). The knowledge of a particular state (encoding

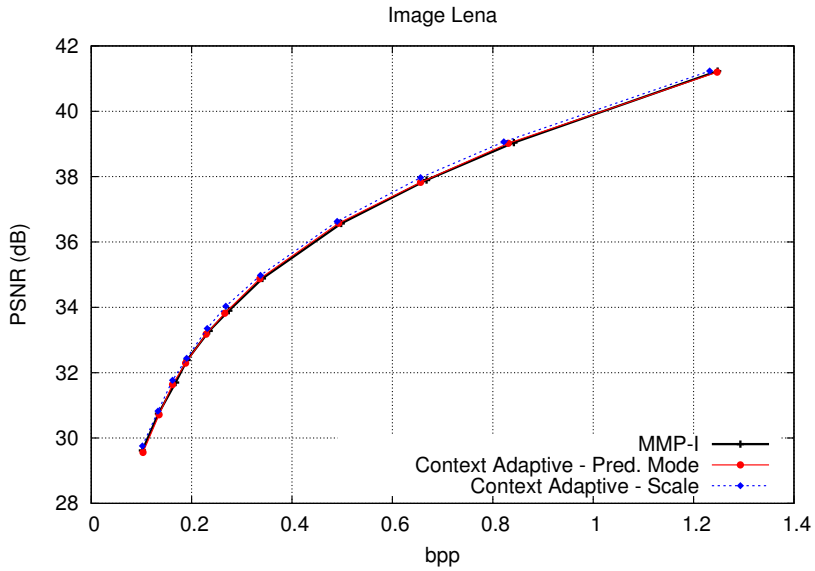


Figure 6.3: Results for the use of context conditioning with MMP-I, for image Lena.

context) is then used by the arithmetic encoder, that adapts the probability histograms to take into account this information and generate a more efficient code for the symbol that is currently being encoded.

Figures 6.3 and 6.4 compare the results of the original MMP-I with those attained when context adaptive dictionary partitioning is used with both discussed criteria[1]. We may observe that both schemes achieve a consistent improvement in RD performance over the original algorithm. Nevertheless, the use of the block's original scale as the coding context produces better overall results than the use of the prediction mode.

6.2 Dictionary redundancy control

The discussion presented in Chapter 3 shows that the existence of a large number of blocks in each dictionary $\mathcal{D}^l$ has the unfavourable effect of increasing the average entropy of the index symbols. A similar experimental study, performed for MMP-I, led to identical conclusions, despite a marginal reduction of this effect. Figure 6.5 shows the same linear relation between the cardinality of the MMP-I dictionaries and the target compression ratio that was observed for MMP (see Figure 3.9).

Figure 6.6 compares the final number of elements for $\mathcal{D}^8$ for MMP-I and MMP. A

[1] The results for other test images are presented in Figure C.19 of Appendix C.

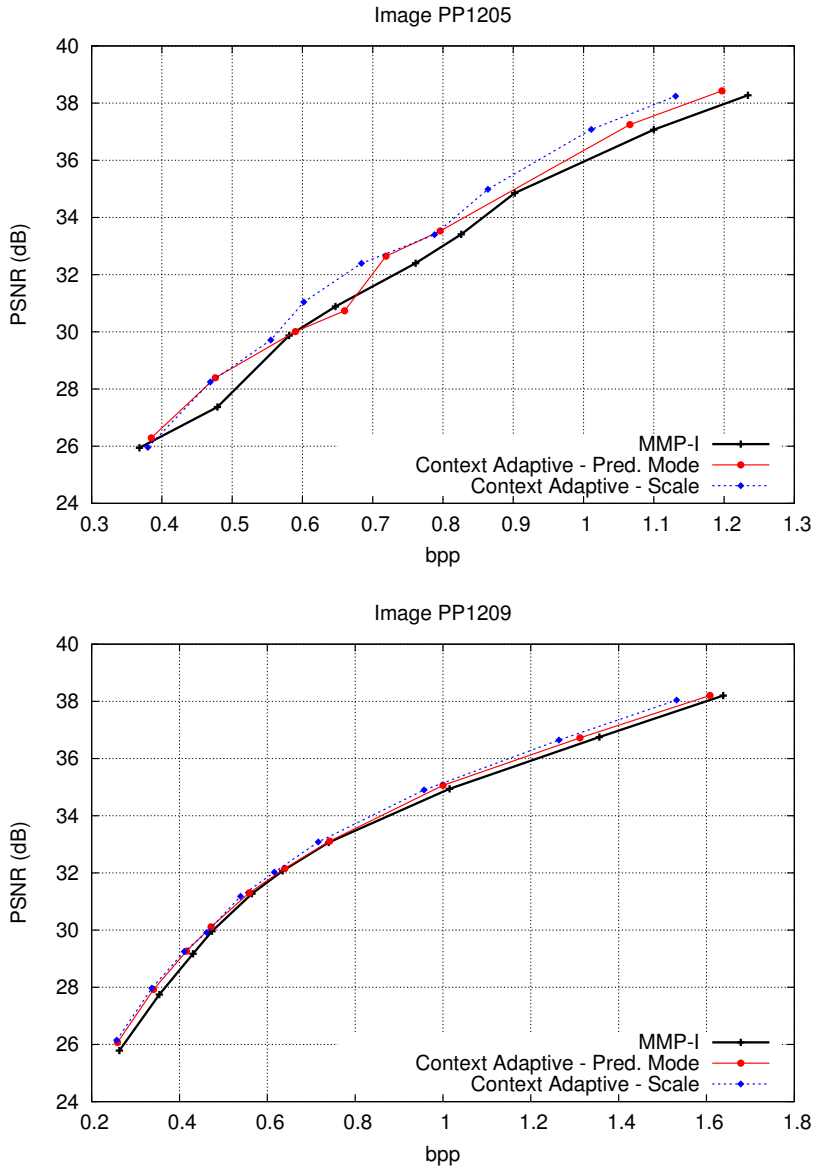


Figure 6.4: Results for the use of context conditioning with MMP-I, for images PP1205 and PP1209.

smaller growth of the MMP-I dictionaries is notorious. This observation validates once again the conclusion that the predictive techniques employed by MMP-I increase the efficiency of the dictionary usage process. We can also observe that the most significant differences exist in the plots of the smoother images, for which the prediction process has a greater efficiency. In fact, for text image PP1205 the dictionary sizes are approximately identical, as are the final performances for MMP-I and MMP, due to the low efficiency of the prediction process. In spite of the gains observed in these figures, the final cardinality

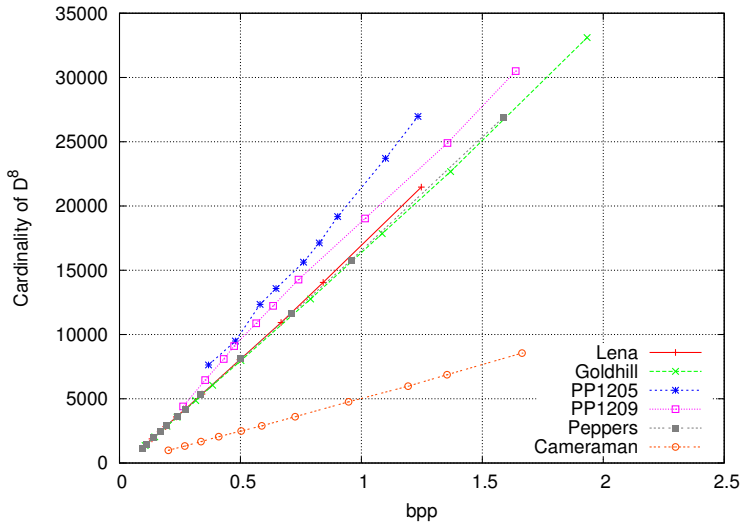


Figure 6.5: Final number of elements in dictionary $\mathcal{D}^8$ (16×16 blocks) for MMP-I.

of the MMP-I dictionaries is still much larger than the actual number of elements that is used in the encoding process. As for MMP, the existence of dictionary vectors that end up being useless for the approximation of image patterns compromises the performance of the encoder and should be avoided. This section deals with enhancing the performance of the MMP-I encoder by selectively eliminating blocks from the dictionaries.

Section 6.2.1 investigates the use of independent dictionaries, while Section 6.2.2 presents an efficient redundancy control strategy for the insertion of new vectors, based on a distance measure relative to the existing dictionary blocks.

6.2.1 Multiple dictionaries

The MMP-I dictionary may be regarded as a melting pot where all blocks are included, so that they can be used in a future approximation. In Section 3.4.1 we observed that reducing the number of dictionary elements (*i.e.* the size of the pot) compromises the efficiency of the method, because one ends up excluding patterns that would be favourable in future matches. In this section we propose the use of several independent dictionaries (the pots in our metaphor) that store unassociated sets of words. At each given moment, only one of the smaller dictionaries is used, both as a source for approximating patterns and as a destination where new blocks are stored.

This test was inspired by the observation of the used prediction modes, most of which use directional prediction. We hypothesised that the predicted residue blocks used by each

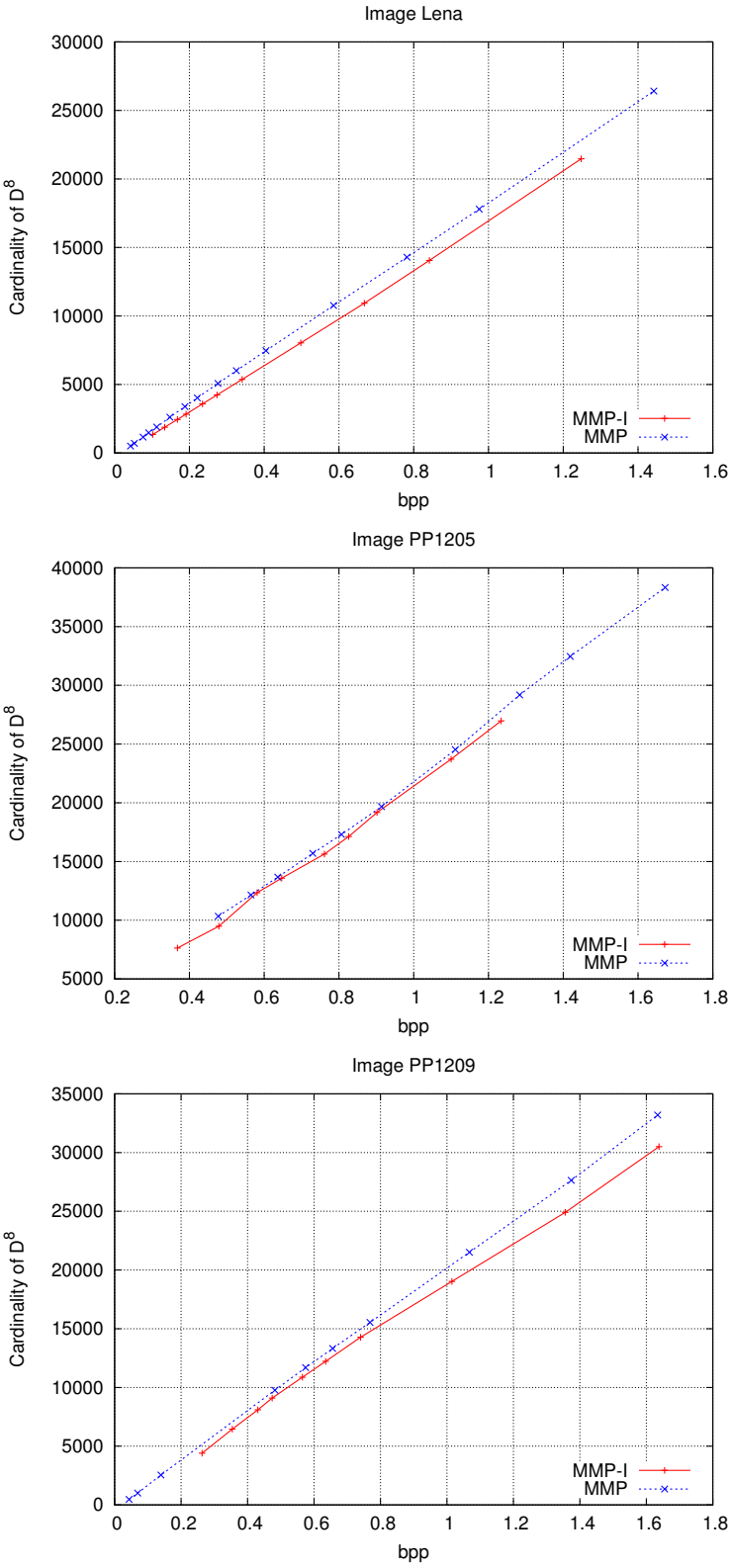


Figure 6.6: Comparison between final dictionary size of scale 8 for MMP-I and MMP, for images Lena, PP1205 and PP1209.

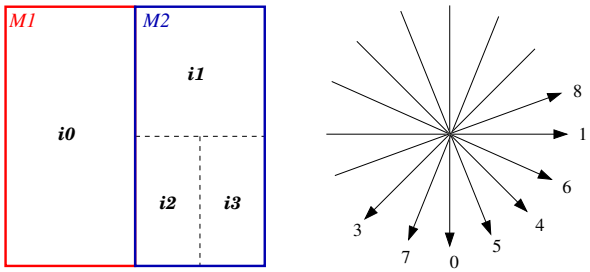


Figure 6.7: a) Example of block prediction using two modes; b) the directional prediction modes

mode should have particular features, associated to the direction used in the prediction process, *e.g.*, the residue blocks generated using horizontal prediction should be highly related, and different from the ones used by vertical prediction. The original MMP-I dictionary mixes all these patterns, meaning that, at all times, all of the patterns contribute to increasing the symbols' entropy. In this test we have created an independent dictionary for each prediction mode. At each moment, a residue block that has been determined using a predictive mode M may only be approximated by vectors from codebook $\mathcal{D}_M^l$, that is composed only by those blocks that have been previously created while encoding residues determined with prediction mode M. As an advantage, each dictionary $\mathcal{D}_m^l$ will be composed by a set of patterns which would be more correlated than the ones of the original dictionary $\mathcal{D}^l$, because they would have all been determined by using the same prediction mode. These vectors would also be correlated with all future blocks that use the same prediction mode, meaning that the approximation power would not be compromised. Besides this, each $\mathcal{D}_m^l$ would have a smaller size, which would favour the performance of the adaptive arithmetic encoder. A potential drawback would be that the restriction of the available patterns might cause a reduction on the coding efficiency. In fact, a given residue could be better approximated by a pattern generated by a different prediction mode. The algorithm's adaptability assures that, in the worst case scenario, the required pattern can also become available at the dictionary for the current prediction mode, but this is achieved by using the concatenation of smaller blocks from the current dictionary, at the cost of having to re-transmit all the segmentation and coding data that was previously used to generate the pattern at the other dictionary.

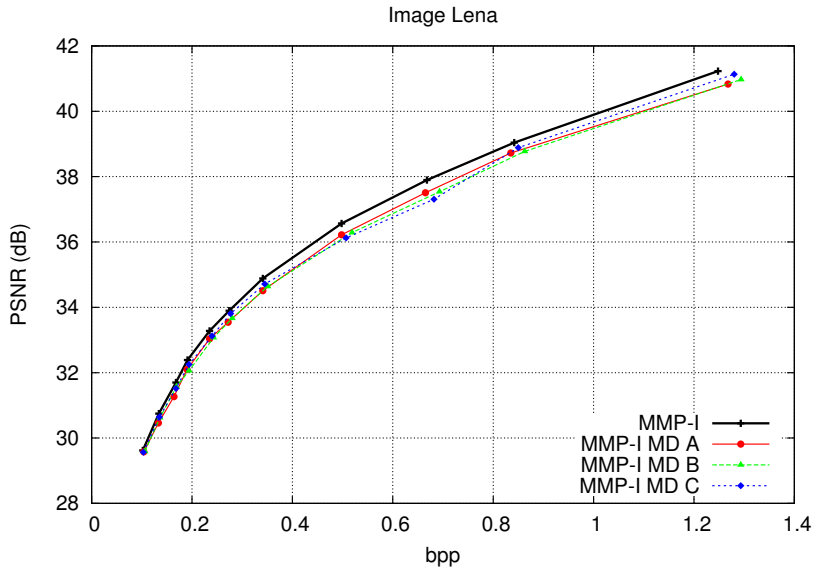


Figure 6.8: Results for original MMP-I and new techniques that use several independent dictionaries (MMP-I MD), for image Lena.

We conducted a set of experimental tests in order to investigate this technique. Some interesting issues were brought forth by this work, namely concerning the configuration of the several independent dictionaries. When an image block is predicted using two different modes, for instance modes M_1 and M_2 represented in Figure 6.7 a), some factors must be considered related with the dictionary updating process. When using independent dictionaries for each prediction mode, each block is approximated using only patterns from the corresponding dictionary. In the example, index i_0 belongs to dictionary $\mathcal{D}_{M_1}$ and indexes i_1 to i_3 are chosen among the elements of dictionary $\mathcal{D}_{M_2}$. The updating process related with the coding of each block half poses no problem: all new patterns created for each half have a single dictionary associated with them. Nevertheless one must tackle the issue of dictionary updating for the concatenation of the two segments that used different prediction modes. Several options were tested:

- When we discarded the new patterns that did not correspond to a specific prediction mode, we observed severe quality losses, because the dictionary updating process is compromised;

- We then used the new pattern to update both $\mathcal{D}_{M_1}$ and $\mathcal{D}_{M_2}$, in a test that will be referred to in this section as MMP-I MD A;

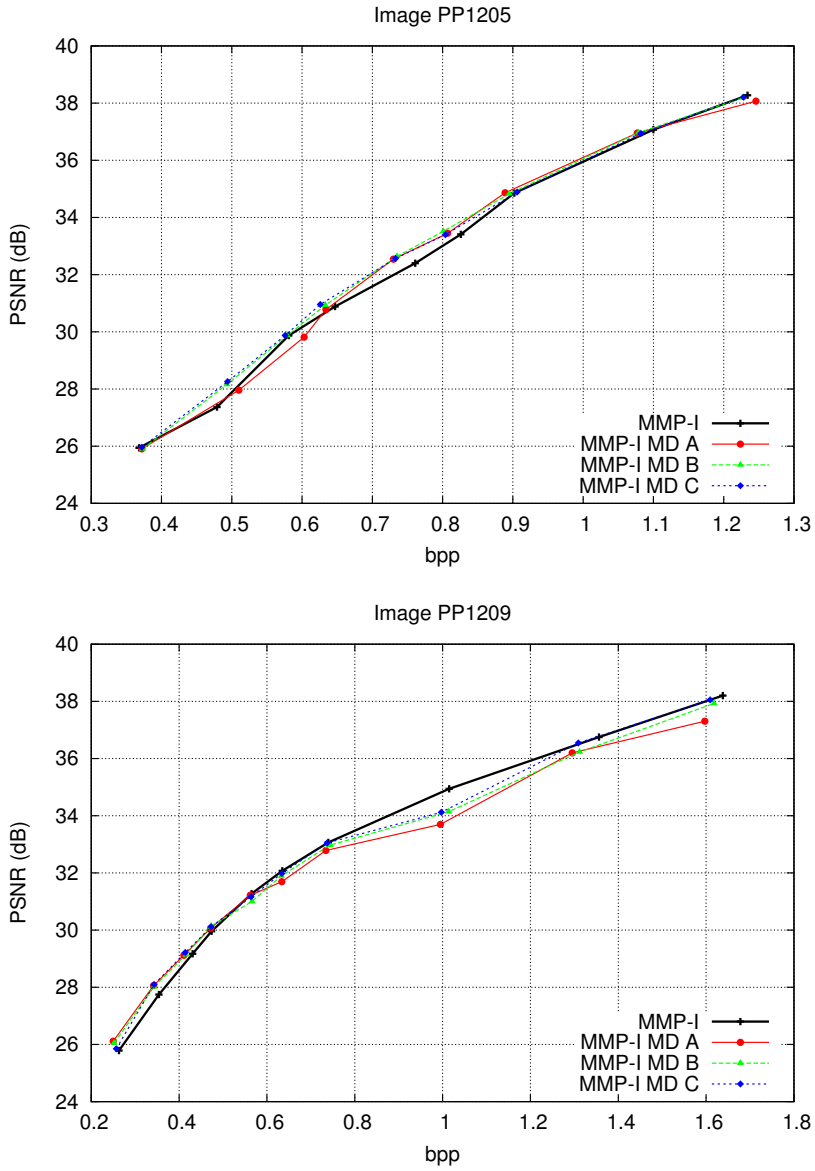


Figure 6.9: Results for original MMP-I and new techniques that use several independent dictionaries (MMP-I MD), for images PP1205 and PP1209.

- Because more than two prediction modes may be used within a single block, a new method was implemented, where not only $\mathcal{D}_{M_1}$ and $\mathcal{D}_{M_2}$ are updated but also the dictionaries related with all the prediction modes that have been previously used in the current block (MMP-I MD B);

- In all these tests a dictionary is used for each prediction mode. In a final test (MMP-I MD C) we associated several prediction modes (related by their similar orientations)

to one single dictionary: independent dictionaries were used for the Vertical mode (mode 0 in Figure 6.7 b)), the Horizontal mode (mode 1), the MFV mode and the Plane 16×16 mode; a fifth dictionary was used for modes 3, 7 and 8 while a sixth dictionary was assigned to modes 4 to 6.

The experimental results for these techniques, presented in Figures 6.8 and 6.9[2], demonstrate that they do not achieve the best overall performance for all image types. Some configurations are favourable for text image PP1205 and also, for some cases, for compound image PP1209, when compared with the original MMP-I. Nevertheless, these results show that this technique is disadvantageous for smooth images. This may be explained by the fact that the prediction process for these cases tends to generate similar patterns, even when different prediction modes are used. The highly structured patterns of the text and compound images tend to either consistently use a reduced set of prediction modes or generate different residues for different prediction modes. Both these factors allow the use of independent dictionaries to cause an increase in the algorithm's performance in these cases. In spite of the interesting gains observed for the non-smooth images, the losses observed for smooth images, that are prioritised in this work due to the comparatively worse performance of MMP-I, compromise the use of independent dictionaries with MMP-I. Nevertheless, future developments of this topic may be worth investigating, namely in text or compound image coding oriented applications.

6.2.2 Avoiding the inclusion of redundant elements

All dictionary limiting techniques that were previously investigated led to some inefficiencies in the dictionary adaptation process. When a restriction on the maximum dictionary size is imposed, one is unable to conveniently define a rule to eliminate appropriate blocks in the dictionary (see Section 3.4.1). The use of independent dictionaries, described in the previous section, restricts the blocks that are available for future approximations and introduces some losses for the compression of smooth images. Because of this, a new technique was investigated that uses a different paradigm: avoid the inclusion of redundant vectors, that do not introduce a significant gain in the approximation power of the dictionary, *i.e.* the ability to represent the image patterns with a low distortion. In general terms, this is achieved by avoiding the inclusion of a new element that is too close to a vector already

[2]The results for other test images are presented in Figure C.20 of Appendix C.

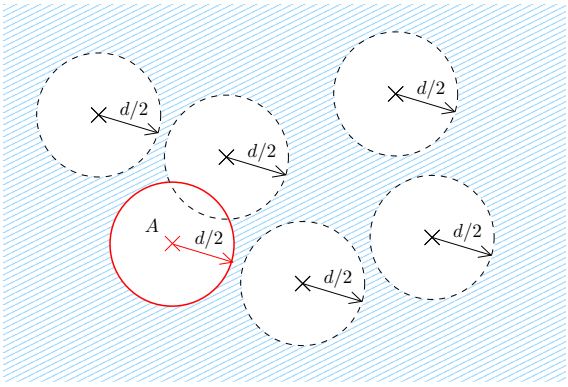


Figure 6.10: Use of a minimum distance condition in the dictionary update procedure. In this example, block A is not inserted in the dictionary.

present in the dictionary. This is done by testing each new block before inserting it in the corresponding dictionary. Each new block $\hat{\mathbf{X}}^l$, of scale l, is only used to update the dictionary $\mathcal{D}^l$ if the distortion between $\hat{\mathbf{X}}^l$ and any block of $\mathcal{D}^l$ is not inferior to a given threshold d^2. In other words, a new block is only inserted in the dictionary if, for every element $\mathbf{S}_i^l$ of dictionary $\mathcal{D}^l$,

$$\sum_{m,n} \left(\hat{\mathbf{X}}^l(m,n) - \mathbf{S}_i^l(m,n) \right)^2 > d^2. \tag{6.1}$$

This introduces a minimum distortion condition between any two vectors of each scale of the dictionary. In fact, this guarantees that, for each scale l, the distance between two or more blocks is at least d, or equivalently, that there is never more than one block inside any hypersphere of radius d in the l-dimensional space. This process is represented in Figure 6.10 for the two-dimensional case.

The threshold value, d, controls the redundancy among the elements of the dictionary and must be carefully determined. If the hyperspheres' radius is too small, the dictionary vectors will be too close to each other, and the aim of reducing redundancy will not be achieved. On the other hand, if d it is too large, the dictionary space will have large void areas, corresponding to patterns that will not be accurately represented, leading to a decrease in the algorithm's performance.

In the case of MMP-I, we can intuitively understand that the optimum value for d is a function of the target bit rate: for lower bits rates the distortion level will be higher, which corresponds to larger distances between the elements of the dictionary, that will have fewer elements. For higher rates, the code-vectors should be closer, providing a more

accurate representation of the image patterns. The optimum value of d will then depend on the target bit rate for the image. The final compression ratio is not available at the beginning of the encoding process, but it is related to the parameter λ, used in the RD control algorithm. Because of this, an heuristic relating d to λ was determined. In order to achieve this relation, we have encoded a set of test images using different pairs (d, λ). We then plotted the corresponding values of rate and distortion on an RD chart and determined its convex hull, that is composed by the points that minimise the value of the function $D(R)$. When we considered these points for a representative set of images, we were able to determine a simple model for the function $d(\lambda)$:

$$d(\lambda) = \begin{cases} 5, & \text{if} \quad \lambda \leq 15; \\ 10, & \text{if} \quad 15 < \lambda \leq 50; \\ 20, & \text{otherwise.} \end{cases} \tag{6.2}$$

Using equation (6.2), the encoder is able to determine the value of d, given the input parameter λ. Because λ is not available at the decoder, the value of d must be transmitted, by using a (negligible) two bits overhead. The decoder is thus able to replicate the dictionary updating procedure used by the encoder. Equation (6.2) is the result of a compromise among all used test images, and it almost achieves the best result for all images. From this one can conclude that the optimum relation $d(\lambda)$ does not strongly depend on the type of input image.

As was previously mentioned, the distortion control is made using a fixed squared distortion for the blocks at all levels of the dictionary (see equation (6.1)). The radius of the hypersphere does not vary with the block size, meaning that the maximum allowed distortion per pixel for the dictionary blocks decreases exponentially with the increase in dictionary scale. As a consequence, the effects of this process will be much more noticeable for the smaller scales of the dictionary, because they are less likely to be updated. Also, the cardinality of larger scale dictionaries will tend to increase faster than the ones of smaller scales. This effect may be observed in Figure 6.11, where the final number of elements for all scales of the dictionary for image Lena is presented, both for the original MMP-I encoder and MMP-I combined with the redundancy control algorithm. In this figure it is possible to observe a large reduction on the final number of elements of the dictionaries for smaller scales (4×4 and below). Nevertheless, there is no reduction in the number of

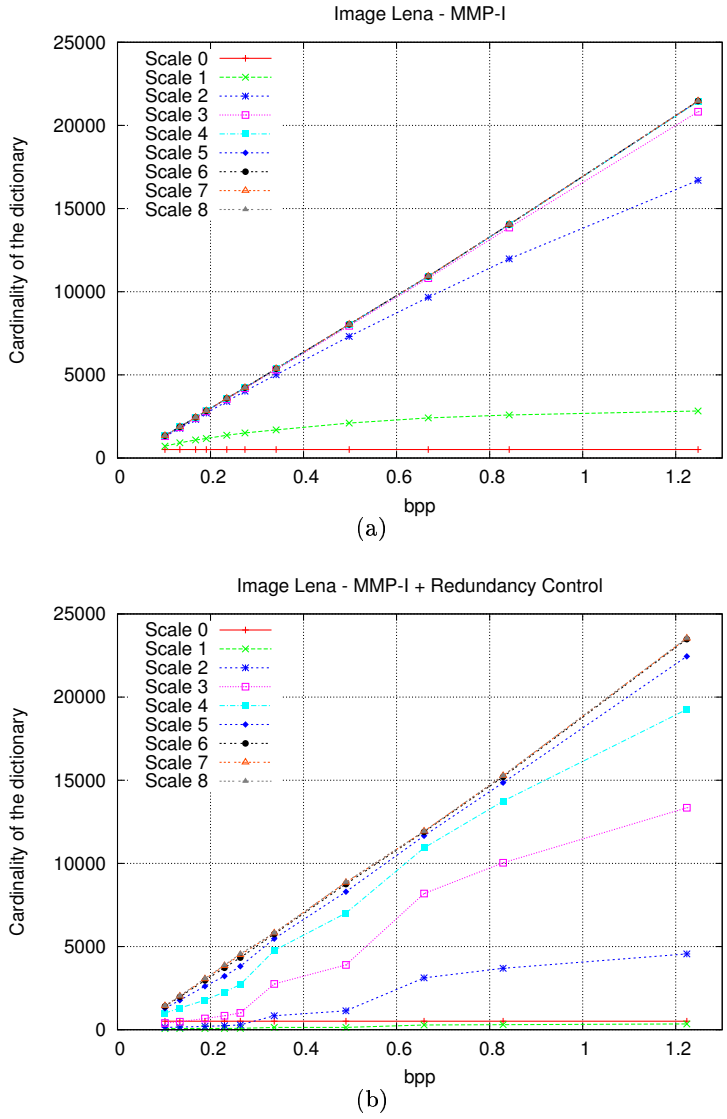


Figure 6.11: Final number of elements for the dictionaries of each scale *vs.* the compression ratio, for image Lena: a) MMP-I; b) MMP-I with Redundancy Control.

elements for the higher scales. In fact, there is a slight increase in the number of elements for these scales.

Alternative ways of controlling the minimum distance between blocks were considered and experimental tests were conducted in order to investigate both the comparative performances and the effects of each method is the dictionary growth. The tests included, among others, the cases for which the average distortion per pixel was fixed or for which the sphere radius increases linearly or exponentially with the value of l. However, the best

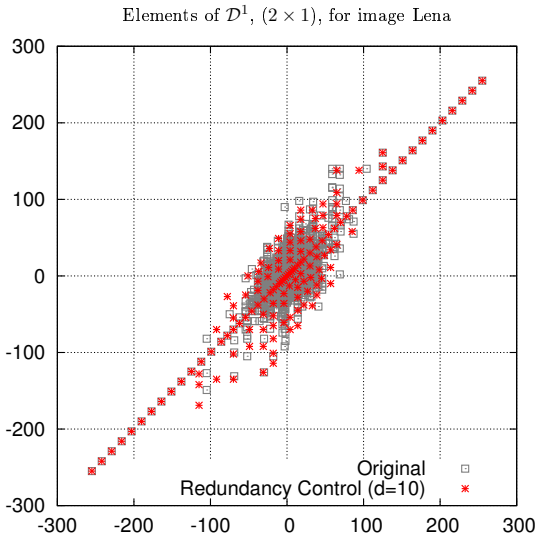


Figure 6.12: 2D plot of the 2×1 code-vectors of dictionary $\mathcal{D}^1$, for the original MMP-I (2045 elements) and MMP-I with redundancy control (147 elements), for image Lena encoded at approximately 0.5 bpp.

results were achieved when equation (6.1) was used, because, from a RD point of view, it is advantageous to have greatly populated dictionaries for the large scale dictionaries, while performing a severe redundancy reduction for the smaller scales. This is so because in MMP methods, larger blocks provide a very efficient way of encoding image pixels, while the use of smaller scales implies higher bit rate cost. Furthermore, MMP-I, as MMP, tends to use mostly blocks of the smaller scales (block sizes from 2×2 to 4×4 pixels). Therefore, the reduction of the redundancy of the dictionary elements for these scales contributes more to increase the coding efficiency. On the other hand, having richer, highly populated dictionaries for the larger scales increases the chances of a successful match, which also benefits compression performance, since it allows for the encoding of a large number of pixels using one single code-vector.

Figure 6.12 shows the effect of using of redundancy control scheme on the dictionary of scale 1 (2×1 pixel blocks), for image Lena encoded at 0.49 bpp. A 2D plot of the 2×1 code-vectors of the dictionary $\mathcal{D}^1$ is presented, both for the original MMP-I encoder and for an MMP-I encoder that uses redundancy control, with $d = 10$. The reduction on the total number of code-vectors is clear (from 2045 to 147), but it can also be observed that the code-vectors that are maintained by the redundancy reduction algorithm cover the same space as the original dictionary, but with a much smaller density. This means that

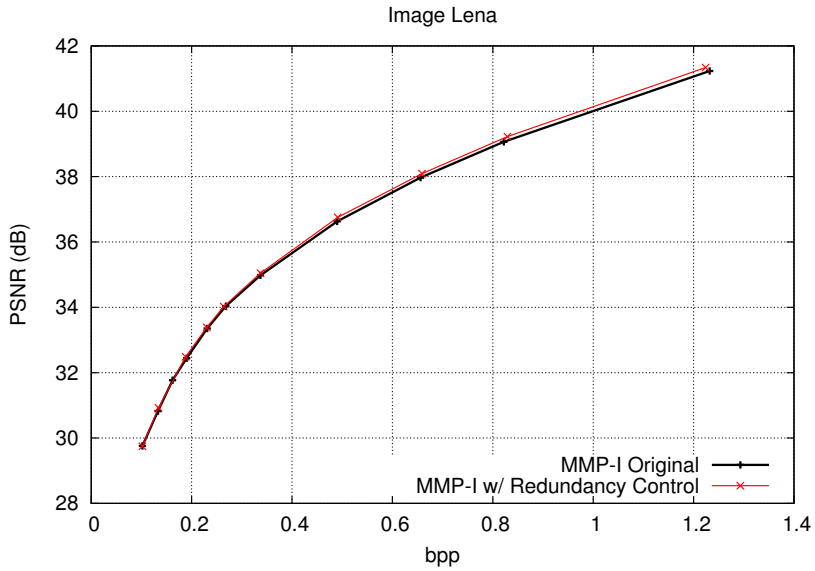


Figure 6.13: Results for original MMP-I and MMP-I using redundancy control (both methods also use dictionary partitioning), for image Lena.

the relevant patterns for encoding the input signal are not eliminated from the dictionary, but one avoids overpopulating the dictionary space with redundant code-vectors.

Figures 6.13 and 6.14[3] compare the performance of the MMP-I algorithm when the original dictionary updating procedure is used and when the new redundancy control technique is applied (both encoders also use the dictionary partitioning technique described in Section 6.1). We observe that the proposed technique consistently increases the RD performance of MMP-I encoder for all tested images and all target compression ratios. These results demonstrate the efficiency of the new redundancy reduction algorithm, that is able to filter the patterns that introduce a negligible gain in terms of approximation power but generate an unfavourable penalty in the overall entropy of the index symbols.

6.3 Improving dictionary approximation power

The previous section described a set of techniques that attempt to improve the overall performance of the dictionary adaptation process by removing inefficient patterns. If the code-vector exclusion is performed is a way that does not compromise the dictionary's ability to produce good approximations for future image blocks, then a rate reduction is

[3]The results for other test images are presented in Figure C.21 of Appendix C.

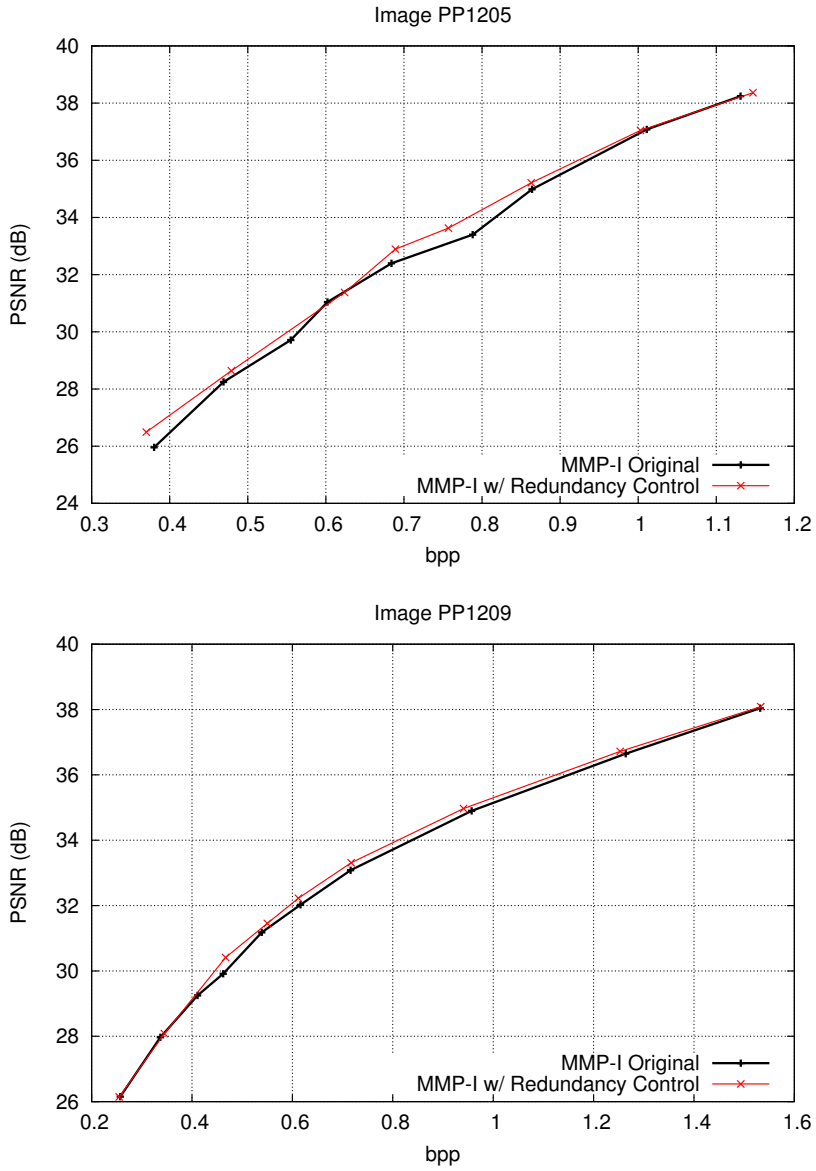


Figure 6.14: Results for original MMP-I and MMP-I using redundancy control (both methods also use dictionary partitioning), for images PP1205 and PP1209.

achieved with no distortion penalty, and the overall performance of the encoder is increased. In this section we investigate a new paradigm: the insertion of additional vectors in the dictionary, in order to increase its approximation power, *i.e.* the ability to achieve a low distortion value when approximating image patterns.

Adaptive approximate pattern matching methods, like MMP and MMP-I, explore the self similitude properties of digital images. They rely on the supposition that each block used to approximate one region of an image has a fair probability of being useful in approxi-

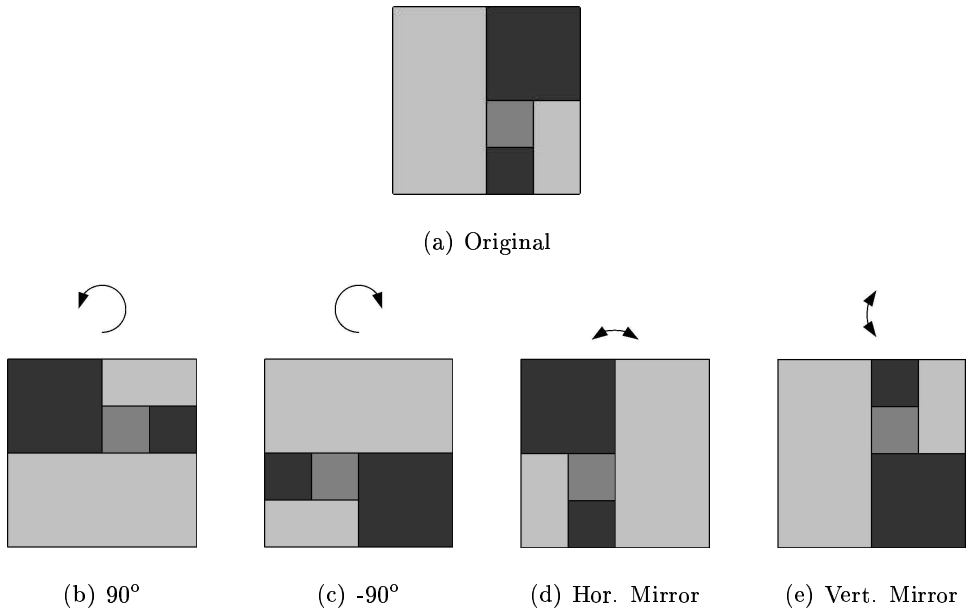
mating other regions of the same image. The MMP-based methods expand this concept for blocks of different dimensions, through the use of scale transformations. The "multiscale" feature means that each new block is inserted in all scales of the dictionary, providing it with a richer set of patterns. In this section we discuss the use of other updating strategies, that enhance the dictionary approximation power by inserting more than one vector per approximation.

When using such updating strategies, one would expect the cardinality of the dictionary to increase at a much faster rate. An increase in cardinality implies an increase both in rate and computational complexity, *i.e.* an opposite effect to that achieved by the redundancy control techniques. Since this may be unfavourable in terms of the overall performance, the new blocks used in the updating procedure must be carefully chosen. In this section we show that, by using carefully chosen updating strategies and combining them with the previous described redundancy control methods, this new updating procedure increases the efficiency of the dictionary adaptation process.

MMP-I uses scaled versions and concatenations of the approximated residue blocks, $\hat{\mathbf{R}}^l_M(m,n)$ to update the dictionary. In this section we consider some methods for generating new patterns for the dictionary updating step, other than just the scaled versions of $\hat{\mathbf{R}}^l_M(m,n)$. Some of these strategies have been previously proposed for the MMP encoder [18, 79], like the use of *geometric transforms* of the original block and *displaced* versions of the image residue. These methods were adapted to the new paradigm of MMP-I, namely the use of predicted residue blocks. Furthermore, novel updating strategies resulted from this work, like the use of the *additive symmetric* of the original vector. In the following sub-sections we describe individually each of the proposed updating methods and present their impact in the performance of the encoder. In Section 6.3.4 we evaluate the combined use of these techniques, in order to determine the most favourable methods to use for dictionary adaptation.

6.3.1 Geometric transforms

The prediction modes used in MMP-I are mainly directional modes, that try to predict patterns which occur in the image with different orientations. Spatial relations between these orientations can be easily observed, as has been previously discussed in Section 6.2.1. An obvious example is the connection between the horizontal and vertical predictions, that



(a) Original

(b) 90° (c) -90° (d) Hor. Mirror (e) Vert. Mirror

Figure 6.15: New patterns created by using geometric transformations.

may be related by the use of a 90^o rotation. To exploit this fact, geometric transforms are applied to $\hat{\mathbf{R}}_M^l$, originating four new versions of the original block, that are used in the dictionary updating process. The chosen transforms are two rotations, by 90^o and -90^o, and two mirroring operations, using horizontal and vertical axes (see Figure 6.15). The rotations transform the horizontal direction into the vertical one and *vice versa*. The mirroring operations relate some of the prediction directions of the third and fourth quadrants. The use of these four new blocks means that the new updating strategy may lead to the insertion of five times more patterns than the dictionary adaptation method. Based on preliminary experimental results for this method, the four blocks resulting from the combination of rotations and mirroring operations (*e.g.* the rotation of the block from Figure 6.15 d)) were not considered.

Figures 6.16 and 6.17[4] show the results of original MMP-I and MMP-I using geometric transforms in the dictionary updating process. Both versions also use the dictionary partitioning technique with scale contexts described in Section 6.1. The dictionary redundancy algorithm is nevertheless disabled, and its use will be independently evaluated in a future section. From Figure 6.16 we have that the use of extra patterns in the dictionary updating process marginally increases the performance of MMP-I for smooth images. In Figure

[4]The results for other test images are presented in Figure C.22 of Appendix C.

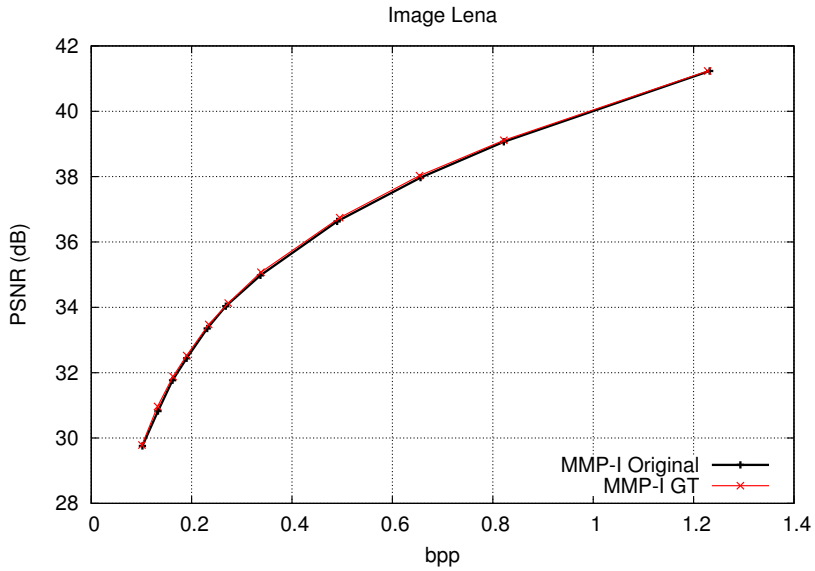


Figure 6.16: Results for MMP-I with the use of new dictionary updating with geometric transforms, for image Lena.

6.17 we may notice that for the text image the results are overall equivalent, despite some fluctuations that give a marginal gain for both techniques at specific compression ratios; for compound image, PP1209, one also notices a marginal performance gain, that results from the compression of the smooth areas of the image.

6.3.2 Displaced blocks

The use of displaced versions of the reconstructed residue block exploits the fact that the patterns in a digital image may occur at any position. This is relevant because the binary segmentation process used by MMP only allows for the exploitation of similar patterns that occur at some fixed image points. One example is the fact that any 8×8 block of the dictionary may only be used to match an image block with upper left corner pixel coordinates given by $(8i, 8j)$, where $i = 0, \ldots, N_{lines}/8$ and $j = 0, ..., N_{rows}/8$. To overcome this limitation, we have tested the use of displaced versions of the reconstructed block as a source for new dictionary patterns. This is achieved by using a sliding window that is displaced along the previously encoded area of the image. When one applies this to MMP, the sliding window originates regions of reconstructed image patterns with the same size as the original block, but located at neighbouring positions [18]. As for the case of the

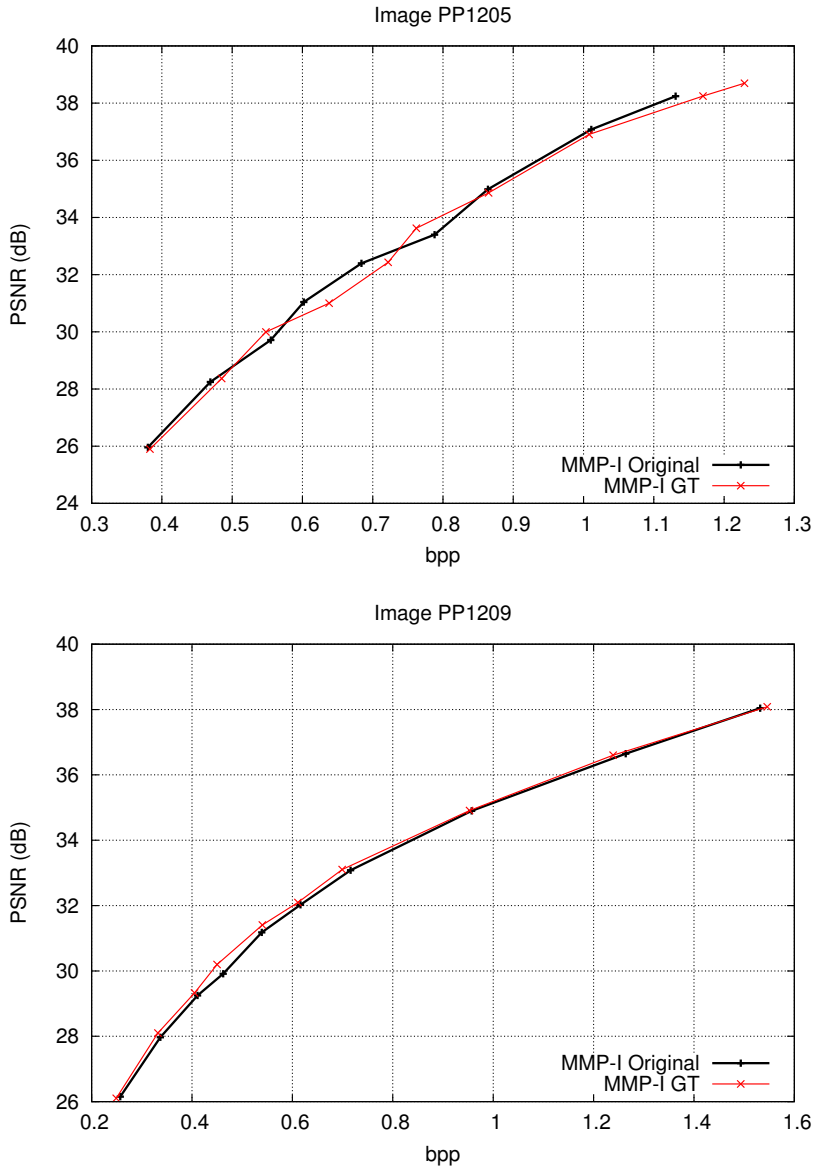
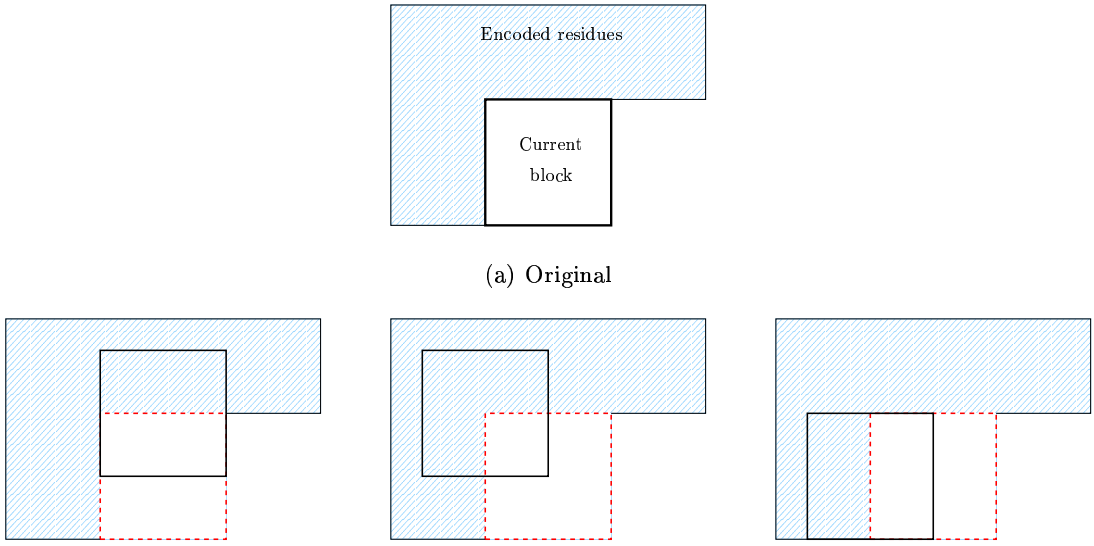


Figure 6.17: Results for MMP-I with the use of new dictionary updating with geometric transforms, for images PP1205 and PP1209.

geometric transforms, one must consider that, when using this method with MMP-I, the dictionary patterns correspond to predicted residues and not to the input image blocks, meaning that the sliding window must be used on the previously *reconstructed residues*.



(a) Original

(b) New block ($p = 0$, $q = 1$) (c) New block ($p = q = 1$) (d) New block ($p = 1$, $q = 0$)

Figure 6.18: New patterns created by using displaced versions of the original patterns, with $s = 2$.

This means that the new blocks used in the updating process are given by[5]:

$$\hat{\mathbf{R}}^l_{\delta_m,\delta_n}(m,n) = \hat{\mathbf{X}}^l(m-\delta_m, n-\delta_n) - \mathbf{P}^l(m-\delta_m, n-\delta_n), \tag{6.3}$$

where $\mathbf{P}(m,n)$ represents the prediction pixels used for pixel (m,n), using the corresponding prediction mode. In the previous equation one must bare in mind that the prediction mode for all possible positions $(m-\delta_m, n-\delta_n)$ may vary, because these pixels may have been predicted in different steps. Because of this, one has to keep a record of the prediction values that have been used within the search window.

In our tests we have used displacement steps that correspond to one half or one quarter of the block dimensions and maximum displacement values that have the same dimensions as the block $\hat{\mathbf{R}}^l_{\delta_m,\delta_n}(m,n)$. This means that, for a block of size $2^M \times 2^N$, we include in the dictionary all blocks

$$\hat{\mathbf{R}}^l_{p\delta_m,q\delta_n} = \mathbf{R}^l(m-p\delta_m, n-q\delta_n), \tag{6.4}$$

[5]In order to simplify the notation, we dropped the mode subscript form the residue and prediction blocks.

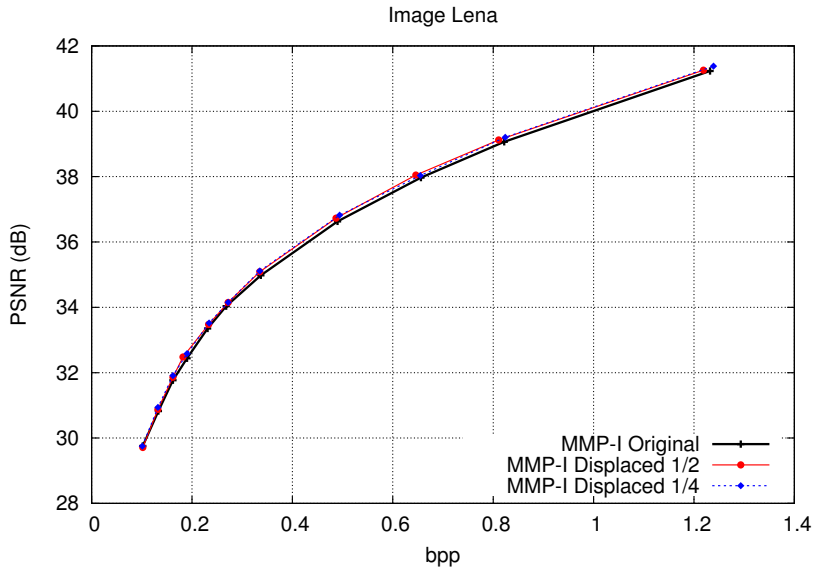


Figure 6.19: Results for MMP-I with the use of new dictionary updating with displaced blocks, for image Lena.

with

$$\begin{cases} p = 0, \ldots, s-1 \\ q = 0, \ldots, s-1 \\ \delta_m = \left\lfloor \frac{2^M}{s} \right\rfloor \\ \delta_n = \left\lfloor \frac{2^N}{s} \right\rfloor \\ s = 2, 4 \end{cases} \tag{6.5}$$

where s is the fraction of the block dimensions corresponding to the displacement steps, *e.g.*, $s = 2$ corresponds to displacements of half the block size in each dimension, whereas $s = 4$ corresponds to displacements of a quarter of the block size. Figure 6.18 represents this process for $s = 2$. The shaded area represents the previously encoded pixels of the residue image. The current block is represented, as well as the three positions that correspond to the residue areas which are also used to update the dictionary. For $s = 4$, each new block originates fifteen new other blocks, corresponding to all possible (δ_m, δ_n) positions defined by equations (6.4) and (6.5).

The experimental results for the described techniques are presented in Figures 6.19 and 6.20[6]. It is possible to observe that the use of displaced versions of the reconstructed

[6]The results for other test images are presented in Figure C.23 of Appendix C.

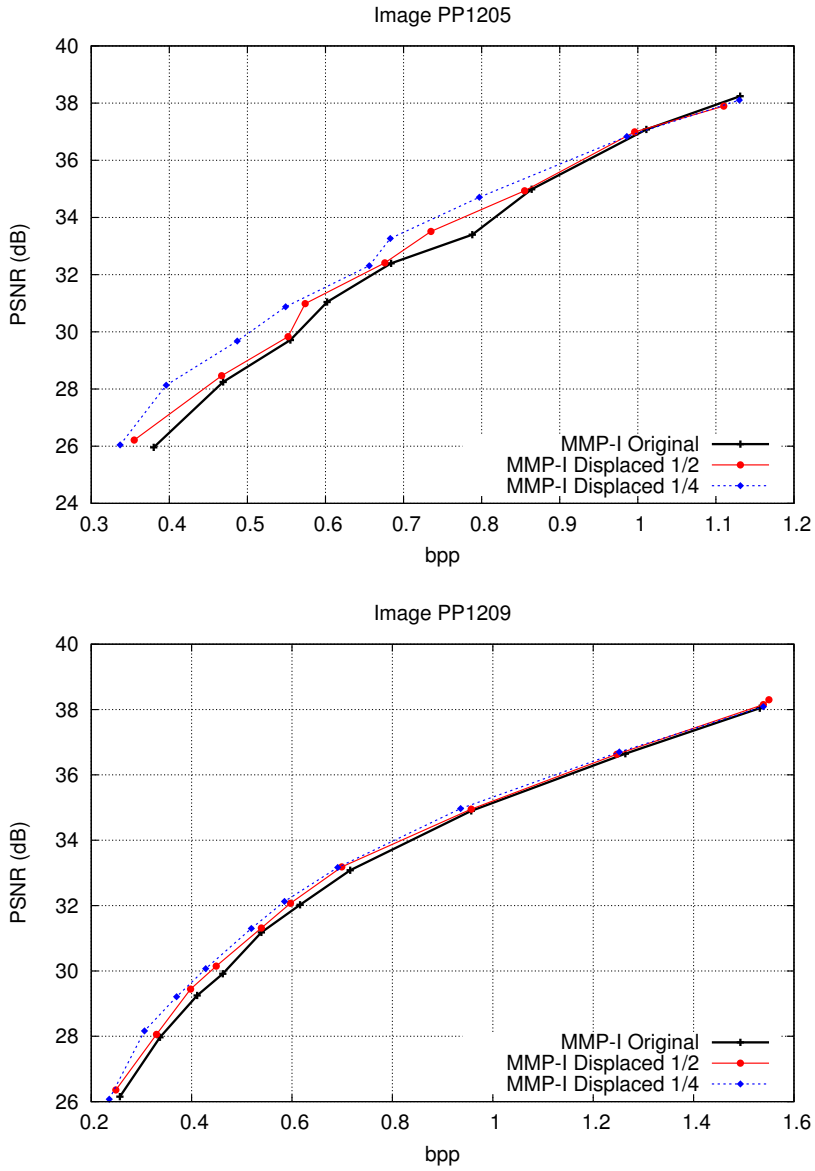
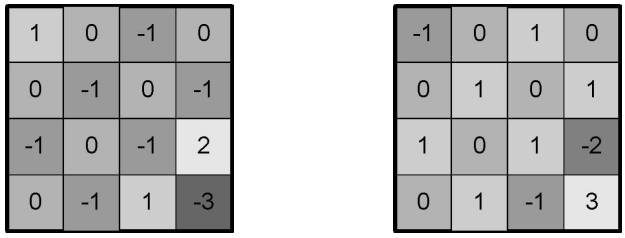


Figure 6.20: Results for MMP-I with the use of new dictionary updating with displaced blocks, for images PP1205 and PP1209.

residue patterns achieves performance gains for all tested images and all compression ratios. The gains are particularly relevant for the compression of text patterns, like those of image PP1205. This may be understood due to the structure of the compressed patterns, that have a lot of symbols (characters or character segments) often repeated, but not necessarily at the positions defined by the original MMP "grid", *i.e.* the positions that may be determined by binary segmentations of the original macroblock. The use of displaced versions allows the dictionary to "learn" these symbols at arbitrary locations, favouring



(a) Original Block (b) Symmetric Block

Figure 6.21: New pattern created by using the additive symmetric of the original block.

future approximations. For text regions the MFV prediction mode is predominately used, preserving the structure of the input image patterns. In this case, the use of displaced blocks allows the exploitation of the self similitude of the patterns in the original image.

For natural images, the residue patterns are not as correlated with the input image, and the experimental results do not show the same significant gains. Nevertheless, the new updating technique is consistently able to improve the quality of MMP-I encoding across all compression ratios. For compound images we observe a combination of the previous phenomena: high gains for the text regions combined with more moderate gains in the smoother areas, resulting in intermediate increments of the PSNR values. Generally we can observe that the use of displacements of one quarter of the blocks dimensions ($s = 4$) is only advantageous over the use of $s = 2$ for non-smooth images. This means that, for smooth images, the gains originated by the extra patterns introduced when this technique is applied are only sufficient to counterbalance the additional rate associated with the extra dictionary indexes.

6.3.3 Additive symmetric

In our investigation, we have also used the *additive symmetric* of each block to update the dictionary, *i.e.* at each dictionary adaptation step we also include the block $-\hat{\mathbf{R}}^l(m,n)$. This technique can only be applied on MMP-I, that processes predicted residue samples which may have positive or negative values. The additive symmetric of a predicted block (Figure 6.21) tends to have the same directional structure as the original block, meaning that it has the potential of being useful to encode future residues determined with a similar prediction mode, or even with a different one.

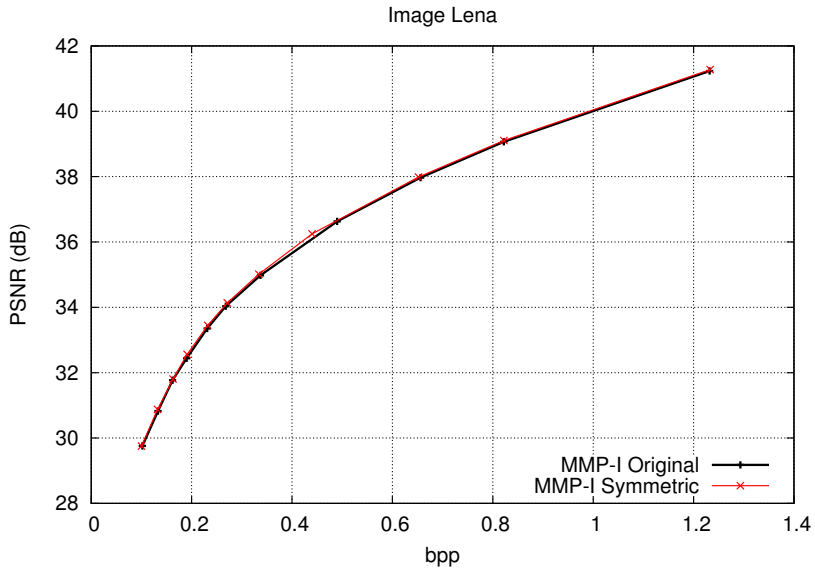


Figure 6.22: Results for MMP-I with the use of new dictionary updating with the additive symmetric block, for image Lena.

Figures 6.22 and 6.23[7] presents the experimental results of the simulations conducted with this new updating strategy. For smooth images, the overall results are equivalent to those of the original method, but for text image we observe significant losses in the PSNR value. This means that the symmetric residue blocks introduced in the dictionary do not improve the approximation power of the stored patterns. For the highly structured text images, that tend to use a prediction mode which produces patterns with consistently negative values (see Figure 5.24), the insertion of the symmetric counterparts only has the negative effect of increasing the indexes' entropy, leading to severe losses in the PSNR values.

6.3.4 Combining the new updating strategies

In the previous sections we have assessed each new technique independently. In order to optimise the use of these strategies we have also tested the performance of their joint use. We have thus tested each possible combination of these schemes, and compared the results with those of each isolated technique. This was done to evaluate if the individual contributions of the updating methods still have beneficial effects when they are combined.

[7]The results for other test images are presented in Figure C.24 of Appendix C.

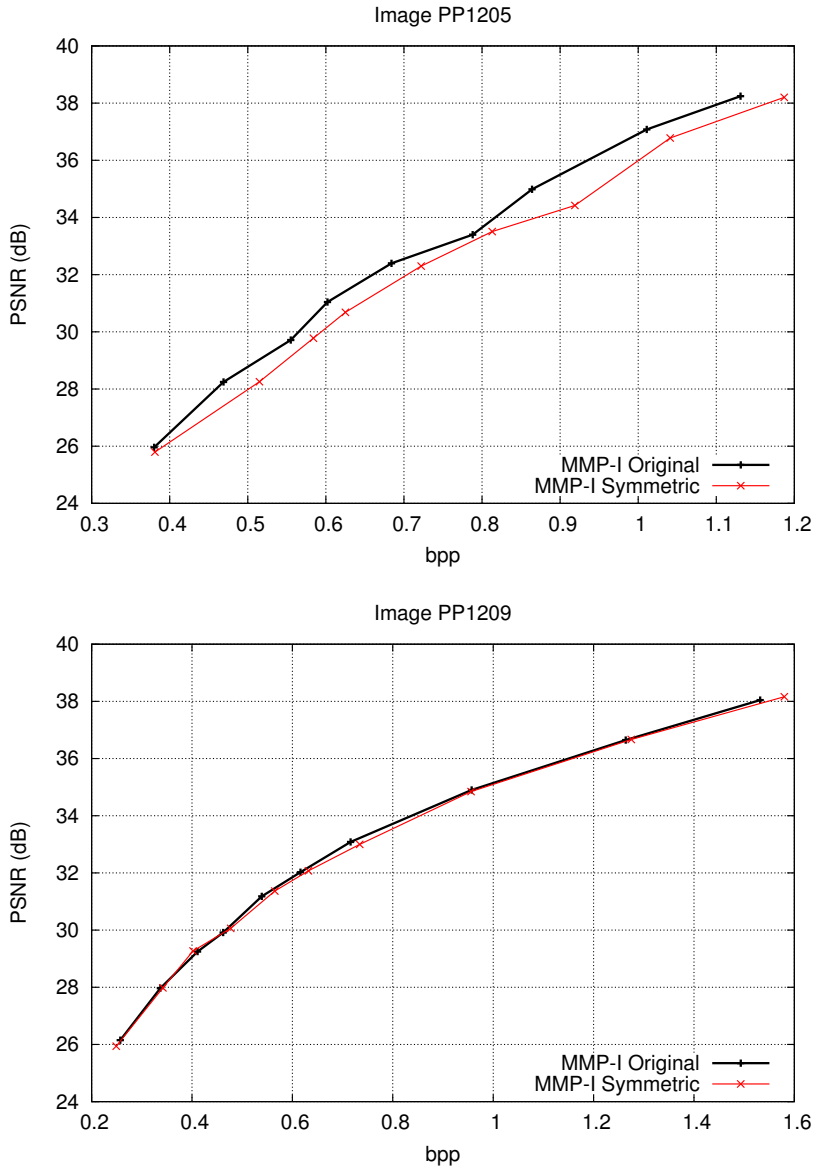


Figure 6.23: Results for MMP-I with the use of new dictionary updating with the additive symmetric block, for images PP1205 and PP1209.

Based on these experimental observations, the best combination of updating methods was chosen.

A detailed description of all tests needed to cover the possible combinations of the discussed methods would be very extensive and unnecessarily tedious. Therefore, we chose to briefly describe the main conclusions of these tests and present only the most relevant results. The best results were achieved by the combination of all updating strategies, which confirms the usefulness of the studied techniques. Nevertheless, two other combinations

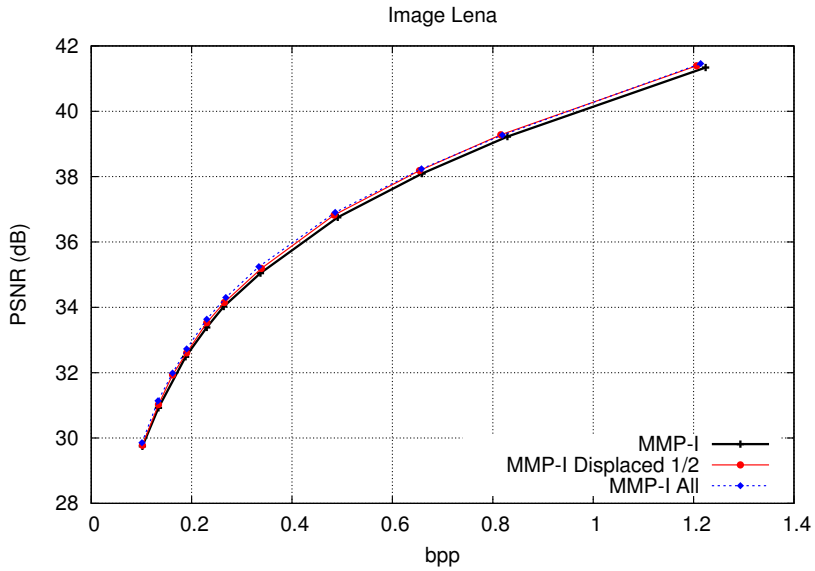


Figure 6.24: Results for MMP-I using combinations of the new dictionary updating techniques, for image Lena .

achieved almost equivalent results, namely a) the use of geometric transforms combined with block displacements using $s = 2$ and b) the isolated use of displacements with $s = 4$. However, for all these cases, the PSNR gains corresponding to the use of just the displaced blocks with $s = 2$ were marginal. Therefore, taking into account the increased computational complexity resulting from the introduction of extra dictionary blocks, we conclude that the isolated use of the displaced blocks with $s = 2$ is a better option. Figures 6.24 and 6.25[8] compare the results for MMP-I using the original dictionary updating technique (MMP-I), the new updating strategies using only displaced blocks with $s = 2$ (MMP-I Displaced 1/2) and the new updating strategies when all of the studied methods are used (MMP-I All).

Other important conclusion of this study is that the effects of the proposed redundancy control methods become even more relevant when the updating techniques are used. In fact, the use of multiple blocks to update the dictionary generates a large number of new patterns, thus increasing the redundancy among the dictionary vectors. To reduce this effect, the redundancy control techniques, developed in Section 6.2.2, were applied. This combination resulted in a very efficient compromise between two opposite goals: increasing the approximation power of the dictionary, while avoiding a compromising redundancy

[8]The results for other test images are presented in Figure C.25 of Appendix C.

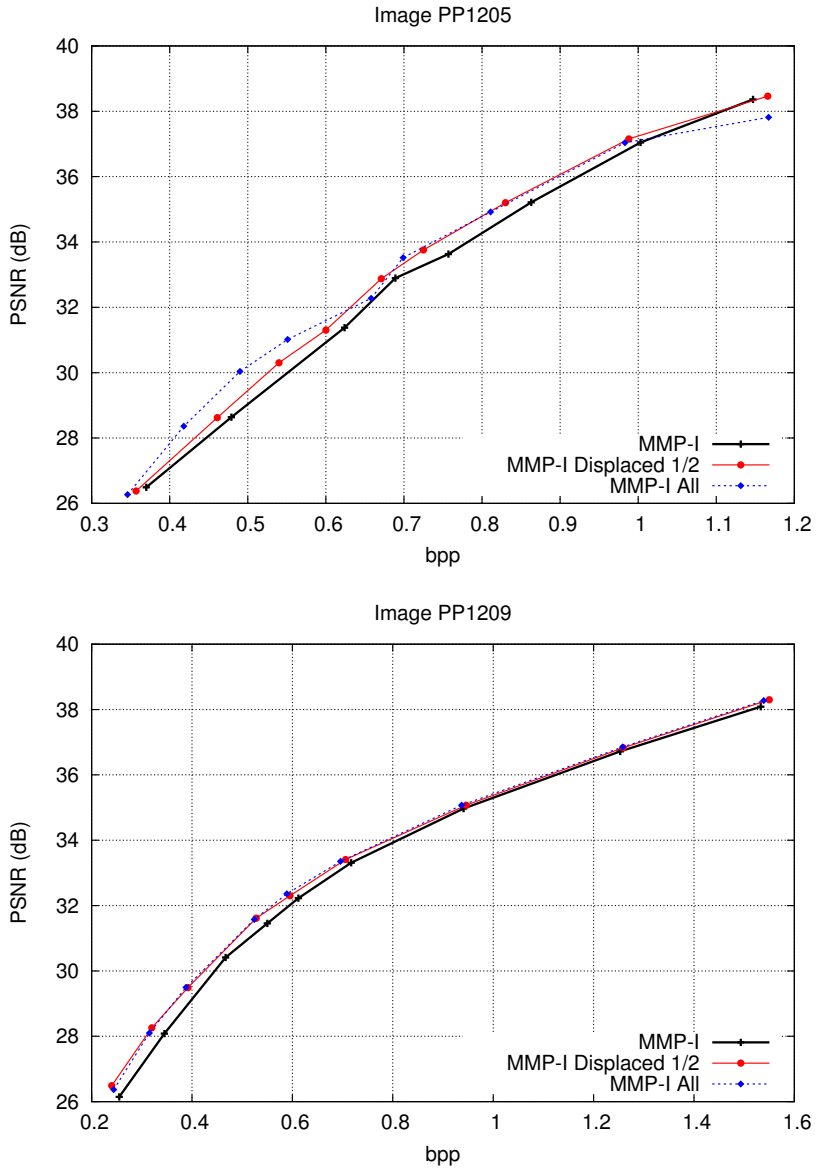


Figure 6.25: Results for MMP-I using combinations of the new dictionary updating techniques, for images PP1205 and PP1209.

among its patterns. Another interesting observation is that some of the updating techniques, that originally introduced coding losses, became favourable when the redundancy removal was applied. Moreover, the comparative results among the techniques discussed in this section change slightly when redundancy control is used. Therefore, all of the results presented in Figures 6.24 and 6.25 already use the redundancy reduction method described

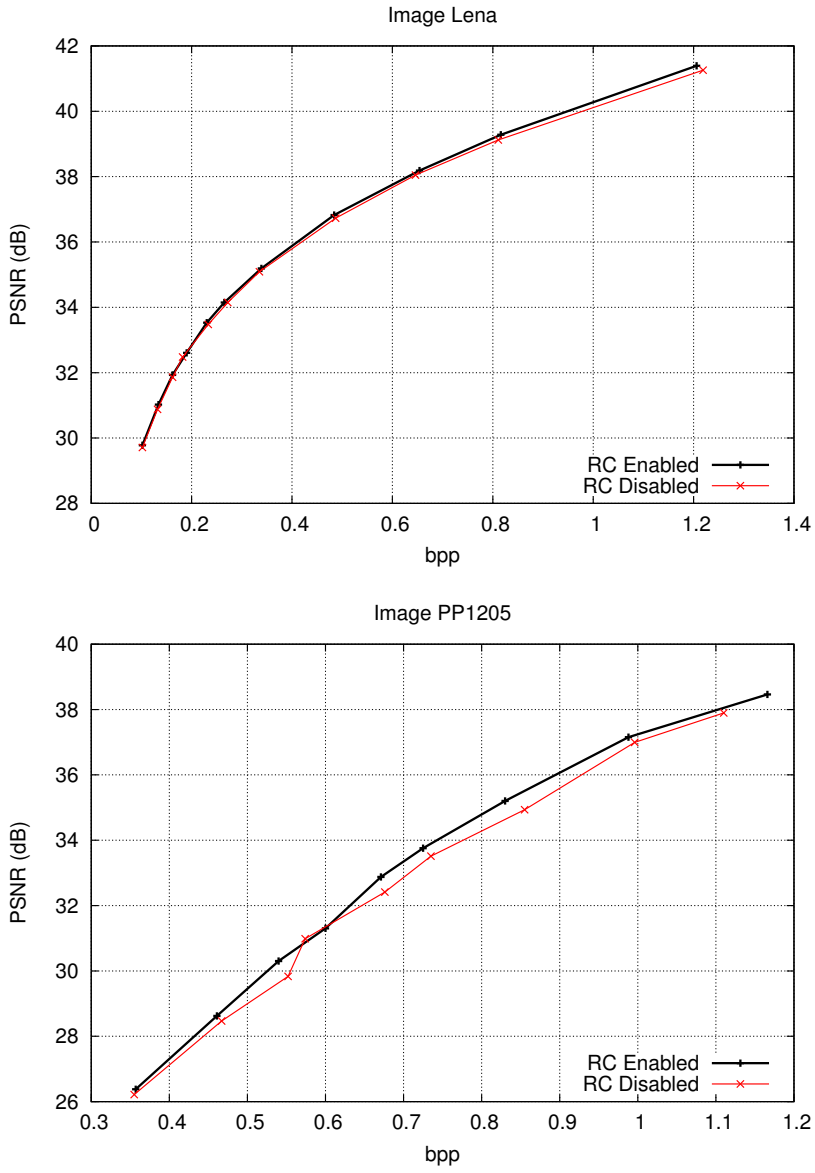


Figure 6.26: Effects of using redundancy control with new dictionary updating techniques (using displaced blocks with $s = 2$), for images Lena and PP1205.

in Section 6.2.2. Figure 6.26[9] presents an example of the effect of using redundancy control for the most favourable updating strategy, *i.e.* the use of displaced blocks with one half of the block dimension.

The updating strategies described in this section were also evaluated as a context conditioning criterion, to be used in the dictionary partitioning scheme described in Section

[9]The results for other test images are presented in Figure C.26 of Appendix C.

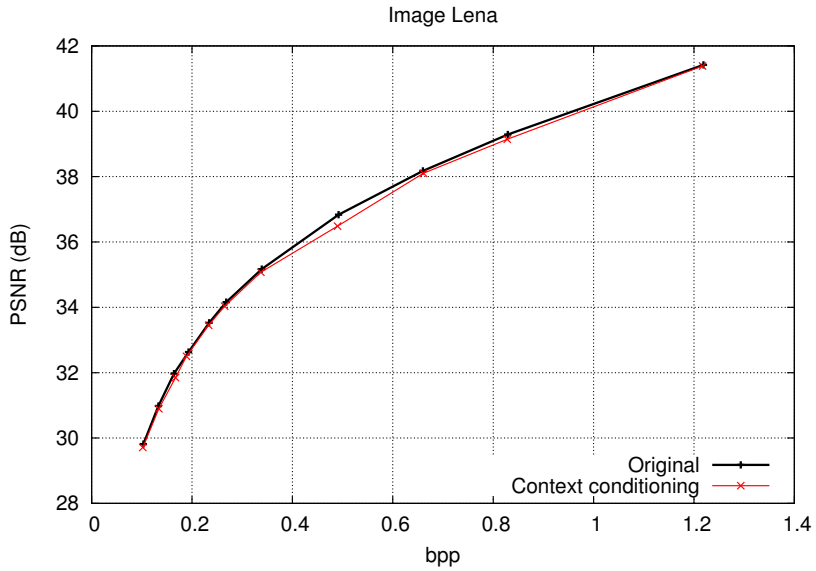


Figure 6.27: Results for MMP-II for image Lena, when context conditioning associated with the code-vectors' origin is used.

6.1. In this case, separate partitions were created to distinguish the blocks that were originally created using each different updating criterion (displaced blocks, geometrically transformed blocks, etc.). However, experimental results have shown that these extra partitions lead to no performance improvements, as can be seen in Figure 6.27 for the case of image Lena. This is mainly due to two reasons: first, the probability distribution associated with the use of different dictionary updating processes does not favour the two-step encoding; second, the increased number of partitions (arithmetic encoding contexts) makes it harder for the arithmetic coder to learn the distribution statistics associated with each context.

The increased efficiency of the new updating procedure described in this section is accompanied by a larger number of dictionary vectors, that severely increases the complexity of the coding algorithm. This is because the redundancy reduction performed by the hypersphere restriction is in fact able to avoid the inclusion of redundant blocks, but is not capable of avoiding the insertion of new patterns, that are sufficiently different from all other dictionary vectors, but are not useful to approximate future image patterns. A discussion on this topic is presented in the following section, together with the proposal of techniques that significantly reduce, or even cancel, this complexity escalation.

6.4 Avoiding the inclusion of unnecessary blocks

In Section 6.2.2 we developed a redundancy control algorithm that avoids the insertion of blocks that are very similar to those already present in the dictionary. The performance gains of this process come from the concentration of all similar patterns into one single index of the dictionary. In this section we also deal with the problem of removing code-vectors from the dictionary, but we tackle this problem from another point of view: we try to avoid the inclusion of new patterns that will not be useful in future approximations (*i.e.* code-vectors that will never be used). The insertion of these code-vectors does not necessarily reduce the compression performance, since by updating the probability histograms of the coding symbols, the arithmetic encoder tends to eliminate the effects of the non-used elements. Therefore, a significant impact in the coding performance of the method should not be expected.

This is very different from what happens with the elimination of *redundant* blocks, *e.g.*, through the use of the hypersphere technique. In this case relevant blocks, *i.e.* blocks with relevant probability counts, are condensed in one single element, meaning that we are in fact eliminating vectors that would cause an increase in the average entropy of the used index symbols. One would expect the hypersphere to also eliminate some vectors that correspond to useless patterns, but this has a less significant impact on the performance of the encoder.

The previous discussion led us to consider the elimination of unnecessary blocks as a secondary objective, since at this stage, as we have already pointed out, the main focus of our work is to maximise coding performance. Furthermore, as we have observed in previous attempts to restrict the dictionary growth, trivial block elimination methods tend to decrease the coding performance. Nevertheless, the large escalation of the dictionary size for the new updating methods, described in the previous section, has the effect of severely increasing the computational complexity of the method. This motivated the study of these techniques, as a way to efficiently control the computational complexity without compromising the algorithm's efficiency.

In an initial study of the MMP algorithm (see Section 3.4.1), the use of simple methods that limit the maximum number of dictionary elements was investigated, but these schemes were shown to degrade the algorithms' performance. This losses are caused by the elimination of code-vectors that would be useful in future approximations. In this section

Scale	$l_o = 0$	$l_o = 1$	$l_o = 2$	$l_o = 3$	$l_o = 4$	$l_o = 5$	$l_o = 6$	$l_o = 7$	$l_o = 8$
$l = 0$	**100,0%**	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
$l = 1$	**32,2%**	**16,1%**	**19,3%**	**28,1%**	1,1%	0,3%	0,2%	0,0%	2,6%
$l = 2$	12,2%	2,7%	**27,6%**	**37,7%**	15,8%	0,1%	0,0%	0,7%	3,3%
$l = 3$	10,0%	0,7%	7,5%	**14,2%**	**58,7%**	3,8%	1,4%	3,5%	0,3%
$l = 4$	0,2%	0,1%	1,5%	9,5%	10,8%	**17,0%**	**42,9%**	3,8%	14,2%
$l = 5$	**26,1%**	0,2%	0,7%	5,6%	**15,6%**	6,5%	**34,1%**	7,7%	3,5%
$l = 6$	5,5%	0,0%	1,3%	0,3%	6,9%	10,1%	11,9%	**30,3%**	**33,9%**
$l = 7$	**58,3%**	0,0%	0,5%	1,4%	0,5%	3,1%	**14,3%**	2,4%	**19,5%**
$l = 8$	**43,6%**	0,0%	0,0%	0,0%	0,5%	0,5%	11,3%	15,9%	**28,2%**

Table 6.1: Percentage of used code-vectors of each scale, l, that were created from blocks with an original level, l_o, for image Lena coded at 0.5 bpp.

we investigate two techniques that successfully eliminate dictionary blocks without causing performance losses. The first approach limits the range of the scale transforms used in the updating step of the dictionary. The second method adaptively chooses the maximum block dimensions. Both methods were optimised in order to achieve the expected computational gains without compromising the encoding performance.

6.4.1 Limiting the range of scale transforms

In the original algorithm, when a new block $\mathbf{S}_{new}^{l_o}$ of level l_o is created, scaled versions of $\mathbf{S}_{new}^{l_o}$ are used to update the dictionary at *all* used scales. In this process, an original 2×1 block is expanded to patterns that may reach a size of 16×16 pixels. Due to the large difference in the blocks dimensions, the scale transformations for which the final scale is very different from l_o may create new blocks that are not closely related to the original patterns. Because of this, the usefulness of these new code-vectors may be reduced. In order to study this phenomenon, we have observed the number of used blocks of each scale and the level l_o, at which they were originally created. Table 6.1 shows, for each dictionary scale l (represented in the table's lines), the percentage of used code-vectors that were created from blocks of each original scale l_o. These data were collected from an MMP-I encoder used on image Lena compressed at 0.5 bpp. From table 6.1 we notice that for each dictionary scale, the encoder tends to use mainly dictionary patterns that were created at a scale l that is close to l_o. The exception for this is the use of blocks created at scale 0,

that correspond to uniform elements that were created with the dictionary's initialisation. This behaviour was also observed for other test images and compression ratios.

The previous result shows that a 16×16 block created from the expansion of a 2×2 block has a very small probability of ever being used (0% in the example of table 6.1). This knowledge was exploited by adapting the scale transform procedure used for dictionary updating. In the new procedure, each new block created at an original scale l_o is only used to update dictionary scales l that are "close" to l_o. Namely, only scales in the range

$$\max\{0, l_o - L_{low}\} \leq l \leq \min\{l_{Max}, l_o + L_{high}\} \tag{6.6}$$

are updated, instead of all available scales. In this equation, the values of L_{low} and L_{high} define the bounds of the scale interval that is used in the dictionary adaptation. Several tests were performed in order to determine the best values for these bounds. The optimum values for these parameters varied slightly for different image types. However, as a general rule, the use of $L_{low} = L_{high} = 2$ allowed for a significant reduction on the computational complexity, without a relevant loss in the final quality of the compressed images. One interesting fact is that, for some particular images and compression ratios (mainly text and compound images), such a restriction on the updated scales leads to some marginal PSNR performance gains. This happens because, in such cases, a large number of useless patterns are introduced by the original updating process.

Figures 6.28 and 6.29[10] present a summary of the effects of scale restriction in the dictionary adaptation process. It shows the RD performance for:

- MMP-I combined with new updating techniques using extra displaced blocks with $s = 2$, dictionary segmentation and redundancy reduction (from this point on, this combination will be referred to as MMP-II A);
- The previous technique combined with scale restriction (from this point on, this combination will be referred to as MMP-II B).

In spite of the equivalent compression results when all scales are used and when one restrict the scales using $L_{low} = L_{high} = 2$, the effects of this technique in the dictionary growth (and hence in the computational complexity of the method) are very significant, as will be discussed in section 6.4.3.

[10] The results for other test images are presented in Figure C.27 of Appendix C.

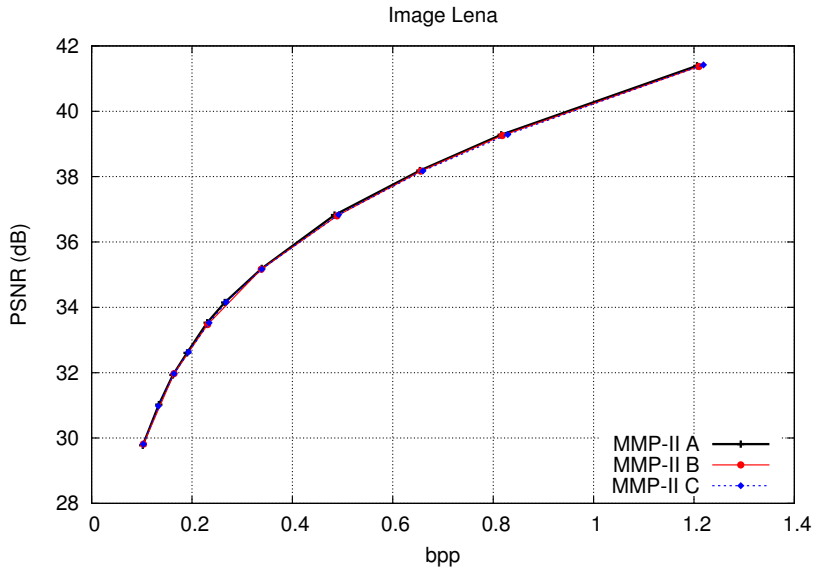


Figure 6.28: Results for MMP-II when scale restriction in dictionary adaptation (MMP-II B) and scale restriction plus adaptive block size (MMP-II C) are used, for image Lena.

6.4.2 The use of adaptive block sizes

Another factor that affects the computational complexity is the maximum block size used by MMP, that corresponds to the highest level of the MMP segmentation tree. Smaller block dimensions mean that segmentation trees with smaller depths have to be used, both in the optimisation of the prediction mode and in the search of the best match, decreasing the overall complexity. Originally, initial 16×16 blocks were used for all cases, since they provided the best rate-distortion performance gains, especially for high compression ratios, where the use of large blocks allows bit rate savings for coarser approximations. However, for low compression ratios, where smaller distortions are targeted, blocks of the largest scales are almost never used. This phenomenon was previously discussed for the original MMP algorithm (see Figure 3.13 and the its discussion) and similar results were also observed for MMP-I-based encoders.

We have exploited this fact by using an adaptive rule that controls the initial MMP block size. The idea is to use larger blocks when higher compression ratios are targeted, while smaller blocks are progressively used for lower compression ratios. We have again related the target compression ratio with the input parameter λ, that is used in RD op-

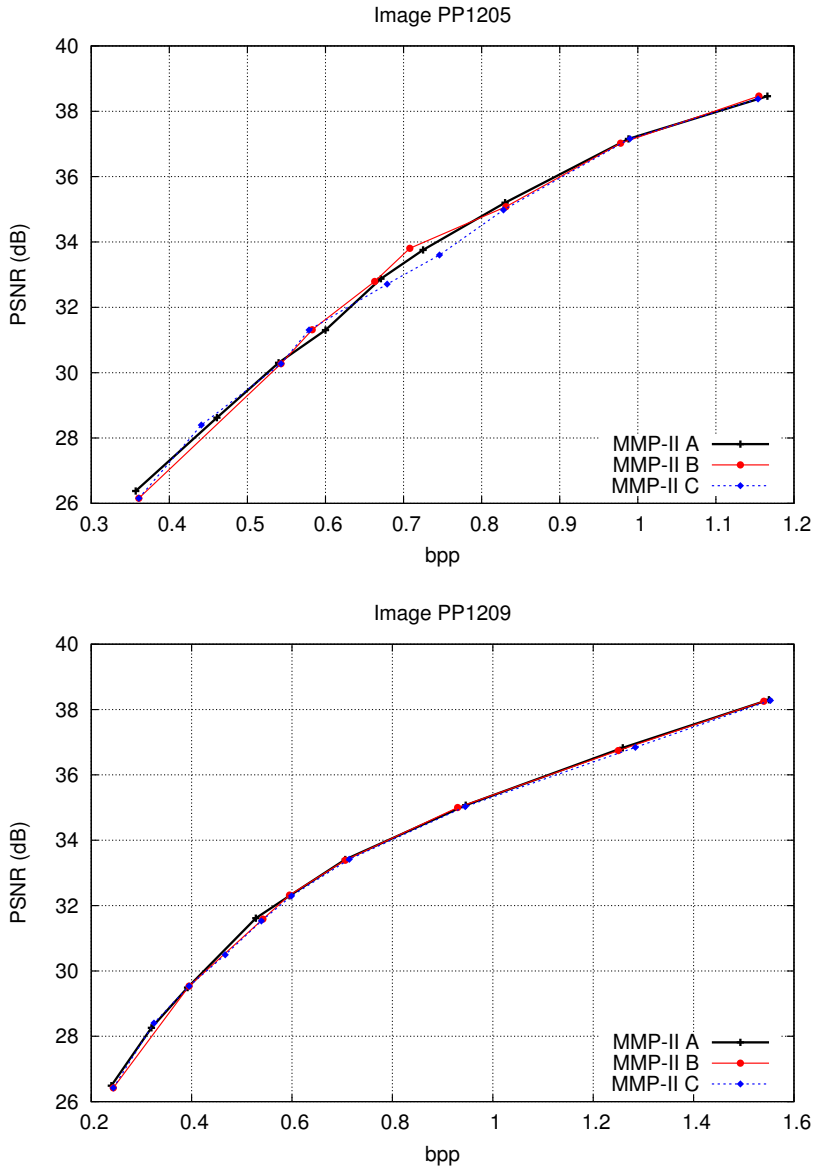


Figure 6.29: Results for MMP-II when scale restriction in dictionary adaptation (MMP-II B) and scale restriction plus adaptive block size (MMP-II C) are used, for images PP1205 and PP1209.

timisation (see Section 6.2.2 for a description of this process). An experimental study of both the computational complexity and coding efficiency of MMP-I, for various initial

block sizes, led to the following equation:

$$L(\lambda) = \begin{cases} 8 & (16 \times 16 \text{ blocks}), \quad \text{if} \quad \lambda > 150; \\ 7 & (16 \times 8 \text{ blocks}), \quad \text{if} \quad 25 < \lambda \leq 150; \\ 6 & (8 \times 8 \text{ blocks}), \quad \text{if} \quad \lambda \leq 25; \end{cases} \tag{6.7}$$

where L is the scale of the used macroblocks. As for the case of the distortion threshold rule, described in Section 6.2.2, a negligible overhead (two bits) must be used to transmit the macroblock size to the decoder. The use of this rule avoids any relevant quality loss for *all* tested images at *all* target bit rates, due to the current focus on the algorithms' performance. The performance of MMP-II B with adaptive block size (ABS), referred to as MMP-II C, is presented in Figures 6.28 and 6.29. The experimental results show an equivalent RD performance when ABS is used. Nevertheless, a significant reduction in the computational complexity was achieved when this technique was applied. This effect will be discussed in Section 6.6.2.

6.4.3 The effects on dictionary adaptation

In this section we discuss the effects of scale restriction and the use of ABS on the dictionary growth. Figure 6.30 shows the final number of elements for all scales of the dictionary for image Lena, encoded with the original MMP-I (a), MMP-II A (b), MMP-II A combined with scale restriction (MMP-II B) (c) and MMP-II B with ABS (MMP-II C) (d). In this figure it is possible to observe the severe effects of the additional blocks used by MMP-II A in the dictionary growth. A large increase in the final number of elements of the dictionary is noticeable, even with the use of the redundancy control algorithm. The scale limiting procedure (MMP-II B) also has a noticeable effect on the growing pattern of the dictionary. One observes a large reduction on the number of elements for all scales, with a larger effect on the smaller (lower than 4) and larger (higher than 6) scales. This is so because most used blocks have scales in the $[2, 4]$ interval. This means that the updating procedure will be more frequent for the scales in the $[0, 6]$ interval, which accounts for the reduction of the cardinality of $\mathcal{D}^7$ and $\mathcal{D}^8$. The use of the redundancy reduction scheme explains the decrease of the number of code-vectors for the smaller scales (see Section 6.2.2). Finally, the effects of using of ABS with MMP-II (MMP-II C) are only noticeable for the higher scales of the dictionary. Scales 7 and 8 of the dictionary are only used for the first seven

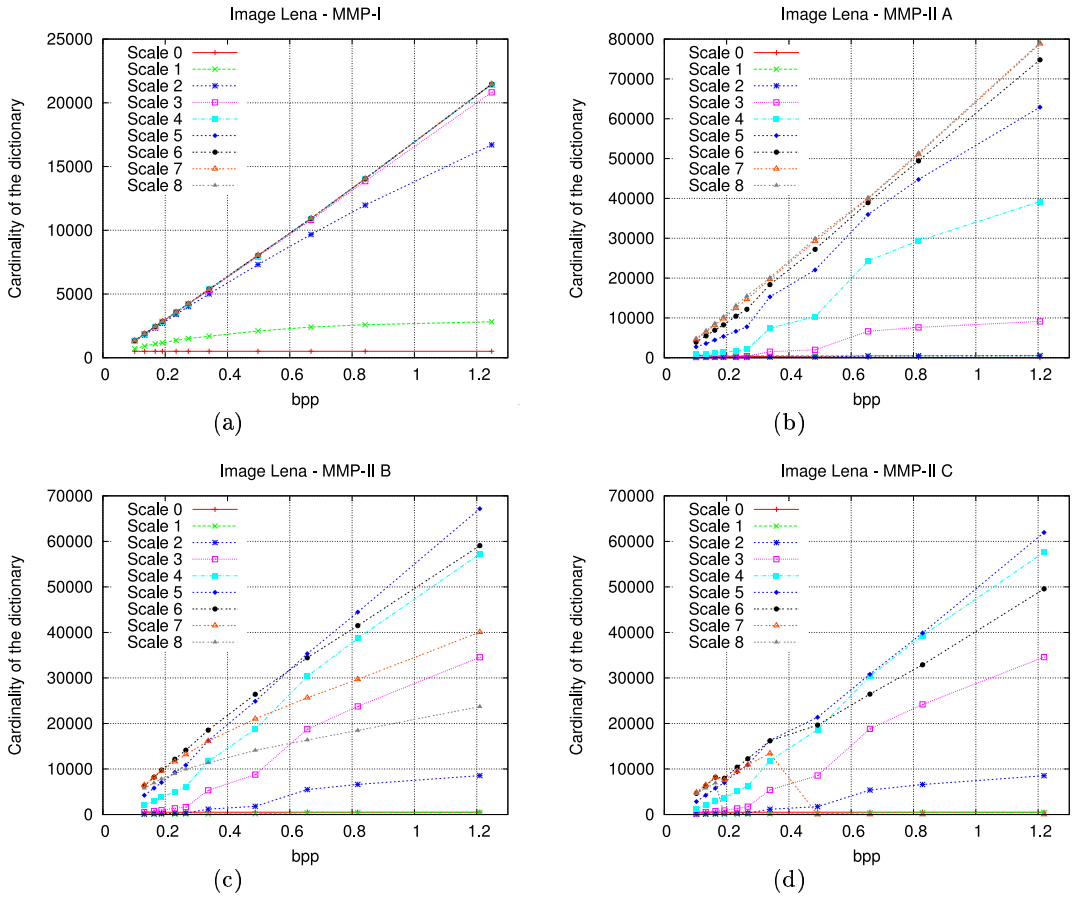


Figure 6.30: Final size of the dictionaries of each scale *vs.* the compression ratio, for image Lena: a) MMP-I; b) MMP-II A; c) MMP-II B and d) MMP-II C.

and three points of Figure 6.30, respectively. This is so because these scales are disabled, depending on the used value for the λ encoding parameter. In terms of the remaining scales, one notices only a small reduction on the number of elements used for scale 6. All other scales have more or less the same number of code-vectors.

6.5 Norm equalisation of scaled blocks

In the previous chapter we have demonstrated the accuracy of the generalised Gaussian model in the characterisation of the predicted residue signal used by MMP-I. In this section we investigate some theoretical results on the problem of encoding GG sources with vector-quantisation. Previous works describe the properties of the GG sources that are useful in the project of VQ encoders [74, 78]. Based on some generic premises described in these

references and experimental tests aiming the analysis of the statistical properties of the MMP-I's source vectors, we introduce a norm equalisation process in the MMP's dictionary update procedure.

Theoretical results for GG sources with a shape parameter α describe the stability of the L^α norm of the input vectors $\mathbf{x}$, when their dimension tends to infinity [78, 74]. The L^α norm (also called the α-norm) of a vector $\mathbf{x} = (x_0, x_1, \ldots, x_{n-1})$ is defined as:

$$|x|_\alpha = \left(\sum_i |x_i|^\alpha \right)^{1/\alpha}. \tag{6.8}$$

A simple case with particular interest is the *Euclidean* or L^2 norm of a vector, defined as

$$|x|_2 = ||x|| = \sqrt{x_1^2 + x_2^2 + ... + x_n^2}. \tag{6.9}$$

The L^2 norm of a vector corresponds to the common measure of the vector's length. When applied to the difference of two vectors, $||\mathbf{x} - \mathbf{y}||$ returns the Euclidean distance between $\mathbf{x}$ and $\mathbf{y}$.

In [78] a study of the properties of a Laplacian source is presented and a new source coding theorem is described. Similar results are then demonstrated for GG sources in [74]. In sum, these results show that, for high rate and dimension, N, the GG source vector's, $\mathbf{x}$, will cluster on a thin "shell" of constant probability, meaning that the optimum source code can be asymptotically designed based only on a N-dimensional shell corresponding to the locus of the vectors with a constant L^α norm, where α is the shape parameter of the GG model of the source. Both theoretical and experimental observations show, nevertheless, that the standard deviation of the L^α norm increases with dimension, meaning that the considered shell is not infinitely thin. However, from a distortion point of view, the shell is of practical interest when one considers that its thickness relative to its distance from the origin tends to zero, for large vectors [74].

This result suggests that the use of dictionary blocks with a uniform L^α norm could provide a better approximation for the residue blocks used by MMP-I. However, one has to consider the limited size of the blocks used by MMP-I, as well as the skew between the theoretical model and the actual predicted residue signal, that may limit the accuracy of this model. Because of this, an experimental evaluation of the regularity features of the predicted residue blocks used by MMP-I was conducted. This study demonstrated a regularity in the norm of the predicted residue blocks used by MMP-I. Based on these

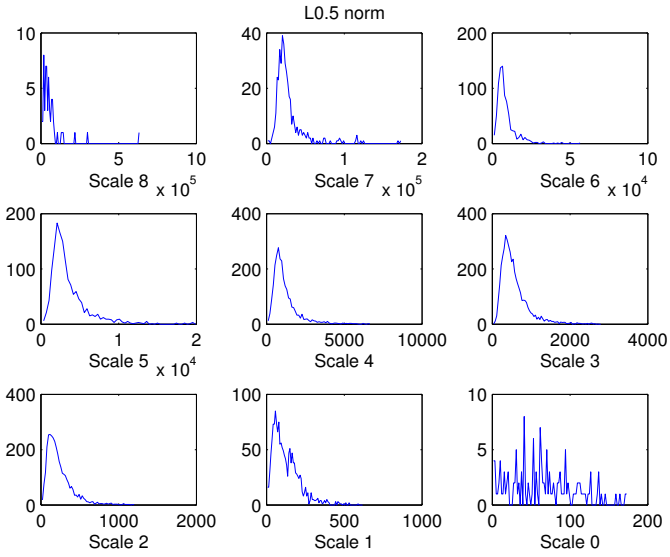


Figure 6.31: Histograms of the $L^{0.5}$ norm of the predicted residue blocks approximated by MMP-I for each scale of image Lena coded at 0.83 bpp.

observations, Section 6.5.2 describes a new norm equalisation procedure that is consistently able to increase the encoding efficiency.

6.5.1 Probability distribution of the predicted residue norms

In this section we study the distribution of the L^{α} norm of the MMP-I predicted residue blocks. In order to do this, the actual prediction residue blocks approximated by MMP-I at different scales were analysed and the value of the L^{α} norm for each of these blocks was determined. Figure 6.31 represents the histograms of the $L^{0.5}$ norm of the residue blocks of each scale, for image Lena encoded at 0.83 bpp with MMP-I. The abscissa axes of the representations were limited from zero to five times the average value of the corresponding distribution (the remaining values are sufficiently low to be negligible). The $L^{0.5}$ norm was chosen because the estimated value for the shape parameter of the generalised Gaussian model of the predicted error distribution is 0.56 (see table 5.2).

In Figure 6.31 we may observe a concentration of the vectors' norms. This originates narrow distributions, that are very similar except for the higher and the lower scales. For the higher scales, one does not have sufficient blocks in order to be able to build a good distribution model. This happens because, at this relatively low compression ratio, MMP-

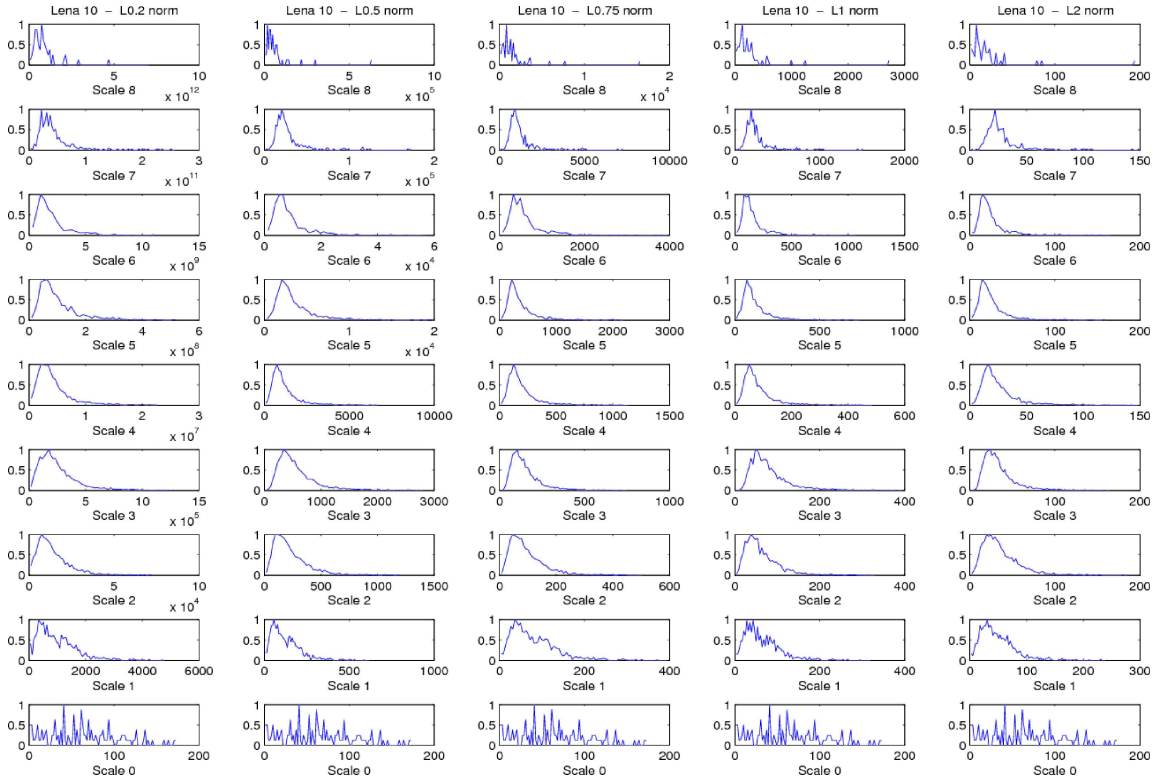


Figure 6.32: Normalised histograms of the L^{α} norm of the MMP-I predicted residue blocks for each scale, for image Lena coded at 0.83 bpp.

I tends to use smaller blocks to approximate the original image. At the lower scales we observe a greater scattering of the blocks norms. These scales are used by MMP-I when the prediction process is not efficient enough to create a smooth residue pattern, that can be approximated by a larger block. These cases generally correspond to residue samples with a greater variance. Also, for these smaller blocks the infinite size condition is severely violated and the norm concentration property does not hold. Despite this, from scale 2 (2×2 blocks) to scale 6 (8×8 blocks), one may observe a reasonably narrow distribution of the $L^{0.5}$ norm distribution, that seems to be uniform across these scales.

In order to evaluate if this phenomenon depends on the used norm, Figure 6.32 compares the plots of the previous figure with equivalent histograms, determined using different L^{α} norms. All plots are normalised using the maximum frequency value of each histogram, in order to facilitate the comparison for values of $\alpha = 0.2$ (left column), $\alpha = 0.5$, $\alpha = 0.75$, $\alpha = 1$ and α=2 (right column). We may see that narrow distributions are observed also for values of α different from the estimated shape parameter for the corresponding generalised

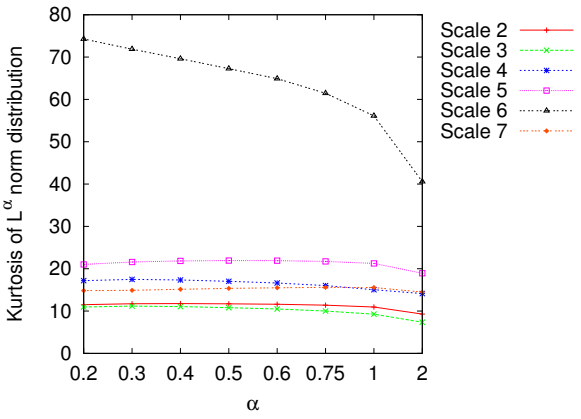


Figure 6.33: Kurtosis of L^α norm distributions for the represented block scales and different values of α.

Gaussian model (equal to 0.56, see table 5.2).

Given this apparent similarity of distributions for the several values of α, we need to further evaluate the regularity of the L^α norm across all scales, for a given value of α. In terms of coding efficiency, one would desire similar distributions, perfectly centred around a constant value of L^α, at all scales. In order to evaluate what is the best value for α, two factors were studied: first, how narrow are the distributions of the norm values for each scale, and second, to which extent do the histograms' higher frequencies occur around a constant value. In order to evaluate the first factor, we plotted the value of each distributions' kurtosis [80]. The kurtosis of a probability distribution is a measure of its concentration around its peak. The reference value is zero for the normal distribution. Positive values correspond to narrower distributions than the Gaussian, while negative values result from frequent high variations. The kurtosis of the probability distribution of the L^α norm of the MMP-II residue blocks is represented in Figure 6.33, for several values of α and for the block scales with a relevant number of used blocks. This figure shows that the value of the distributions' kurtosis tends to be generally higher for values of α between 0.5 and 1. Nevertheless, the observed variations are two small for distinguishing the distributions for each value of α, because the "peakedness" does not have a linear variation with the kurtosis.

Figure 6.34 plots the most frequent value of each L^α norm (that corresponds to the peak of the corresponding histogram), as a function of the block scale. In order to evaluate the accuracy of this value, two bound lines corresponding to the distribution's standard

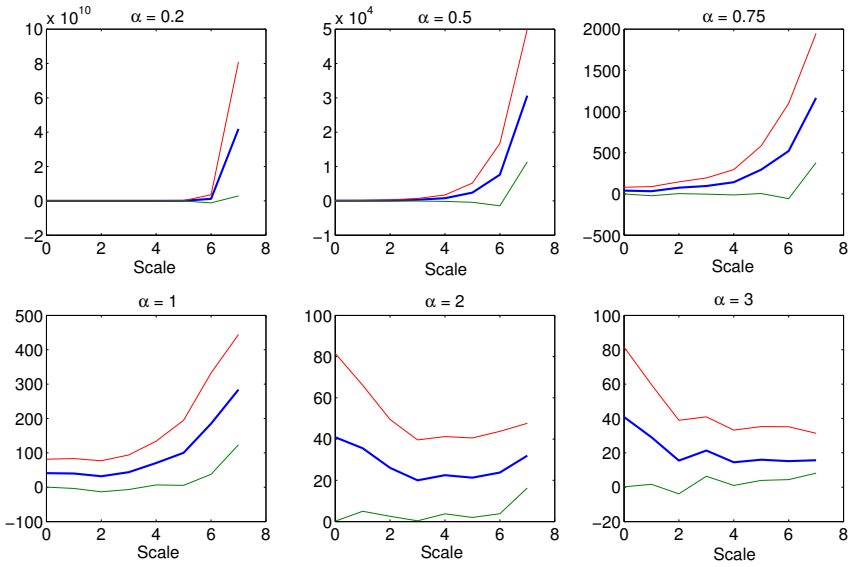


Figure 6.34: Most frequent value of each L^α norm, as a function of the block scale (bold line), for the represented values of α.

deviation added and subtracted from the considered value are also represented. In these plots we can see that the peak of the norm histogram has a smaller variation for larger values of α, which seem to indicate a grater regularity for these cases, and thus an advantage of using higher values of α. Nevertheless, this regularity is partly a consequence of the much smaller dynamic range of the norm values, and this advantage must be confirmed with experimental tests. Generally, the L^α distributions are concentrated in a small range of values, except for the very small or very large scales, where the number of observed blocks, or the number of vector elements, are too small. In spite of this, for α values below 2, we observe a growing tendency of the norm value with the considered scale.

The presented data are representative of what happens generally, but some variations are observed depending on the considered image and target compression ratio. For non-smooth images, like text and compound images, the prediction process tends not to be as efficient as for smooth images. In these cases, the predicted residues tend to have wider distributions, as do the residue blocks' norms. This also happens for smooth images, when higher compression ratios are used. In these cases we observe the same tendency described in Figure 6.34, but with a higher variation of the norm value. Nevertheless, this higher variation is also accompanied by larger standard deviations, meaning that the "confidence"

interval, represented by the two bound lines of Figure 6.34, is also larger.

Furthermore, depending on these same factors (image type and compression ratio), MMP-I tends to use mostly blocks of a given scale range: for higher compression ratios and smoother images MMP-I uses larger blocks, while for lower compression ratios or more detailed images, MMP-I uses mainly blocks from the lower scales. In these cases one has that for the least used scales the observations loose significance. Nevertheless, for other scales one may observe an identical behaviour to that presented in the previous discussion of a particular case.

6.5.2 A new norm equalisation procedure

The experimental investigation performed in the previous section demonstrated that, in spite of the reduced dimensions of the used blocks, there is a consistency in the norm of the predicted residue vectors of different scales. This means that when a new pattern is generated by MMP-I at a scale l_o, the new blocks used to update the dictionary at different scales, resulting from the scale transformation of $\mathbf{R}^{l_o}$, should have a similar norm. This norm preserving property is not present in the original scale transforms. The scale transformations, that are basically related with linear interpolations of the original blocks, tend to transform the block's norm in the same proportion as the relation between the number of elements of the original and scaled blocks.

In order to exploit the norm regularity, a new version of the scale transformations was implemented. It uses the original scaling procedure, *i.e.*

$$\mathbf{R}^l = T^l_{l_o}(\mathbf{R}^{l_o}), \tag{6.10}$$

but introduces a norm equalisation step, where one assures the equivalence of the L^α norms of the original ($\mathbf{R}^{l_o}$) and the scaled block ($\mathbf{R}^l$), *i.e.*

$$\mathbf{R}^l = s^{l_o,l}_\alpha T^l_k(\mathbf{R}^{l_o}), \tag{6.11}$$

where

$$s^{l_o,l}_\alpha = \frac{|\mathbf{R}^{l_o}|_\alpha}{|\mathbf{R}^l|_\alpha}. \tag{6.12}$$

The study presented in the previous section indicates that this procedure should be adequate for several norms. Experimental results confirmed this result and demonstrated that the L^α norm used in the new scale transformation step does not have to be very

accurate. The use of norm equalisation generally improved the final quality of the reconstructed image, for values of α in the interval $[0.2, 2]$. Changing the value of α causes some variation in the final result, but the L^1 norm represents a good compromise for all tested images, because it achieves either the best result or very nearly that, for all tested images and values of α. Besides this, the L^1 norm also has the advantage of being computationally more efficient to implement.

Geometrically, the use of L^1 norm is also a compromise between the actual value of the GG shape parameter of the signal and the reduced dimensions of the residue vectors used by MMP-I. From Table 5.2 we have that the GG distributions of the input signal are characterised by a value of α in the range $[0.3, 1]$, suggesting that the best value of α would be $L^{0.5}$. Nevertheless, experimental tests revealed a small advantage when the L^1 norm is used. This may result from the fact that, for blocks of the same dimension, the block shell is thinner for higher values of α [74].

One interesting fact observed in the simulations is that the described procedure is in fact effective, but only when its use is restricted to block expansions, maintaining the original contractions without the use of norm equalisation. One may easily understand the reason for this fact by observing once again the Figure 6.34. Considering the plot for L^1 as an example, we can see that the use of the norm equalisation procedure corresponds to a shift in the abscissa axis, maintaining the same ordinate value. A block expansion corresponds to a left-to-right movement in the plot. In this case, the resulting norm will most probably fall within less than one standard deviation from the most frequent value of the L^1-norm at the final level. Nevertheless, for block contractions, where we use a shift form right-to-left, the probability of the norm of the new block belonging to this "tolerance" interval is very small. This means that, in this case, we are in fact creating a new block that will not be efficient in approximating the residue vectors of the new scale. Also, the use of scale restriction in dictionary updating (see Section 6.4.1) limits the amplitude of the scale shift, increasing the chances that the norm of the new block belongs to the interval defined by the bound lines in Figure 6.34.

Figures 6.35 and 6.36[11] show the quality gains for MMP-I when the norm equalisation procedure is used. We can see that this technique generally allows for quality gains for smooth images (that range up to 0.2 dB), without compromising the quality of non-smooth

[11] The results for other test images are presented in Figure C.28 of Appendix C.

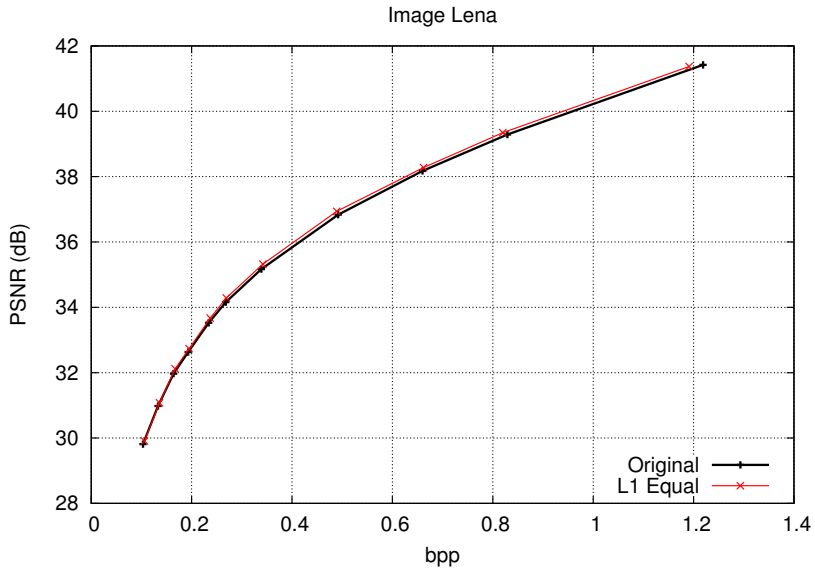


Figure 6.35: Results when L^1 norm equalisation is used with MMP-II A, for image Lena.

images, where the smaller efficiency of the predictive step generates a set of predicted residue blocks that do not correspond to the used model as accurately as those of smooth images.

6.6 Some considerations on the computational complexity of MMP-based methods

In this section we start by presenting a brief discussion on the computational complexity of the proposed algorithms and on the factors which have the most relevant effects on this measure. We then describe some implementation techniques that effectively reduce the complexity of MMP-based methods. Finally, we evaluate the effects on the computational complexity of the several techniques that were proposed along this thesis.

Since MMP-based encoders use an approximate pattern matching scheme, they have a computational complexity similar to that of standard VQ methods, that is traditionally higher than that of transformed-based encoders. The computational complexity of these encoders is mainly associated with the calculations of the sum of square differences (SSD), used in the search for the best match for a given image block. Because of this, the complexity closely depends on the number of blocks stored on the dictionary. Besides this, the

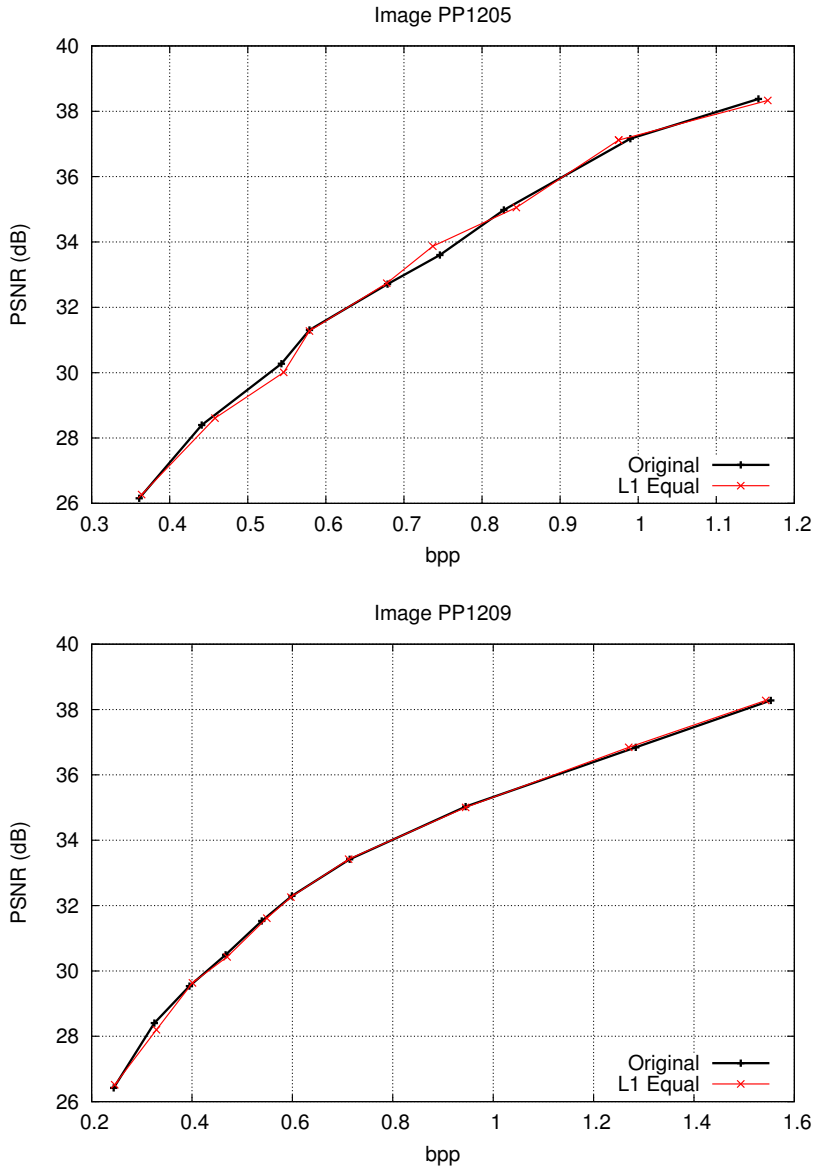


Figure 6.36: Results when L^1 norm equalisation is used with MMP-II A, for images PP1205 and PP1209.

MMP encoder also has another relevant time consuming task, that is related with the computation of the rate associated with every encoded symbol. This involves the computation of a logarithmic function, that is traditionally a computationally expensive task.

The use of predictive techniques does not impose a significant additional computational cost, since it involves only a few additions. Nevertheless, the use of prediction in MMP-I implies the optimisation of the encoding process for the residue blocks determined for each of the M considered prediction modes. This means that one should expect MMP-I to be

about M times more complex than the original MMP algorithm. In fact, this is only the case for MMP-I using FBS. Because MMP-I uses hierarchical prediction, every prediction level must be optimised for all of the available prediction modes. This causes a severe increase on the computational effort, when compared with the original MMP. As we have stated before, the computational complexity of the algorithms developed in this thesis was a secondary concern, when compared with the improvement of the compression performance of MMP for smooth images. Despite this, some of the described techniques enable an interesting reduction of the complexity of these algorithms, as will be discussed later. Also, a significant reduction can be achieved by using some appropriate implementation strategies. A brief summary of some of these strategies is presented in the next section.

6.6.1 Implementation issues

As has been previously discussed, the operations that must be considered for the complexity analysis of MMP are SSD computations, used for the search of the best match, and the computation of the Lagrangian cost function for each block. Determining the Lagrangian cost for a block involves processing a logarithmic function, since the rates involved in the cost equations are generally estimated by:

$$R(i) = -log_2(P(i|l_i)), \tag{6.13}$$

where i is the symbol for which the rate is being estimated and $P(i|l_i)$ is the probability of i in the considered probability context (in this case, the one associated with scale l_i). Roughly speaking, we have to compute one *log* function value for each attempted block match. A comparatively small number of *log* values are also determined for the segmentation and prediction mode flags.

The most time consuming task is the determination of the SSD in the search for the best match for a given block. In a straightforward implementation, this process implies one integer addition and one integer multiplication for each pixel of each code-vector in the dictionary. This means that the number of *add* and *multiply* operations used to calculate the SSD values when searching for the best match, is given by the number of dictionary elements at each scale, multiplied by the corresponding number of block pixels. In most computer systems, the multiplication of two numbers requires a much greater computational effort than their sum. In the case of MMP-based methods, the multiplications are

used to determine the squared differences for the SSD computations. A simple technique to avoid the computation of these operations is the use of a table with the squared values of each possible difference. This replaces each multiply operation by one memory fetch operation. Similarly, a table that stores the values of the base 2 logarithms may be used in the computation of the Lagrangian cost functions. This allows a considerable reduction in the computational complexity of the methods, at the cost of additional memory requirements, associated with the square and *log* tables. These memory requirements are, however, tolerable in a standard personal computer (PC) based implementation.

A significant reduction in the number of operations used in the search for the best match can also be achieved through the use of fast search methods, like those described in [81, 82]. These methods efficiently limit the number of dictionary blocks that must be tested when searching for the best match, reducing the number of operations associated with this process. Due to the focus on compression efficiency, only optimum schemes (*i.e.* schemes that always return the best match) were considered in our implementation. Much larger gains in computational efficiency have been reported in the literature for sub-optimum search methods. Nevertheless, the simple techniques described in this section led to interesting gains in complexity. We feel that further gains could be introduced by some complexity oriented implementations of the algorithms, in a future work.

6.6.2 On the computational complexity of the proposed algorithms

In spite of the complexity gains achieved by the above implementation techniques, the complexity of the proposed algorithms is still strongly dependent on the number of codevectors. Because of this, the changes in the dictionary adaptation procedure also impact on the computational complexity of the method. Some of the proposed techniques have almost no effect on the computational complexity of the algorithm, while others have a strong impact on this value. In this section we briefly discuss the complexity implications of each of the proposed dictionary adaptation algorithms.

The use of dictionary partitioning, described in Section 6.1, is one example of a technique that has no noticeable effect in the computational complexity of the encoder. Another case is the use of norm equalisation, described in Section 6.5.2. This is because the computation and equalisation of the L^1-norms of the new blocks is a very simple procedure that has no relevance when compared with the other tasks involved in MMP-based compression.

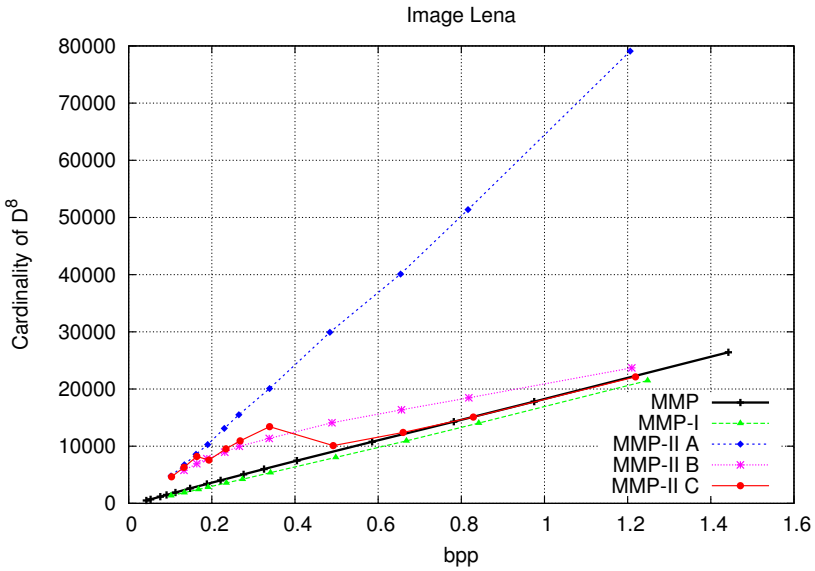


Figure 6.37: Final number of elements for scale 8 of the dictionary *vs.* rate for image Lena.

The redundancy reduction technique, proposed in Section 6.2.2, has some beneficial effects on the complexity of the MMP-II encoder, since it avoids the introduction of some elements in the smaller dictionary. The measured complexity gains depend on the reduction of the number of dictionary elements, and increase with the value of parameter d.

Other design techniques have a significant impact on the computational complexity of the method. The techniques proposed in Section 6.3 are based on the inclusion of extra elements in the dictionary. The extra code-vectors cause a significant increase of the dictionaries' cardinalities (relate to Figure 6.30 and the corresponding discussion, on Section 6.4.3). The escalation of the dictionary size for the new updating procedure severely increases the complexity of the method. Figure 6.37 summarises the effects of the use of extra displaced blocks with $s = 2$ (MMP-II A) for image Lena. Only the size of $\mathcal{D}^8$ is presented. The comparison between the number of code-vectors for MMP-II A *vs.* that of MMP-I is well representative of the additional complexity introduced by the new dictionary blocks.

On the other hand, the use of scale restriction in dictionary updating (Section 6.4.1) and the use of an adaptive block size with MMP-II (Section 6.4.2) have the opposite effect. As we have discussed in Section 6.4.3, these techniques cause a considerable reduction on the number of elements of the dictionary. This reduction can also be observed in Figure

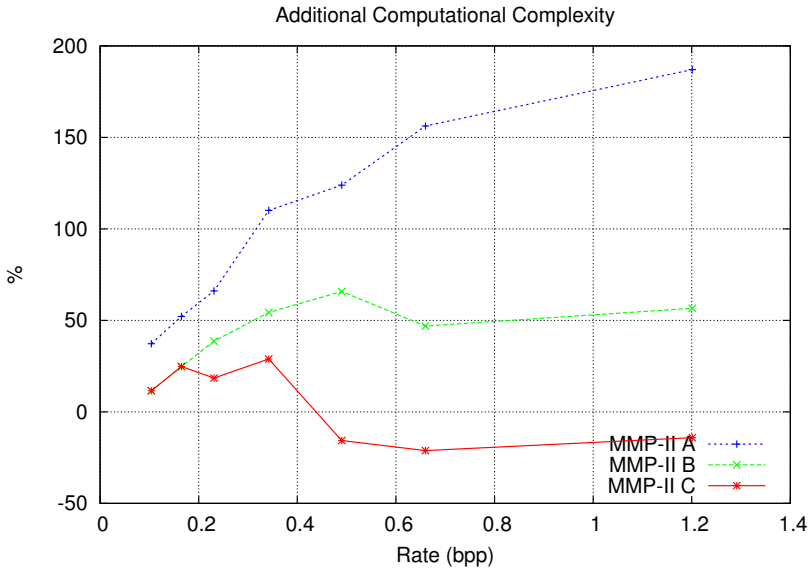


Figure 6.38: Increase in computational complexity of the new updating techniques relative to MMP-I, for image Lena.

6.37, that shows a reduction of about 75% on the dictionary size for MMP-II B and MMP-II C when compared with MMP-II A (see Figure 6.30 for an analysis of all dictionary scales).

Due to the computational optimisation techniques discussed in the previous sub-section, it is not a trivial task to theoretically evaluate the relation between the computational complexity and the number of dictionary elements, or the used block size. Because of this, an experimental study was performed, in order to evaluate the impact of the proposed techniques in the reduction of the methods' complexity. Figure 6.38 shows the percentage of additional computational complexity, relative to MMP-I, for: MMP-I with extra displaced blocks (MMP-II A); MMP-II A with scale restrictions (MMP-II B) and MMP-II B with blocks of adaptive dimensions, using equation (6.7) (MMP-II C). The experimental results have shown that:

- For MMP-II A, the computational complexity grows at a smaller rate than the number of dictionary words;
- Limiting the range of the updated dictionary scales allows for an important reduction of the computational complexity for all compression ratios;
- The combined use of scale restriction and adaptive block size allows for a significant

reduction of the complexity. In fact, for some rates we observe a reduction in the total computation time for MMP-II using these techniques (MMP-II C), when compared with the original MMP-I.

One should note that, although the above techniques provide a large reduction in computational complexity for MMP-II, it is still, at least, one order of magnitude higher than that of MMP, that already was much more complex than the one of transform-based methods. Also, all tests were performed using C-based implementations of the algorithms that were mainly intended for compression performance evaluation, as well as for the extraction of several coding statistics, such as those that have been presented in the previous sections. The implemented MMP-based encoders have only been subjected to a minor effort to optimise them for execution performance, corresponding to the implementation techniques described in Section 6.6.1. Future work will focus on other efficient ways to reduce the computational burden of the proposed methods. Some of the described techniques give a good indication that good computational gains can be achieved at the cost of minor performance losses. However, at this point of our investigation the main focus is still on optimising the algorithm's rate-distortion performance.

6.7 Experimental results for MMP-II

In the previous sections of this chapter we have presented several techniques for MMP dictionary adaptation. Experimental results were analysed for each independent technique, in order to assess its individual contribution to the compression performance of the MMP-I encoder. In this section we present a summary of the experimental results for the combined effects of the methods described in this chapter, as well as a brief discussion on their effects.

Along the presentation of the dictionary adaptation methods we were able to determine which techniques contributed to a better encoding performance, namely:

- The context conditioning for the adaptive arithmetic encoder used on the dictionary indexes' symbols, described in Section 6.1, results in a more efficient lossless compression of the MMP symbols;
- The redundancy control scheme, proposed in Section 6.2, allows for an efficient elimination of redundant code-vectors, achieving a more efficient compromise between the average entropy of the indexes' symbols and the dictionary's approximation power;

- The enhanced dictionary updating strategies, described in Section 6.3, allow for the inclusion of new code-vectors that improve the approximation power of the dictionary's patterns. In this case, we have used displaced blocks with $s = 2$ (see Section 6.3.2), because they represent the best compromise between compression efficiency and computational complexity;

- The use of scale restriction and ABS, proposed in Section 6.4, allow for a significant reduction of the computational complexity of the method with a negligible decay in encoding performance;

- The L^1-norm equalisation procedure, defined in Section 6.5, allows for a greater concentration of the dictionary patterns, increasing the coding efficiency of the algorithm.

As was previously mentioned, we refer to MMP-I combined with these techniques as MMP-II. Figures 6.39 to 6.41 compare the results of MMP-II with those of MMP and MMP-I, in order to evaluate the effects of the dictionary adaptation techniques on the algorithm. The results for the JPEG2000 [12] and H.264/AVC high profile Intra-frame image encoder [13] are also presented, in order to evaluate the relative performance of MMP-II when compared with the current state-of-the-art image coding algorithms.

The depicted experimental results demonstrate that the use of the new techniques increases the performance of the MMP-I encoder for all test images for all compression ratios. For text and compound images (see Figure 6.39), the introduced gains add to the original performance advantage of MMP-I, over the transform-based encoders. For text image PP1205, MMP-II achieves an advantage of about 6 dB over JPEG2000 and 5 dB over the H.264/AVC *high profile* Intra frame encoder. For compound image PP1209, MMP-II also outperforms all other tested methods, presenting gains over MMP-I of about 0.5 dB and an advantage over MMP of more than 1 dB. When compared with the state-of-the-art encoders, MMP-II achieves gains over JPEG2000 of about 2 dB, while the advantage of the proposed method over the H.264/AVC encoder goes up to about 1 dB.

The usefulness of the dictionary adaptation process is also evident for natural scene images, where MMP-II nearly eliminates the performance gap that exists between MMP-I and the transform-based encoders. In fact, MMP-II is generally able to achieve a coding performance equivalent to that of JPEG2000 for this type of images. In some cases it is

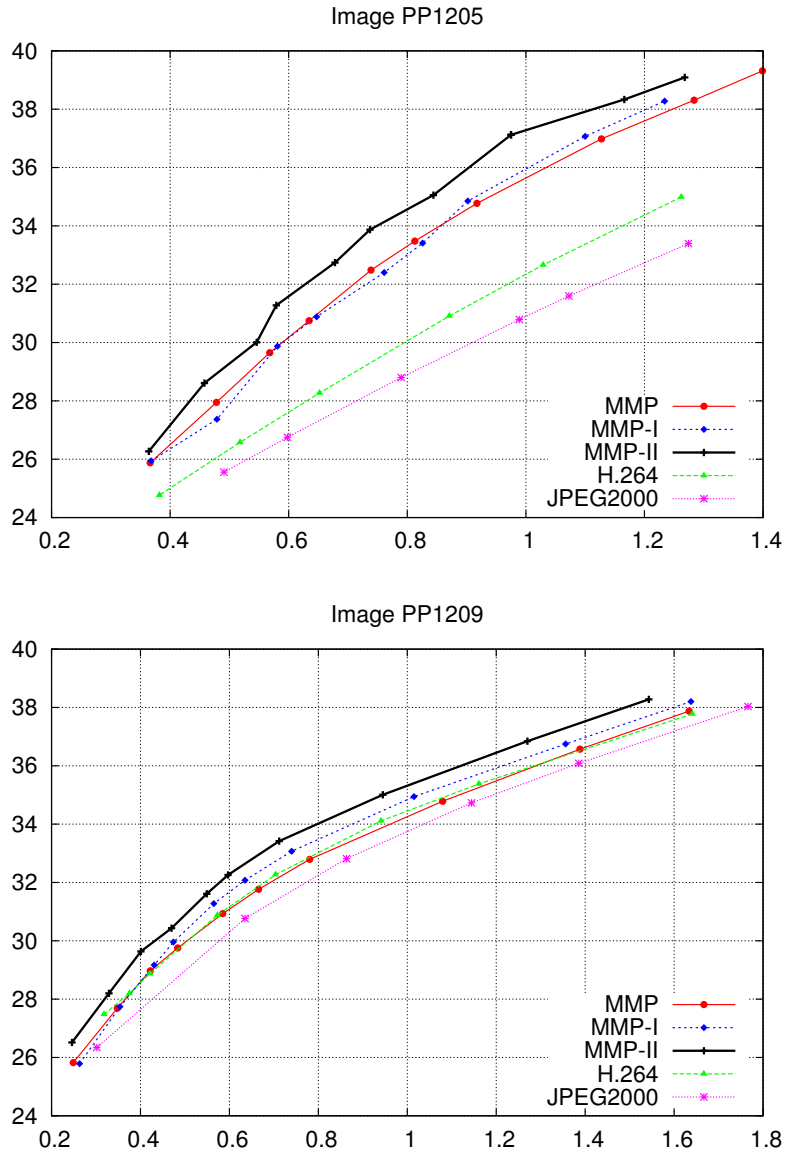


Figure 6.39: Experimental results for MMP-II, for images PP1025 and PP1209.

even able to beat the DWT-based encoder, as for the Cameraman image. When compared with H.264/AVC, the relative performance of MMP-II depends on the used test image. MMP-II tends to achieve a close result for the higher rates, but still presents a small performance disadvantage for higher compression ratios. Image Cameraman serves once again as an exception, but for image Peppers MMP-II is equivalent to H.264/AVC for compression ratios below 0.35 bpp (see Figure 6.41).

A brief perceptual evaluation of MMP-II is presented in Figures 6.42 and 6.43. For the

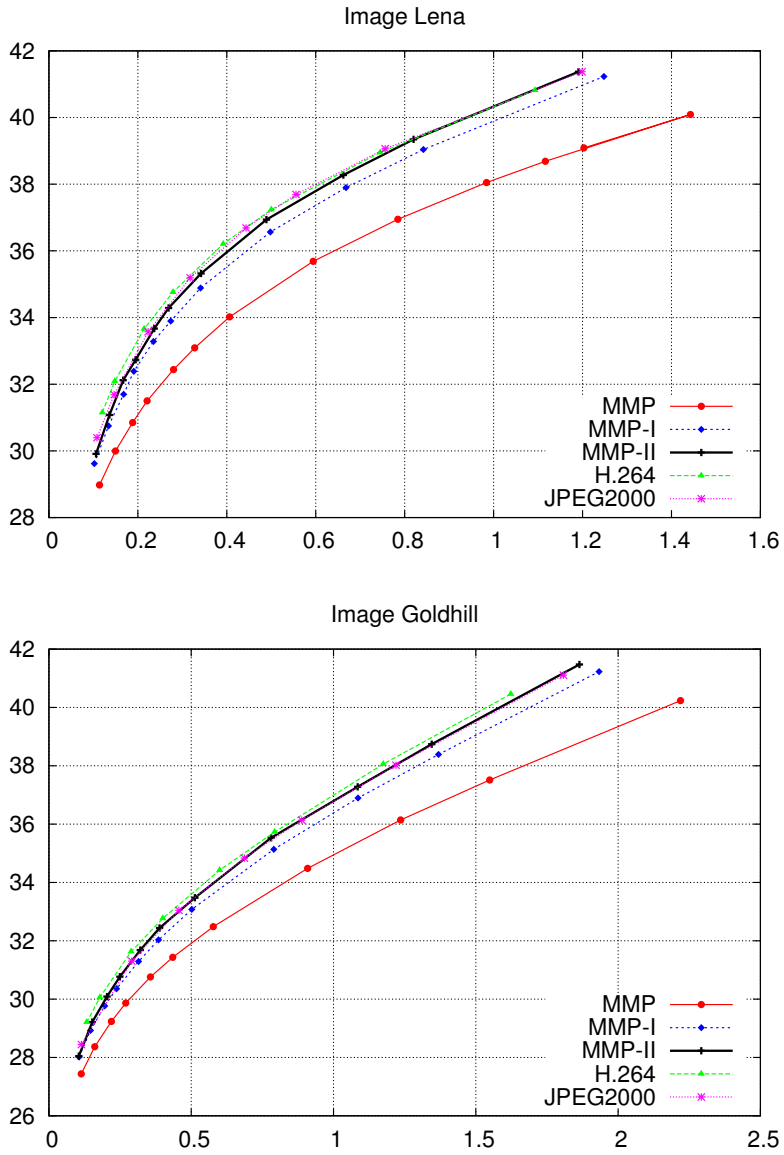


Figure 6.40: Experimental results for MMP-II, for images Lena and Goldhill.

smooth image Lena (Figure 6.42[12]), we observe perceptual gains for MMP-II over MMP-I. Smoother images are produced, with a more accurate representation of areas with a slow variation in luminance. We also notice that the perceptual quality achieved by MMP-II is now very close to that of JPEG2000 and H.264/AVC. JPEG2000 introduces a slight blurring effect, while MMP-II still suffers from mild blocking artifacts. H.264/AVC is able to avoid these blocking artifacts, trough the use of its post-filtering technique [83].

[12]Refer to Figures C.29 to C.31 of Appendix C for a representation of the entire images.

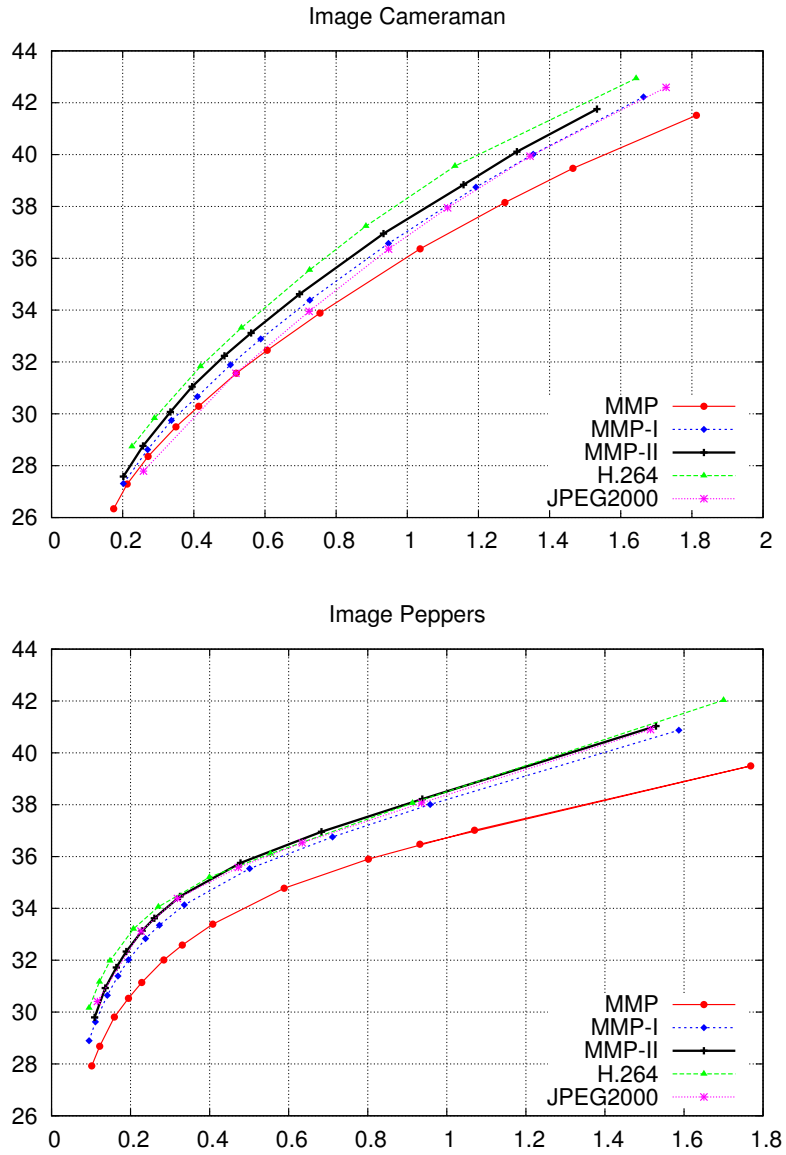


Figure 6.41: Experimental results for MMP-II, for images Cameraman (256×256) and Peppers.

For text image PP1205 (Figure 6.43[13]), the perceptual quality advantage of MMP-II is easily observed. When compared with MMP-I, the use of the new dictionary adaptation techniques cause a reduction in some artifacts that appeared in the background areas located near the text characters. JPEG2000 introduces some disturbing ringing and blurring artifacts as a consequence of the quantisation of the DWT coefficients. This tends

[13]Refer to Figures C.32 to C.34 of Appendix C for a representation of the entire images.

(a) MMP-I (36.57 dB)

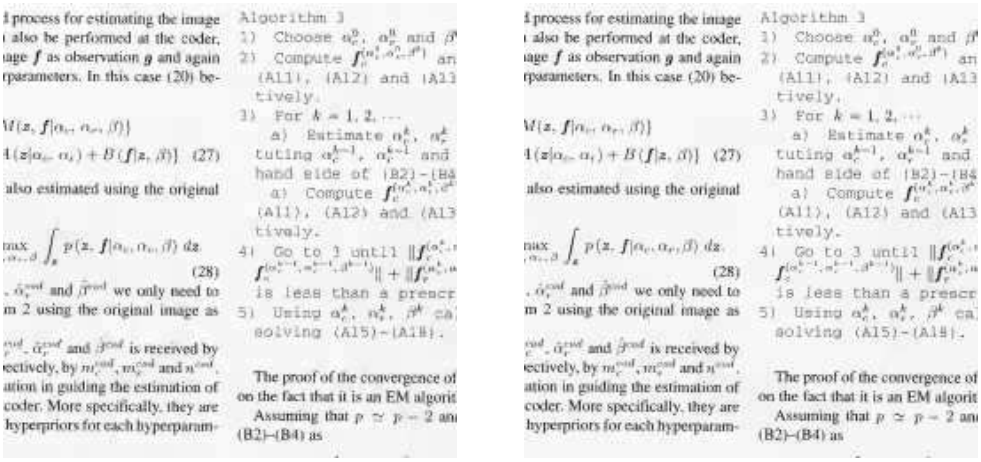
(b) MMP-II (36.94 dB)

(c) JPEG2000 (37.15 dB)

(d) H.264/AVC (37.24 dB)

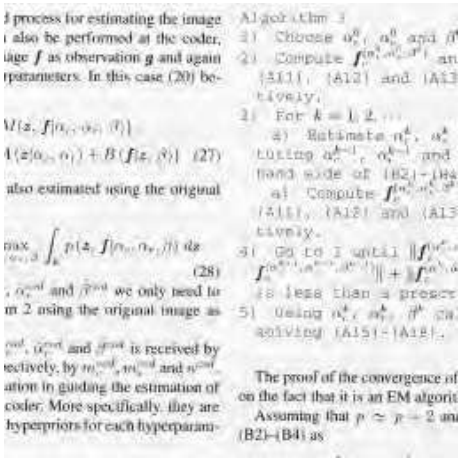
Figure 6.42: Detail for image Lena compressed at approximately 0.5 bpp using a) MMP-I; b) MMP-II; c) JPEG2000; and d) H.264/AVC.

to eliminate higher frequency coefficients of the DWT image spectrum, resulting in the well known Gibbs phenomenon for the very steep transitions on the signal. H.264/AVC efficiently avoids the ringing artifacts. This results from the higher adaptability of this block-based encoder and, once again, from the use of the post-processing adaptive deblocking filter. Nevertheless, we observe a clear perceptual advantage of MMP-II in terms of the distortion in the image's characters.

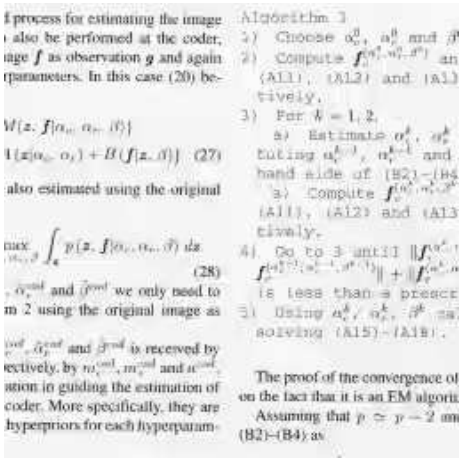


(a) MMP-I (30.89 dB)

(b) MMP-II (32.74 dB)



(c) JPEG2000 (27.21 dB)



(d) H.264/AVC (28.27 dB)

Figure 6.43: Detail for image PP1205 compressed at approximately 0.65 bpp using a) JPEG2000 and b) H.264/AVC.

6.8 Conclusions

In this chapter we have described several new dictionary adaptation techniques for multiscale recurrent pattern matching image coding. The use of these methods in MMP-I resulted in an enhanced coding scheme that we refer to as MMP-II. Each of the methods described in this chapter explores some feature of MMP-based encoding, in order to improve its efficiency. One example of this is the use of context conditioning for the adaptive arithmetic encoder (see Section 6.1), that achieves a better coding performance by improving the lossless compression of the MMP dictionary indexes' symbols. Other useful

method introduces a L^1-norm equalisation procedure (see Section 6.5), that changes the new dictionary patterns according to a probability distribution criterion, thus increasing the coding efficiency of the algorithm.

The combination of the several dictionary adaptation methods also results in some interesting synergies. One example is the combination of the redundancy reduction techniques, described in Section 6.2.2, with the enhanced dictionary updating strategies (see Section 6.3), that add extra blocks to the dictionary. In the first case we try to selectively remove dictionary blocks that are not useful in improving the matching distortion. This reduces the indexes' average entropy, and increases the performance of the encoder. In the second case, we aim to over-populate the dictionary space, by introducing new code-vectors that will help in future block approximations. The joint use of these processes results in an efficient compromise between their two antagonistic goals, identified in the experimental study presented in Chapter 3. In this analysis we observed the disparity between the dictionary size and the number of used blocks, but also the advantage of using fast growing dictionaries.

Other methods were also developed, to eliminate unnecessary dictionary vectors. These methods, presented in Section 6.4, have proved to be advantageous, allowing a significant reduction in processing time, with only negligible performance losses.

One interesting point is that, in spite of the obvious connection of these dictionary adaptation methods with MMP-based algorithms, most of the proposed schemes are generic enough to be easily used with other adaptive pattern matching algorithms.

In terms of coding performance, MMP-II is able to achieve quality gains over both MMP and MMP-I for every tested image and all compression ratios. For non-smooth images, these gains further increase the advantage of MMP-based algorithms over the state-of-the-art, transform-based encoders. For smooth images, the methods proposed in this chapter allow MMP-II to reach a performance level similar to that of the JPEG2000 algorithm.

In terms of perceptual quality, MMP-II suffers from some blocking artifacts, that result from the independent optimisation of each of the image blocks. The use of an adaptive deblocking filter justifies some of the perceptual advantage of the H.264/AVC algorithm over MMP-II. In the next chapter we study a post-processing deblocking scheme for MMP-II, along with some other techniques that aim at further improving the method's performance.

Chapter 7

Further improvements to MMP-II

In the previous chapters we have described new schemes that are able to significantly improve the performance of the MMP algorithm, especially for natural images. The MMP-I algorithm, described in Chapter 5, uses adaptive predictive techniques to exploit the spatial redundancy of the images. MMP-II achieves its performance gains through the use of enhanced dictionary adaptation methods, that were described in Chapter 6.

In this chapter we investigate other ways to improve the performance of MMP-II, inspired by some observations described in previous chapters. In Section 7.1 we study the prediction process used by MMP-II. The performance of the current prediction modes is evaluated and the usefulness of some new methods is tested. In Section 7.2 we describe a post-processing deblocking method for MMP-II, that increases both the perceptual and the objective quality for smooth images. This method is based on the MMP deblocking scheme, previously described in Section 4.3.3. Tests revealed that the original method leads to severe losses in both objective and subjective image quality, when applied to MMP-II. The new deblocking process not only corrects these issues, but also enhances the compression performance MMP-II.

7.1 The prediction process

MMP-II uses a set of prediction modes inspired by the ones defined for H.264/AVC [13]. A discussion on the prediction process used by MMP-II was presented in Section 5.2.3. We now return to this subject, motivated by some previous observations on the MMP-II encoding process, in search for changes that could lead to a better encoding performance

or a reduction of the computational complexity of the method.

The use of several prediction modes aims a reduction of the predicted error signal's activity. Nevertheless, each used prediction mode adds complexity to the method, because it represents one more option that has to be processed by the encoder. Therefore, the removal of prediction modes that do not lead to any performance gain will be beneficial in terms of computational complexity. In Section 7.1.1 we test several configurations for this purpose. In Section 7.1.2 we investigate the effects of using different block sizes for prediction, as a way to improve the performance of MMP-II, especially for higher rates. Finally, in Section 7.1.3 we study the effects of using a new prediction mode, based on an algorithm originally proposed for texture synthesis, called template matching.

7.1.1 Altering the available prediction modes

In this section we study the effects of disabling some of the prediction modes. The objectives of this study are twofold: first, we intend to further understand the impact of the used prediction modes in the performance of MMP-II; second, we are searching for nonessential methods, that may be removed with no loss of encoding performance. From the analysis of the prediction mode usage, like the one performed in Figure 5.15 of Section 5.3.1, no obvious conclusions could be reached. Therefore, we have used the knowledge gathered along the previous sections of this work, to make some educated guesses on what would be good policies for altering the set of used prediction modes.

The importance of the MFV mode became evident from the discussion presented in Section 5.2.3. Therefore we tried to eliminate some directional prediction modes. From Figure 7.1 we notice the use of several prediction directions that are located in the same angle quadrant. In our tests we have eliminated the prediction modes that are represented by the dashed lines, creating a new version of the encoder that we refer to as MMP-II Pred A. This was done because we hypothesised that if one of these directions should be necessary, it could be replaced by a neighbour orientation. As an example, a prediction block using mode 7 could eventually be replaced by the usage of either mode 3 or mode 0.

MMP-II uses nine prediction modes for all block scales except scale 8, *i.e.* 16×16 blocks, for which only four modes are used. The restriction of the prediction modes at this higher scale is justified by the smaller accuracy of the prediction process for larger blocks. This means that the extra complexity caused by the use of additional modes is expected

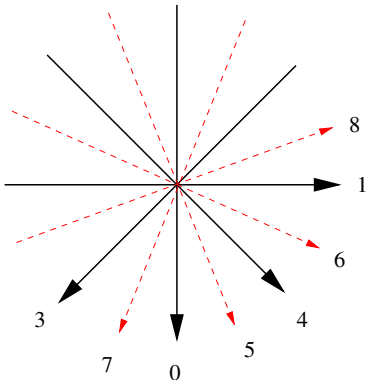


Figure 7.1: The prediction directions used by the MMP-II prediction modes.

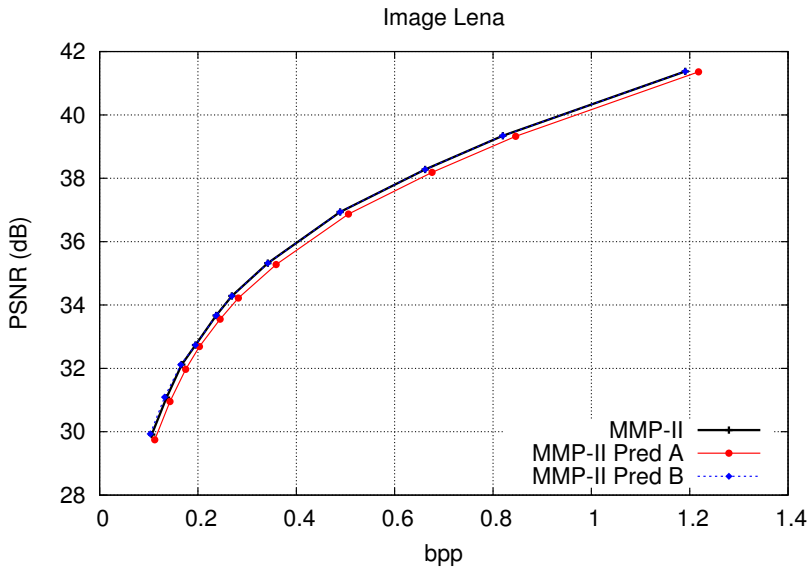


Figure 7.2: Effects of changing the MMP-II prediction modes, for image Lena.

to have no noticeable return in terms of quality gains. An investigation was conducted in order to determine the influence of this procedure in MMP-II. In this test, that we refer to as MMP-II Pred B, the use of all available prediction modes for 16×16 blocks was investigated.

Figures 7.2 and 7.3 show the results of the described MMP-II Pred A and MMP-II Pred B tests[1]. The experimental results show that eliminating some of the directional prediction modes has a negative impact on the performance of MMP-II, especially for smooth images. On the other hand, the use of additional prediction modes for MBs of scale

[1] The results for other images are presented in Figure C.35.

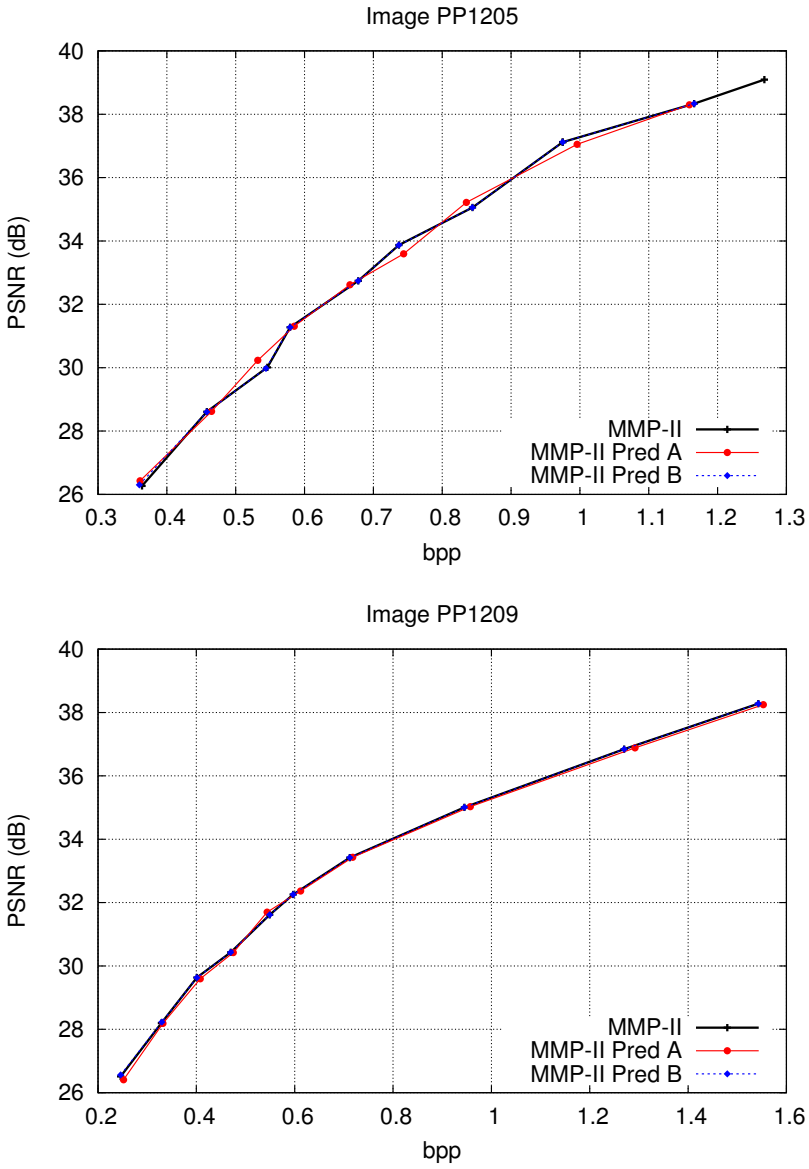


Figure 7.3: Effects of changing the MMP-II prediction modes, for images PP1205 and PP1209.

8 introduces almost unnoticeable gains, but only for smooth images and high compression ratios, due to the use of adaptive block sizes on MMP-II (see Section 6.4.2). For text images one notices that the performance of the encoding process is almost unaltered by the changes in the prediction process, which has a smaller influence than for the case of smooth images.

Other tests were implemented regarding the prediction modes used in MMP-II. In one test (MMP-II Pred C), we only used the MFV prediction mode for 16×16 blocks. In

another test (MMP-II Pred D), only the first three modes (vertical, horizontal and MFV) were used for scales 16×16 and 16×8 blocks, assuming that the Plane prediction mode could be efficiently replaced by one of these modes. The results for these tests[2] showed only negligible changes in the encoding performance for all image types.

The previous tests show that some computational complexity reduction may be attained by restricting the used prediction modes. The observed impact on coding performance was negligible when the restriction was made only for the blocks of the higher scales. Therefore, we confirmed the consensus that led to the choice of prediction modes currently used by the MMP-II prediction scheme. Future investigations could be carried out to determine if eliminating some prediction modes is a favourable method to achieve a computational complexity reduction, at the cost of some losses in coding efficiency.

7.1.2 Altering the prediction block sizes

In the definition of the prediction stage of MMP-I we have used the same convention as H.264/AVC, that uses prediction only for block partitions with at least 4×4. In H.264/AVC, this threshold was set because this block size would result in a sufficient accuracy for prediction and because of the use of 4×4 blocks for DCT-like transform. In MMP-I and MMP-II we maintained this threshold, inspired by the H.264/AVC convention. Nevertheless, in the analysis of the prediction process (see, for example, Figures 5.16 and 5.17 and the corresponding discussion, in Section 5.3) one notices that MMP-I often uses 4×4 blocks in the prediction step. This means that it could be advantageous to decrease the minimum size of the prediction blocks used for the prediction step. The potential benefits of this change in terms of encoding performance come from a more accurate prediction. This would result in a more regular prediction error that would be more efficiently compressed by MMP.

In order to test this hypothesis, we implemented several versions of the MMP-II encoder using different values for the minimum prediction block size. Experimental results showed that a minor gain is in fact achieved by this change in the prediction process. Nevertheless, this gain is only relevant for smooth images, since for other image types the prediction process does not play an equally important role in the compression efficiency. Also, the

[2]These results present only negligible variations when compared with the original method and are therefore presented only in Figures C.36 and C.37.

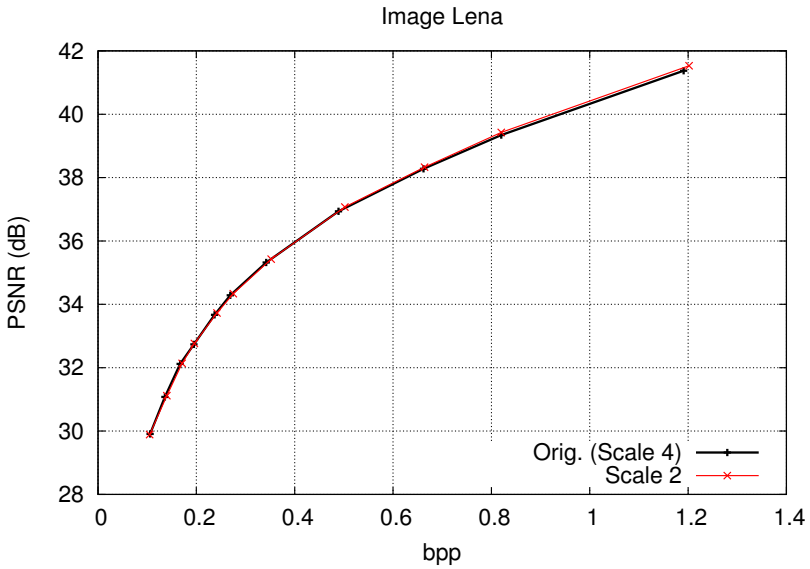


Figure 7.4: Effects of changing the minimum block scale used in the prediction stage of MMP-II, for image Lena.

gain is only observed for higher rates, since for smaller rates the encoder cannot afford the small prediction block sizes. Experimental results for several tested scales were analysed, in search for a compromise between performance gains and computational complexity. The use of scale 2 (2×2 blocks) as the minimum prediction scale was chosen. Figures 7.4 and 7.5[3] show the experimental results for this case. In this figure we can see that PSNR gains occur only for the medium to the higher rates of the smooth and compound images. For text images we have an equivalent performance. The small gains observed for these tests demonstrate that the initial MMP-I threshold is a good compromise between compression efficiency and computational complexity.

7.1.3 Block prediction with template matching

The previous section revealed that the use of a more flexible prediction process generally leads to a better encoding performance for MMP-II. The removal of some of the original prediction modes led to performance losses, while the use of new block sizes for prediction allowed for some PSNR gains. In this section we study the introduction of a new prediction method, as a way to improve the flexibility of the prediction process.

[3]The results of this test for other images are presented in Figure C.38.

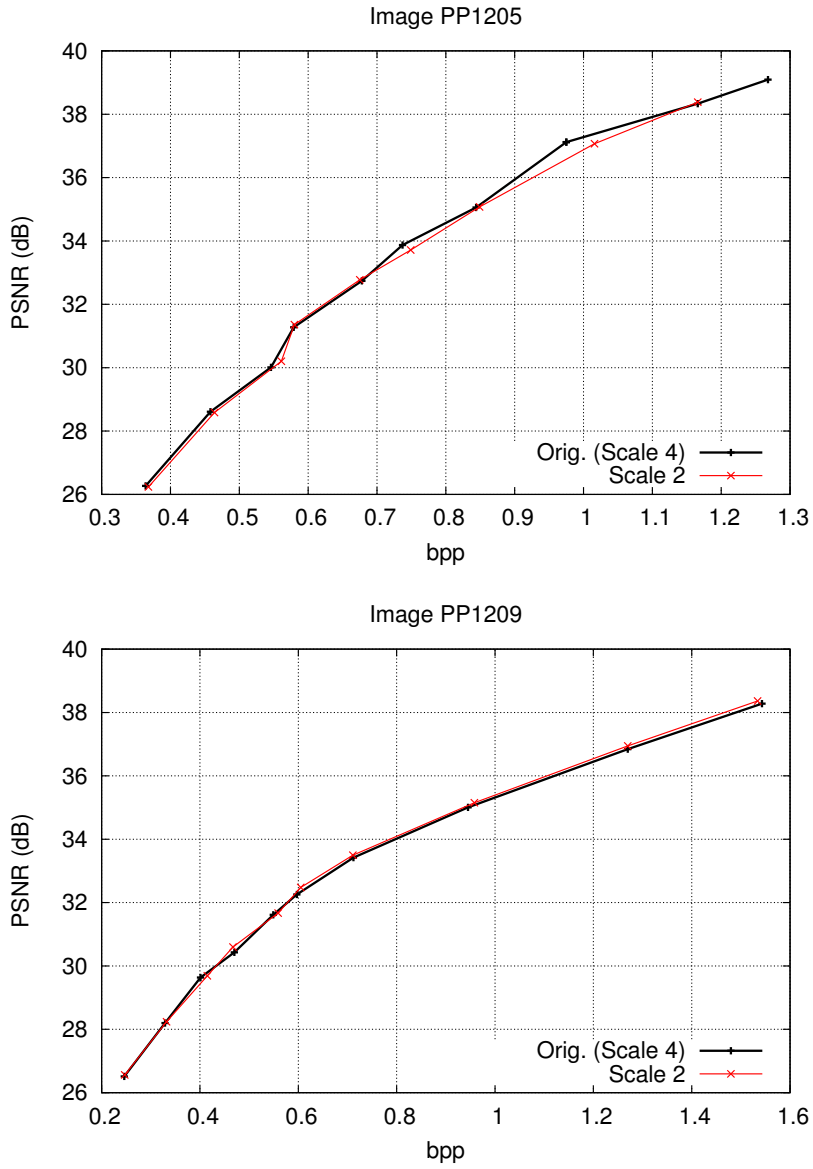


Figure 7.5: Effects of changing the minimum block scale used in the prediction stage of MMP-II, for images PP1205 and PP1209.

The efficiency of the H.264/AVC prediction process is widely recognised and its modes have proved their efficiency [61, 75]. Nevertheless, two problems were observed [84]: first, they often loose efficiency in predicting blocks with complex texture and second, pixels of the current target block that are positioned far from the prediction borders, and are thus less correlated with the prediction values, are usually poorly predicted. Because of this, the new prediction method uses a different prediction paradigm from those of the H.264/AVC prediction modes. It is based on template matching (TM). TM was originally

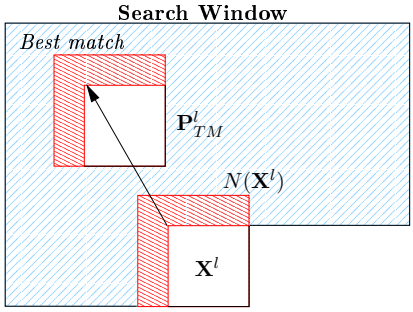


Figure 7.6: Intra prediction using template matching.

proposed as a texture synthesis algorithm [85], but has been successfully applied to a slightly different goal, namely spatial prediction in a video encoding framework. Previous works describe advantageous implementations of this method as inter-frame [86] and intra-frame [84] prediction schemes, in a H.264/AVC video encoder.

When using template matching for intra prediction, the current image block, $\mathbf{X}^l$, is estimated by comparing its spatial neighbourhood, $N(\mathbf{X}^l)$, with all identical regions in an adjacent area of the reconstructed portion of the image. $N(\mathbf{X}^l)$ is called the *template* of $\mathbf{X}^l$ and is composed by its causal neighbouring pixels. The search for a best match for the pixels of $N(\mathbf{X}^l)$ returns a position in the causal search window. The block of the reconstructed area of the image with the most similar neighbourhood to the template $N(\mathbf{X}^l)$ is assigned as the prediction block, $\mathbf{P}^l_{TM}$ (see Figure 7.6). The sum of absolute difference (SAD) is used to access the best match between the pixels of the candidate neighbourhood and the block template.

Template matching prediction can be related with motion compensation using previously reconstructed pixels from within the intra picture, but with the advantage of not needing additional overhead associated with the motion vectors. The prediction step is performed in the encoder and the decoder, that uses the same search area of the reconstructed image to determine the same prediction block used by the encoder.

In [84], the use of TM for intra prediction in a H.264/AVC encoder is proposed using a fixed block size of 4×4, but applying template matching prediction on four 2×2 target sub-blocks. The predictor block for the target 4×4 block is then made up from the concatenation of the four best match 2×2 candidate sub-blocks. Because MMP-II uses an adaptive block size for prediction, with blocks that range from 4×4 to 16×16, several tests were performed regarding the use of sub-blocks for the template matching

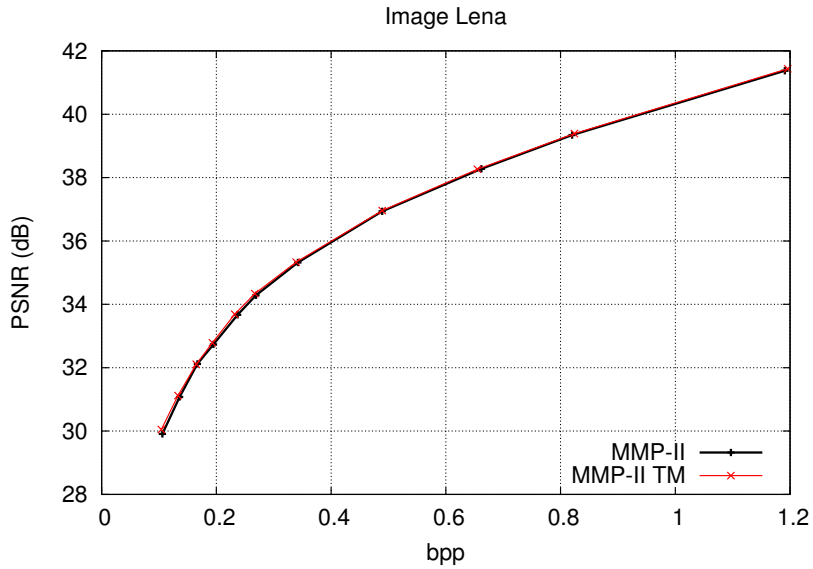


Figure 7.7: Effects of using template matching in the prediction stage of MMP-II, for image Lena.

step. These experiments showed that the best results for MMP-II are achieved when no sub-blocks are used, *i.e.* when the used template corresponds to the entire predicted block, independently of its size. A search region of 24 columns and 32 rows, centred around the current block (see Figure 7.6), was used for all prediction block scales. Experimental tests showed that such a search region represents a good compromise between prediction efficiency and computational complexity.

Figures 7.7 and 7.8[4] compare the compression performance of MMP-II with and without the use of template matching prediction. A small but consistent increase in the PSNR level for smooth and compound images can be observed, when one uses TM prediction. For text images, the use of the new prediction mode has an equivalent performance to that of the original MMP-II method.

7.2 Efficient deblocking for MMP-II

The experimental results of the MMP-II encoder, presented in Section 6.7, show relevant improvements on both the objective and subjective qualities of the encoded image, when compared with the original method. Nevertheless, MMP-II compressed images still suffer

[4]The results of this test for other images are presented in Figure C.39.

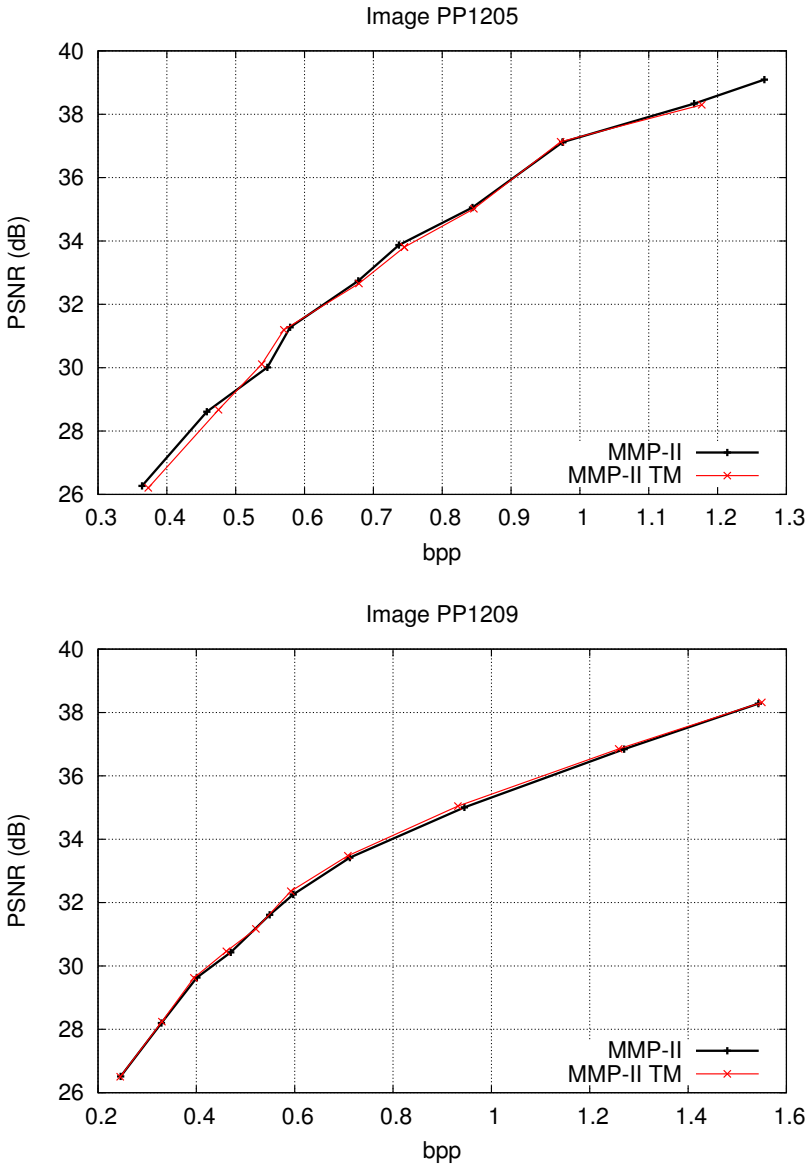


Figure 7.8: Effects of using template matching in the prediction stage of MMP-II, for images PP1205 and PP1209.

from some some blocking artifacts, especially for smooth images. In this section we describe an adaptation of a deblocking scheme that was originally proposed for MMP in [1, 2, 65]. A study of the original method revealed some unexpected drawbacks, that compromise its efficiency when we use it with MMP-II. This motivated the investigation of several improvements to the original method, that not only enable its use on MMP-II, but also allow for both objective and subjective quality gains for smooth images. At the same time, the limitations of the original scheme, that introduced highly destructive artifacts on

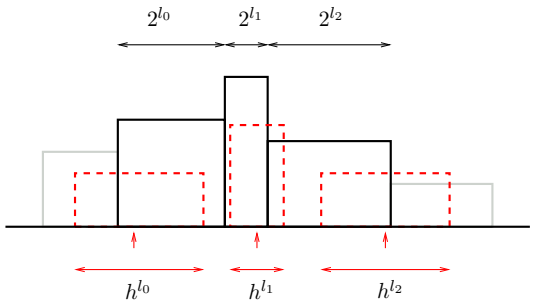


Figure 7.9: Use of FIR filters with adaptive support according to the MMP block scale.

non-smooth images, are also eliminated.

The original (MMP) deblocking method was described in Section 4.3.3, where we related it with an MMP synthesis scheme that uses overlapping basis vectors. As we have seen, the method consists on the use of a two-dimensional FIR kernel on each block of the reconstructed image. For each image block the deblocking filter's dimensions are successively adapted, by using information about the scale used by MMP to approximate the corresponding image area. This causes blocks of larger scales to be filtered by a kernel with a larger support region, while blocks of smaller scales use shorter smoothing filters (see Figure 7.9, repeated here for convenience). Large scale blocks tend to be used in smooth image areas, so the filtering with longer, more smoothing kernels is appropriate. On the other hand, blocks of smaller scales are used on image areas with higher activity. The use of shorter filters attempts to preserve the detail of these regions.

In [1, 65], a rectangular kernel with length $N + 1$ was used for the deblocking, where N is the length of the block that is being filtered. This resulted in an improved objective performance, due to the reduction of the blocking artifacts, that was accompanied by a decrease on the PSNR values for the reconstructed image. In [2], the use of a Gaussian impulse response was proposed. This method reduces the blocking artifacts, increasing the perceptual quality for smooth images, but also introduces PSNR gains. Nevertheless, both methods have one important drawback: they introduce highly disturbing blurring artifacts in non-smooth images, that result in a severe decrease in both subjective and objective quality. This fact limits the applicability of this filter, because there is no practical way of avoiding the blurring of non-smooth images.

In order to assess the performance of the original method, we tested both these schemes

in MMP-II. Both methods lead to a severe decrease in the objective quality of the reconstructed image. For the rectangular kernel, the drop in PSNR value was far larger than the one observed for MMP. When the Gaussian kernel was used, the PSNR gains observed for MMP were replaced by noticeable losses, that went up to almost 1 dB for the case of image Lena. An analysis of the filtered image revealed disturbing blurring artifacts, meaning that the filtering process was too strong. Also, unexpected artifacts were observed, that lead not only to PSNR losses but also to a decrease in the objective quality of the reconstructed image. In the next sections we describe the problems revealed by the use of the original scheme in MMP-II and propose a new method, that is able to avoid these problems and perform efficiently for MMP-II.

Adapting shape and support for the deblocking kernel

In order to increase the performance of the original methods for MMP-II, different kernels with various support regions for the deblocking filter were tested. Experimental results confirmed the advantage of using Gaussian kernels, as was the case for MMP [1]. Besides this, these tests also demonstrated that the quality of the deblocked image strongly depends on the dimensions of the support region of the used filter. In the original method, this support is set to $2^{l_k}+1$, where l_k is the MMP scale of the original rectangular pulse that is used to approximate the current image block. We varied this value and discovered that it is in general the best option for the running average filter, but this is not the case when we use a Gaussian kernel.

Instead of explicitly adjusting the support region of the Gaussian kernel, we have set the filter length to a constant value (the original $2^{l_k}+1$ samples), but adapted the Gaussian's variance. This method produces filter kernels with different shapes.

Consider a Gaussian filter, with variance σ^2 and length L, with an impulse response (IR) given by:

$$g^{l_k}(n) = e^{-\frac{\left(n-\frac{L-1}{2}\right)^2}{2\sigma^2}}, \tag{7.1}$$

where l_k is the MMP scale, $L = 2^{l_k}+1$ and $n = 0, 1, ..., L-1$. The shape of the filter is controlled by changing a filter parameter α, that controls the variance of the Gaussian, by using the expression:

$$g^{l_k}(n) = e^{-\frac{\left(n-\frac{L-1}{2}\right)^2}{2(\alpha L)^2}}, \tag{7.2}$$

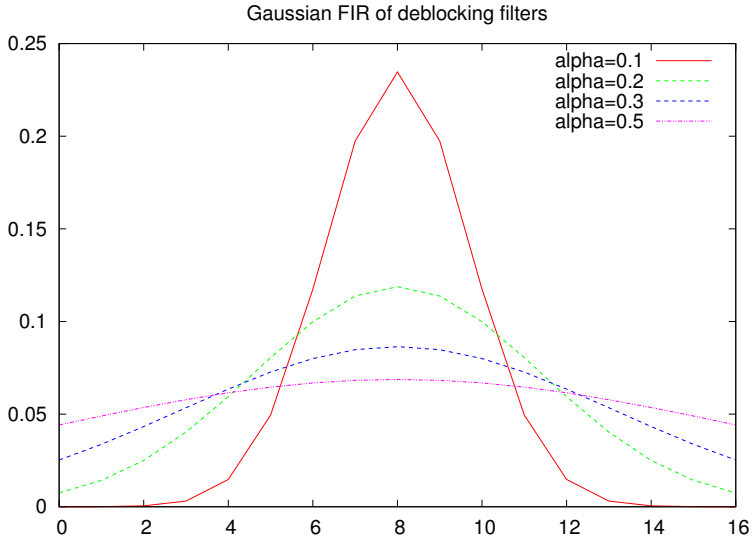


Figure 7.10: Shape of the impulse responses of the adaptive filters used for deblocking.

to determine the filter's impulse response.

Figure 7.10 represents the shape of a 17 tap filter, $g^8(n)$, for several values of parameter α. This figure clearly demonstrates the relation between the filter's shape and its approximate support. By varying the value of the α parameter, one is able to efficiently adjust the IR of the filter from a nearly rectangular filter, with a support region $2^{l_k}+1$, to a Gaussian filter with different lengths. Also, when α tends to zero, the IR of the filter becomes a simple impulse. This disables the deblocking effect, for those cases were it is not beneficial.

In order to chose the adaptive kernel that optimises the filtering result, the value of the parameter α is controlled by the MMP-II encoder. At the end of the encoding stage, the deblocking process is tested, using different values for the α parameter. Figure 7.11 shows the effects of using different values of α for the deblocking methods in the rate-distortion curve, for image Lena. Using a trial and error scheme, the encoder is able to determine the value of α that maximises the PSNR of the reconstructed image. This value is appended at the end of the encoded bit-stream, by using a 3 bit code, that corresponds to one of eight possible values for α, represented in table 7.1. This introduces a marginal computational cost in the encoder, as well as negligible rate overhead. The eight values of α were determined experimentally by using a set of several test images. Code *000* corresponds to $\alpha = 0$, meaning that no filtering should be applied to the decoded image.

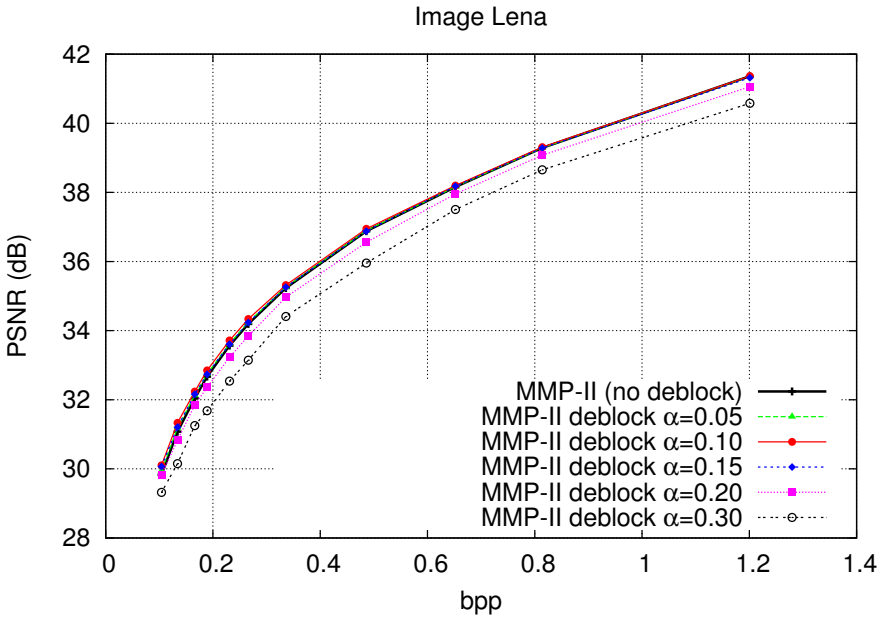


Figure 7.11: Effects of the MMP-II deblocking method on the image's PSNR, when different values of parameter α are used, for image Lena.

Value of α	Binary code
0	000
0.05	001
0.10	010
0.15	011
0.20	100
0.25	101
0.30	110
0.40	111

Table 7.1: Three bit binary codes used to represent the parameter α.

Eliminating artifacts introduced by the deblocking process

The original method only takes into account the dimensions of the block currently being filtered to set the filter support. Nevertheless, the filter may be used across several neighbouring blocks. In our investigation we noticed that this may introduce an unexpected artifact, for example when wide and narrow blocks, with very different intensity values, occur in neighbouring regions. This case is represented in Figure 7.12, where a wide dark

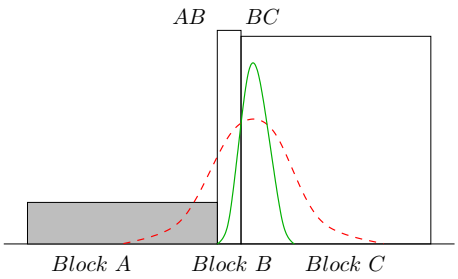


Figure 7.12: The concatenation of blocks with very different supports and pixel intensities causes the appearance of an image artifact after the deblocking filtering.

block, A, is concatenated with two bright blocks: one narrow block, B, followed by one wide block, C. When blocks A and B are filtered, a smooth transition appears that eliminates the blocking effect in the AB border. When the block C is filtered, because the used filter has a very wide support region, the pixels near the BC border will suffer from the influence of some of the dark pixels of block A. This causes a dark "valley" to appear in the BC border, that introduces a visible artifact in the deblocked image.

In order to avoid these artifacts, the new method controls the filter length, so that the deblocking filter does not take into consideration pixels that are neither from the current block nor from its immediate neighbours. In the example of Figure 7.12, the length of the filter used in the C block's pixels that are near the BC border is limited, so that the left most pixel that is used in the deblocking is always the first (left most) pixel of block B, eliminating the described artifact. In Figure 7.12, this means that the new method uses the filter represented by the solid line, instead of the original one, represented by the dashed line.

Another artifact caused by the original method is the introduction of smooth variations in regions of the image that originally have very steep transitions (from low to high pixel intensity values, or vice versa). This problem was corrected by monitoring the differences in the frontiers' pixels' intensities, in order to avoid filtering steep variations that do not correspond to blocking artifacts. This is again controlled by the MMP-II encoder, using an adaptive method. A step intensity threshold, s, is used. This value corresponds to the maximum intensity difference between the two border pixels for which filtering is still performed. This process is represented in Figure 7.13, where two blocks A and B with very different intensity values are concatenated. In this case, the AB border is only filtered if the absolute difference between the border pixels is inferior to the defined value for s, *i.e.*

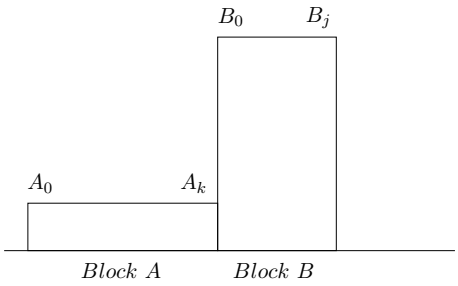


Figure 7.13: A steep variation in pixel intensities may be a feature of the original image, that should not be filtered.

Threshold s	**Binary code**
0	000
16	001
32	010
64	011
96	100
128	101
192	110
255	111

Table 7.2: Three bit binary codes used to represent the parameter s.

if $|A_k - B_0| < s$.

The value of s is chosen in order to maximise the PSNR for the image that is being deblocked. The encoder tests a set of different values and transmits the code corresponding to the chosen step threshold. A three bit code is used to represent eight possible values for s, determined experimentally. The code assignment is described in table 7.2. In this process $s = 0$ corresponds to no filtering and $s = 255$ corresponds to the case where all blocks are filtered.

7.2.1 Experimental Results

Experimental tests were performed using the new and the original deblocking methods. Figure 7.14[5] presents a detail of image Lena, encoded using MMP-II, and compares it with

[5]The entire images corresponding to these cases are presented in Figures C.40 and C.41. Figure C.42 presents the deblocking results for image Peppers.

(a) No deblocking (31.08 dB)

(b) MMP deblocking rectang. [1] (28.94 dB)

(c) MMP deblocking Gaussian [2] (30.25 dB)

(d) MMP-II deblocking ($\alpha = 0.10$ and $s = 256$) (31.35 dB)

Figure 7.14: A detail of image Lena, encoded with MMP-II at 0.135 bpp.

the results of using the original deblocking techniques, referred to as MMP deblocking. Perceptually, we can observe that the new deblocking filter is able to efficiently eliminate the blocking artifacts in Lena's face and hat, without compromising the image quality at regions with high detail, like Lena's hair and the hat's feathers. In this case, the used filter has $\alpha = 0.10$ and $s = 255$. When compared with the original deblocking methods, we can observe that the blurring artifacts that are noticeable in Figures 7.14 b) and c) (especially in the areas with finer details) are avoided. The perceptual advantage of using Gaussian kernels instead of rectangular ones is also clear from these figures.

In Figures 7.14 b) and c) we can also observe the first type of artifacts, explained in the previous section. They appear in Lena's shoulder, where the described "dark valleys" are easy to observe. We can see that the proposed method efficiently eliminates these artifacts.

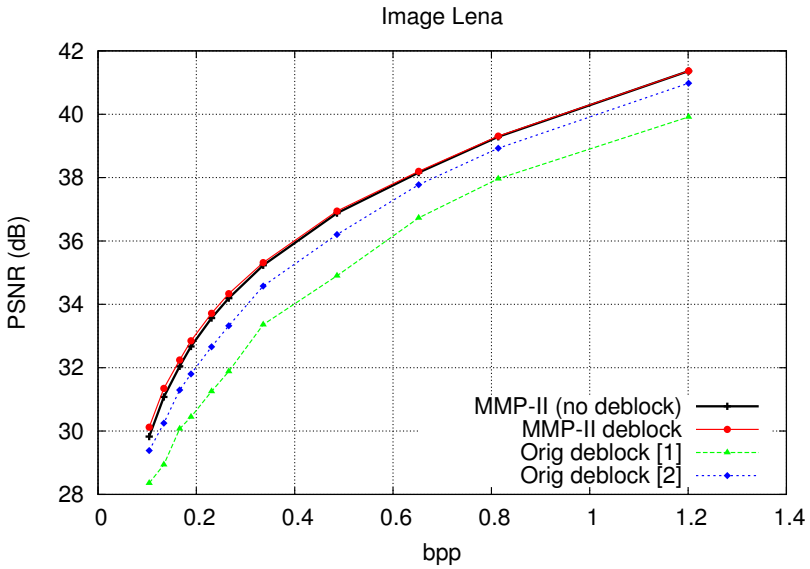


Figure 7.15: Objective quality gains for image Lena (adaptive deblocking filter used with $\alpha = 0.10$ and $s = 256$).

These figures also show why the methods developed originally for the MMP encoder suffer from a unexpectedly high performance loss, when used with MMP-II. Because MMP-II uses predictive coding, the used dictionary blocks approximate *residue* patterns. In some cases, where the prediction step is particularly efficient, some detailed areas are approximated by large, uniform, residue blocks and a detailed prediction block. In this case, the deblocking process uses a wide filter to deblock an image area that is not necessarily smooth. When this happens, the use of the original deblocking method originates noticeable artifacts, like the one observed in Lena's lip. This results in a severe reduction both in subjective and objective quality measures, thus compromising the performance of the original method for MMP-II. However, due to its adaptability, the new MMP-II deblocking procedure does not seem to suffer from this disturbing factor. This can be observed in Figure 7.15[6], that shows the rate-distortion performance of the described methods, for image Lena. Figure 7.16 highlights the PSNR quality gains introduced by the new deblocking method. As would be expected, these gains are more relevant for higher compression ratios, where the blocking artifacts are more noticeable.

We also performed experimental tests using non-smooth images, like text image PP1205 and compound image PP1209. The use of any of the MMP deblocking strategies in text

[6]The corresponding results for image Peppers are presented in Figure C.43.

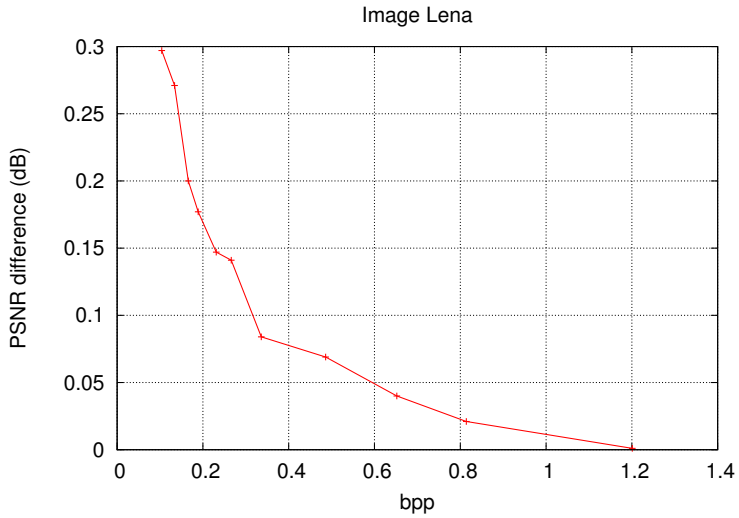
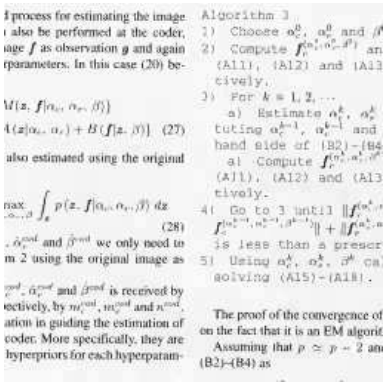


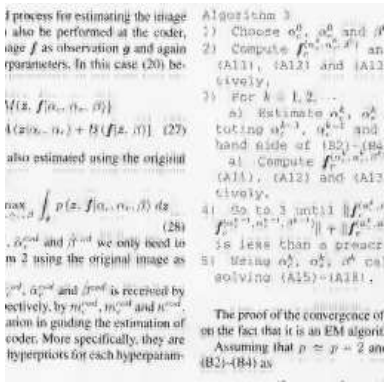
Figure 7.16: PSNR gains for the MMP-II deblocking method, for image Lena.

regions introduces highly disturbing blurring artifacts. These artifacts are noticeable in Figure 7.17, that shows a detail of the text image PP1205 processed using the various deblocking methods. However, the use MMP-II deblocking enables the encoder to adapt the filtering strength of the kernel. By setting the α parameter to 0, the encoder is able to turn off the deblocking process, thus eliminating the blurring artifacts from the reconstructed image, as may be observed in Figure 7.17. This feature solves one of the main drawbacks of the original deblocking methods.

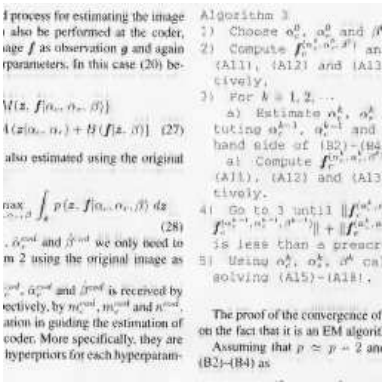
When the deblocking schemes are used for compound images we get a combination of the previously described effects: an improved quality is observed for the smooth areas of the image, but with a loss of sharpness in the text regions. Nevertheless, the optimisation procedure used on the MMP-II encoder always chooses the best parameter set for the deblocking filter, avoiding the PSNR losses that were introduced by the MMP deblocking methods. Figure 7.18 presents the perceptual results for compound image PP1209. In this case, the MMP-II deblocking uses $\alpha = 0.05$ and $s = 32$. These values cause a very mild deblocking effect, that is not able to eliminate the blocking artifacts on smooth areas. Nevertheless, stronger deblocking procedures would lead to noticeable quality losses in the text regions. As a comparison, Figure 7.18 e) shows the effects of using MMP-II deblocking with sub-optimal parameters, in terms of PSNR. From this figure it is clear that the post processing method is able to efficiently reduce the blocking artifacts of the smooth regions without compromising the quality of the text regions.



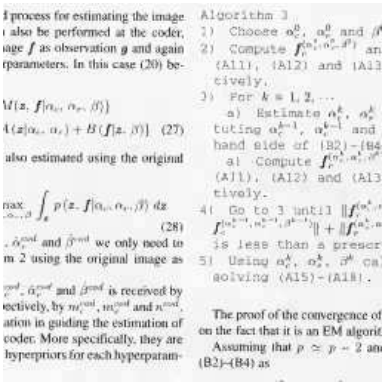
(a) Original image (no deblocking) (30.22 dB)



(b) MMP deblocking rectang. [1] (26.15 dB)



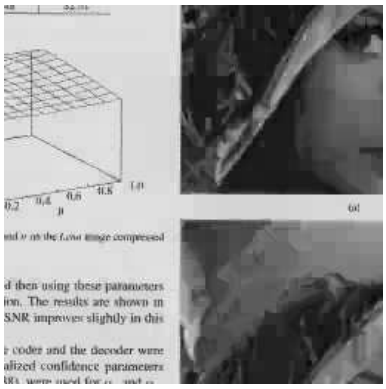
(c) MMP deblocking Gaussian [2] (28.42 dB)



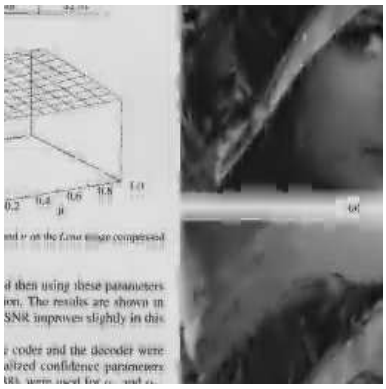
(d) MMP-II deblocking ($\alpha = 0$) (30.22 dB)

Figure 7.17: A detail of text image PP1205, encoded with MMP-II at 0.54 bpp.

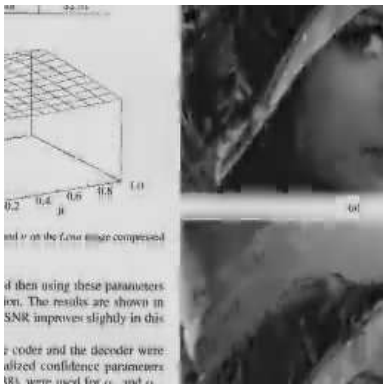
Figure 7.18 also exposes the second type of artifacts introduced by the original MMP methods, that were previously explained. These artifacts consist of a smoothing ramp in areas of the image that originally had an abrupt variation, like the boundaries of the smooth areas in image PP1209. Because the MMP-II deblocking method uses an adaptive step threshold, it is able to avoid the introduction of these artifacts. However, even if one ignores these artifacts for the original methods, the severe loss in subjective quality compromise the suitability of these algorithms as generic deblocking schemes. This is not the case for the new deblocking scheme.



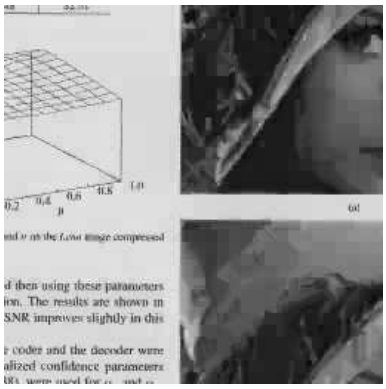
(a) Original image (no deblocking) (28.35 dB)



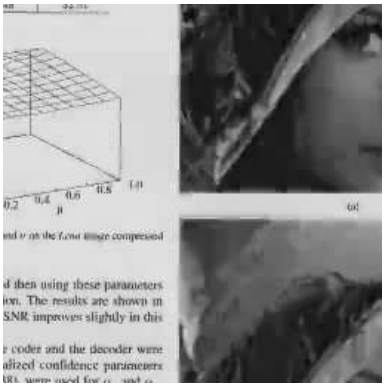
(b) MMP deblocking rectang. [1] (25.21 dB)



(c) MMP deblocking Gaussian [2] (26.72 dB)



(d) Optimum MMP-II deblocking ($\alpha = 0.05, s = 32$) (28.35 dB)



(e) Sub-optimum MMP-II deblocking ($\alpha = 0.15, s = 64$) (27.95 dB)

Figure 7.18: A detail of image PP1209, encoded with MMP-II at 0.32 bpp.

7.3 Conclusions

In this chapter we presented the results of a more detailed investigation on the prediction process used by the MMP-II encoder and discussed a new MMP-II post-processing deblocking method.

The study of MMP-II prediction determined the usefulness of a flexible prediction scheme, originally defined in Section 5.2.3, that is composed by eight directional prediction modes plus the MFV mode. Experimental tests showed performance losses when some of the directional modes are disabled for the smaller scales. Furthermore, a study of the MMP-II prediction block sizes, revealed that the use of intra-frame prediction for blocks down to 2×2 pixels allows a better performance than the use of the original 4×4 block size limit, inherited from the H.264/AVC encoder. Finally, the use of a different prediction mode, that is not based on directional prediction but on template matching, was also investigated for MMP-II, with favourable results.

The investigation of a deblocking method for MMP-II was motivated by the observation of some blocking artifacts, especially for highly compressed smooth images. The use of two deblocking methods originally proposed for MMP was studied for the MMP-II case, revealing several compromising inefficiencies. A new deblocking method was then investigated, based on the original MMP schemes. The MMP-II deblocking algorithm uses an adaptive FIR filter to process each image block. The filter's shape is adapted according to the image region that is being processed. Two filtering parameters are determined by the encoder, in order to maximise the deblocking performance for smooth images and eliminate the severe PSNR losses for non-smooth ones. The parameters are transmitted to the decoder using a negligible overhead. Experimental results show that the new deblocking method is suitable for any image type. Additionally, for the smooth regions, its use results in an improvement of both objective and subjective quality levels.

Chapter 8

Pattern matching-based video compression

In the previous chapters we observed the good performance of MMP for image coding, namely in the compression of the intra-predicted error signal. In this is chapter we investigate the use of the pattern matching and MMP paradigms in video compression.

Several decades of video coding standards have confirmed the hybrid model as the preferred architecture for video encoding algorithms. In these methods, a motion compensation stage reduces the temporal redundancy of the signal. The motion compensated residue is then compressed using traditional transform-quantisation-entropy coding methods, that efficiently exploit the data's spatial correlation.

This general architecture was maintained in the successful H.264/AVC [13] standard, that achieves more than twofold gains over its predecessors. A relevant fact is that these performance gains are not the result of a change of coding paradigm; they come mainly from the exploitation of a richer set of tools for each of the encoders' modules, resulting in a more complex, but highly efficient method [75].

As for the case of image compression, there have been rare proposals to adapt pattern matching-based algorithms to video coding applications. Some exceptions can be found in the work of [30, 52, 53, 87], but, to the authors knowledge, none of these methods has been able to achieve a performance near to that of current state-of-the-art methods.

In this chapter we develop two pattern matching-based approaches to video compression. In Section 8.2 we investigate a version of the H.264/AVC video compression standard

that uses no residual error encoding. Experimental results demonstrate that this technique can provide significant improvements in the performance of the original encoder for Bi-predictive (B) slices, especially at low to medium bit rates. A formal analysis of this process relates it to coding paradigms not usually associated with video coding, like vector quantisation and Lempel-Ziv encoders.

In Section 8.3 we investigate an MMP-based video encoding algorithm, named MMP-Video. It uses MMP to encode the motion compensated residue, in a hybrid video coding scheme. MMP-Video naturally inherited some of the dictionary updating strategies described in previous chapters. Nevertheless, a study on the particular aspects of using MMP for video compression also led to new dictionary design techniques for MMP-Video.

Both of the previous methods were developed based on the original H.264/AVC standard [13]. Because of this, this standard is briefly described in the next section.

8.1 The H.264/AVC video encoding standard

In this section we present a brief overview of the H.264/AVC video coding standard [13]. This overview is justified by the relevance of H.264/AVC video coding for the discussion of the algorithms described in this chapter. The following discussion is therefore limited to the most relevant aspects of the H.264/AVC encoding algorithm. For a detailed description of the H.264/AVC standard, the reader is advised to refer to one of the many available texts on this subject, namely [13, 14, 61, 76].

In spite of the ever growing available band-width and network speeds, the increasing number of services, as well as the growing popularity of high definition television, make the need for efficient video encoding algorithms as important now as it has ever been. This fact led to the combination of efforts from the two main video coding standards developing groups: the ITU-T's Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG). This unified the H.26x and the MPEG-x families of standards, that had been developed by these groups. As a product of this joint effort, the H.264/AVC video coding standard [13] emerged as the current state-of-the-art video compression scheme. The H.264/AVC standard defines a set of new coding tools, that improve its efficiency to about two times that of previous standards (namely the highly successful MPEG-2 standard [88]), at the cost of an increased computational complexity.

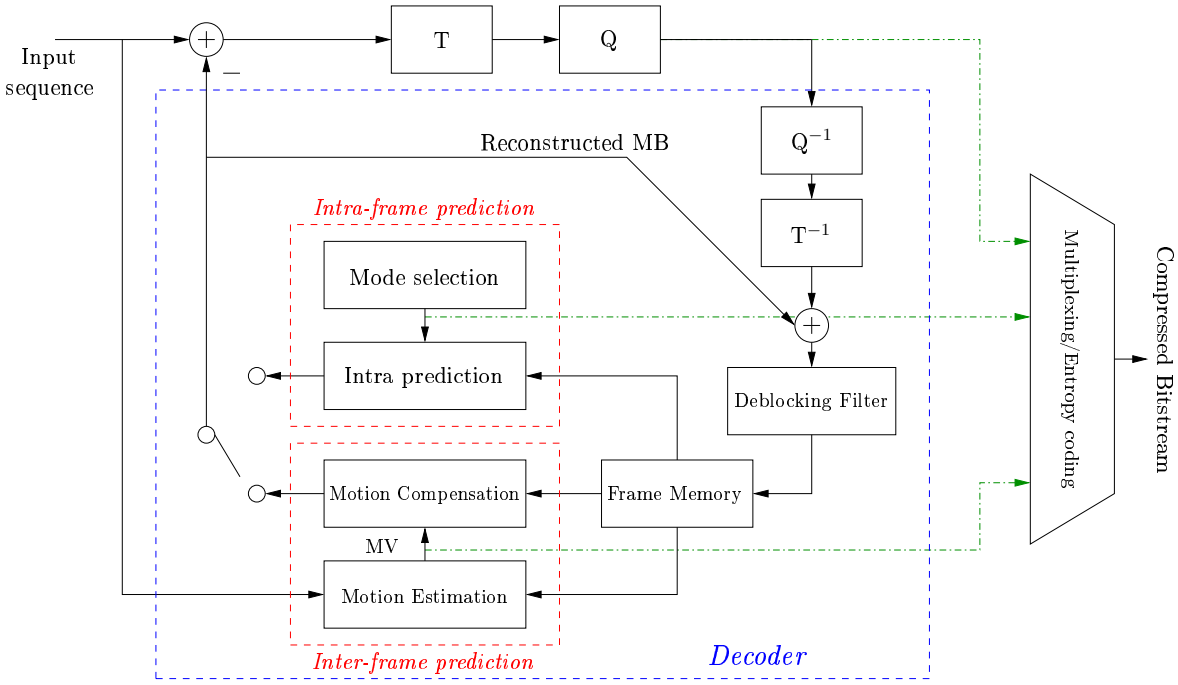


Figure 8.1: Architecture of an H.264/AVC video encoder.

In spite of the importance of the new H.264/AVC coding tools, this standard maintains the traditional hybrid coding architecture, as well as a transform-quantisation based encoding scheme to compress the motion compensated residue patterns. The hybrid video coding paradigm, represented in Figure 8.1, uses a combination of signal prediction and transform-quantisation-based encoding. Previous video encoding standards used mostly temporal prediction, by exploiting the similarities among consecutive sequence frames (inter-frame prediction) through the use of motion compensation (MC). H.264/AVC also uses a highly efficient intra frame prediction scheme, that uses several prediction modes to exploit the redundancies among the neighbouring pixels of a frame (see Section 5.2.3). A slice that uses only macroblocks coded with intra prediction is called an intra (I) slice.

The prediction process of H.264/AVC represents a very significant evolution in relation to the previous standards. For intra macroblocks (MB) a set of directional spatial prediction modes is used to estimate the image data. Improvements have also been introduced for inter-frame, or motion-compensated, prediction. The fixed MC block size has been replaced by a more flexible scheme, that uses adaptive block sizes. H.264/AVC allows for seven different segmentation modes of the motion compensated block, organised into two levels: in the first level, MBs may be partitioned into luma blocks with 16×16, 16×8, 8×16

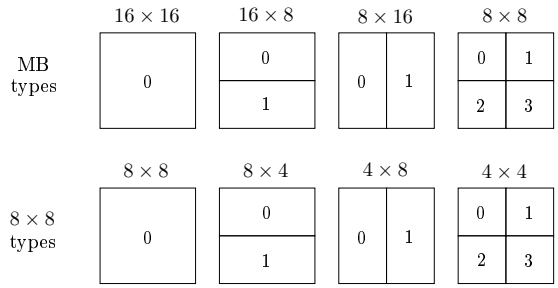


Figure 8.2: Adaptive block sizes used for partitioning each MB for motion compensation.

or 8×8 pixels; for 8×8 partitions, each block can be further divided into partitions of 8×4, 4×8 or 4×4 luma samples (second level). The segmentation sizes used for MC are represented in Figure 8.2. Independent translational motion vectors (MV) and reference frame indexes are assigned to each luma partition, meaning that each inter MB can be encoded using a number of MVs ranging from 1 (for a 16×16 partition) to 16 (when 4×4 partitions are used). Each MV can be associated to a different reference frame, among a set of pictures stored in a dedicated buffer. Besides the performance improvements resulting from the use of smaller blocks, each MV can be represented up to quarter-pixel accuracy, by using interpolation techniques to determine the half and quarter-sample positions. Moreover, MVs may point beyond picture boundaries, by repeating the edge samples before interpolation occurs. Additional details on this process may be found in [89].

H.264/AVC also allows the simultaneous use of several reference frames for MC. The reference frames used for MC are organised into one or two lists, maintained in a synchronised manner by both the encoding and decoding algorithms. For each MB, a reference index parameter is transmitted for each motion-compensated partition. In addition to the I slices, H.264/AVC also defines *P* and *B* slices. Both P and B slices use mainly inter-frame prediction, but they may also use intra predicted MBs. While P slices use inter prediction with at most one MC prediction signal per prediction block, B slices may use inter prediction with two MC prediction signals. Figure 8.3 represents the multiframe MC used by H.264/AVC.

For P slices, the prediction modes corresponding to the several partition sizes may be used (see Figure 8.2). Additionally, a P macroblock may be encoded using a P skip mode, for which no MC data is transmitted. Instead, the reconstructed MB is obtained

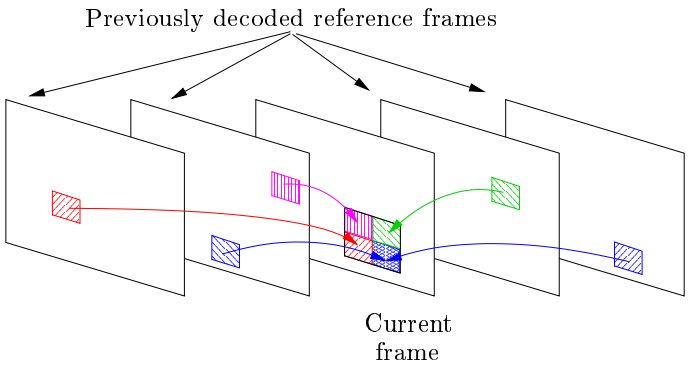


Figure 8.3: Motion compensation using multiple reference frames and bi-predictive motion compensation.

by using the 16×16 prediction block associated with the MV predictor and frame 0 of the picture buffer. For B slices, four types of inter prediction may be used: list 0, list 1, bi-predictive and direct prediction. In bi-predictive coding, the prediction signal for each MB is composed by a weighted average of motion compensated prediction signals from list 0 and list 1. The direct prediction mode uses information inferred from previously transmitted syntax elements to estimate the current block. Additionally, a B skip mode is also defined, that uses a similar technique to that of P skip MBs. The reference frames stored in list 0 and list 1 may include future frames and not only causal ones. This is possible due to a highly flexible ordering scheme for referencing and display purposes.

As its predecessors, H.264/AVC uses transform-coding to compress the predicted residue signal. Nevertheless, as for the case of spatial and temporal prediction, several novelties were introduced, that allow for an improved coding performance. The adoption of a separable integer transform (IT), with similar properties to the DCT; the use of 4×4 transform blocks and a two step transform coding for the DC coefficients of each MB are the main differences, when compared with the previous standards. Additionally, the encoder may select a special coding mode, that extends the length of the transform block to 16×16, for low-frequency residue information (a 8×8 block size is also available for the fidelity range extensions (FREXt) profiles [61]). Despite these changes, the same basic paradigm is used: at the encoder, a forward transform is used, followed by zig-zag scanning, quantisation and entropy coding (see Figure 8.1); the decoder uses the inverse procedure, in order to reconstruct the predicted error signal. Further details on the H.264/AVC transform coding and quantisation may be found in [77, 90].

Another improvement factor for the encoding performance of H.264/AVC is the introduction of an adaptive in-loop deblocking filtering, that reduces the blocking artifacts introduced by the block-based processing of each input frame. Both the encoder and the decoder control the filtering strength, based on the values for several syntax elements that were used on each block. The filtering is applied inside of the prediction loop (see Figure 8.1), meaning that the prediction signal is also affected by the process. The improvement in subjective quality is accompanied by a 5 to 10% reduction on the bit-rate for the same objective quality. A detailed description of the adaptive deblocking filter can be found in [83].

The encoding parameters of H.264/AVC are compressed using a variable length code (VLC) compression algorithm. H.264/AVC combines VLC with binary arithmetic coding (BAC), in a context-adaptive framework. This results in the definition of two highly adaptive and efficient compression algorithms: context-adaptive variable length encoding (CAVLC) and context-adaptive binary arithmetic coding (CABAC). A simple universal VLC code table is also used for some higher-layer syntax elements, like sequence and slice headers. Other syntax elements, like reference frame indexes, MVs and quantised transform coefficients may be encoded using CAVLC or CABAC. The encoding performance is also favoured by the use of predictive techniques for encoding several syntax elements, like the used MVs. Both CAVLC and CABAC use context modelling to choose which probability model should be applied in the VLC step. This choice is based on local statistics of the current stream. CABAC uses an additional binarisation step, to transform the original symbols into a set of bits that are then compressed using the adaptive arithmetic encoder. The use of CABAC allows for an increased compression efficiency, when compared with CAVLC, at the cost of an additional computational complexity. Detailed information about H.264/AVC's VLC coding can be found in [91].

One important reason for the performance gains of H.264/AVC is the very high number of available encoding strategies for each block. A number of prediction modes (intra, inter and bi-directional) may be used in conjunction with blocks with adaptive sizes. Because of this, efficient encoder control strategies are essential for determining the best encoding option for each block. H.264/AVC (namely its verification model software, used in this thesis) uses a Lagrange optimisation technique to perform rate-distortion optimisation. For each block, the encoder tests all available encoding modes and block sizes and deter-

mines the corresponding distortion and estimated rate values. Based on the value of the Lagrangian cost function (see equation (3.2)) for each possibility, the encoder chooses the most efficient compression mode for the current block, in an RD sense. A description of the coder control strategies used in H.264/AVC is presented in [92].

Several other encoding tools are defined by H.264/AVC, namely related with: robustness to data errors and information losses; flexible operation for a variety of network environments; use of flexible macroblock and slice ordering and lossless macroblock modes, among others (see [76, 14, 13] for information on these methods). As for the case of the previous encoding standards, H.264/AVC has a set of encoding *profiles*, that define which coding tools are available for the encoder. A *baseline* profile was defined to minimise complexity while the *main* profile was designed with an emphasis on coding performance. The *extended* profile was design in order to combine the coding performance of the main profile with a set of techniques related with extra network robustness. A fidelity range extensions (FRExt) amendment was also published, that includes a set of new features, used by a new family of profiles, called the *high* profiles. These tools are mainly related with: supporting higher sample resolution video signals (up to 12 bits per sample) and other chroma sampling formats (namely the 4:4:2 and the 4:4:4 formats), using adaptive block sizes for residual spatial frequency transform and the use of encoder-specific perceptual-based quantisation matrices. Information on the FRExt coding techniques and profiles may be found in [61].

8.2 On overriding H.264/AVC's motion-compensated residue coding

In H.264/AVC, as in most video coding standards, the transmission of the MC-prediction error is generally accepted as crucial for the performance of video encoders. This results from the fact that the prediction process alone is not able to eliminate the prediction error for the current segment of the message. The coding error is thus propagated, since it also compromises the performance of the prediction of the following block. Despite this, in this section we describe a successful H.264/AVC-based compression scheme that does not use the transmission of the motion-compensated residue. The described method has an interesting relation with the pattern-matching coding paradigm, as described in Section 8.2.2.

Experimental results, presented in Section 8.2.3, demonstrate that, unlike other pattern-matching video coding algorithms proposed in the literature, the described method is able to achieve compression gains when compared with state-of-the-art H.264/AVC standard.

8.2.1 The Overriding Process

In Section 8.1 we described the inter-frame prediction process used by H.264/AVC. As was previously discussed, the use of the highly adaptive MC algorithm creates a residual error information that can be efficiently compressed. For B slices, that use information from both past and future reference frames, the combination of these techniques provides even larger gains in compression performance. The gains introduced in H.264/AVC by the motion compensation process and its efficient RD optimisation scheme were impressive. They led us to hypothesise that, at least in some cases, motion compensation is efficient enough to obviate the need for MC-predicted error coding. Furthermore, the flexible use of several prediction schemes and reference frames allows the encoder to efficiently determine the best prediction block, among a wide set of possibilities. This allows the prediction process to reduce, or even eliminate, the prediction error. In this section we investigate the performance of a H.264/AVC-based encoder[1], that does not transmit the information for the motion-compensated residue block.

Figure 8.4 presents a schematic of the structure of the tested encoder. We have called it H.264/AVC No MPE (no motion-compensated prediction error). The original optimisation scheme tests each available prediction mode and then uses transform coding to compress the predicted error block, in order to determine the encoding distortion. Therefore, the optimisation process always takes into account the distortion and rate values associated with the transmission of the predicted error. In order to create a coding scheme that is compliant with the original standard, instead of simply removing all the residual error information from the bitstream, the samples of the MC residue block are set to zero, prior to the residual encoding step. This is also done during the optimisation stage, meaning that the encoder chooses the prediction mode that optimises the Lagrangian cost function in the absence of the encoded error signal. The H.264/AVC encoding syntax efficiently compresses the overhead associated with the null coefficients. The encoder signals the

[1]The H.264/AVC reference software [93] (version *JM9.3*) was used for the development of the new algorithm.

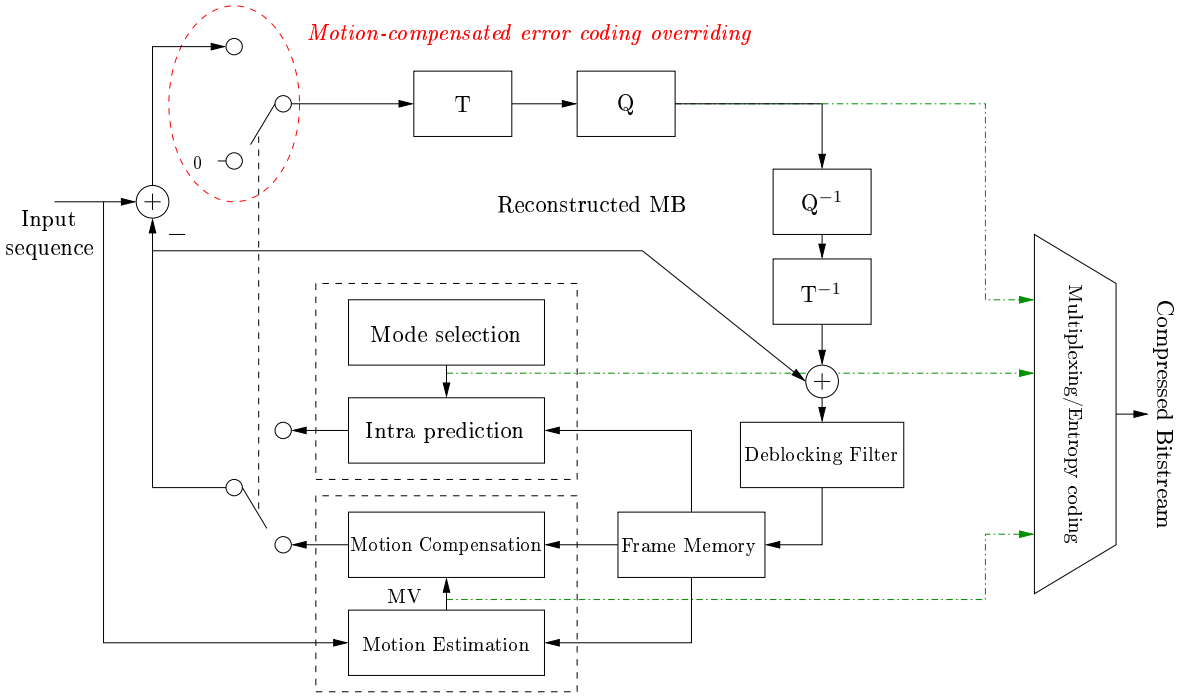


Figure 8.4: Architecture of a hybrid video encoding scheme that uses no compression of the motion-compensated error.

null IT coefficient block by setting the *Coded Block Pattern* (CBP) parameter to zero and transmits no further information. This fact is efficiently exploited by the entropy coding tools defined in the H.264/AVC standard. On the decoder side, the used prediction mode and the null error block are decoded and used for the reconstruction of the MB, using exactly the same process as for the case of non-null residue patterns. Therefore, the resulting bit-stream is fully compliant with the original H.264/AVC reference decoder [94].

It is important to note that for intra-frame prediction, the compression of the intra-predicted error is maintained. This is so because, for the intra case, the encoder has no way to avoid the prediction error propagation. Also, for P or B slices, the intra prediction process ensures that the encoder always has a viable option to approximate the original block, in case that the motion prediction fails, even if it much more costly in terms of used rate.

8.2.2 A pattern matching video encoding point-of-view

In this section H.264/AVC No MPE is regarded as an implementation of a lossy pattern matching-based video encoder and is related both to LZ and VQ-based video compression

schemes.

As was previously discussed in Chapter 2, in Lempel-Ziv based encoders, each new block of the input signal is approximated by using segments of the previously encoded part of the message. In LZ77 [3] encoders, pointers are used to identify the longest match for the current block within a search buffer, composed by the recently encoded data. In this case the length of the matched string is also transmitted. LZ78 [4] implementations use an explicit dictionary, composed by an indexed list of previously used portions of the message. Vector quantisation algorithms [5] use a set of codewords, stored in a fixed or adaptive dictionary, to approximate the input signal. Each message block is thus encoded by replacing it with a dictionary index. More details on both LZ and VQ algorithms may be found in Section 2.2.

When we override the compression of the MC-prediction error, the best approximation of the current block is given solely by a block in a reference frame, pointed to by the MV determined by the motion-estimation algorithm. As in LZ schemes, the used block corresponds to a previously encoded portion of the message. We can thus think of the MVs in H.264/AVC No MPE as LZ77 pointers, that are encoded by using a predictive scheme and a context adaptive arithmetic encoder. The length of the message used in each approximation is implicitly encoded by the partition size used in the MC process. The LZ search buffer (see Section 2.2.1) is defined by the reference frames that are used for each slice, together with the motion vectors' range. H.264/AVC No MPE also presents an interesting addition to this basic LZ scheme: the use of B slices allows the use of a combination of blocks of the reference frames. This means that each segment of the message may be encoded as a combination of two previously encoded segments of the search window.

We can also relate the patterns stored in the reference frames to an adaptive dictionary. Because the dictionary is composed by previously encoded segments of the video sequence, this may be related either to a LZ78 or a VQ algorithm (see Sections 2.2.1 and 2.2.2, respectively). For this dictionary, each MV acts as an index that identifies the chosen codeword. The use of different partition sizes in the MC process can be regarded as the use of several dictionaries, that store blocks with different dimensions. The dictionary indexing process is implicitly determined by the choice of the partition size. Furthermore, the dictionary adaptation process consists in the use of a variable set of codewords, that are

chosen according to temporal (related to the choice of reference frames) and neighbourhood (represented by the search window) criteria. This increases the dictionary's efficiency, because it tends to use codewords that are likely to be similar to the current frame's blocks. As for the LZ77 analysis, the use of B slices can be interpreted as an extension of the dictionary-based coding paradigm, to the case were a weighted combination of two codewords is employed.

8.2.3 Experiment description and simulation results

A set of experimental tests was performed using the first 100 frames of several well known test sequences Foreman (CIF), Mobile & Calendar (CIF), Akiyo (QCIF) and Flower Garden (SIF), with 4:2:0 colour sub-sampling[2]. These sequences were chosen due to their common use as benchmarks for video encoding and also because they represent a wide variety of video content: from a typical head and shoulders sequence to highly detailed sequences, with large motion.

Experimental tests were performed for a wide set of input parameter values, in order to assess their effect in the performance of H.264/AVC No MPE. However, some configurations are common to all simulations, namely: RD optimisation is enabled, 30 Hz frame-rate, variable bit-rate (VBR) mode, only the first frame is intra, 5 reference frames are used, CABAC and deblocking filter are enabled and single level bi-predictive sub pixel motion estimation was used. The H.264/AVC high profile was used both for the original version and for the new encoder.

We first evaluated the results when residual error coding is disabled for non-intra blocks of B slices only, with an *IPBP* pattern. In this test, the coding strategies for the I and P slices were left unaltered. Different target bit rates were achieved by varying the values of the input quantisation parameter (QP), according to the values of table 8.1. Note that there is a direct relationship between the values of QP and the Lagrangian multiplier parameter λ, used in the RD optimisation [92].

Figure 8.5[3] presents the results for this test. Since the coding strategies for the I and P slices were unaltered, the results for these slice types are similar. Therefore, only the RD results for B slices are presented. In this figure we observe that disabling the com-

[2] An overview of these sequences is presented in Section A.2.

[3] Results for other test sequences are presented in Figure C.44.

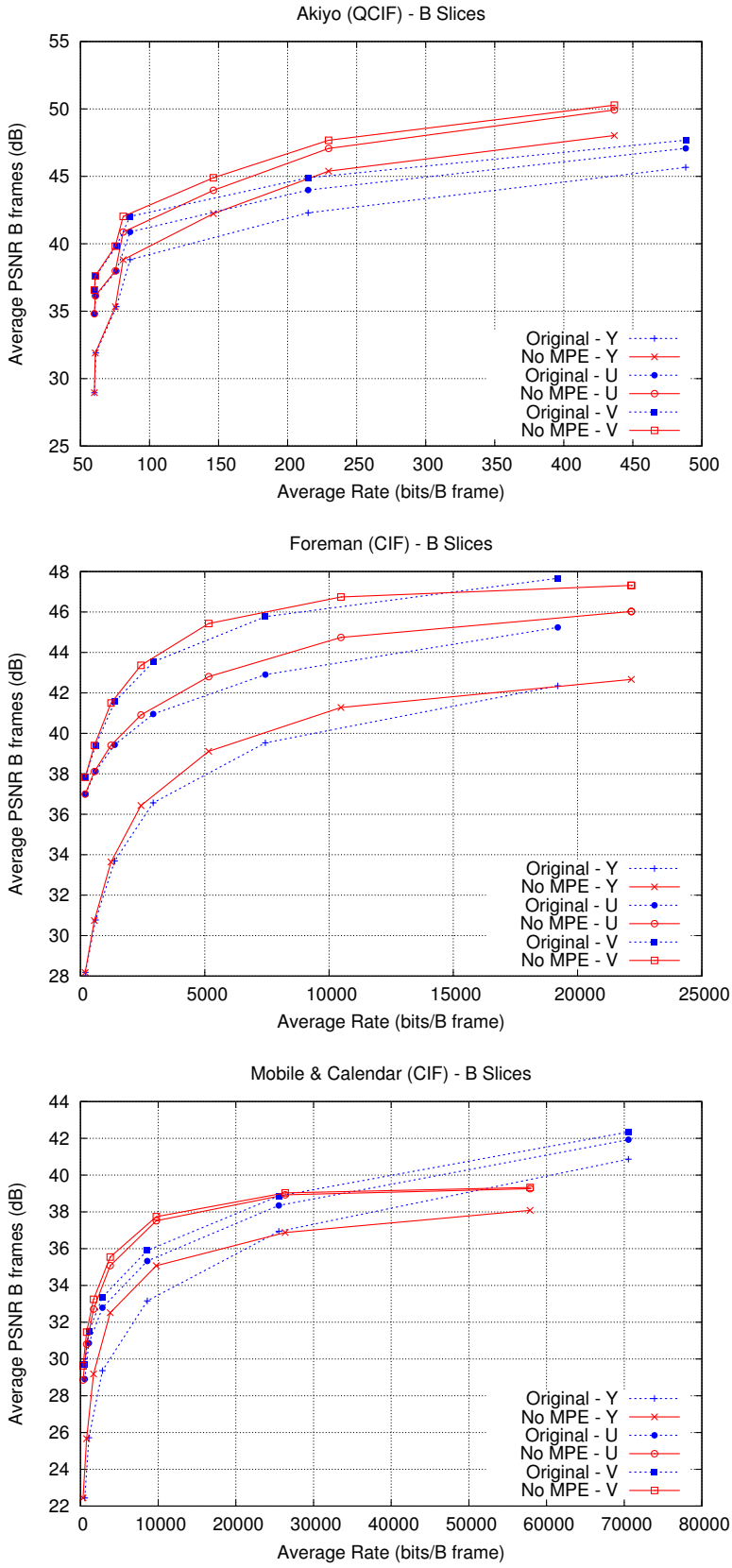


Figure 8.5: Experimental results of the original H.264/AVC encoder and the new encoder, that disables the compression of motion-predicted error (for B slices only).

QP for B slices	45	40	35	30	25	20	15*
QP for I and P slices	43	38	33	28	23	18	13*

(* used only for the version with no residual coding)

Table 8.1: QP values used in the experimental tests.

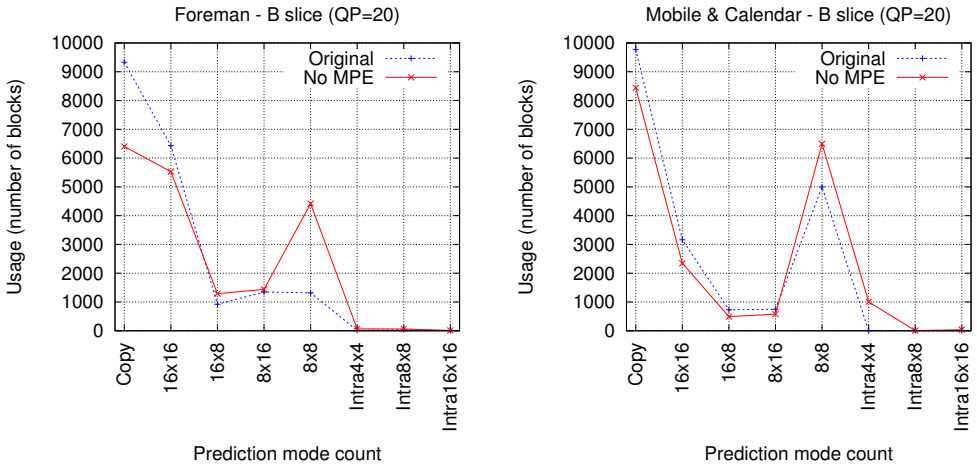


Figure 8.6: Number of MB's that use each of the available prediction modes, for the B slices.

pression of the inter residue error increases the performance of the original H.264/AVC algorithm, for all components of all tested sequences. In these cases the motion compensation procedure is able to efficiently reconstruct the bidirectionally predicted frames from the previously encoded slices, with no need to encode the MC-prediction error. For sequences with small motion (like Akiyo sequence), not using residual coding for B slices is always advantageous, achieving, in some cases, the same PSNR using less than half the bit rate of the original H.264/AVC encoder. For sequences with complex or high speed motion, the H.264/AVC No MPE method still outperforms the original encoder, but only for low to medium qualities/bit rates.

Figure 8.6 represents the total number of MB's encoded with each prediction mode, for the B slices of sequences Foreman and Mobile & Calendar, using QP= 20. We may observe that, even for this relatively low value of the QP parameter, the copy mode is used frequently. Also, when we deactivate the residual error encoding, the motion estimation uses more partitions of a small size[4]. These smaller blocks allow for a reduction of the

[4]Note that, due to a limitation in the H.264/AVC reference software, the 8×8 column has the number

distortion values, favouring the RD performance of these modes when no residue is encoded. For the original encoder, the use of these prediction modes is penalised by the higher bit-rate. For the Mobile & Calendar sequence we can also observe an increase in the number of used intra predicted MB's. As was previously discussed, intra blocks are used when the motion-estimation is not able to efficiently reconstruct the original signal without the use of the motion-predicted error, resulting in a high distortion and therefore a higher value for the RD cost function.

Figure 8.6 shows that, when no residual information is used, the encoder is able to compensate the distortion losses by using smaller block sizes for MC. Nevertheless, this effect is limited by the minimum allowed block size for H.264/AVC, set to 4×4. As a consequence, for higher rates the encoder is not able to provide the needed fidelity. In fact, the plots of figure 8.5 show that the PSNR of the B slices tends to be bounded for rates above a certain point. For higher rates we observe that the original encoder is able to increase the PSNR value, as the bit-rate increases, but this is not the case for the altered version of the algorithm. In this version, the PSNR remains approximately constant, even for higher rates. This phenomenon is more evident for sequences with higher motion. In these cases, the use of residual error information is advantageous, because it allows the encoder to reduce the distortion of the reconstructed sequences, increasing the objective quality levels. Nevertheless, from these observations one may conjecture that the use of smaller block sizes in the MC process would increase the performance of inter-prediction. This would increase the performance of the altered encoder, possibly allowing it to achieve good performances also in the high quality/bit-rate regions.

In order to better evaluate the proposed technique, other tests were performed by varying the most relevant encoding parameters. The relative gains presented previously were also observed when the main profile was used (see Figure C.45), when the bi-predictive coding mode was turned off (see Figure C.46) and also when the width of the motion search window was increased to 32×32 pixels (see Figure C.47). For all tested cases, the proposed method maintained an advantage for all sequences and bit rate scenarios.

In one other test, we increased the number of B slices. This decreases the accuracy of the prediction step, since the key frames for the B slices (the I and P frames) are "further away" (from a temporal point of view), from each encoded B slice. Nevertheless, the same

of all 8×8 blocks and their partitions

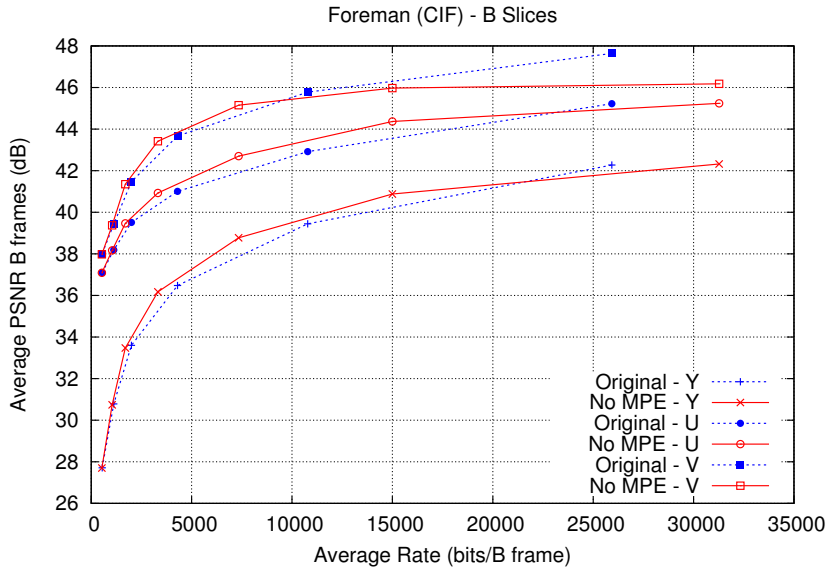


Figure 8.7: Experimental results of original H.264/AVC and new encoder, that disables the compression of motion-predicted error (for B slices only), using *IPBBBP...* slice pattern.

tendency described for the *IPBP* case has been observed. The difference is that the points for which the absence of residual data becomes a disadvantage happen at a lower rate than for the *IPBP* case. However, for low to medium rates it is still advantageous to override the transmission of residual data, as can be seen if Figure 8.7[5], that presents the comparative results of the new algorithms, when B slices are used with a *IBBBPBBBP...* slice pattern.

The overriding of motion-predicted error coding was also tested for P slices. In this case, the performance of the encoder decreases significantly, as can be seen in Figure 8.8[6], that shows the experimental results for H.264/AVC No MPE, when the transmission of the motion predicted error is disabled for the P slices. These losses result from the use of only past I and P slices as references for the P frames. This means that, after the initial frames are compressed, each P slice uses only past P slices as reference. This fact not only limits the patterns that may used to approximate each new image block, but also means that, because no prediction error is transmitted for the reference signal, the reconstruction error is propagated from each P frame to the next. Furthermore, one should consider the inferior performance of the unidirectional MC process, due, for example, to problems with

[5]Results for other test sequences are presented in Figure C.48.
[6]Results for other test sequences are presented in Figure C.49.

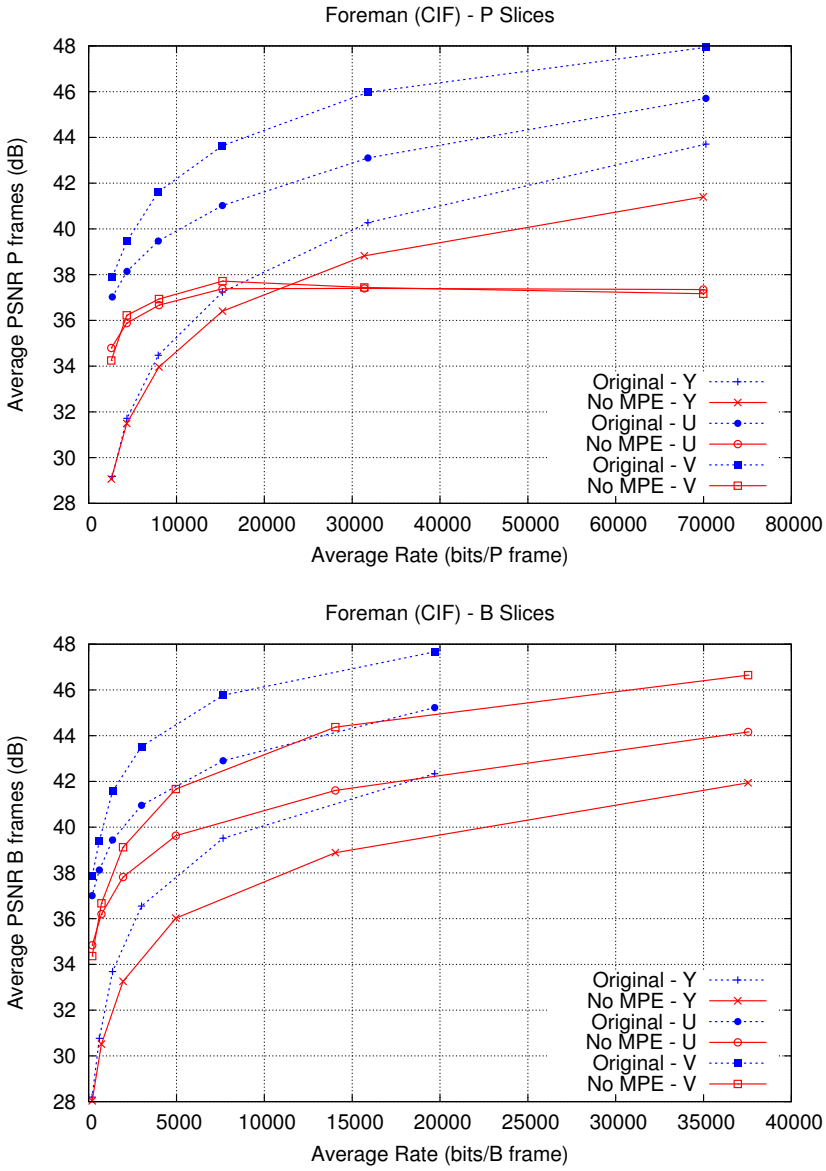


Figure 8.8: Experimental results of original H.264/AVC and H.264/AVC No MPE, when the compression of motion-predicted error is disabled for P slices (results for P and B slices, of sequence Foreman).

uncovered background. Figure 8.9 shows the performance of H.264/AVC No MPE when no residue is used for both P and B slices. As one would expect from the previous discussion, in this case the prediction error is propagated across all inter frames and the performance suffers considerably from this fact.

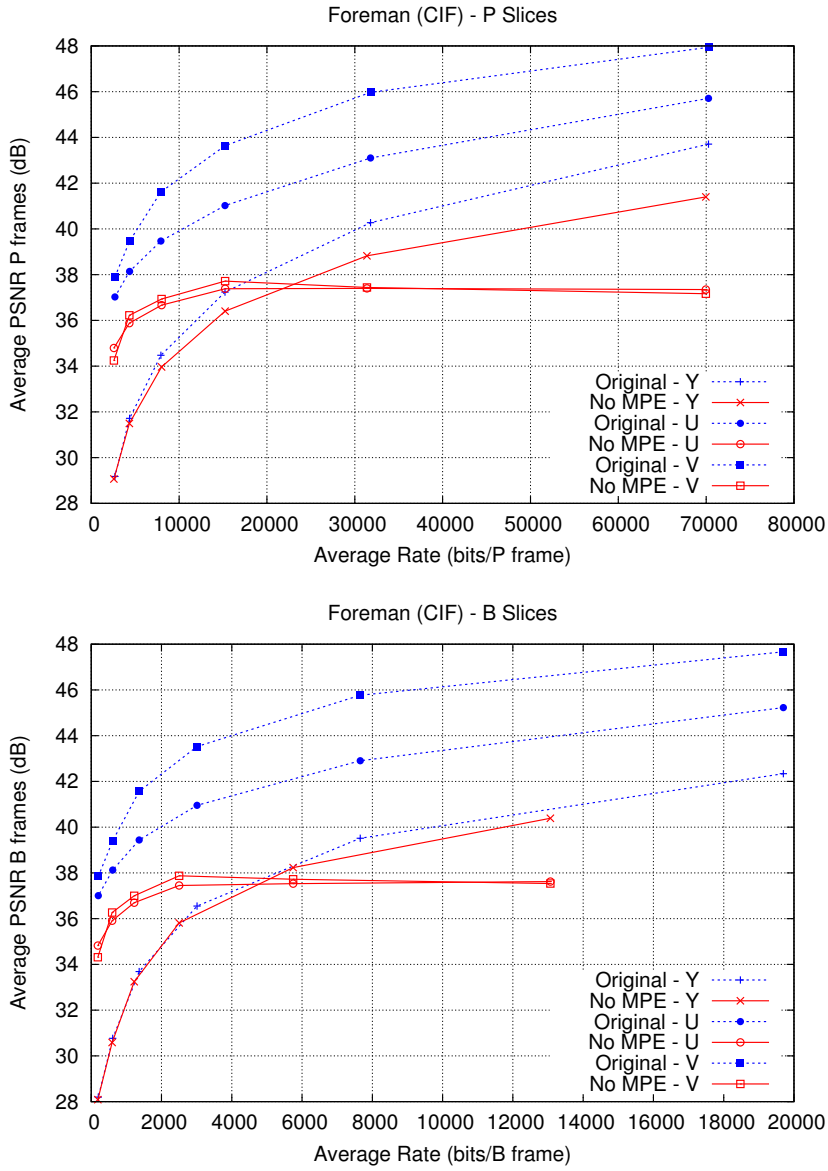


Figure 8.9: Experimental results of original H.264/AVC and H.264/AVC No MPE, when the compression of motion-predicted error is disabled for both P and B slices (results for P and B slices, of sequence Foreman).

8.3 Video coding with multiscale recurrent patterns: the MMP-Video algorithm

In the previous section we have presented an H.264/AVC-based video compression algorithm that overrides the transmission of the MC-prediction error. The described algorithm revealed an interesting relation with traditional pattern-matching compression schemes.

Experimental results for the proposed scheme revealed advantageous performances for high compression ratios, when the MC error is not transmitted for the B slices. Nevertheless, the maintenance of a satisfactory performance for higher rates demands the compression of the error information.

In this section we investigate a second pattern matching-based paradigm for video compression, namely the use of MMP to encode the motion compensated residual data, in a hybrid video coding scheme. We refer to this algorithm as MMP-Video. The good results achieved by MMP-I and MMP-II (see Chapters 5 and 6) demonstrate that MMP is an efficient method to encode image residue patterns, that result from a prediction process (in those cases, intra-frame prediction). Furthermore, the comparison with H.264/AVC intra-frame encoder shows that approximate pattern matching with scales can, in some cases, be advantageous over the H.264/AVC's integer transform. In the following sections we first present an overall description of the MMP-Video encoder, followed by a discussion on some functional improvements and architectural adaptations, namely related with dictionary adaptation for MMP-Video. Experimental results are also presented.

8.3.1 MMP-Video architecture

The MMP-Video architecture is based on that of H.264/AVC, but uses MMP for the compression of the motion compensated residual data. Because of this, the MMP-Video implementation shares its basic structure with H.264/AVC reference software[7]. Figure 8.10 presents a diagram of the architecture of an MMP-Video encoder. The implementation of MMP-Video according to this architecture raises some interesting issues that were analysed during the definition of the method. Some of these general features of the algorithm are discussed in this section. The specific optimisations, that resulted from a thorough investigation on the MMP-Video, including the corresponding experimental tests, will be described with more detail in the following sections.

In this investigation we are mainly interested in assessing the performance of MMP for inter residual coding, since the performance of MMP for intra MBs has already been documented in the previous chapters of this document. In the previously presented results we have shown that the performance of MMP and transform coding for intra frames is different. Because MMP-Video, as H.264/AVC, uses the I slices as references for the MC

[7]The H.264/AVC reference software [93] (version *JM9.3*) was used for the development of MMP-Video.

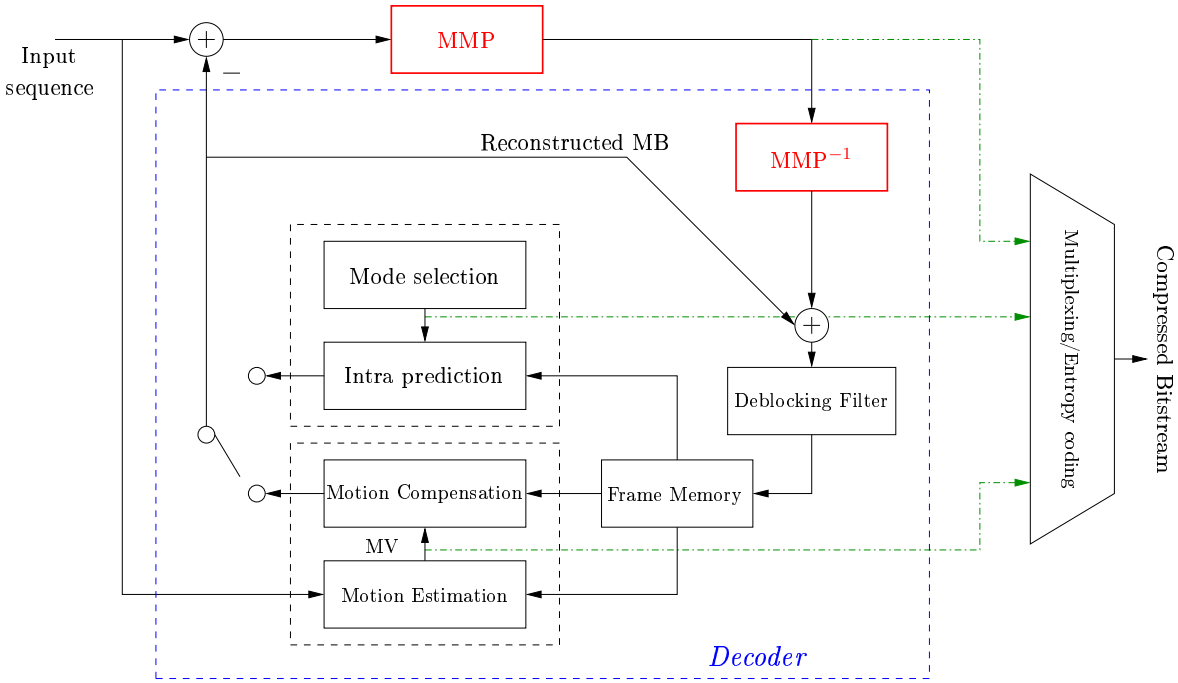


Figure 8.10: Basic architecture of an MMP-Video encoder.

frames, the use of different intra compression schemes could compromise the comparison of the performance of MMP-Video and H.264/AVC, for coding motion compensated residual errors. Because of this we have used an encoder that processes all the intra MB using exactly the same procedure has H.264/AVC, including the same integer transform.

As H.264/AVC, the MMP-Video encoder also uses adaptive block sizes (ABS) to perform the motion estimation/compensation of the inter-predicted MBs. The MMP-Video MC process optimises the partition sizes and corresponding MVs, through the use of the same Lagrangian RD cost function. MMP-Video uses the same scheme as the H.264/AVC reference software to control the RD compromise for the compression of a sequence. Two quantisation parameter values are defined in order to control the target compression ratio of the encoder. The values of the QP parameter are set independently for the I/P slices and for the B slices. These values have a direct correspondence to the value of the Lagrangian multiplier, λ [92]. MMP-Video uses the same value of λ as the one defined for H.264/AVC, to perform the RD optimisation of the MMP encoder.

During the optimisation loop, the encoder tests exhaustively all of the coding options. Because of the high complexity of MMP, the computation of the RD cost function in MMP-Video is performed using the same measurements as in the H.264/AVC encoder.

This means that the distortion is estimated based on the transform coding of the residues, either by using the sum of absolute differences (SAD) or the sum of absolute transformed differences (SATD). It is important to note that, unlike the case of H.264/AVC's, the residue block with minimal SAD or SATD is not necessarily the block that is more efficiently encoded by MMP-Video. This means that the MC parameter search is sub-optimal, from the MMP coding point-of-view. Nevertheless, our simulations have demonstrated that the use of these error measures still allows MMP-Video to perform efficiently, with a significant reduction in computational complexity. This complexity reduction results from two factors: first, the simplification inherited from the original encoder, that replaces a whole MB's residue direct and inverse transform evaluation with a simpler SAD or SATD calculation; and second, the replacement of the full dictionary search, required by MMP encoding, by the simpler transform-based algorithm. Nevertheless, once the optimum coding parameters are determined, the MC data is encoded and transmitted, followed by the MMP compression of the MC residue block. This has the interesting side effect of allowing for a fair comparison between the encoding efficiency of MMP and H.264/AVC's transform coding, because the residue patterns, generated by the motion compensation process, tend to be approximately the same for both encoders.

The use of ABS motion-compensation means that each MC-prediction error macroblock (with 16×16 luma samples) can be the result of the concatenation of several smaller segments, depending on the used partition (see Figure 8.2). H.264/AVC encodes this residue MB with a block size for the transform coding that depends on the MC partition size. In MMP-Video, this information can also be used, by considering each partition independently in the MMP compression step. Nevertheless, the MMP encoder may also disregard the MC partitioning and process the entire 16×16 residue block, composed by the concatenation of the several MC partition blocks. In this case, MMP segments the original 16×16 block in a way that optimises the compression cost of the MC residue. Considering the original partition of the MB can be regarded as a efficient way to encode the first few segmentation decisions of the 16×16 block, which could lead to some performance gains. Nevertheless, experimental tests showed a marginal performance gain when the entire MB was processed by MMP, especially for B slices, so this was adopted in the final versions of MMP-Video.

Apart from the motion compensated residual data, all information transmitted by

MMP-Video is encoded using the original H.264/AVC techniques [13]. This includes all MC information, like the partition modes and the motion vectors for each block, as well as sequence, slice and MB headers. In these cases, the original H.264/AVC options were maintained, namely the use of CAVLC or CABAC, depending on the used encoding profile. Furthermore, the in-loop deblocking process was maintained in MMP-Video. This decision was justified by the performance gains observed in some experimental tests, that demonstrated the efficiency of this method also for MMP-Video.

8.3.2 Dictionary design for MMP-Video

In a video coding framework, the MMP dictionary design possibilities increase significantly, because it is possible to exploit additional signal features. MMP-Video, as H.264/AVC, divides the video sequence into I, P and B slices (for all our tests, each sequence frame corresponds to an individual slice). Due to the different encoding tools used for each slice type, one may expect the residue signals to vary accordingly. Furthermore, because we are dealing with colour sequences, each frame is composed by a luma (Y) and two chroma (U and V) components[8]. The information about the frame type and the colour component that is being processed can be used in the MMP-Video dictionary design process. Another relevant feature is the longer time available for dictionary adaptation, when compared with image coding. Depending on the number of compressed sequence frames, the MMP-Video dictionary has much more time to "learn" the new patterns.

Several tests were performed in order to evaluate which was the best configuration for the MMP-Video's dictionary. In these tests we have used much of the knowledge gathered from the work on MMP dictionary adaptation for image coding, described in the previous chapters of this document. One relevant aspect of the dictionary design process for MMP-Video is the possibility of using independent dictionaries for different image components or slice types. In its initial version, MMP-Video used six independent dictionaries to encode the MBs of each colour (YUV) component of the P and B slices. In this case, each dictionary only "learns" the specific residue patterns of each type of source data, *e.g.* the P-Y dictionary is only used to approximate luma patterns of P slices and is exclusively updated for such cases. On one hand, this can be a performance improvement

[8] In this thesis we follow the H.264/AVC convention, that uses the terms luma and chroma instead of luminance and chrominance [13].

factor, because it results in highly specialised dictionaries. On the other hand, this means that a MB of a given component/slice type can only be approximated by blocks of the corresponding dictionary, that has a smaller approximation power than a more general (and thus more complete) dictionary.

Our tests demonstrated that the use of six independent dictionaries limited the performance for the chroma components. Since chroma residue blocks tend to be very smooth, the number of segmentations performed by MMP for these blocks is small. This causes the chroma dictionaries to learn very few new patterns, which compromises their coding efficiency. Therefore, several other dictionary configurations were tested.

Another important tool for MMP-Video dictionary design is the use of context conditioning techniques, like the ones discussed in Section 6.1. In these cases, the dictionary elements are organised into different partitions, which use independent probability contexts for the arithmetic encoding of the indexes. The following criteria were tested as the probability contexts:

- The original block scale, in which each dictionary partition contains the vectors that were originally created at a given scale;
- The slice type, meaning that each partition contains the vectors that were created for the I, P or B slices;
- The image component, where each partition contains the vectors that were created for the Y, U or V colour components.

Experimental tests were conducted, in order to evaluate the performance of using a different number of dictionaries and different context conditioning criteria. In these tests there was not a unique configuration that proved to be the best for all frame types and colour components, at all compression ratios. Nevertheless, two options achieved the best overall results:

- In the first option, MMP-Video uses two *independent* dictionaries: one for the P slices (all colour components) and another for the B slices. Each dictionary is thus able to learn the residue patterns that correspond to all three components of every MB of each slice type, allowing MMP to use a richer dictionary to encode the chroma components, improving the coding efficiency of these residues. On the other hand,

luma MBs have to "share" the dictionary with the chroma patterns, causing a slight efficiency loss for the luma component. Nevertheless, this configuration achieved the best overall results when we used independent dictionaries, because the small loss for the luma components is compensated by the relevant gains achieved for the chroma components.

- In the second option, a single dictionary is used to approximate all residue blocks, independently of their corresponding component and slice type. This dictionary is, however, segmented considering the original level of each new block of the dictionary, using a similar procedure to the one described in Section 6.1. The results for this method were almost equivalent to the use of the two independent dictionaries of the previous case, segmented by level. The use of a single dictionary means that all dictionary blocks are always available, regardless of the slice type and colour component that is being encoded. Once more, this has a beneficial effect in the compression of the chroma components, at the cost of a small reduction in the efficiency of luma compression.

The dictionary redundancy control scheme, discussed in Section 6.2, and the scale restriction technique (see Section 6.4.1), were also used in MMP-Video. The use of redundancy control introduces consistent quality gains for all sequences. As for the case of image encoding, the best value for the used distortion threshold, d, depends on the target rate and is related with the value of λ. The parameter λ of the MMP-Video encoder is set based on the QP parameter, using the same relation that was defined for the original H.264/AVC encoder [92]. An experimental optimisation of the $d(\lambda)$ rule was performed, according to a procedure similar to the one described for MMP image coding in Section 6.2.2. In spite of the greater variance of the results for video sequence coding, the original rule defined for MMP-II (see equation (6.2)) was found to be appropriate also for MMP-Video. As for MMP-II, the restriction of the scale transforms used in dictionary updating also achieves relevant computational complexity gains, without a noticeable impact on the performance of MMP-Video.

A maximum dictionary size was set for both versions of MMP-Video. We used a value that did not generally limit the dictionary growth (100.000 elements). Nevertheless, this limit was sometimes reached, especially for more complex sequences (like Mobile & Calendar) and for higher rates. The index frequency-based elimination rule, determined

in Section 3.4.1 for image coding, was also used for MMP-Video. As for the case of image compression, experimental tests revealed that the performance of the MMP-Video encoder does not strongly depend on this threshold, unless the maximum dictionary size is severely reduced.

The norm-equalisation procedure, described in Section 6.5, was also tested. Experimental results revealed that the benefits of this technique depend on the input video sequence and on the features of the encoder. For some cases, the use of norm equalisation improves the coding performance, but, for some sequences, it has a negative impact. Because of this, this technique was disabled for MMP-Video.

8.3.3 The use of a CBP-like flag

As was previously described, the H.264/AVC standard uses a coded block pattern (CBP) parameter to characterise the encoded residue block transform coefficients (for instance the absence of non-zero AC coefficients), for each MB. This allows the encoder to avoid transmitting information associated with some null residual coefficients, saving overhead bits. Since MMP does not use encoded coefficients, the direct use of the CBP value is not possible. Nevertheless, an analysis of the MMP-Video coding data revealed that a large number of null 16×16 and 8×8 blocks were transmitted. This can be a source of inefficiency for the MMP-Video encoder. For a null residual block, MMP has to transmit one no-segmentation flag followed by the index that corresponds to the null block, for each of the three components, i.e., a total of six symbols.

Two techniques for reducing this inefficiency in the MMP-Video encoder were studied. Both of them use the transmission of CBP-like information. In the first case, a binary flag signals the existence of non-zero residue blocks for any of the YUV components. If all component residues are null, this flag (we maintained the CBP designation, as a reference to its purpose in the original H.264/AVC encoder) will be zero and the encoder simply does not transmit any MMP information for the current MB, thus saving bits for the same reconstruction quality. When there are non-null residues, the CBP flag is set to one and its transmission is followed by the encoded MMP residue blocks. The second version of this process uses a CBP value with three bits, one for each of the Y, U and V components. As an example, a CBP with value 5 is followed by the MMP encoded residues for the Y and V components and explicitly signals the decoder that the U residue is null.

In both cases, the CBP value is encoded using an adaptive arithmetic encoder and transmitted for every MB, immediately before the MMP data that encodes the residue patterns. The use of the CBP parameter by MMP-Video thus requires an additional rate overhead. Because of this, both options were tested for MMP-Video, in conjunction with the investigation of the most favourable dictionary architectures. Some simulations were performed in order to determine which of the CBP parameter configurations were more efficient, for the two dictionary architectures described in the previous section. Experimental results demonstrated that the best option for the use of a CBP parameter in MMP-Video depends on the dictionary configuration:

- When two independent dictionaries are used, the best option is to encode a CBP parameter with three bits and use it for both P and B slices. In this case, the three-bit CBP is able to efficiently avoid the transmission of null residue for some of the components and was shown to be more efficient than the use of a binary valued parameter;
- When MMP-Video uses only one dictionary, it is more efficient to use a binary CBP and transmit this value only for the B slices. In this case, the overhead introduced by the use of a three-bit CBP results in a loss of coding efficiency, as does the use of the binary CBP for the P slices.

8.3.4 Experimental results

In this section we present the experimental results of the performance of the MMP-Video encoder. The two versions discussed in the previous section were tested:

- MMP-Video 2Dic: uses two independent dictionaries and a three bit CBP, applied on P and B slices;
- MMP-Video 1Dic: uses one dictionary, with context conditioning, and a one bit CBP, used only of B slices.

The results of MMP-Video were compared against those of the H.264/AVC high profile video encoder (best coding performance). The first 99 frames of the 4:2:0 test sequences Foreman (CIF), Akiyo (QCIF) and Mobile & Calendar (CIF) where used in the tests.

A set of common encoding parameters was used for all the encoders, namely a IBPBP pattern with only one intra reference frame and the high profile. The slices were set to

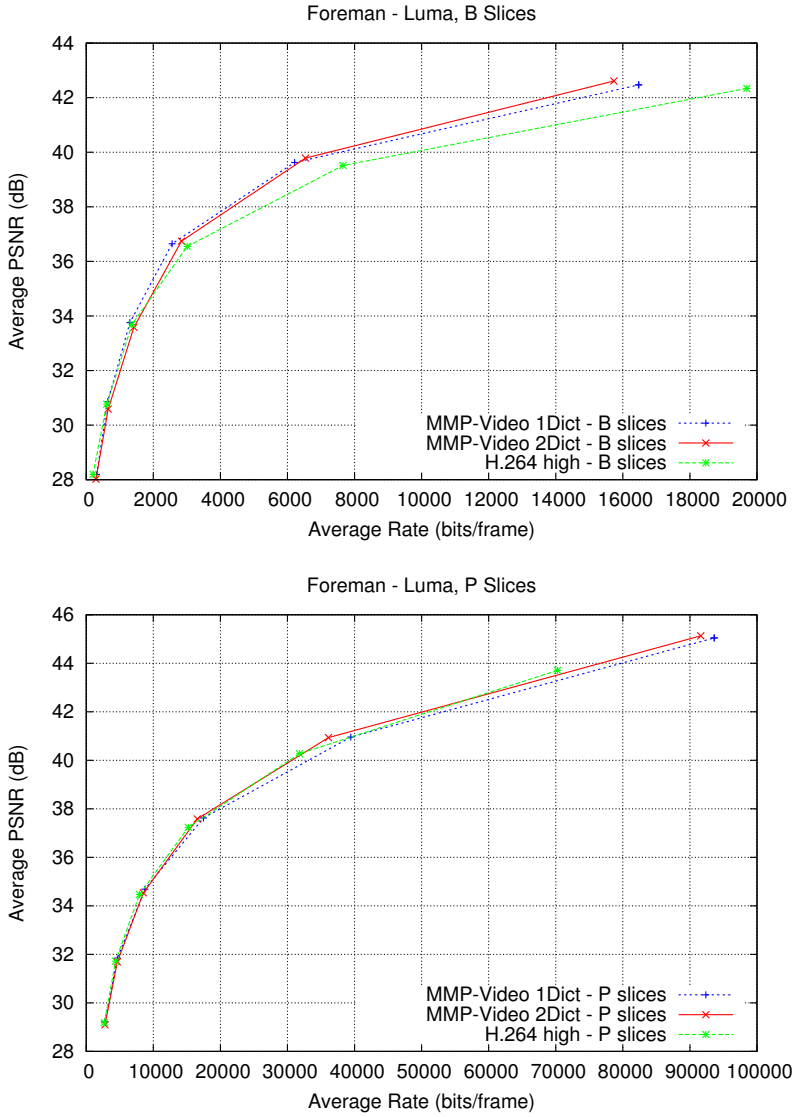


Figure 8.11: Comparative results for MMP-Video encoder and the H.264/AVC high profile video encoder, for the B slices of sequence Foreman (CIF).

the entire frames. RD optimisation and the use of I MBs in inter-predicted frames were enabled, while no error resilience and no weighted prediction for B frames were used. The context-based adaptive arithmetic coder (CABAC) option was set for all encoders. Variable bit rate mode was used and the encoders were tested for several quality levels of the reconstructed video sequence, by fixing the QP parameter for the I/P and B slices.

Figures 8.11 to 8.16 show the results for the P and B slices of three test sequences. The results for the frame I are not presented, because they are equivalent for all the encoders.

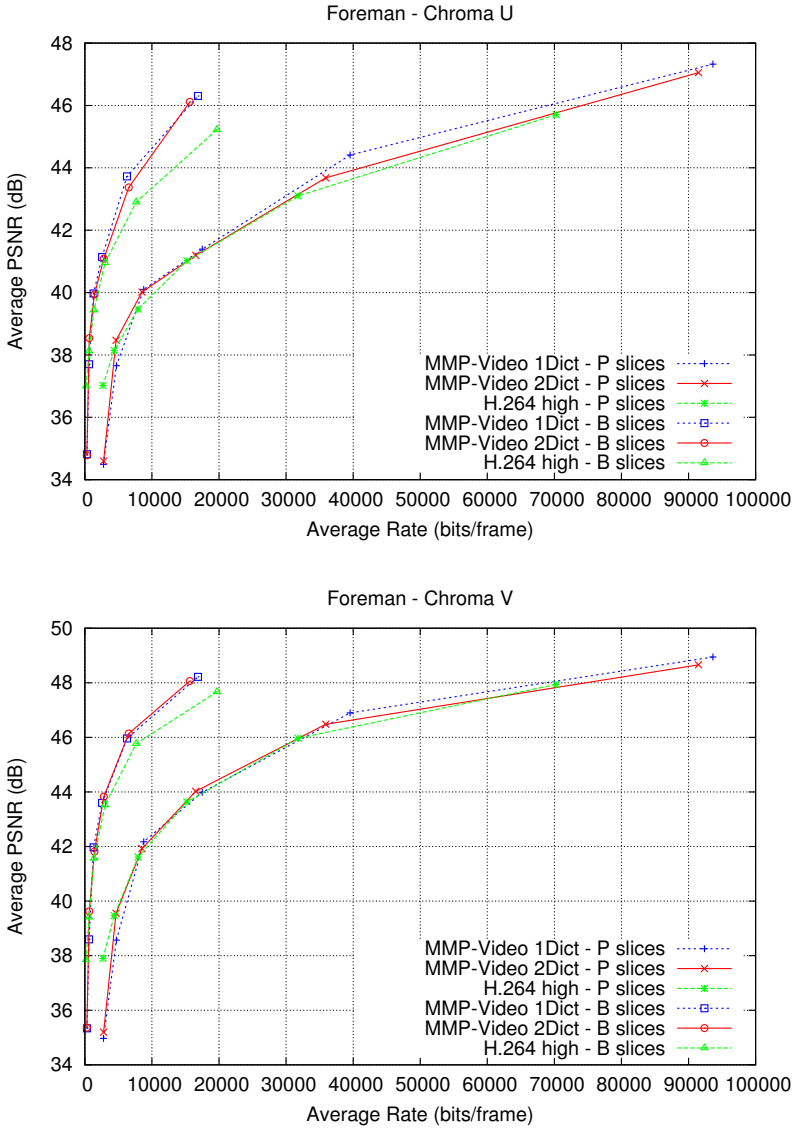


Figure 8.12: Comparative results for MMP-Video encoder and the H.264/AVC high profile video encoder, for sequence Foreman (CIF).

This results from the use of the original H.264/AVC intra coding techniques in the tested versions of MMP-Video, as was previously explained. The RD lines represent the average PSNR values *vs.* the average number of bits per frame. The results are represented for the luma component of the P and B slices (Figures 8.11, 8.13 and 8.15) and for the U and V chroma components of the B and P slices (Figures 8.12, 8.14 and 8.16).

When we compare the two versions of the MMP-Video encoder we observe that they vary in performance but have generally the same behaviour. MMP-Video 1Dict tends to

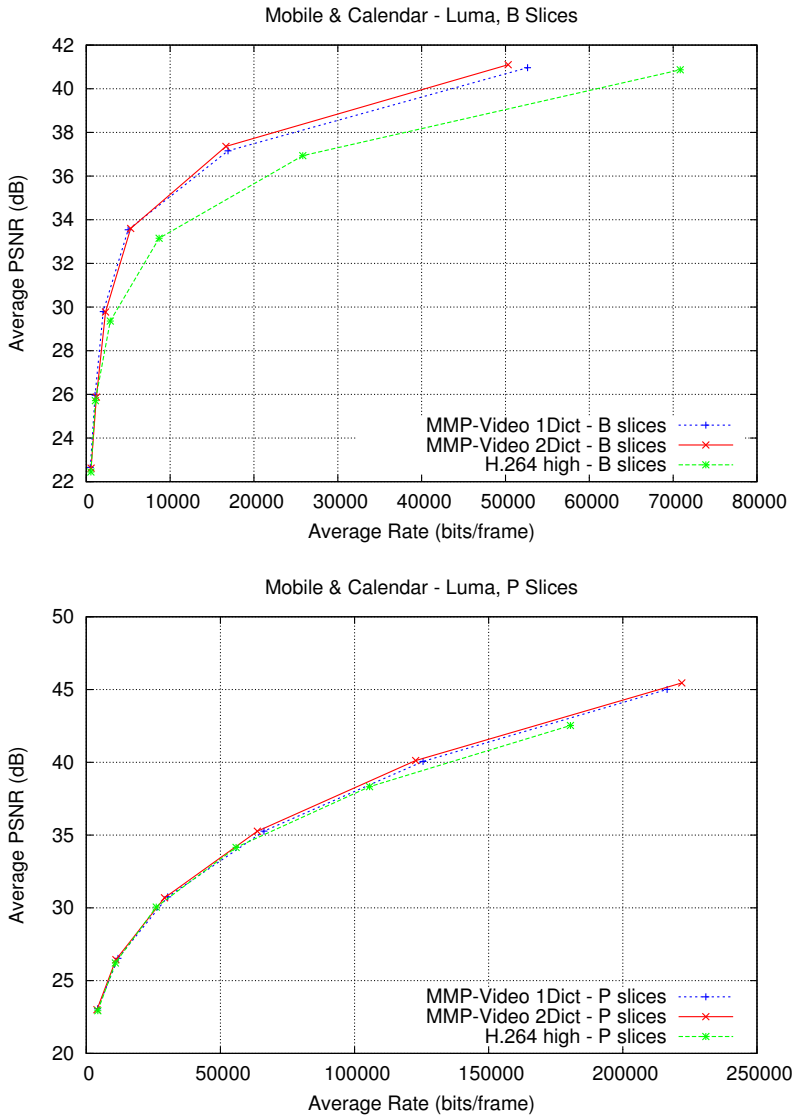


Figure 8.13: Comparative results for MMP-Video encoder and the H.264/AVC high profile video encoder, for sequence Mobile & Calendar (CIF).

be marginally better for the chroma components of the P slices, while MMP-Video 2Dicts presents some advantage for the luma components of both P and B slice types. When compared with the H.264/AVC high profile video encoder, the performance of the MMP-Video encoder is consistently better for all components of B slices, especially at higher bit rates, for CIF sequences Foreman (see Figures 8.11 and 8.12) and Mobile & Calendar (see Figures 8.13 and 8.14). For these rates we may observe PSNR gains that range up to 2dB. For the P slices we observe that MMP-Video is able to achieve some coding gains over the

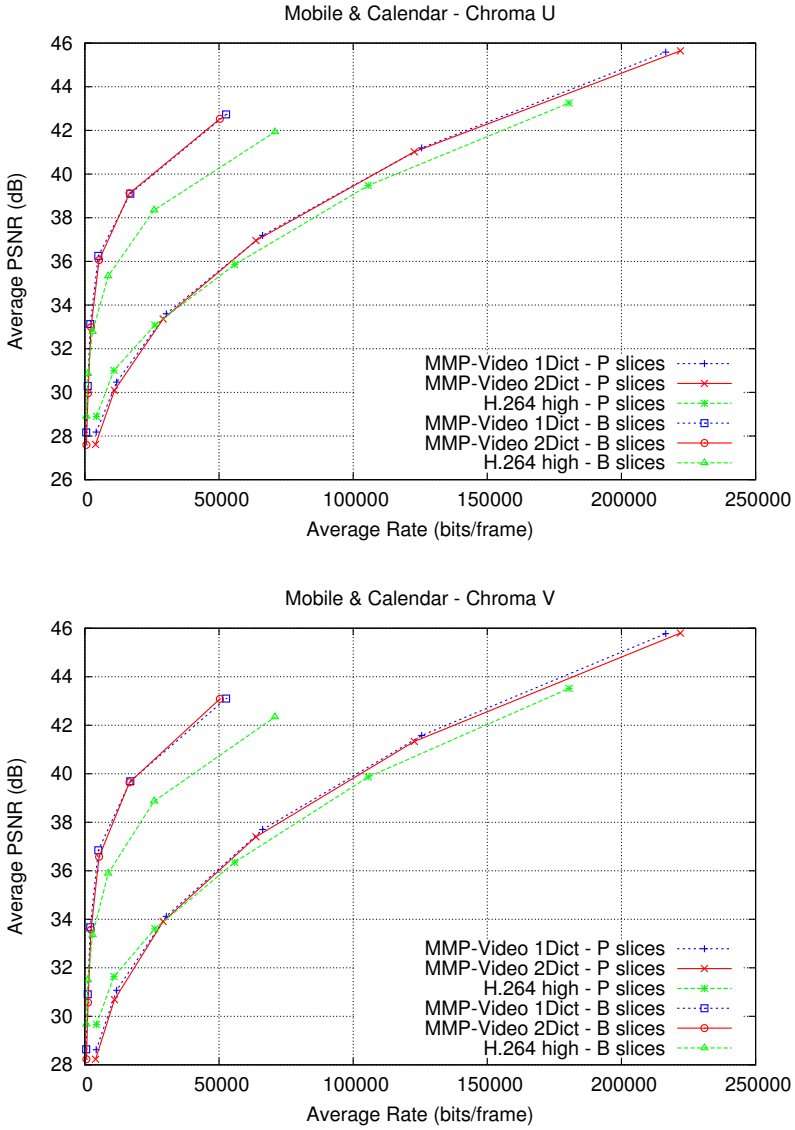


Figure 8.14: Comparative results for MMP-Video encoder and the H.264/AVC high profile video encoder, for sequence Mobile & Calendar (CIF).

H.264/AVC encoder for the chroma components, for all sequences except for Akiyo (see Figures 8.15 and 8.16). For the luma component of the P slices, the relative performance of MMP-Video varies with the tested sequence: it is better than the H.264/AVC for the Mobile & Calendar sequence, equivalent for the Foreman sequence and slightly worse for Akiyo.

For QCIF sequence Akiyo, one may also observe quality gains for the B slices, but only at higher rates. For the lower rates one observes some quality losses. The sequence Akiyo

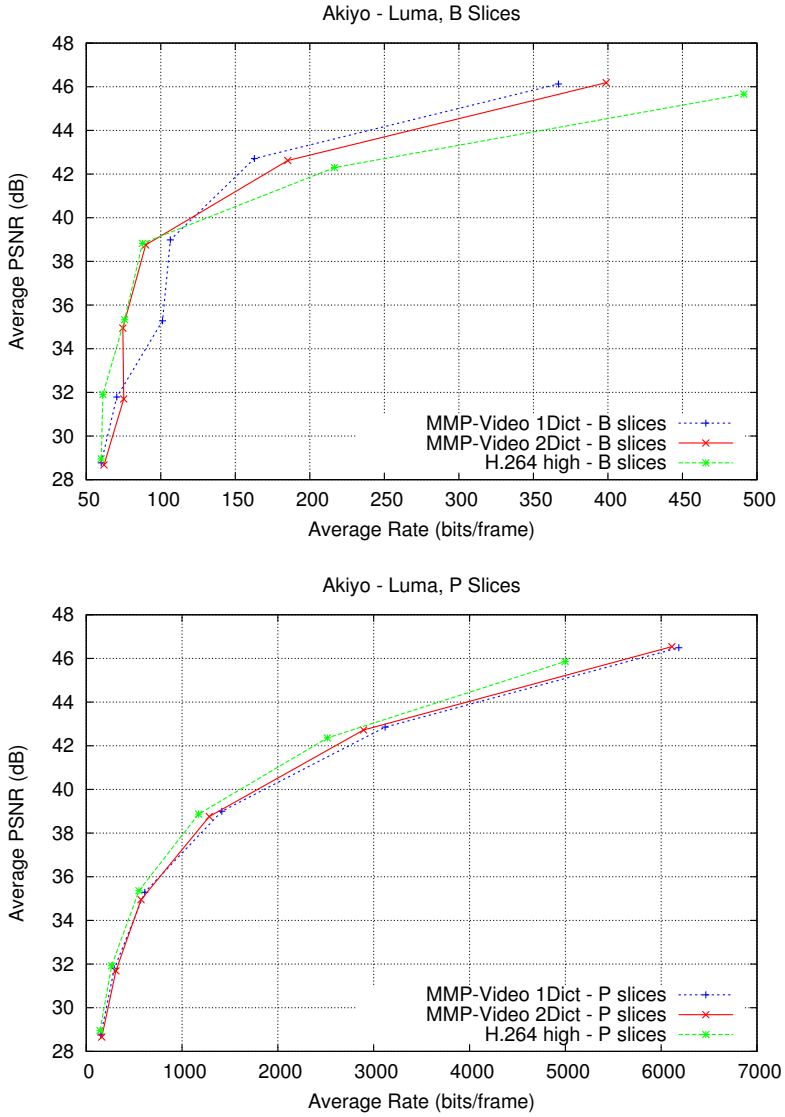


Figure 8.15: Comparative results for MMP-Video encoder and the H.264/AVC high profile video encoder, for sequence Akiyo (QCIF).

is a typical head-and-shoulders sequence, with very small motion. This means that the motion-compensated residue generally has a very small, or null, amplitude, particularly for the B slices. When high compression ratios are used, the encoder tends to use the null blocks very often. This has an unfavourable effect on the comparative performance of MMP-Video, for two reasons: first, the use of the original H.264/AVC tends to be more efficient in compressing the null residues, due to its highly optimised choice of coding parameters and adaptiveness of the CABAC entropy encoder; and second, the repetitive

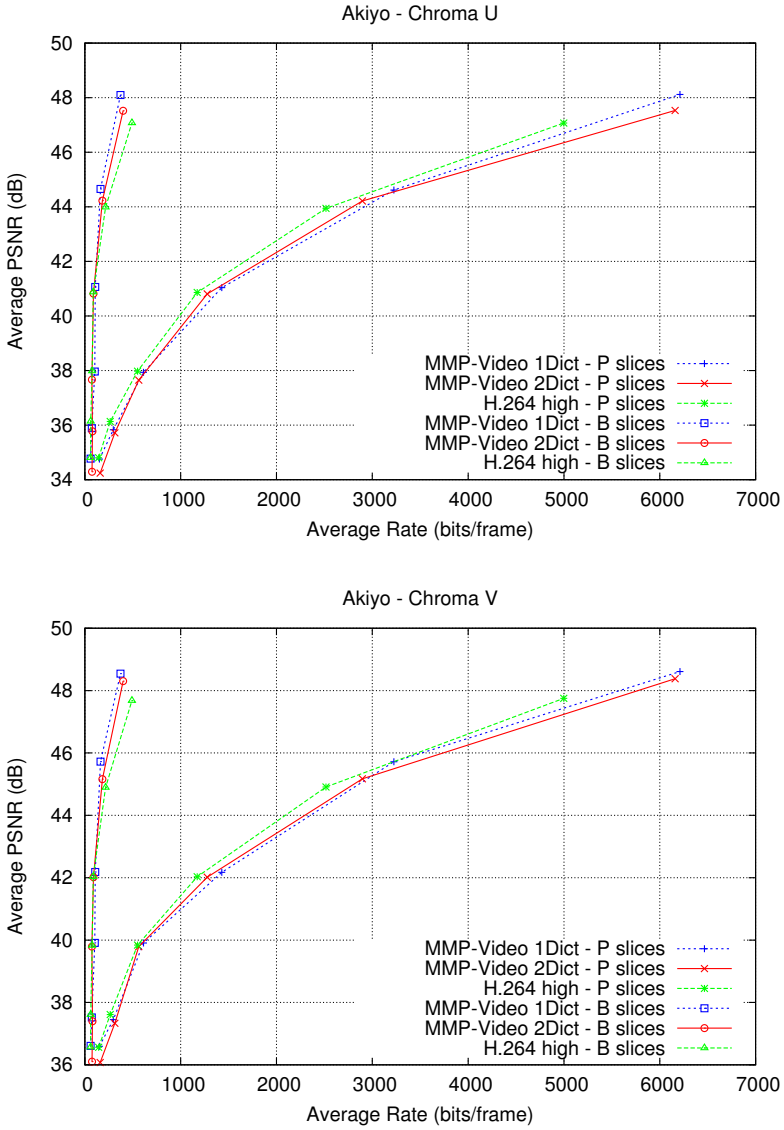


Figure 8.16: Comparative results for MMP-Video encoder and the H.264/AVC high profile video encoder, for sequence Akiyo (QCIF).

use of the null codeword by MMP-Video tends to limit the dictionary growth, meaning that, when the MC fails to produce a near null residue, the encoder has more difficulty in finding a good match in the dictionary.

Figure 8.17 presents a comparison between the performances of MMP-Video, H.264/AVC and H.264/AVC No MPE, described in Section 8.2. We may observe that MMP-Video is able to achieve the same good results as H.264/AVC No MPE, for the medium to the higher rates. Nevertheless, the performance limitation that was observed for H.264/AVC

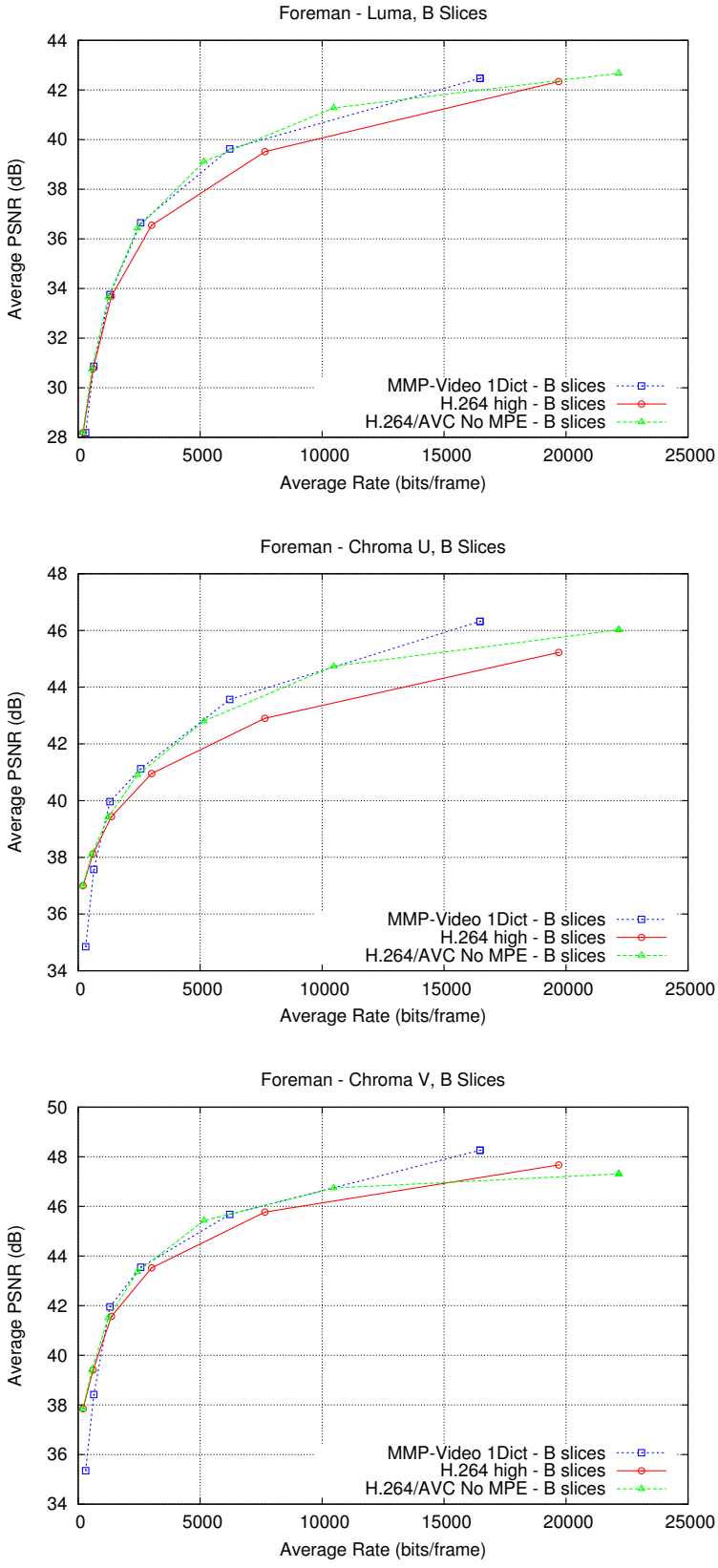


Figure 8.17: Comparative results for MMP-Video encoder and the H.264/AVC No MPE video encoder, for sequence Foreman (CIF).

No MPE for higher rates does not exist for MMP-Video, since, in this case, the encoder is able to achieve the low distortion levels required for these cases.

Figures C.50 to C.53 (see Appendix C) show the results for other previously discussed options of the MMP-Video encoder, namely the use of 6 independent dictionaries, the use of ABS, overriding of the deblocking filter and the use of the norm equalisation procedure in MMP-Video, for sequence Foreman.

8.4 Conclusions

In this chapter we have proposed two successful algorithms for video compression. In Section 8.2 we describe H.264/AVC No MPE: an H.264/AVC-based compression scheme that does not use the transmission of the motion-compensated residue. An interesting relation was established between the described algorithm and traditional pattern-matching coding methods, namely Lempel-Ziv and VQ. Part of the bit-rate that is usually associated with the compression of the motion-compensated predicted residue is used by the proposed encoder in more elaborate prediction modes. Experimental tests show that, when the motion-compensated error is not encoded for B slices, the proposed compression method has compression gains over the original standard, for low to medium rates.

The use of MMP for video coding was also investigated and a new method, referred to as MMP-Video, was presented in Section 8.3. The use of MMP in a hybrid video coder framework took into consideration some important functional optimisations. Some of these techniques resulted from the investigation on MMP as a digital image encoder, presented in the previous chapters. Nevertheless, new techniques were especially designed in order to exploit the particular characteristics of digital video encoding. Experimental results show that the general coding performance of the MMP-Video encoder is better than the one of H.264/AVC high profile, especially for medium to high bit-rates and for the B slice data, where the coding gains range up to 2dB. Moreover, MMP-Video avoids the performance limitations of H.264/AVC No MPE for higher rates, due to the absence of the motion predicted error.

The described results for MMP-Video show that, in spite of its larger computational complexity, the multiscale recurrent pattern matching paradigm may also be used in video compression. These results also show the importance of using appropriate dictionary design

techniques for MMP-Video.

Chapter 9

Conclusions and future work

This thesis investigated new and efficient image and video compression schemes based on the multiscale recurrent pattern matching coding paradigm. A previously proposed compression algorithm, referred to as multidimensional multiscale parser (MMP) was used as the starting point for the development of this paradigm. After an introduction to the pattern matching compression paradigm and the MMP algorithm, the two main goals of this research work were addressed:

- New MMP-based image compression algorithms were proposed, that achieve a similar coding performance to the state-of-the-art image compression schemes for smooth images, while maintaining, or even improving, the MMP gains for non-smooth image coding;

- A new MMP-based video compression algorithm, with state-of-the-art compression results, was developed.

Chapter 2 presents a brief introduction to pattern matching-based algorithms, with a special focus on applications to the compression of image and video signals.

Chapter 3 presented a description of the MMP algorithm for image compression. An experimental evaluation of MMP was also made, in order to assess the compression performance of the original method against that of the state-of-the-art image compression algorithms. Some functional aspects of MMP, related with the bit-stream structure and code-vector usage, were investigated. This knowledge proved to be very useful in the development of new compression techniques, described in later sections of the thesis.

Chapter 4 presented a formal analysis of the MMP algorithm, revealing other insights. MMP was then regarded both as an extension of traditional vector quantisation methods and as a string matching-based algorithm (Lempel-Ziv). Some other analogies were also described, namely between MMP and transform-based decomposition schemes or non-uniform sampling procedures. These insights were at the genesis of some of the work presented in other sections of the document.

As suggestions for future work, one could exploit the non-uniform sampling analogy, in order to develop alternative ways to reconstruct the MMP-compressed signal, namely by using other interpolation methods. The pattern matching nature of MMP could also be explored, namely by adapting some of the traditional VQ dictionary design strategies to MMP.

Chapter 5 first presented a study that demonstrated the advantage of using sources in MMP with more concentrated probability distributions, that were represented by generalised Gaussian functions. Based on this investigation, a new image compression algorithm, that combines MMP's multiscale recurrent pattern matching and predictive coding paradigms, was proposed. The new algorithm, referred to as MMP-I, uses adaptive image prediction techniques, similar to the ones defined for the H.264/AVC compression standard, to transform the input image's pixels into predicted residue samples with highly concentrated probability distributions. The use of the predicted error signal favours the adaptation of MMP's dictionary, leading to a significant improvement on the compression performance, especially for smooth images, where the prediction process has a higher accuracy. An experimental evaluation of MMP-I's performance revealed that the advantage over the state-of-the-art transform-based compression algorithms was maintained for non-smooth images, like text and graphics. For smooth images, where MMP presented a clear performance deficit, the PSNR gains put the MMP-I performance only about 0.5 dB below that of JPEG2000 and H.264/AVC intra frame encoder.

As a suggestion for future work, the generalised Gaussian-like probability distributions of the prediction signal could be exploited by relating MMP-I compression with a pyramid VQ scheme [78], namely for the development of new, more computationally efficient versions of the encoder. Another interesting research line would be the use of fast VQ search algorithms, to improve the computational efficiency of MMP algorithms. A simple method was used for MMP-I, with good results, but the investigation on the performance of faster,

sub-optimum search schemes is an open problem.

Chapter 6 presented some new techniques related with dictionary adaptation for MMP-I. The use of these methods resulted in a new compression algorithm, referred to as MMP-II, that presents a better coding performance for all tested images. The proposed schemes are related with several aspects of the MMP encoding process: the use of context conditioning for the adaptive arithmetic encoder (see Section 6.1) improves the lossless compression of MMP dictionary indexes' symbols; redundancy reduction techniques (see Section 6.2) selectively remove disadvantageous dictionary blocks, reducing the indexes' average entropy and increasing the encoding performance; enhanced dictionary updating strategies (see Section 6.3) introduce new code-vectors that will be useful for future block approximations; scale-adaptive updating schemes for the dictionary (see Section 6.4) allowed a significant reduction in processing time with only negligible performance losses; a L^1-norm equalisation procedure (see Section 6.5) adapts the new dictionary patterns according to a probability distribution criterion.

Despite their MMP-based genesis, most of the proposed schemes are generic enough to be used with other adaptive pattern matching algorithms. As an example, one could consider the use of the proposed redundancy control techniques in a lossy LZ78-based encoder (*e.g.* [35]), in order to avoid the insertion of disadvantageous code-vectors.

Experimental results showed that MMP-II is able to achieve quality gains over both MMP and MMP-I for every tested image and all compression ratios. For non-smooth images, these gains further increase the advantage of MMP-based algorithms over the state-of-the-art, transform-based encoders. For smooth images, the methods proposed in this chapter allow MMP-II to reach a performance level similar to that of the JPEG2000 algorithm, fulfilling one of the main objectives of this thesis.

Future work could include the exploitation of the MMP-II features to compress other signal types. This thesis was mainly focused on increasing smooth image compression efficiency. In spite of the gains that were observed also for non-smooth images, the problem of optimising the encoding performance for these image classes was not specifically addressed. An open research line is the optimisation of MMP-based algorithms for text and compound image compression. Another interesting problem is the optimisation of MMP-II compression methods for other signal types. An example would be the compression of stereoscopic images and video sequences, or electrocardiogram (ECG) signals [15, 16].

In Chapter 7 we revisited the prediction process used by MMP-II and proposed a new MMP-II post-processing deblocking method. The initial proposal of MMP-I inherited the prediction methods of the H.264/AVC standard. In this chapter, an evaluation of the used prediction scheme was performed, and some optimisations were introduced, related with prediction block size and the use of additional prediction modes. Additionally, a new deblocking method for MMP-II was investigated, motivated by some blocking artifacts observed in the MMP-II images, especially for highly compressed smooth images. The proposed deblocking algorithm uses an adaptive filter, that varies depending on the image region that is being processed. The filtering parameters are optimised by the encoder and then transmitted to the decoder using a negligible overhead. This maximises the deblocking performance for smooth images and eliminates the traditional PSNR losses for deblocked non-smooth images, making the proposed scheme suitable for any image type. Additionally, for the smooth regions, experimental results show an improvement in both objective and subjective quality levels.

The investigation on the use of other prediction methods in MMP-II is an interesting future research problem. Several new prediction paradigms have been recently proposed [95, 96, 97, 98], which could provide nice performance gains in MMP-II, especially for smooth image compression.

Chapter 8 investigated the use of MMP for video compression. As a results a new algorithm, MMP-Video, was proposed. Some functional optimisations were developed specifically for MMP-Video. The use of some of the previously mentioned techniques, developed in the MMP-II framework, was also considered. MMP-Video encoder achieved a better encoding performance than the current state-of-the-art encoder, the H.264/AVC high profile, for medium to high bit-rates and for the B slice data, where the coding gains range up to 2dB. These results show that the multiscale recurrent pattern encoding paradigm may also be successfully used for video compression, fulfilling the second major objective of this thesis. Additionally, an H.264/AVC-based compression scheme that does not use the transmission of the motion-compensated residue was also investigated. Some interesting relations were established, in order to regard this algorithm as a pattern matching video compression scheme. Experimental tests showed that this method is able to achieve better results than the original H.264/AVC standard, for low to medium rates.

Future work options include the investigation on the use of fast intra and inter frame prediction methods for MMP-Video. Many related techniques have been proposed for H.264/AVC, with good results [99, 100, 101]. Such schemes are a promising research line in the reduction of MMP-Video's computational complexity. The research on MMP-Video demonstrated good performance on the compression of the chroma components of the video sequences. As future work one could investigate optimised ways to compress colour images, by exploiting the particular features of the several image components, for different colour spaces.

As a summary, one may conclude that the use of the multiscale recurrent pattern matching coding paradigm, and of MMP-based methods, may lead to successful image compression schemes. For smooth images, the proposed techniques were able to achieve a coding performance that is comparable to the one of the current state-of-the-art transform-based image compression standards. Nevertheless, unlike the traditional algorithms, the proposed methods also achieve top performances for non-smooth images, like text and compound images, assigning them a very useful universal feature. Additionally, the proposed MMP-based video compression algorithm also demonstrated an encoding performance above the levels achieved by the current state-of-the-art standard. These results show that, in spite of its larger computational complexity, the multiscale recurrent pattern matching paradigm is not only a viable alternative for image and video compression, that is worth further investigation.

Appendix A

Test Signals

A.1 Test images

Figure A.1: Grayscale natural test image Lena, 512×512.

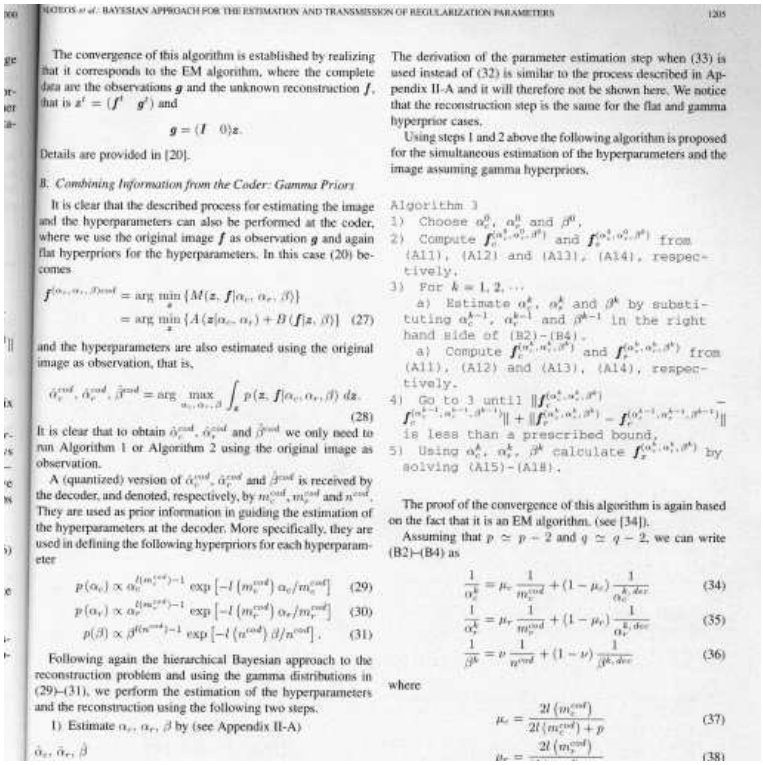


Figure A.2: Grayscale text image PP1205, 512×512.

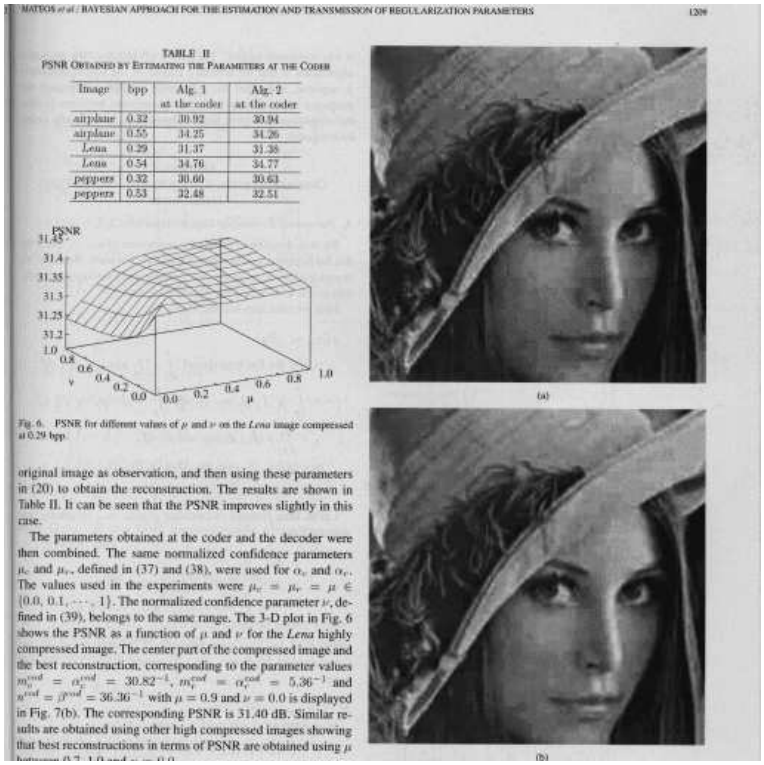


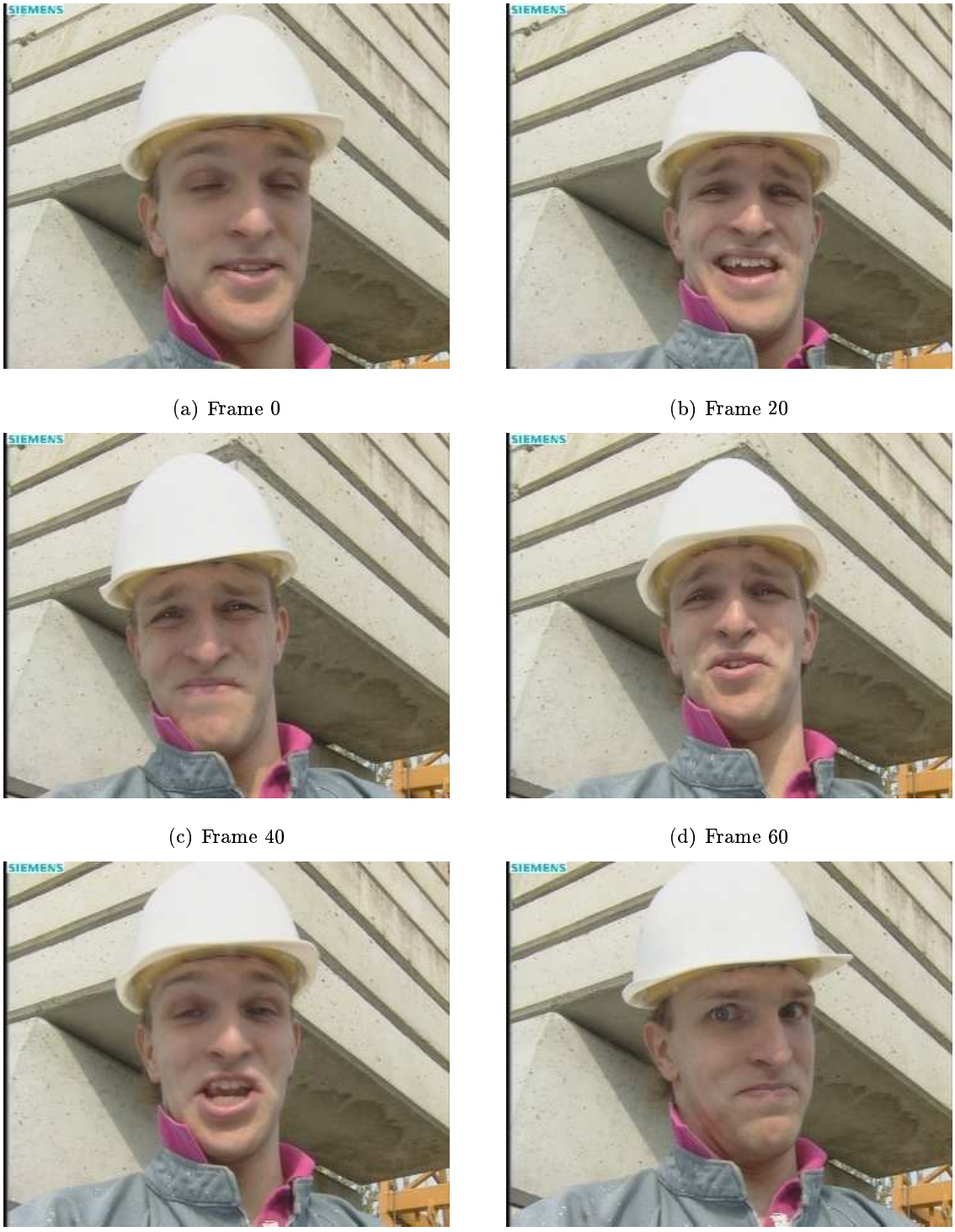
Figure A.3: Grayscale compound (text and graphics) test image PP1209, 512×512.

Figure A.4: Grayscale natural test image Goldhill, 512×512.

Figure A.5: Grayscale natural test image Peppers, 512×512.

Figure A.6: Grayscale natural test image Cameraman, 256×256.

A.2 Test video sequences



(a) Frame 0

(b) Frame 20

(c) Frame 40

(d) Frame 60

(e) Frame 80

(f) Frame 99

Figure A.7: Some frames of the test video sequence Foreman (CIF: 352×288).

(a) Frame 0

(b) Frame 20

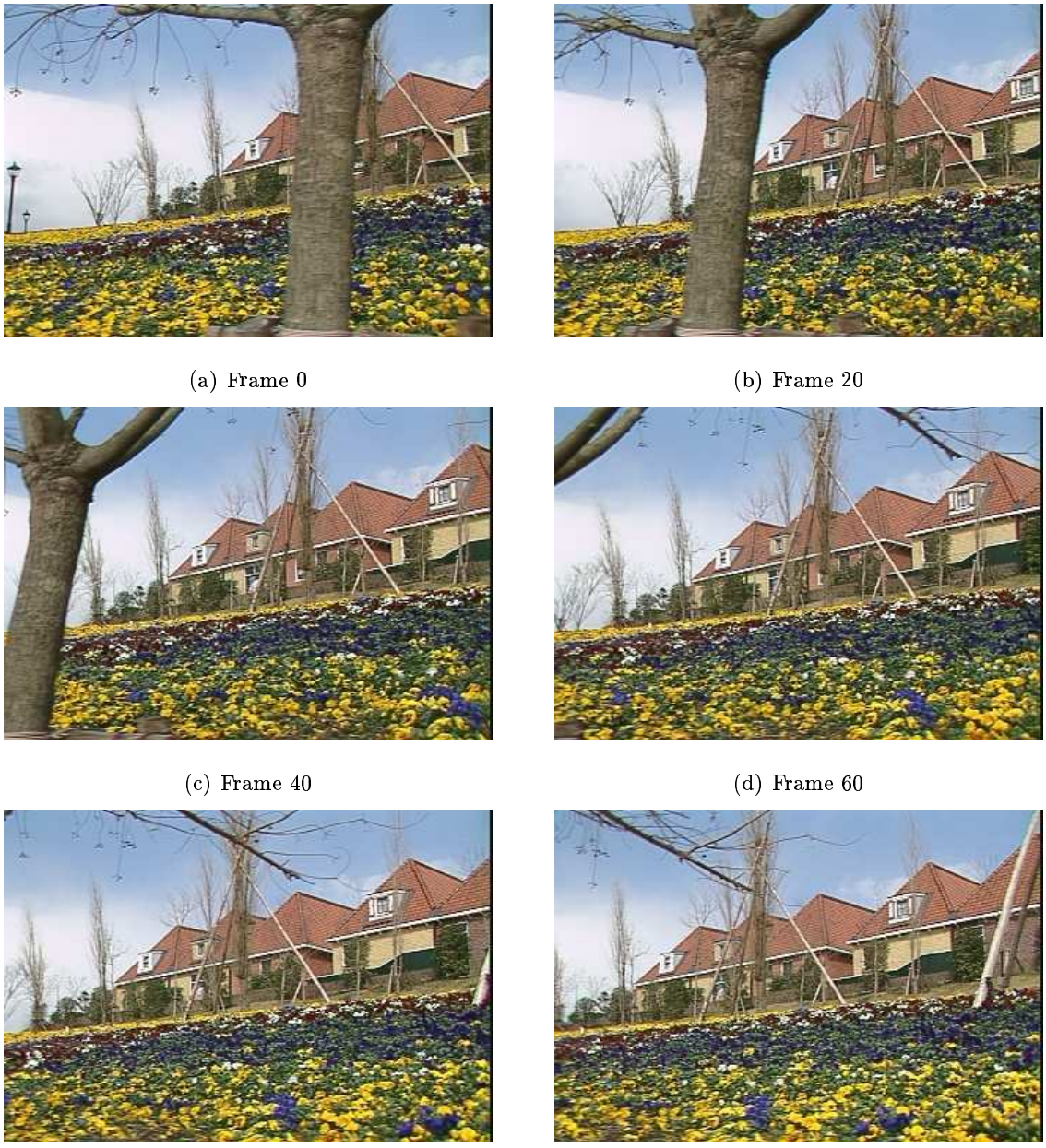
(c) Frame 40

(d) Frame 60

(e) Frame 80

(f) Frame 99

Figure A.8: Some frames of the test video sequence Mobile & Calendar (CIF: 352 × 288).



(a) Frame 0

(b) Frame 20

(c) Frame 40

(d) Frame 60

(e) Frame 80

(f) Frame 99

Figure A.9: Some frames of the test video sequence Flower Garden (SIF: 352×240).

(a) Frame 0 (b) Frame 20

(c) Frame 40 (d) Frame 60

(e) Frame 80 (f) Frame 99

Figure A.10: Some frames of the test video sequence Akiyo (QCIF: 176×144).

Appendix B

Implementation details for the MMP algorithms

B.1 Introduction and basic definitions

In this appendix we present a detailed description of the Multidimensional Multiscale Parser-based algorithms that have been discussed throughout this document. All these methods partition the input image into blocks of fixed size, $\mathbf{X}^l$, that are compressed independently. A standard raster scan order is used for block selection. MMP uses dyadic segmentations of each original image block, to divide the image into variable size texture regions, that are encoded using vectors from a multi-scale adaptive dictionary, $\mathcal{D}$. The decision to perform the bisection of each sub-block is based on rate-distortion criteria and repeated recursively. The segmentation structure that results from this optimisation procedure is represented by a binary segmentation tree, $\mathcal{T}$, as shown in Figure B.1.

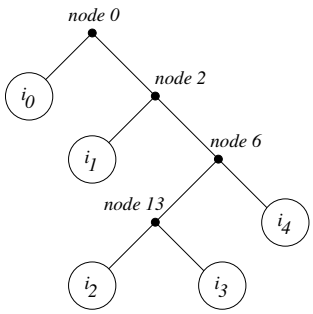


Figure B.1: Segmentation tree for a block encoded with MMP.

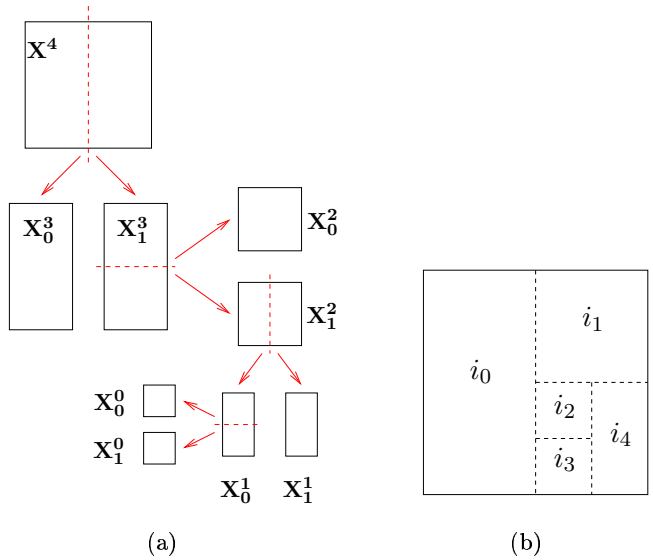


Figure B.2: *Vertical segmentation* of an original 4×4 block (level 4): a) segmentation decisions; b) block partitioning.

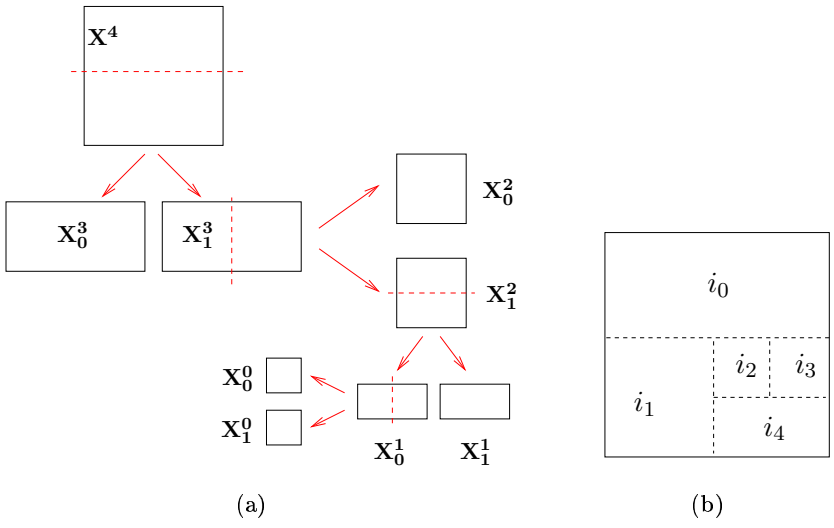


Figure B.3: *Horizontal segmentation* of an original 4×4 block (level 4): a) segmentation decisions; b) block partitioning.

We define $\mathbf{X}^l$ as an input image block of scale l. This scale value (identified by the use of the superscript l) has a direct correspondence with the *level* of the segmentation tree, $\mathcal{T}$, that it corresponds to. There is also a direct relation between this value and the dimensions of $\mathbf{X}^l$:

- If the square blocks, corresponding to even values of l, are segmented into two *vertical rectangles* (Figure B.2), then a block of level l has dimensions:

$$2^{\lfloor \frac{l+1}{2} \rfloor} \times 2^{\lfloor \frac{l}{2} \rfloor}; \tag{B.1}$$

- If the square blocks are segmented into two *horizontal rectangles* (Figure B.3), then a block $\mathbf{X}^l$ has dimensions:

$$2^{\lfloor \frac{l}{2} \rfloor} \times 2^{\lfloor \frac{l+1}{2} \rfloor}. \tag{B.2}$$

Note that both examples of Figures B.2 and B.3 use the segmentation tree represented in Figure B.1. The level 0 of $\mathcal{T}$ (the deepest level) always corresponds to blocks of dimension 1×1. The root of $\mathcal{T}$ is always located at the level that corresponds to the size of $\mathbf{X}^l$. Each node of the binary tree represents a decision to divide the corresponding block, while a tree leaf corresponds to an undivided rectangular sub-block. A tree leaf is represented by a binary flag set to '0' while a segmentation is represented by a '1'. The following sections have an algorithmic representation of the considered methods. A detailed description of these methods may be found in the main text of this thesis.

The rate-distortion optimisation procedures use a *Lagrangian multiplier*, λ [54, 55] in order to weight both the distortion of one block and the rate needed to encode the associated information. This optimisation problem is solved by minimising the value of a cost function $J(\mathcal{T})$, given by:

$$J(\mathcal{T}) = D(\mathcal{T}) + \lambda R(\mathcal{T}), \tag{B.3}$$

where $D(\mathcal{T})$ is the distortion of the approximation represented by $\mathcal{T}$ and $R(\mathcal{T})$ is the rate used to encode this approximation. The sum of squared differences (SSD) is used as the distortion measure between two blocks, $\mathbf{X}^l$ and $\mathbf{S}_i^l$. It is defined as:

$$D(\mathbf{X}^l, \mathbf{S}_i^l) = \sum_{y=1}^{M} \sum_{x=1}^{N} \left(\mathbf{X}^l(y,x) - \mathbf{S}_i^l(y,x) \right)^2. \tag{B.4}$$

Consider a block $\mathbf{X}^l$ of scale l, corresponding to a node n_j in the segmentation tree. n_j can be a tree leaf (in this case $\mathbf{X}^l$ is approximated by a single dictionary vector) or

it can be a tree node, parent of two nodes n_{2j+1} and n_{2j+2} (in this case, n_j signals the segmentation of $\mathbf{X}^l$). This decision is made after comparing the cost functions of the original approximation with the one of the two segments. More precisely, if n_j is a leaf node, its cost is given by

$$J(\text{leaf}(n_j)) = D(\mathbf{X}^l, \mathbf{S}_j^l) + \lambda R(\text{leaf}(n_j)), \tag{B.5}$$

where $R(\text{leaf}(n_j))$ is the rate spent to encode node n_j as a leaf node. $R(\text{leaf}(n_j))$ is the sum of the rate spent to encode the dictionary vector, $R(\mathbf{S_j^l})$, with the rate of the binary flag that signals that the block should not be segmented, $R(f_1^l)$, *i.e.*

$$R(\text{leaf}(n_j)) = R(\mathbf{S}_j^l) + R(f_1^l). \tag{B.6}$$

The rate values are determined based on the probability of the used symbol in the corresponding histogram. Independent probability distributions are kept for the segmentation flags and dictionary indexes of each scale. Thus, we have that

$$R(x) = log_2 \frac{1}{P(x|l(x))} = -log_2 \left(P(x|l(x))\right), \tag{B.7}$$

where x is the current symbol and $l(x)$ is the scale of x, that corresponds to its encoding context. The expression $P(x|l(x))$ represents the probability of symbol x in the context (probability histogram) corresponding to scale $l(x)$. As an example, for the particular case of dictionary indexes we have that

$$R(\mathbf{S}_j^l) = -log_2(P(i^l|\mathcal{D}^l)), \tag{B.8}$$

where $P(i^l|\mathcal{D}^l)$ is the probability of occurrence of the vector with index i of level l of the dictionary.

The Lagrangian cost of approximating block $\mathbf{X}^l$ using the concatenation of two nodes n_{2j+1} and n_{2j+2} is estimated by:

$$J(node(n_j)) = J(n_{2j+1}) + J(n_{2j+2}) + \lambda R(f_0^l). \tag{B.9}$$

where $R(f_0^l)$ corresponds to the rate used to encode flag '0' at level l.

B.2 Algorithm for MMP

The MMP compression[1] of an input image block, $\mathbf{X}^l$, may be defined by two main steps (see Figure B.4):

[1] See Section 3.1 for a detailed description of the MMP algorithm.

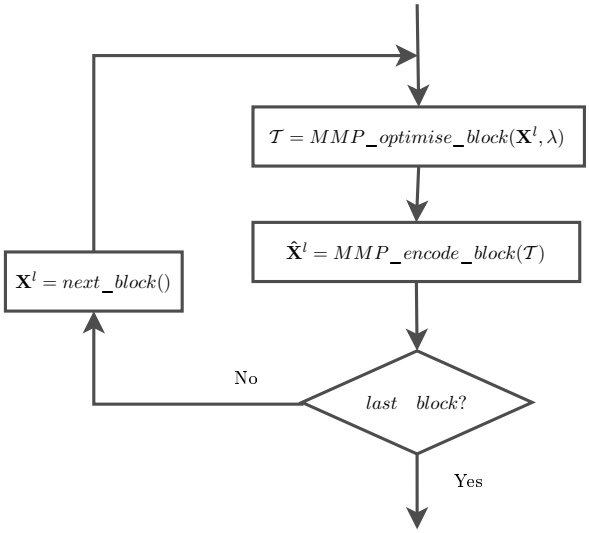


Figure B.4: Flowchart for the MMP algorithm.

- First: procedure $\left\{\hat{\mathbf{X}}^l, \mathcal{T}, J(\mathcal{T})\right\} = MMP\_optimise\_block(\mathbf{X}^l, \lambda)$ performs an *optimisation* of the encoded block. It receives an input block, $\mathbf{X}^l$ of scale l, as well as the value for the Lagrangean RD optimisation parameter, λ. It determines the best MMP representation of the original blocks in a RD sense and returns the MMP tree $\mathcal{T}$ that represents it, as well as the resulting approximated block, $\hat{\mathbf{X}}^l$. $J(\mathcal{T})$ is the Lagrangean cost of the approximation performed by $\mathcal{T}$. This procedure is described in Section B.2.1;
- Second: procedure $\hat{\mathbf{X}}^l = MMP\_encode\_block(\mathcal{T})$ performs the *encoding* of the previously optimised block. This procedure analyses the MMP segmentation tree, $\mathcal{T}$, determined by the optimisation function, generates the MMP string with the symbols for the segmentation flags and dictionary indexes and encodes each symbol using the adaptive arithmetic encoder. Also, for each block segmentation, the encoding procedure performs the relevant dictionary updating operations. This procedure is described in Section B.2.2;

B.2.1 MMP optimisation procedure

Procedure $\left\{\hat{\mathbf{X}}^\mathbf{l}, \mathcal{T}, \mathbf{J}(\mathcal{T})\right\} = \mathbf{MMP\_optimise\_block}(\mathbf{X}^\mathbf{l}, \lambda)$

Step 1: search $\mathcal{D}^l$ for the element $\mathbf{S}_i^l$ that minimises the approximation cost defined by $J(\mathcal{T}') = D(\mathbf{X}^l, \mathbf{S}_i^l) + \lambda . R(\mathbf{S}_i^l)$, where $D(\mathbf{X}^l, \mathbf{S}_i^l)$ is the squared error between

the input block and the dictionary element and $R(\mathbf{S}_i^l)$ is the rate needed to encode index i of $\mathcal{D}^l$;

Step 2: segment $\mathbf{X}^l$ into two blocks: $\mathbf{X}_0^{l-1}$ and $\mathbf{X}_1^{l-1}$ using the appropriate rule (vertical or horizontal);

Step 3: compute $\left\{\hat{\mathbf{X}}_0^{l-1}, \mathcal{T}_0, J(\mathcal{T}_0)\right\} = MMP\_optimise\_block(\mathbf{X}_0^{l-1}, \lambda)$;

Step 4: compute $\left\{\hat{\mathbf{X}}_1^{l-1}, \mathcal{T}_1, J(\mathcal{T}_1)\right\} = MMP\_optimise\_block(\mathbf{X}_1^{l-1}, \lambda)$;

Step 5: determine the cost of approximating $\mathbf{X}^l$ using a single block from $\mathcal{D}^l$:

$$J_{leaf} = J(\mathcal{T}') + \lambda.R(f_0^l),$$

where $R(f_0^l)$ is the number of bits needed to encode the leaf flag '0' of level l;

Step 6: determine the cost of approximating $\mathbf{X}^l$ using a segmentation:

$$J_{node} = J(\mathcal{T}_0) + J(\mathcal{T}_1) + \lambda.R(f_1^l),$$

where $R(f_1^l)$ is the number of bits needed to encode the node flag '1' of level l;

Step 7: if $(J_{leaf} <= J_{node})$, then do

$\hat{\mathbf{X}}^l = \mathbf{S}_i^l$;

$\mathcal{T} = leaf(\mathbf{S}_i^l)$;

$J(\mathcal{T}) = J_{leaf}$;

Step 8: else do

$\hat{\mathbf{X}}^l = \left(\hat{\mathbf{X}}_0^{l-1} : \hat{\mathbf{X}}_1^{l-1}\right)$, where $(\ :\ )$ means the concatenation of two blocks according to the appropriate direction (vertical or horizontal);

$\mathcal{T} = (\mathcal{T}_0 \bigcup \mathcal{T}_1)$, where $(\ \bigcup\ )$ is the combination of two binary trees under a common root node;

$J(\mathcal{T}) = J_{node}$;

Step 9: return $\left\{\hat{\mathbf{X}}^l, \mathcal{T}, J(\mathcal{T})\right\}$.

B.2.2 MMP encoding procedure

Procedure $\hat{\mathbf{X}}^{\mathbf{l}} = \mathbf{MMP\_encode\_block}(\mathcal{T})$

Step 1: if $l = 0$ then go to step 5;

Step 2: set $\mathcal{T}_0 = left\_node(\mathcal{T})$ and $\mathcal{T}_1 = right\_node(\mathcal{T})$;

Step 3: if $\mathcal{T}_0 = 0$ and $\mathcal{T}_1 = 0$ go to step 4; else go to step 7;

Step 4: encode and output flag '0' of level l;

Step 5: encode and output index i of level l, representing element $\mathbf{S}_i^l$ associated with $\mathcal{T}$;

Step 6: return $\hat{\mathbf{X}}^l$;

Step 7: encode and output flag '1' of level l;

Step 8: do $\hat{\mathbf{X}}_0^{l-1} = MMP\_encode\_block(\mathcal{T}_0)$;

Step 9: do $\hat{\mathbf{X}}_1^{l-1} = MMP\_encode\_block(\mathcal{T}_1)$;

Step 10: determine $\hat{\mathbf{X}}^l = \left(\hat{\mathbf{X}}_0^{l-1} : \hat{\mathbf{X}}_1^{l-1}\right)$;

Step 11: for each scale l_i, do:

determine $\hat{\mathbf{X}}^{l_i} = T_{l_i}^l\left(\hat{\mathbf{X}}^l\right)$, where $T_{l_i}^l()$ performs a scaling operation from original scale l to scale l_i;

update dictionary $\mathcal{D}^{l_i}$ with block $\hat{\mathbf{X}}^{l_i}$, unless a similar element already exists in $\mathcal{D}^{l_i}$;

Step 12: return $\hat{\mathbf{X}}^l = \left(\hat{\mathbf{X}}_0^{l-1} : \hat{\mathbf{X}}_1^{l-1}\right)$.

B.3 Algorithm for MMP-I FBS.

The MMP-I FBS (Fixed Block Size) algorithm[2] considers image blocks of constant size for the prediction step. This size corresponds to the top level of the segmentation tree for the MMP algorithm. For each input block, MMP-I FBS first determines all prediction

[2]See Section 5.2 for a detailed description of the MMP-I FBS algorithm.

modes available for $\mathbf{X}^l$. Then it performs the prediction step using each available mode and optimises the MMP compression of the corresponding prediction error block, by using the previously described MMP encoding procedure. Finally, the algorithm chooses the best prediction mode, based on the determined values of the RD cost functions. It is implemented using two procedures:

- procedure $\left\{\hat{\mathbf{X}}^l, \mathcal{T}, J(\mathcal{T}), M\right\} = MMPI\_FSB\_optimise\_block(\mathbf{X}^l, \lambda)$, that determines the best prediction mode, M, among the available modes for the current block and returns the MMP data that optimises the compression of the predicted residue. $\mathbf{R}_M^l$ and $\mathbf{P}_M^l$ are, respectively, the prediction residue and prediction blocks for block $\mathbf{X}^l$ using mode M.
- procedure $\hat{\mathbf{X}}^l = MMPI\_FBS\_encode\_block(\mathcal{T}, M)$ analyses the MMP-I segmentation tree and encodes all the symbols associated with the block representation.

As for the MMP algorithm, each image block is first processed by the optimisation routine and the results of this function are then passed to the encoding procedure. A definition of these two procedures is presented in the following sections, while a flowchart of the algorithm is shown in Figure B.5.

B.3.1 MMP-I FSB optimisation procedure

Procedure $\left\{\hat{\mathbf{X}}^\mathbf{l}, \mathcal{T}, \mathbf{J}(\mathcal{T}), \mathbf{M}\right\} = \mathbf{MMPI\_FSB\_optimise\_block}(\mathbf{X}^\mathbf{l}, \lambda)$

Step 1: determine the set of prediction modes that are available for block $\mathbf{X}^l$, based on the previously encoded image pixels, $\mathcal{M}$;

Step 2: initialise $J_{min} = \infty$ and $M = 0$;

Step 3: for each available prediction mode, $m \in \mathcal{M}$, do:

determine $\mathbf{P}_m^l$ and $\mathbf{R}_m^l$ by performing block prediction using mode m;

compute $\left\{\hat{\mathbf{R}}_m^l, \mathcal{T}_m^l, J(\mathcal{T}_m^l)\right\} = MMP\_optimise\_block(\mathbf{R}_m^l, \lambda)$ (see sec. B.2.1);

set $J(\mathcal{T}_m^l) = J(\mathcal{T}_m^l) + \lambda.R(f_m^l)$, where $R(f_m^l)$ is the rate used to encode prediction flag m of level l;

if $\left(J(\mathcal{T}_m^l) < J_{min}\right)$ then set $J_{min} = J(\mathcal{T}_m^l)$ and $M = m$;

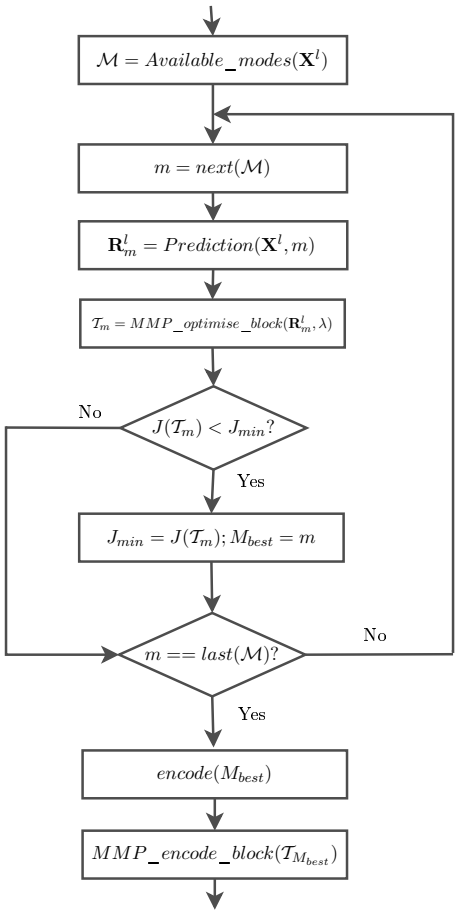


Figure B.5: Flowchart for the MMP-I FBS algorithm.

Step 4: determine $\hat{\mathbf{X}}^l = \mathbf{P}_M^l + \mathbf{R}_M^l$;

Step 5: return $\left\{\hat{\mathbf{X}}^l, \mathcal{T}_\mathcal{M}, J(\mathcal{T}_\mathcal{M}), M\right\}$.

B.3.2 MMP-I FSB encoding procedure

Procedure $\hat{\mathbf{X}}^l = \mathbf{MMPI\_FBS\_encode\_block}(\mathcal{T}, \mathbf{M})$

Step 1: output prediction flag m of level l;

Step 2: do $\hat{\mathbf{R}}_M^l = MMP\_encode\_block(\mathcal{T}_M^l, M)$ (see sec. B.2.2);

Step 3: determine $\mathbf{P}_M^l$ by performing block prediction using mode M;

Step 4: determine $\hat{\mathbf{X}}^l = \mathbf{P}_M^l + \hat{\mathbf{R}}_M^l$;

Step 5: return $\hat{\mathbf{X}}^l$.

B.4 Algorithm for MMP-I.

The implementation of MMP-I[3] is based upon the MMP original optimisation and encoding procedures, combined with an hierarchical, adaptive block size prediction step. The prediction may use blocks with dimensions down to 4×4, corresponding to $l = 4$. In order to perform the joint optimisation of both the prediction mode and prediction block size, the possible combinations are tested recurrently, using the MMP-I FBS optimisation procedure, defined in the previous section. A new procedure is defined for this purpose: $\left\{\hat{\mathbf{X}}^l, \mathcal{T}, J(\mathcal{T})\right\} = MMPI\_optimise\_block(\mathbf{X}^l, \lambda)$. In this case, the segmentation tree structure was altered so that it also holds the information about the prediction procedure. Three flags are used:

- Flag '0' indicates a tree node (block segmentation) for which no prediction is necessary. This flag is used when no prediction mode has been set and both the prediction block and MMP block must be segmented or when the prediction step has already been performed, but the MMP block should be segmented;
- Flag '1' indicates a tree leaf. If no prediction mode has been previously set for the pixels of this block, this flag is followed by a prediction mode flag, that describes the prediction process that should be used at this scale. If prediction data has already been defined for this block, flag '1' signals a simple tree leaf and is followed by the dictionary index of the vector used for the approximation;
- Flag '2' corresponds to a tree node (MMP segmentation) but at a scale where prediction should be performed. It is always followed by the corresponding prediction mode flag. From this point on no further prediction should be used and all flags represent coding decisions related to the MMP process.

A flowchart of the MMP-I algorithm is shown in Figure B.6.

B.4.1 MMP-I optimisation procedure

Procedure $\left\{\hat{\mathbf{X}}^{\mathbf{l}}, \mathcal{T}, \mathbf{J}(\mathcal{T})\right\} = \mathbf{MMPI\_optimise\_block}(\mathbf{X}^{\mathbf{l}}, \lambda)$

Step 1: compute $\left\{\hat{\mathbf{X}}^l, \mathcal{T}, J(\mathcal{T}), M\right\} = MMPI\_FSB\_optimise\_block(\mathbf{X}^l, \lambda)$ (see sec. B.3.1);

[3] See Section 5.2 for a detailed description of the MMP-I algorithm.

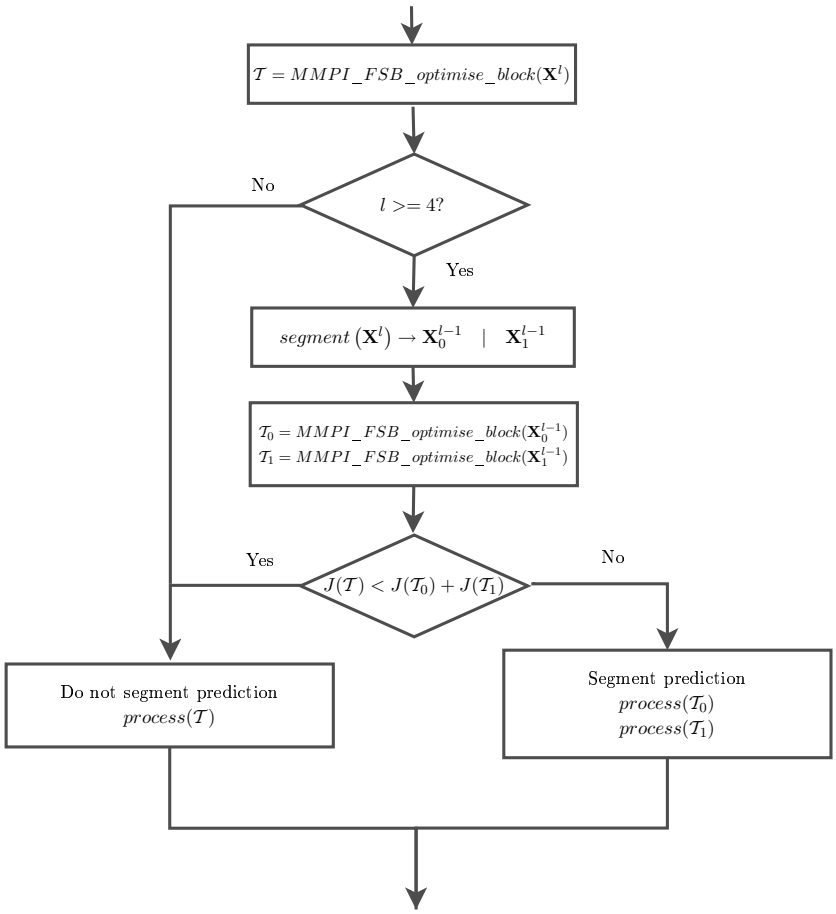


Figure B.6: Flowchart for the MMP-I algorithm.

Step 2: compute $J(\mathcal{T}) = J(\mathcal{T}) + \lambda.R(f_x^l)$, where $R(f_x^l)$ is the rate needed to encode the segmentation flag used for this case:

$f_x^l = 1$ if $(\mathcal{T}_0 = \mathcal{T}_1 = 0)$;

$f_x^l = 2$ if $(\mathcal{T}_0 \neq 0 \wedge \mathcal{T}_1 \neq 0)$;

Step 3: if $l = 4$ then return $\left\{\hat{\mathbf{X}}^l, \mathcal{T}, J(\mathcal{T})\right\}$;

Step 4: segment $\mathbf{X}^l$ into two blocks: $\mathbf{X}_0^{l-1}$ and $\mathbf{X}_1^{l-1}$ using the appropriate rule;

Step 5: compute $\left\{\hat{\mathbf{X}}_0^{l-1}, \mathcal{T}_0, J(\mathcal{T}_0)\right\} = MMPI\_optimise\_block(\mathbf{X}_0^{l-1}, \lambda)$;

Step 6: compute $\left\{\hat{\mathbf{X}}_1^{l-1}, \mathcal{T}_1, J(\mathcal{T}_1)\right\} = MMPI\_optimise\_block(\mathbf{X}_1^{l-1}, \lambda)$;

Step 7: if $\left(J(\mathcal{T}) < J(\mathcal{T}_0) + J(\mathcal{T}_1) + \lambda.R(f_0^l)\right)$, then the prediction should be made using mode M at level l, so return $\left\{\hat{\mathbf{X}}^l, \mathcal{T}, J(\mathcal{T})\right\}$, determined in step 2;

Step 8: else

set $\hat{\mathbf{X}}^l = \left(\hat{\mathbf{X}}_0^{l-1} : \hat{\mathbf{X}}_1^{l-1}\right)$, where (:) means the concatenation of two blocks according to the appropriate direction;

set $\mathcal{T} = \left(\mathcal{T}_0^{l-1} \bigcup \mathcal{T}_1^{l-1}\right)$, where ($\bigcup$) is the combination of two binary trees under a common root node;

set $J(\mathcal{T}) = \left(J(\mathcal{T}_0) + J(\mathcal{T}_1) + \lambda.R(f_0^l)\right)$;

return $\left\{\hat{\mathbf{X}}^l, \mathcal{T}, J(\mathcal{T})\right\}$.

B.4.2 MMP-I encoding procedure

Procedure $\hat{\mathbf{X}}^{\mathbf{l}} = \mathbf{MMPI\_encode\_block}(\mathcal{T})$

Step 1: read first segmentation flag from $\mathcal{T}$ and store it into f_x^l;

Step 2: if $\left(f_x^l = 0\right)$ then segment the block

output flag '0' of level l;

set $\mathcal{T}_0 = left\_node(\mathcal{T})$ and $\mathcal{T}_1 = right\_node(\mathcal{T})$;

compute $\hat{\mathbf{X}}_0^{l-1} = MMPI\_encode\_block(\mathcal{T}_0)$;

compute $\hat{\mathbf{X}}_1^{l-1} = MMPI\_encode\_block(\mathcal{T}_1)$;

set $\hat{\mathbf{X}}^l = \left(\hat{\mathbf{X}}_0^{l-1} : \hat{\mathbf{X}}_1^{l-1}\right)$;

Step 3: else, if $\left(f_x^l = 1\right)$ then:

output flag '1' of level l;

if the prediction process has not yet been performed then

compute $\hat{\mathbf{X}}^l = MMPI\_FBS\_encode\_block(\mathcal{T}, M)$ (see sec. B.3.2);

else

compute $\hat{\mathbf{X}}^l = MMP\_encode\_block(\mathcal{T})$ (see sec. B.2.2);

Step 4: else, if $\left(f_x^l = 2\right)$ then:

output flag '2' of level l;

compute $\hat{\mathbf{X}}^l = MMPI\_FBS\_encode\_block(\mathcal{T}, M)$ (see sec. B.3.2);

Step 5: return $\hat{\mathbf{X}}^l$.

Appendix C

Experimental results for the proposed methods

C.1 Complementary results for Chapter 3

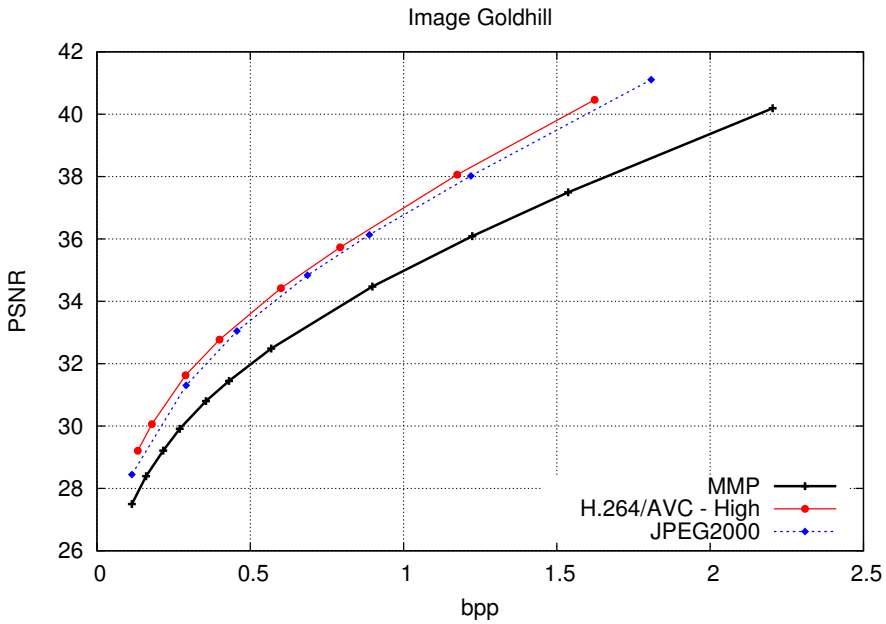


Figure C.1: Experimental results of MMP for natural test image Goldhill.

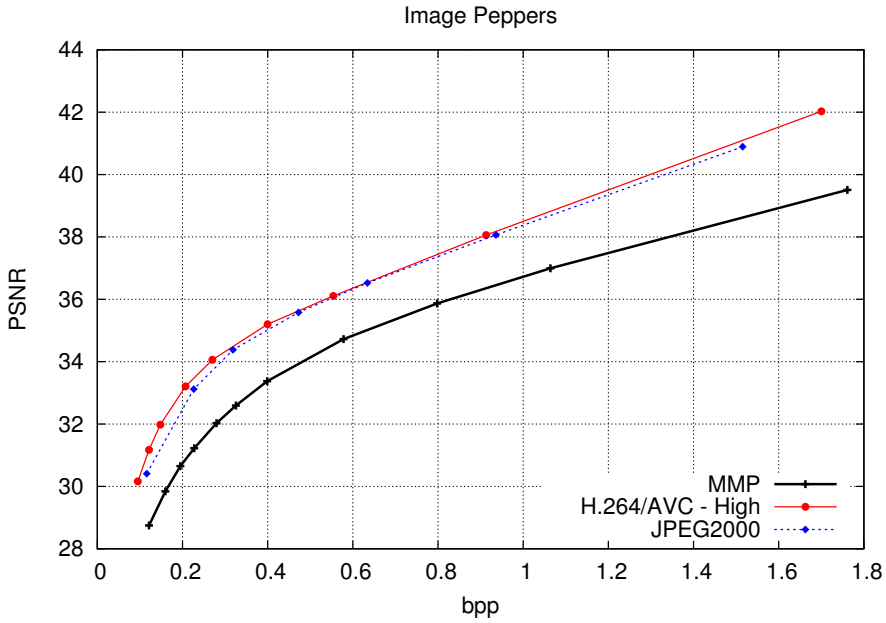


Figure C.2: Experimental results of MMP for natural test image Peppers.

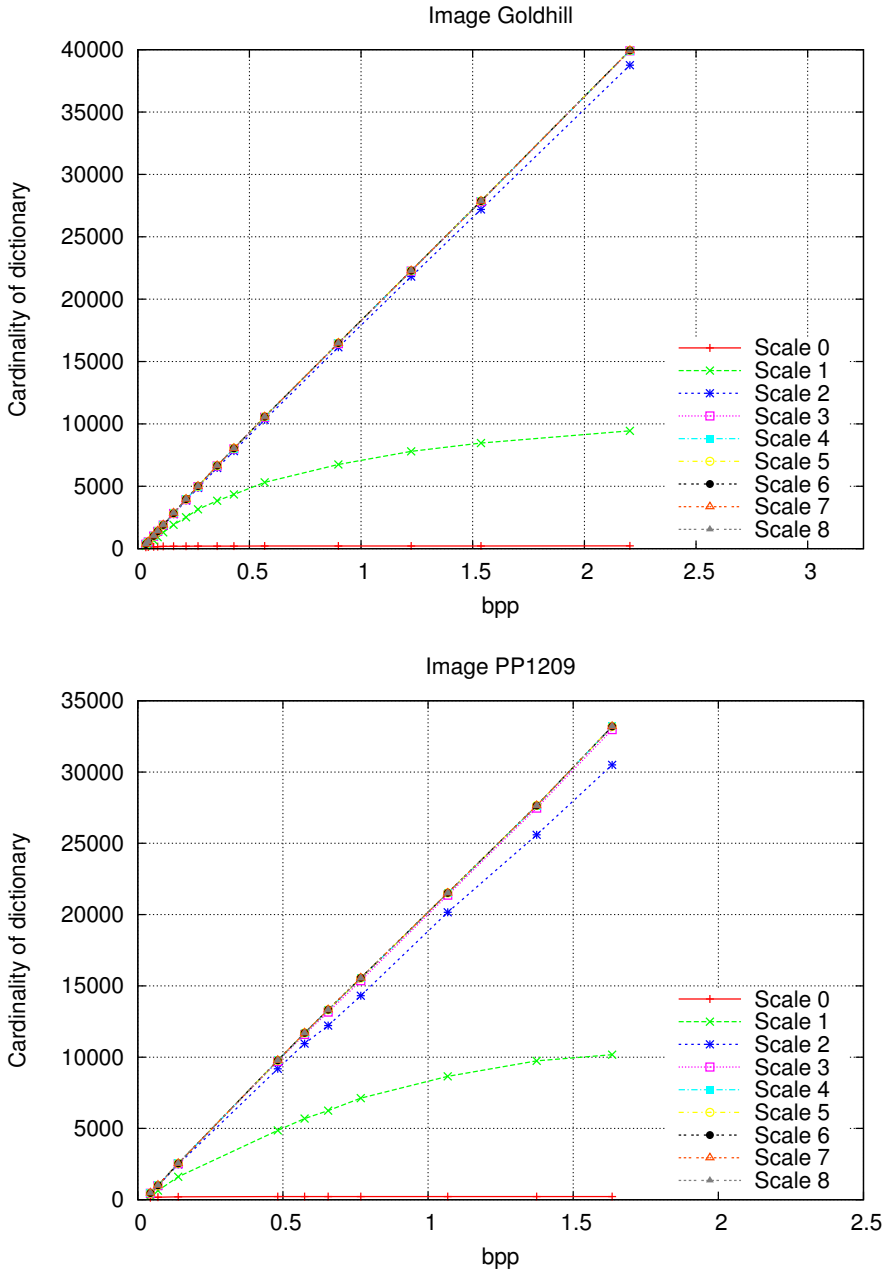


Figure C.3: Final number of elements for the dictionary for MMP, as a function of the image compression ratio, for images Goldhill and PP1209.

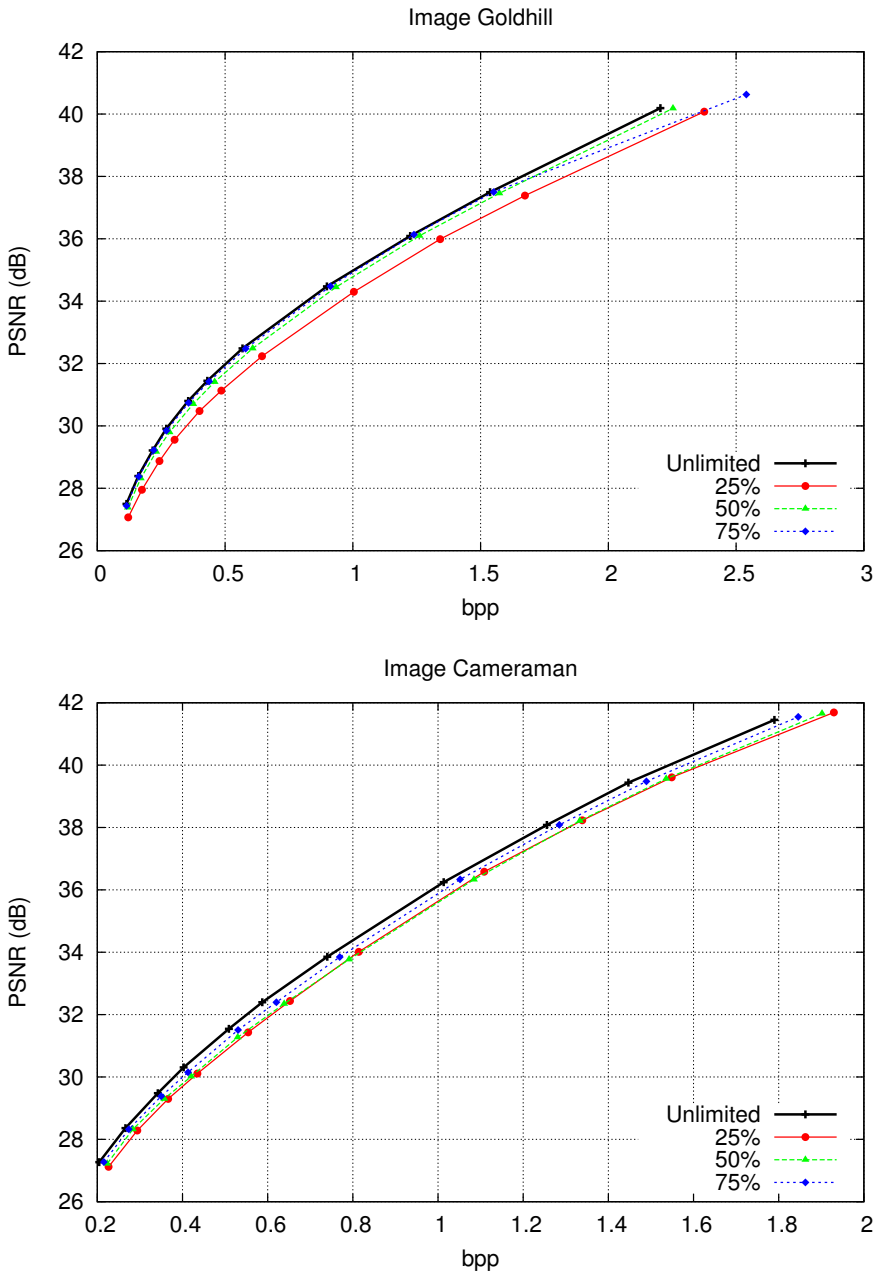
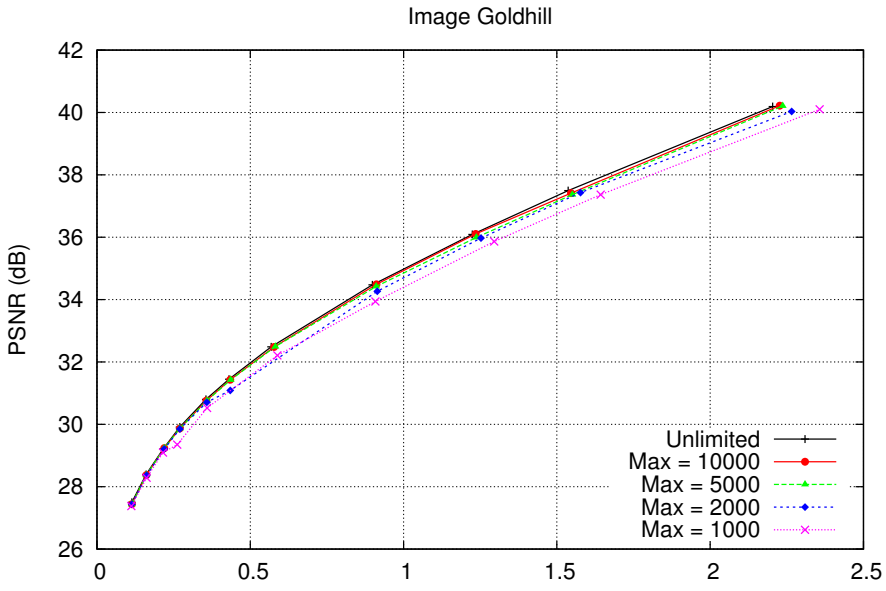
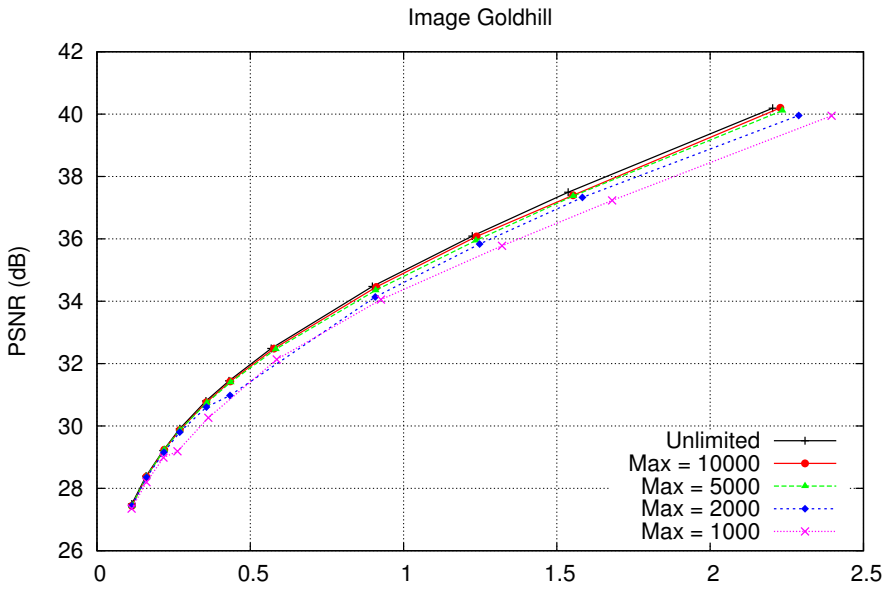


Figure C.4: Results for MMP when dictionary updating is interrupted at a percentage of the total number of blocks, for images Goldhill and Cameraman.

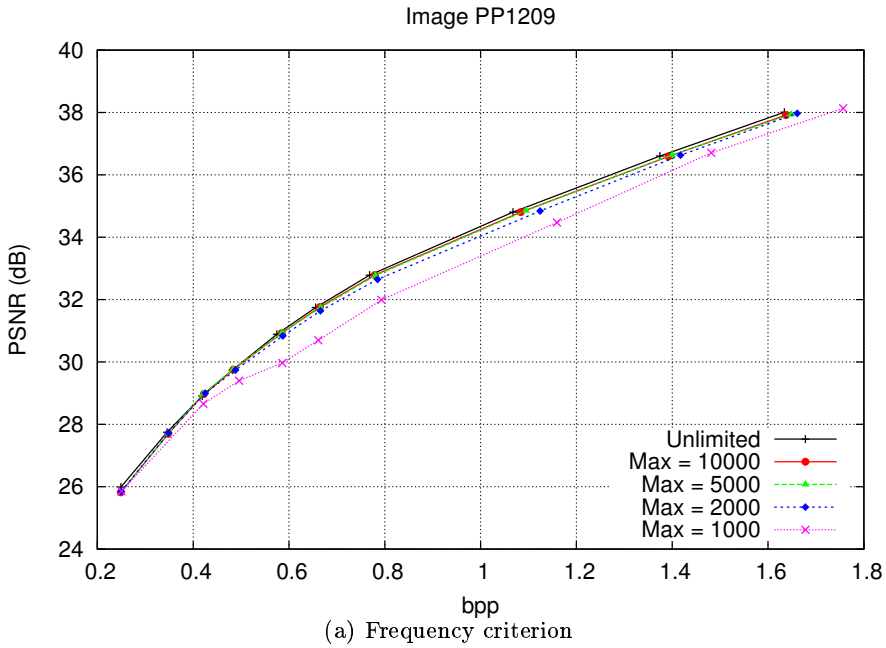


(a) Frequency criterion

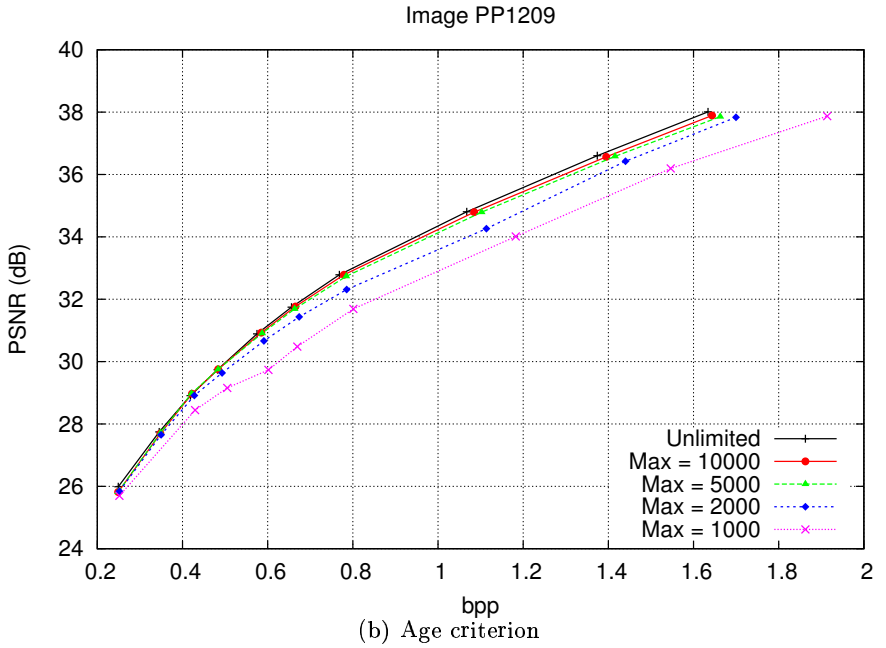


(b) Age criterion

Figure C.5: Results for MMP using a maximum dictionary size and the two tested elimination criteria, for image Goldhill.



(a) Frequency criterion



(b) Age criterion

Figure C.6: Results for MMP using a maximum dictionary size and the two tested elimination criteria, for image PP1209.

C.2 Complementary results for Chapter 4

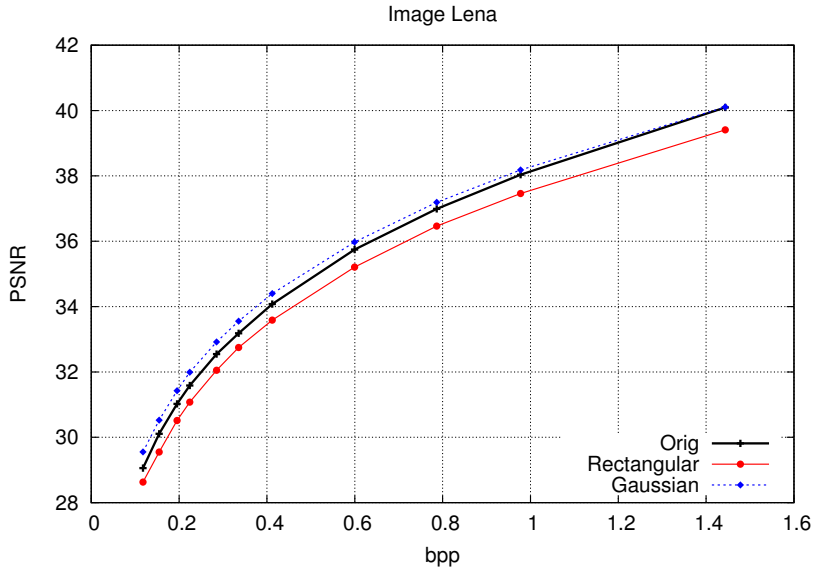


Figure C.7: Results of the deblocking post-processing for image Lena encoded with MMP.

Figure C.8: Image Lena compressed by MMP at 0.29bpp with no deblocking post-processing (PSNR = 32.55dB).

(a) Rectangular (32.65dB)

(b) Gaussian (32.87dB)

Figure C.9: Results of the MMP post-processing deblocking filters for image Lena coded at 0.29bpp for a) deblocking with rectangular kernel and b) deblocking with Gaussian kernel (see Figure C.8 for the original image with no deblocking).

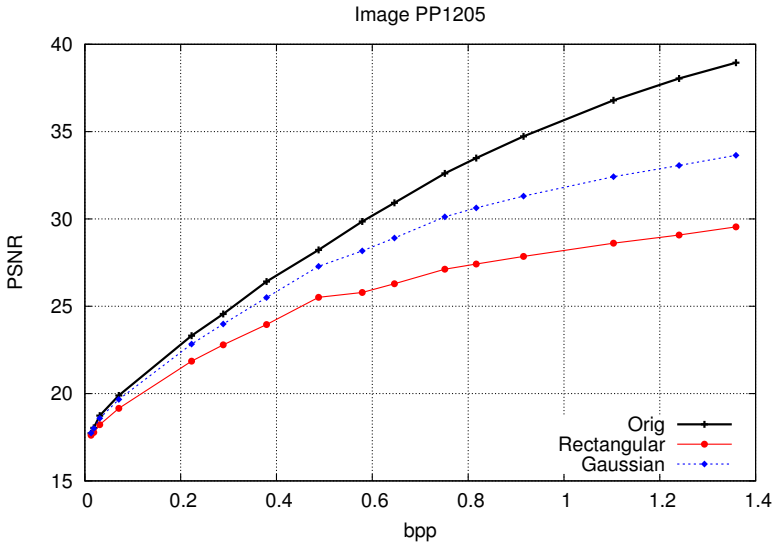
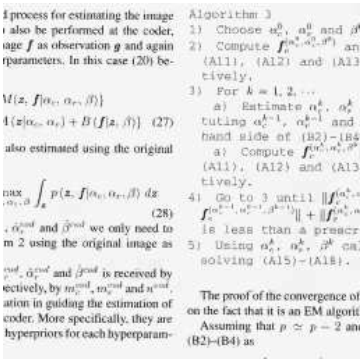
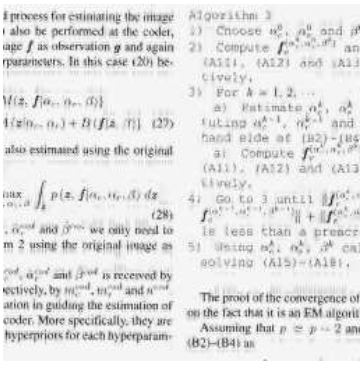


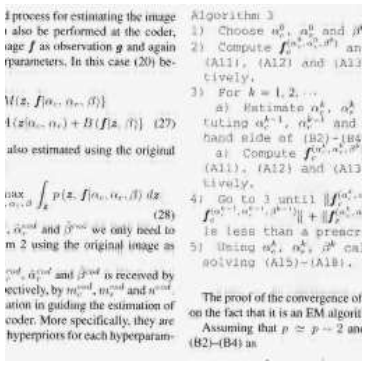
Figure C.10: Rate-distortion results for the post-processing deblocking filters for text image PP1205 encoded with MMP.



(a) Original (32.61dB)



(b) Rectangular (27.12dB)



(c) Gaussian (30.12dB)

Figure C.11: Results of the MMP post-processing deblocking filters for a detail of image PP1205 coded at 0.75bpp.

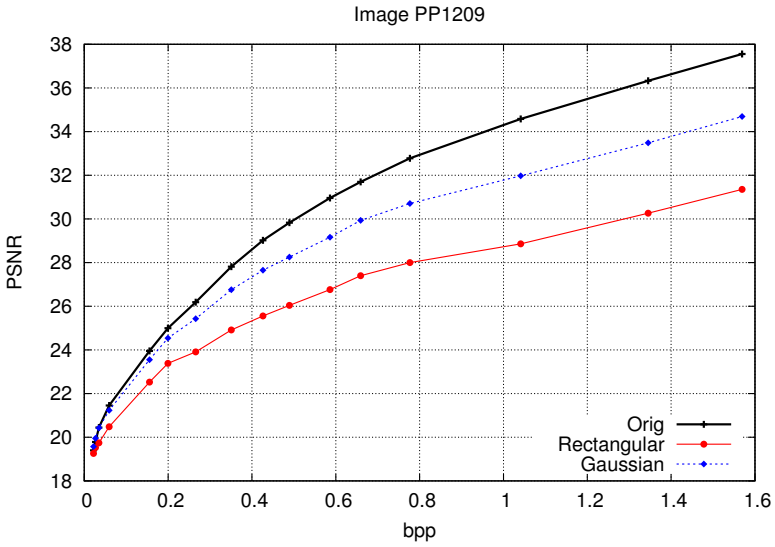
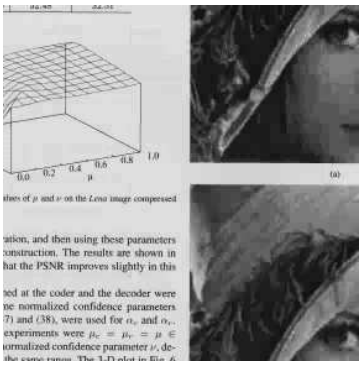
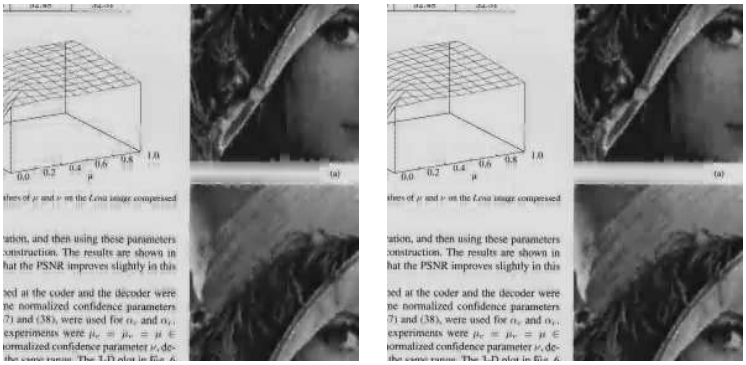


Figure C.12: Rate-distortion results for the post-processing deblocking filters for compound image PP1209 encoded with MMP.



(a) Original (32.78dB)



(b) Rectangular (28.0dB) (c) Gaussian (30.71dB)

Figure C.13: Results of the MMP post-processing deblocking filters for a detail of image PP1209 coded at 0.78bpp.

(a) Independent 64×64

(b) Independent 32×32

Figure C.14: Result of the thin-plate spline interpolation using *independent* optimisation of image blocks with dimensions a) 64×64 and b) 32×32.

(a) Overlapping 64×64

(b) Overlapping 32×32

Figure C.15: Result of the thin-plate spline interpolation using optimisation of *overlapping* image blocks with dimensions a) 64×64 and b) 32×32.

(a) MMP (28.787 dB)

(b) TPS (27.974 dB)

Figure C.16: Image Lena compressed at 0.12 bpp: a) Thin-plate spline interpolation using 16×16 overlapping blocks; b) original MMP synthesis.

(a) MMP (33.855 dB)

(b) TPS (31.496 dB)

Figure C.17: Image Lena compressed at 0.45 bpp: a) Thin-plate spline interpolation using 16×16 overlapping blocks; b) original MMP synthesis.

C.3 Complementary results for Chapter 5

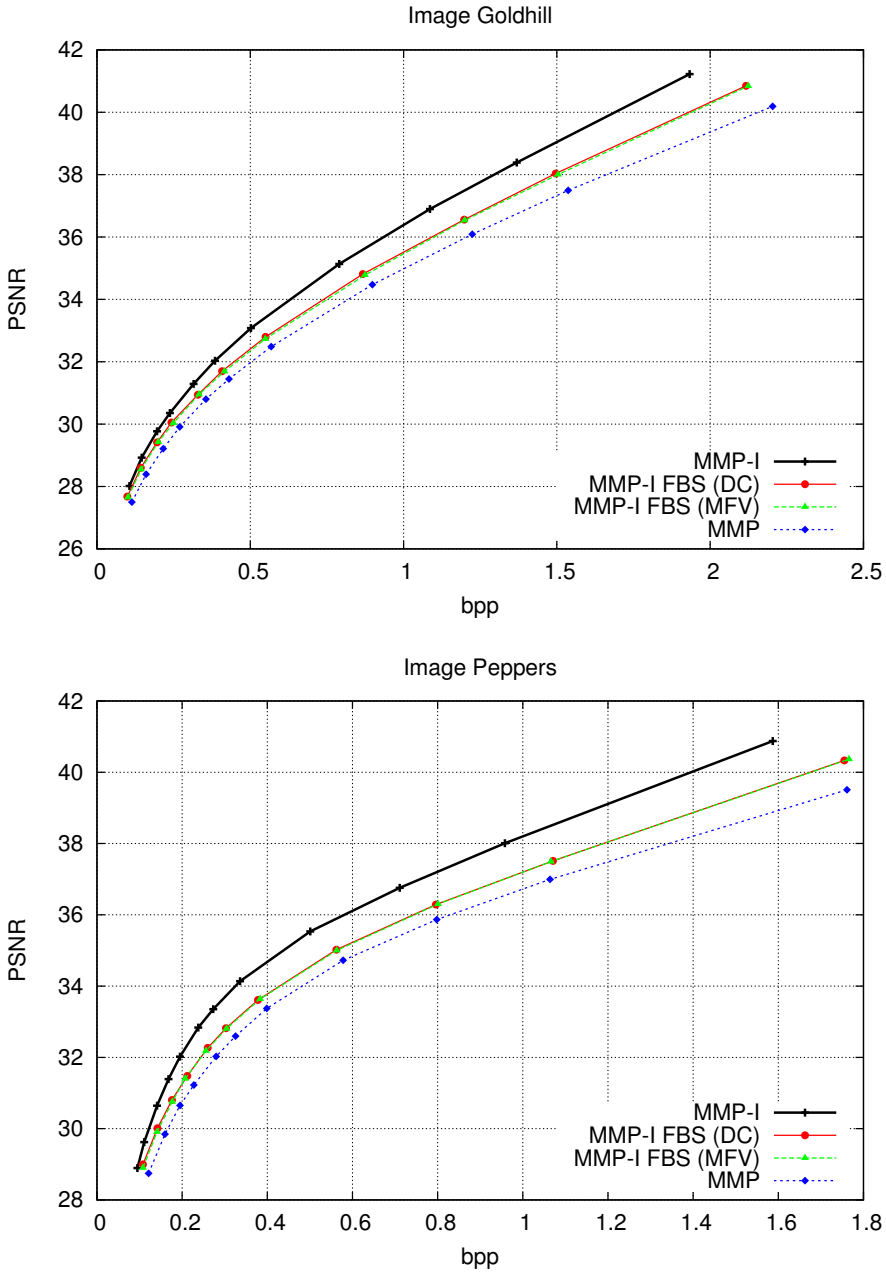


Figure C.18: Experimental results of MMP-I for grayscale test images Goldhill and Peppers.

C.4 Complementary results for Chapter 6

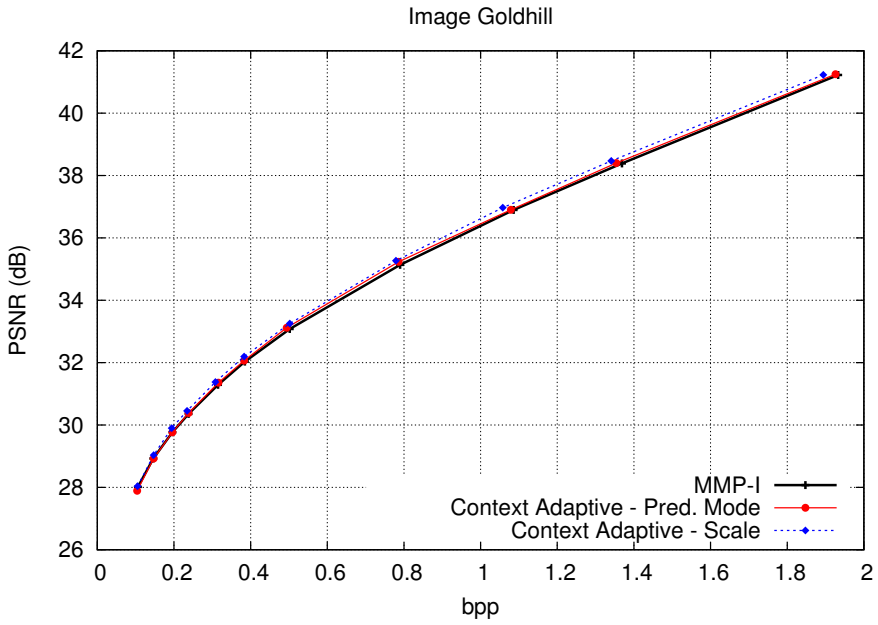


Figure C.19: Results for the use of context conditioning with MMP-I, for image Goldhill.

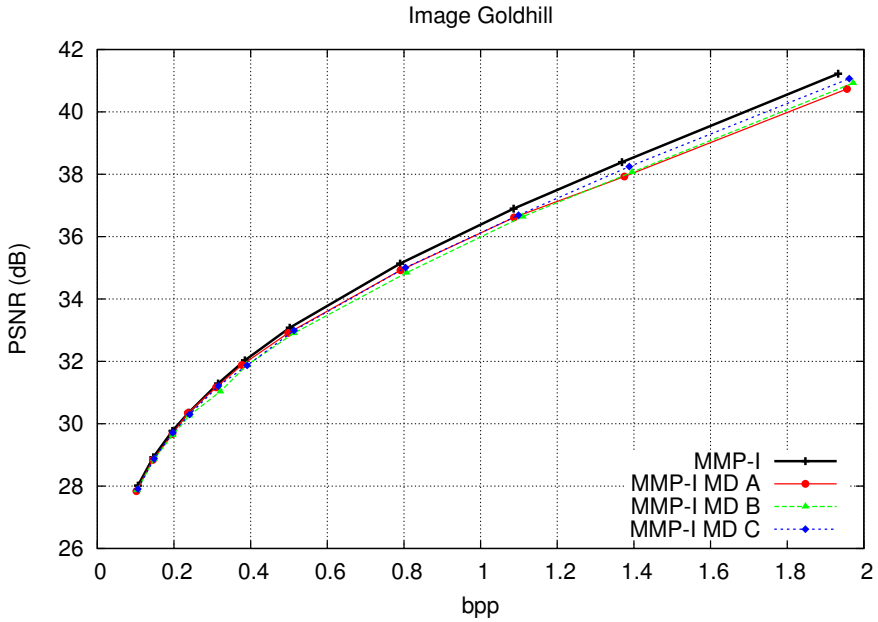


Figure C.20: Results for original MMP-I and new techniques that use several independent dictionaries (MMP-I MD), for image Goldhill.

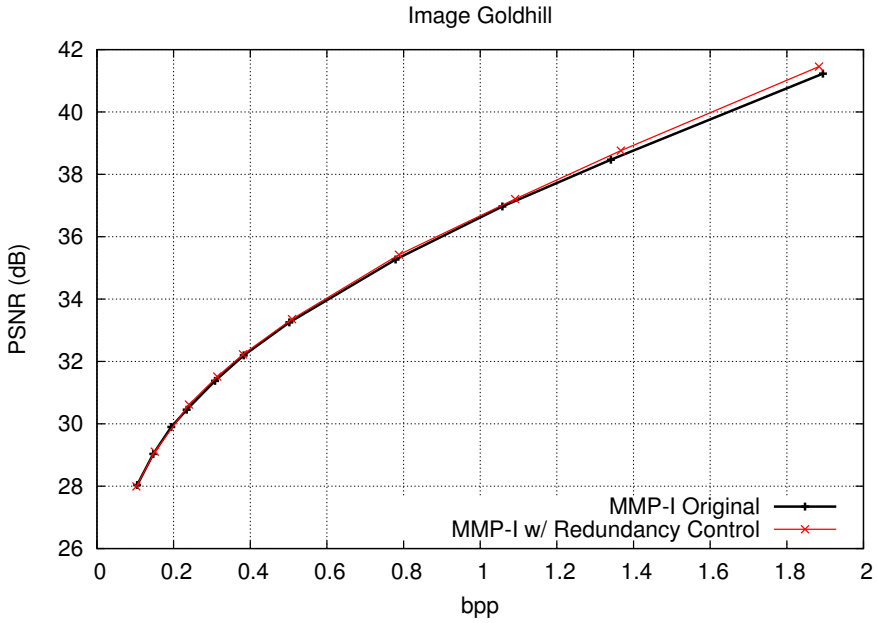


Figure C.21: Results for original MMP-I and MMP-I using redundancy control (both methods also use dictionary partitioning), for image Goldhill.

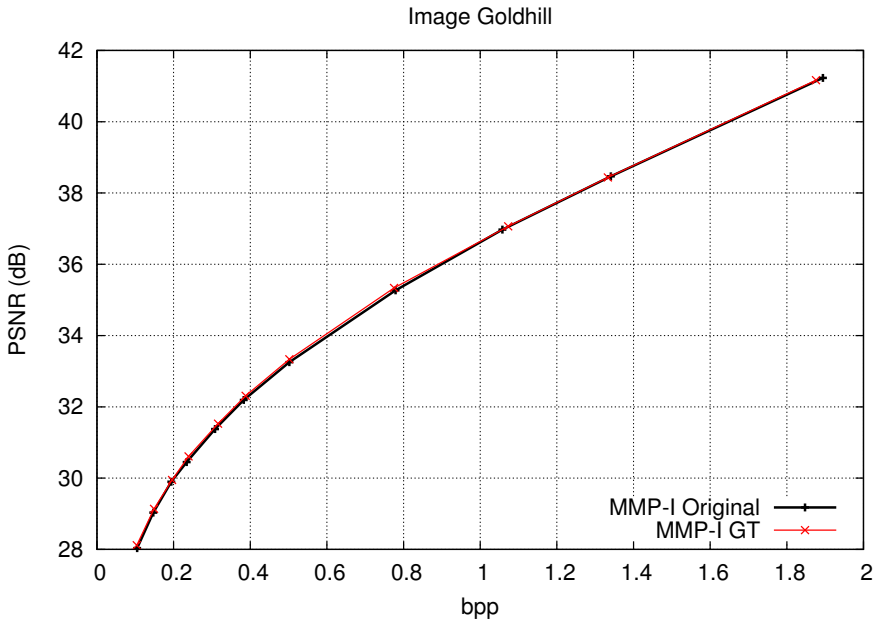


Figure C.22: Results for MMP-I with the use of new dictionary updating with geometric transforms, for image Goldhill.

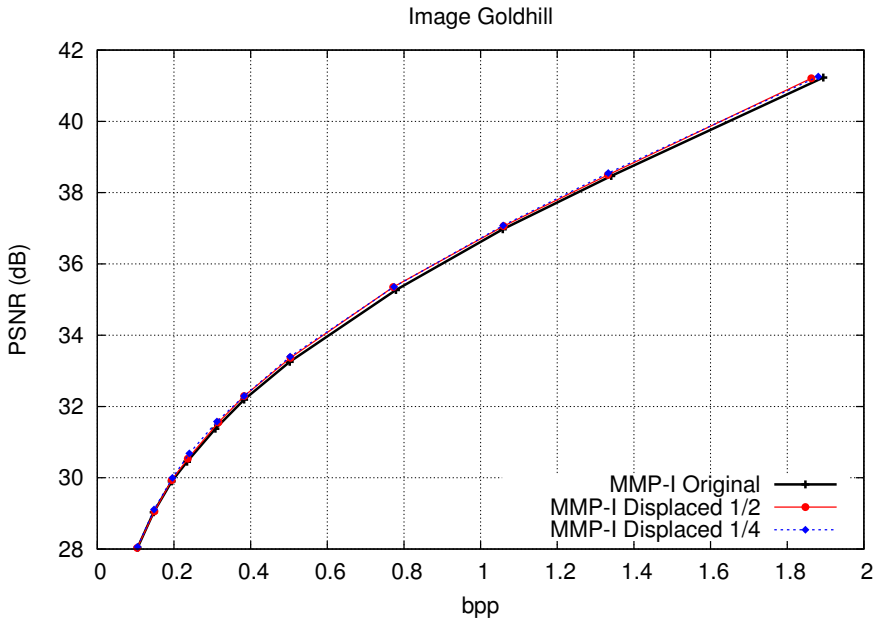


Figure C.23: Results for MMP-I with the use of new dictionary updating with displaced blocks, for image Goldhill.

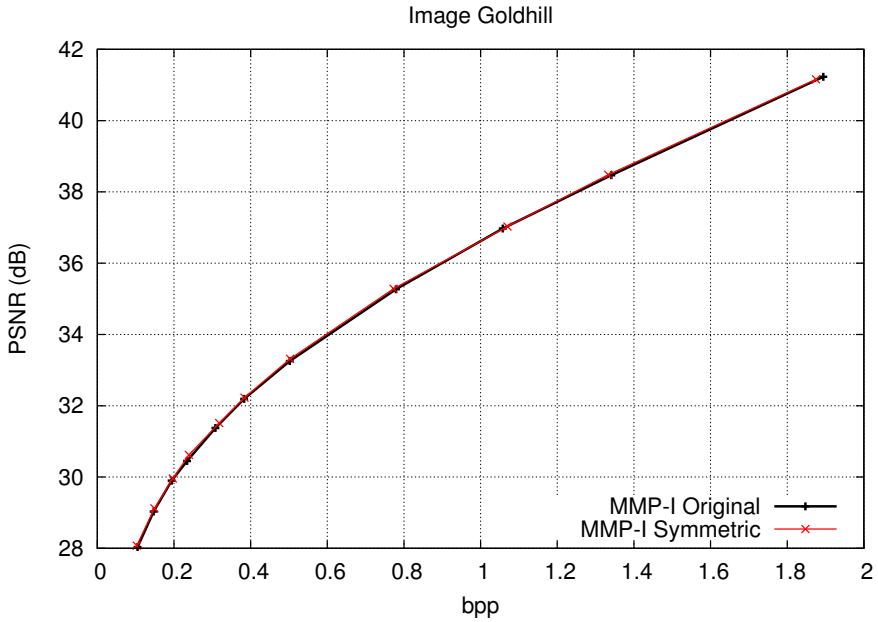


Figure C.24: Results for MMP-I with the use of new dictionary updating with the additive symmetric block, for image Goldhill.

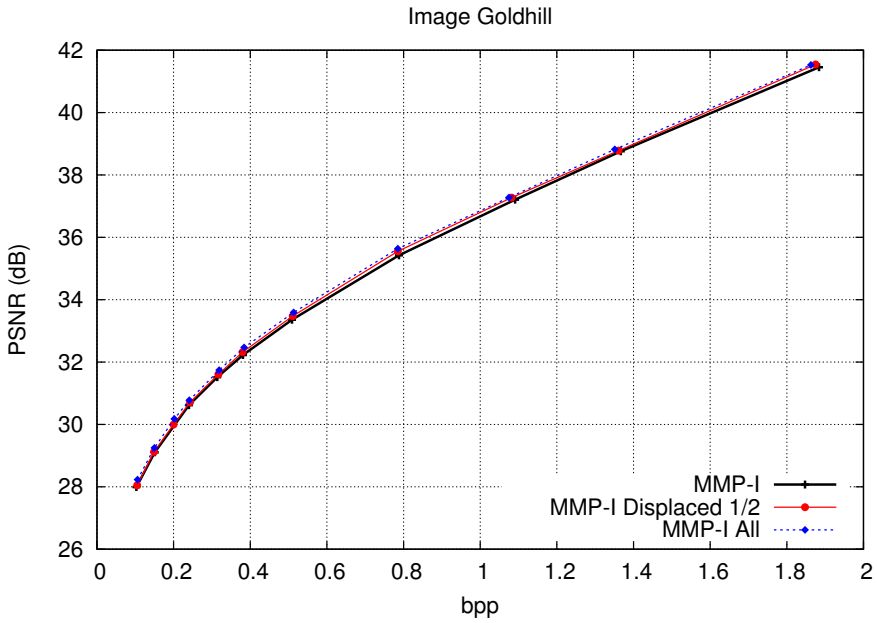


Figure C.25: Results for MMP-I using combinations of the new dictionary updating techniques, for image Goldhill.

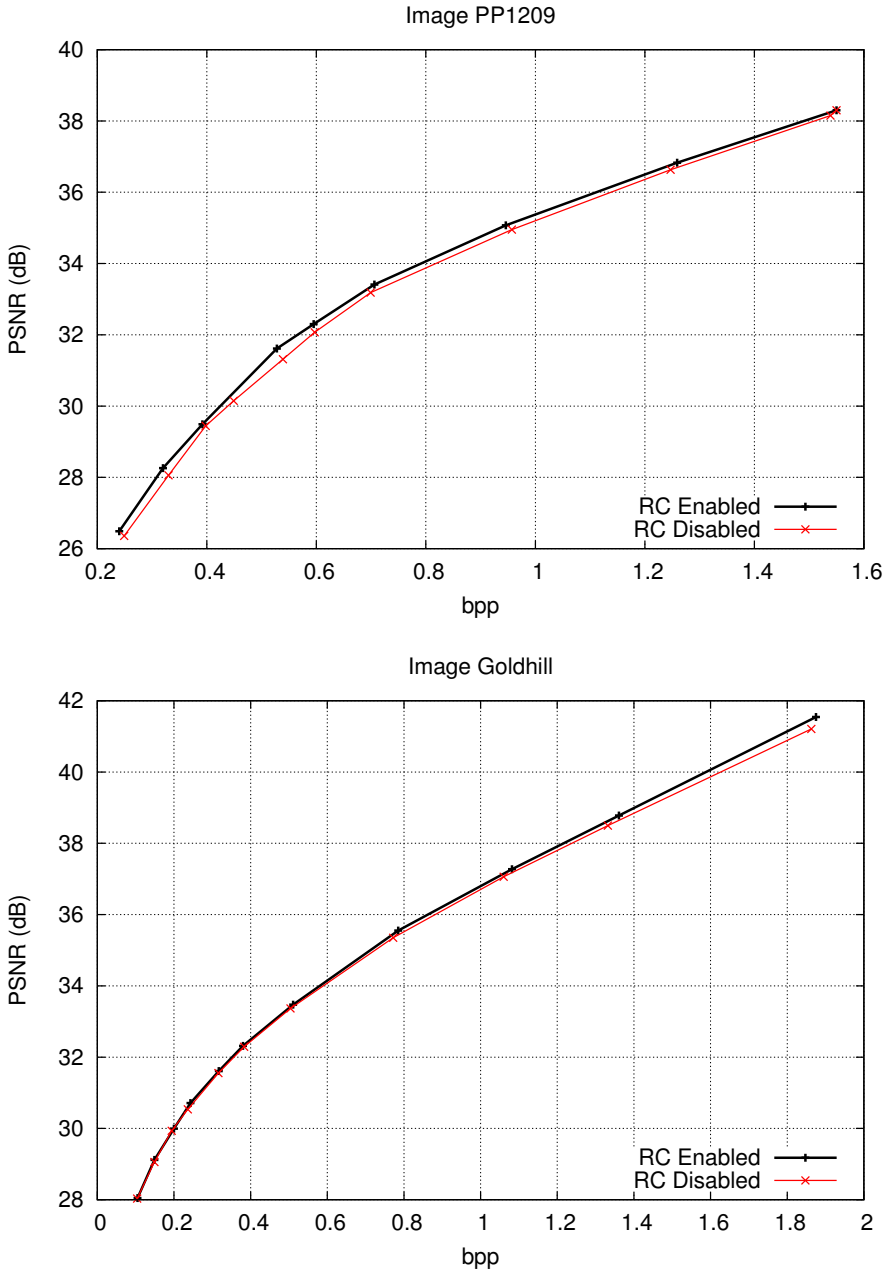


Figure C.26: Effects of using redundancy control with new dictionary updating techniques (using displaced blocks with $s = 2$), for images PP1209 and Goldhill.

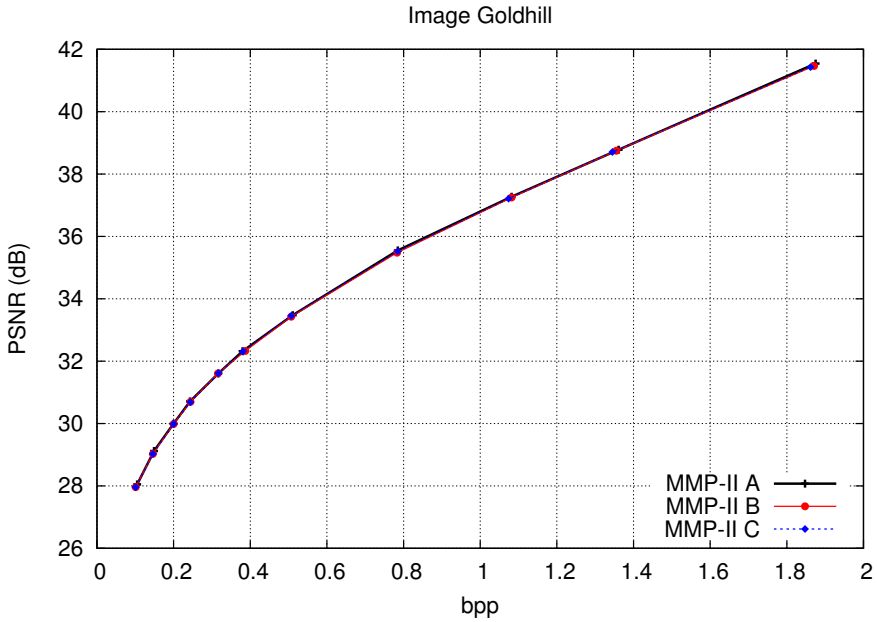


Figure C.27: Results for MMP-II when scale restriction in dictionary adaptation (MMP-II B) and scale restriction plus adaptive block size (MMP-II C) are used, for image Goldhill.

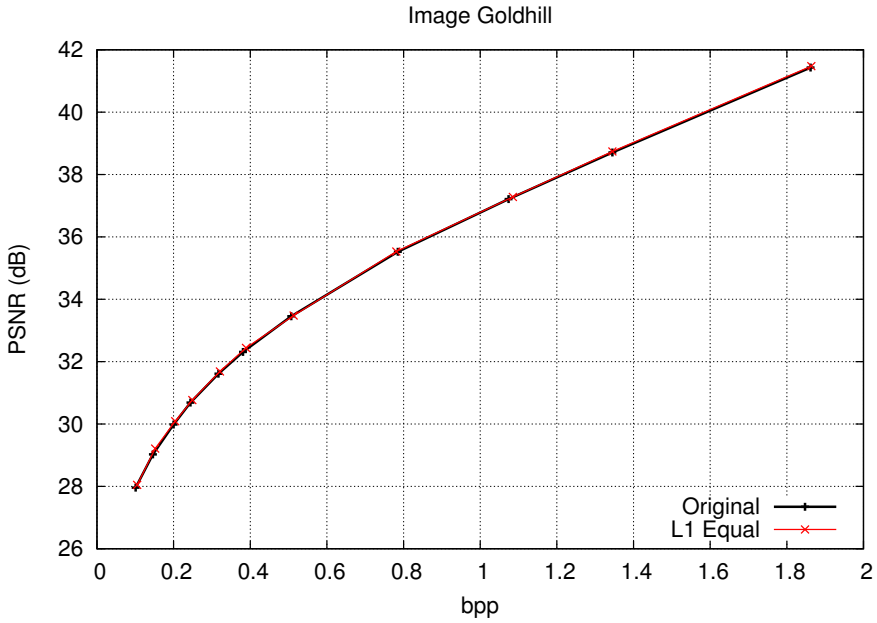


Figure C.28: Results when L^1 norm equalisation is used with MMP-II A, for image Goldhill.

C.5 Perceptual results for the MMP-based methods

(a) MMP (34.82dB)

Figure C.29: Image Lena compressed at approx. 0.5bpp using MMP.

(a) MMP-I (36.57dB)

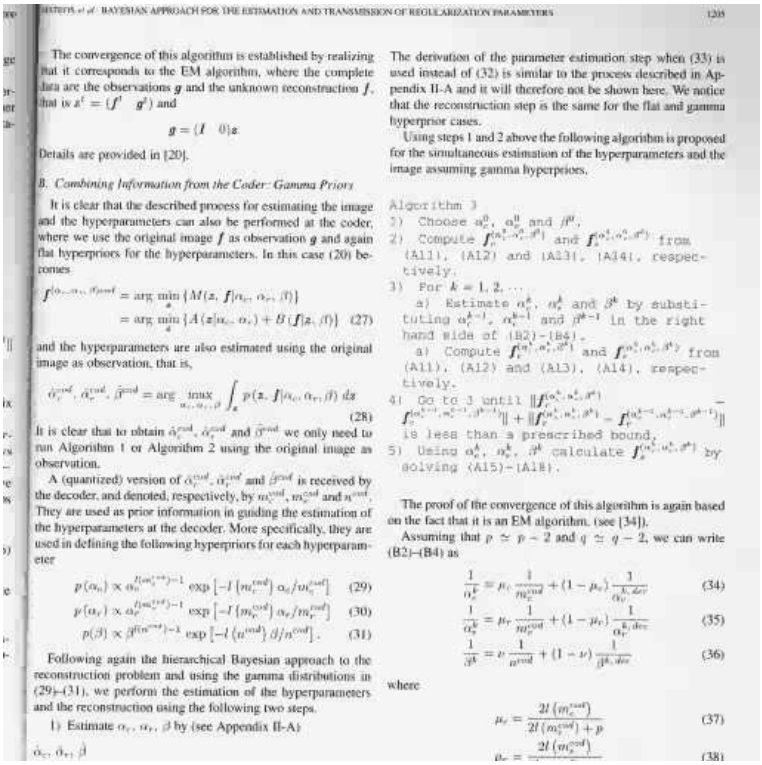
(b) MMP-II (36.94dB)

Figure C.30: Image Lena compressed at approximately 0.5bpp using MMP-I and MMP-II.

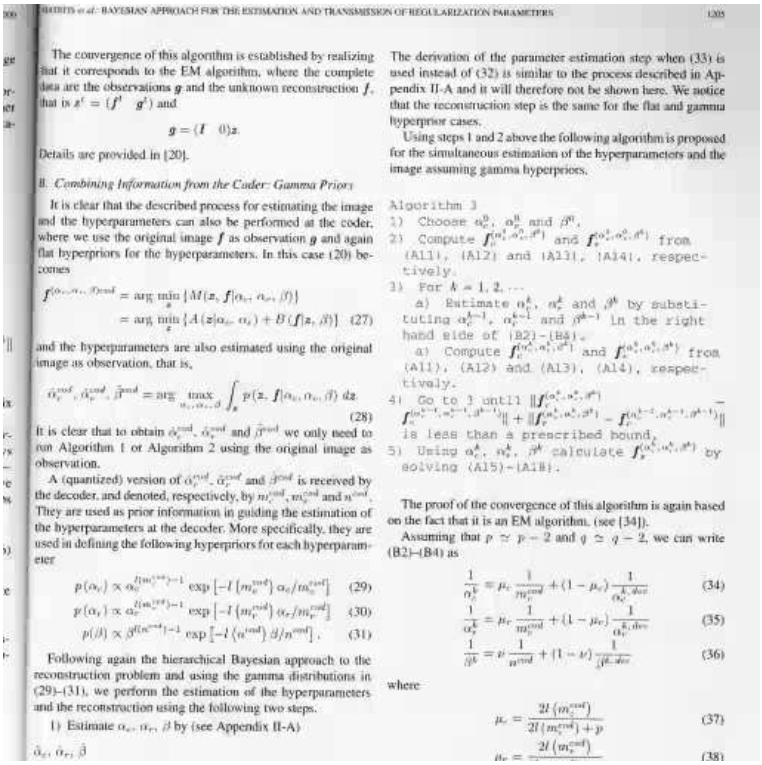
(a) JPEG2000 (37.15dB)

(b) H.264/AVC (37.24dB)

Figure C.31: Image Lena compressed at approximately 0.5bpp using JPEG2000 and H.264/AVC.

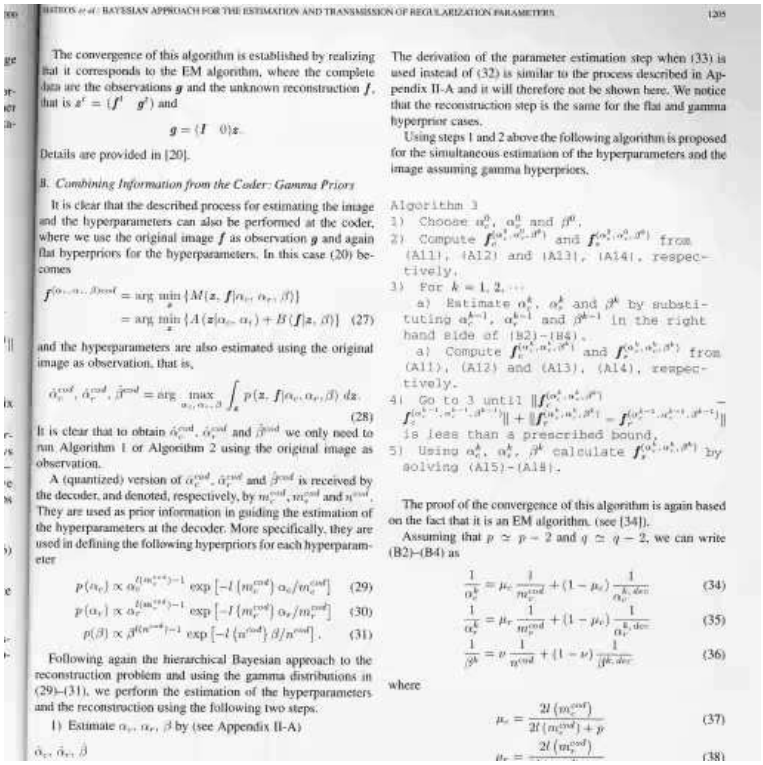


(a) MMP (30.82dB)



(b) MMP-I (30.89dB)

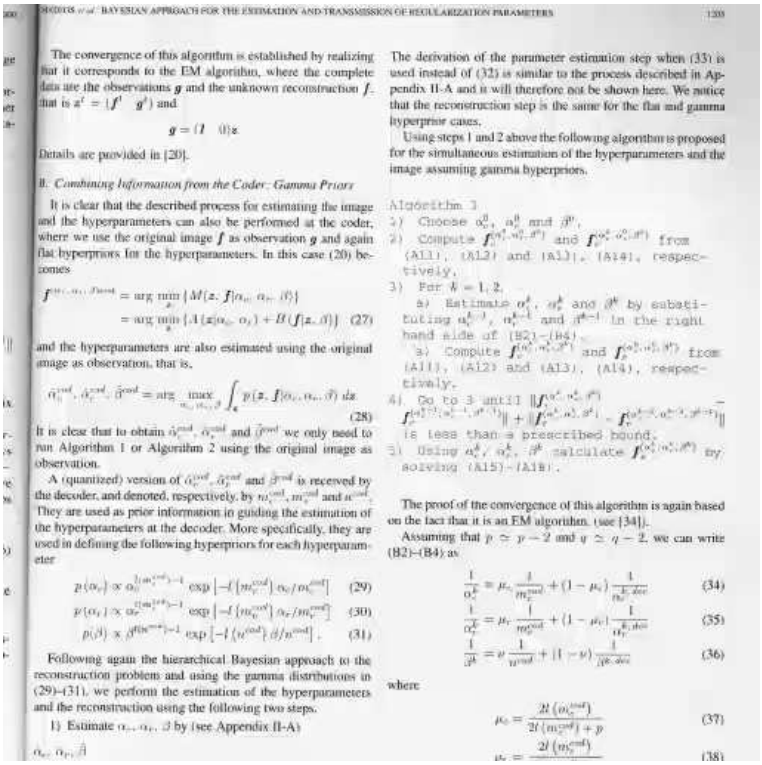
Figure C.32: Image PP1205 compressed at approximately 0.65bpp using a) MMP and b) MMP-I.



(a) MMP-II (32.74dB)

(b) JPEG2000 (27.21dB)

Figure C.33: Image PP1205 compressed at approximately 0.65bpp using a) JPEG2000 and b) H.264/AVC.



The convergence of this algorithm is established by realizing that it corresponds to the EM algorithm, where the complete data are the observations $\boldsymbol{g}$ and the unknown reconstruction $\boldsymbol{f}$, that is $\boldsymbol{z}^t = (\boldsymbol{f}^t \quad \boldsymbol{g}^t)$ and

$$\boldsymbol{g} = (\boldsymbol{I} \quad \boldsymbol{0})\boldsymbol{z}.$$

Details are provided in [20].

B. Combining Information from the Coder: Gamma Priors

It is clear that the described process for estimating the image and the hyperparameters can also be performed at the coder, where we use the original image $\boldsymbol{f}$ as observation $\boldsymbol{g}$ and again flat hyperpriors for the hyperparameters. In this case (20) becomes

$$\boldsymbol{f}^{(\alpha_\epsilon, \alpha_r, \beta)_{\text{coder}}} = \arg\min_{\boldsymbol{f}} \{M(\boldsymbol{z}, \boldsymbol{f}|\alpha_\epsilon, \alpha_r, \beta)\} = \arg\min_{\boldsymbol{f}} \{A(\boldsymbol{z}|\alpha_\epsilon, \alpha_r) + B(\boldsymbol{f}|\boldsymbol{z}, \beta)\} \quad (27)$$

and the hyperparameters are also estimated using the original image as observation, that is,

$$\hat{\alpha}_\epsilon^{\text{coder}}, \hat{\alpha}_r^{\text{coder}}, \hat{\beta}^{\text{coder}} = \arg\max_{\alpha_\epsilon, \alpha_r, \beta} \int_{\boldsymbol{z}} p(\boldsymbol{z}, \boldsymbol{f}|\alpha_\epsilon, \alpha_r, \beta)\, d\boldsymbol{z} \quad (28)$$

It is clear that to obtain $\hat{\alpha}_\epsilon^{\text{coder}}$, $\hat{\alpha}_r^{\text{coder}}$ and $\hat{\beta}^{\text{coder}}$ we only need to run Algorithm 1 or Algorithm 2 using the original image as observation.

A (quantized) version of $\hat{\alpha}_\epsilon^{\text{coder}}$, $\hat{\alpha}_r^{\text{coder}}$ and $\hat{\beta}^{\text{coder}}$ is received by the decoder, and denoted, respectively, by $m_\epsilon^{\text{coder}}, m_r^{\text{coder}}$ and n^{coder}. They are used as prior information in guiding the estimation of the hyperparameters at the decoder. More specifically, they are used in defining the following hyperpriors for each hyperparameter

$$p(\alpha_\epsilon) \propto \alpha_\epsilon^{l(m_\epsilon^{\text{coder}})-1} \exp\left[-l\left(m_\epsilon^{\text{coder}}\right)\alpha_\epsilon / m_\epsilon^{\text{coder}}\right] \quad (29)$$

$$p(\alpha_r) \propto \alpha_r^{l(m_r^{\text{coder}})-1} \exp\left[-l\left(m_r^{\text{coder}}\right)\alpha_r / m_r^{\text{coder}}\right] \quad (30)$$

$$p(\beta) \propto \beta^{l(n^{\text{coder}})-1} \exp\left[-l\left(n^{\text{coder}}\right)\beta / n^{\text{coder}}\right]. \quad (31)$$

Following again the hierarchical Bayesian approach to the reconstruction problem and using the gamma distributions in (29)–(31), we perform the estimation of the hyperparameters and the reconstruction using the following two steps.

1) Estimate $\alpha_\epsilon, \alpha_r, \beta$ by (see Appendix II-A)

$\hat{\alpha}_\epsilon, \hat{\alpha}_r, \hat{\beta}$

The derivation of the parameter estimation step when (33) is used instead of (32) is similar to the process described in Appendix II-A and it will therefore not be shown here. We notice that the reconstruction step is the same for the flat and gamma hyperprior cases.

Using steps 1 and 2 above the following algorithm is proposed for the simultaneous estimation of the hyperparameters and the image assuming gamma hyperpriors.

Algorithm 3

1) Choose α_ϵ^0, α_r^0 and β^0.
2) Compute $\boldsymbol{f}_\epsilon^{(\alpha_\epsilon^0, \alpha_r^0, \beta^0)}$ and $\boldsymbol{f}_r^{(\alpha_\epsilon^0, \alpha_r^0, \beta^0)}$ from (A11), (A12) and (A13), (A14), respectively.
3) For $k = 1, 2, \ldots$
   a) Estimate α_ϵ^k, α_r^k and β^k by substituting α_ϵ^{k-1}, α_r^{k-1} and β^{k-1} in the right hand side of (B2)–(B4).
   a) Compute $\boldsymbol{f}_\epsilon^{(\alpha_\epsilon^k, \alpha_r^k, \beta^k)}$ and $\boldsymbol{f}_r^{(\alpha_\epsilon^k, \alpha_r^k, \beta^k)}$ from (A11), (A12) and (A13), (A14), respectively.
4) Go to 3 until $\|\boldsymbol{f}_\epsilon^{(\alpha_\epsilon^k, \alpha_r^k, \beta^k)} - \boldsymbol{f}_\epsilon^{(\alpha_\epsilon^{k-1}, \alpha_r^{k-1}, \beta^{k-1})}\| + \|\boldsymbol{f}_r^{(\alpha_\epsilon^k, \alpha_r^k, \beta^k)} - \boldsymbol{f}_r^{(\alpha_\epsilon^{k-1}, \alpha_r^{k-1}, \beta^{k-1})}\|$ is less than a prescribed bound.
5) Using α_ϵ^k, α_r^k, β^k calculate $\boldsymbol{f}_p^{(\alpha_\epsilon^k, \alpha_r^k, \beta^k)}$ by solving (A15)–(A18).

The proof of the convergence of this algorithm is again based on the fact that it is an EM algorithm (see [34]).

Assuming that $p \simeq p-2$ and $q \simeq q-2$, we can write (B2)–(B4) as

$$\frac{1}{\alpha_\epsilon^k} = \mu_\epsilon \frac{1}{m_\epsilon^{\text{coder}}} + (1-\mu_\epsilon)\frac{1}{\frac{1}{k}\sum_{\cdots}} \quad (34)$$

$$\frac{1}{\alpha_r^k} = \mu_r \frac{1}{m_r^{\text{coder}}} + (1-\mu_r)\frac{1}{\frac{1}{k}\sum_{\cdots}} \quad (35)$$

$$\frac{1}{\beta^k} = \nu \frac{1}{n^{\text{coder}}} + (1-\nu)\frac{1}{\frac{1}{pk}\sum_{\cdots}} \quad (36)$$

where

$$\mu_\epsilon = \frac{2l\left(m_\epsilon^{\text{coder}}\right)}{2l\left(m_\epsilon^{\text{coder}}\right) + p} \quad (37)$$

$$\mu_r = \frac{2l\left(m_r^{\text{coder}}\right)}{\quad} \quad (38)$$

(a) H.264/AVC (28.27dB)

Figure C.34: Image PP1205 compressed at approximately 0.65bpp using H.264/AVC.

C.6 Complementary results for Chapter 7

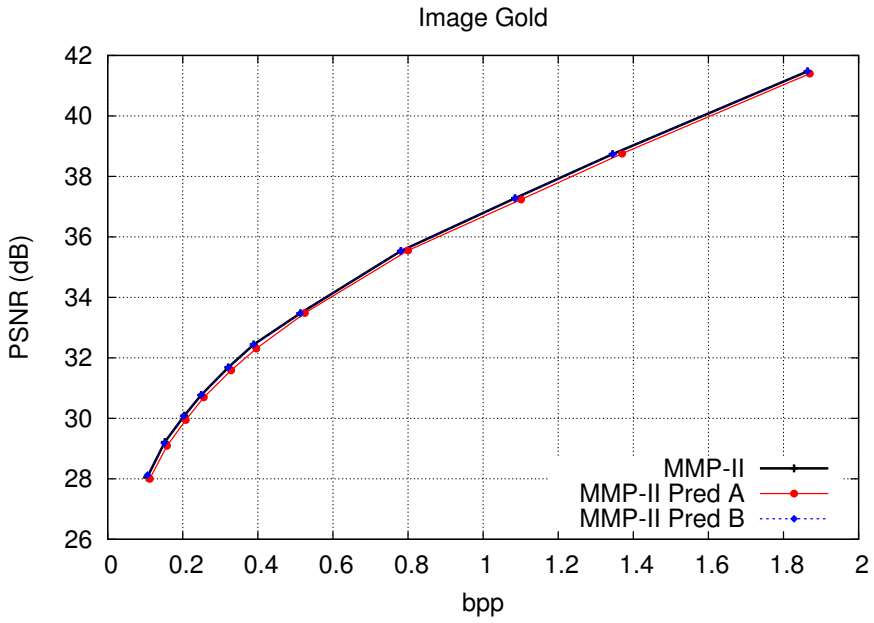


Figure C.35: Effects of changing the MMP-II prediction modes, for image Goldhill.

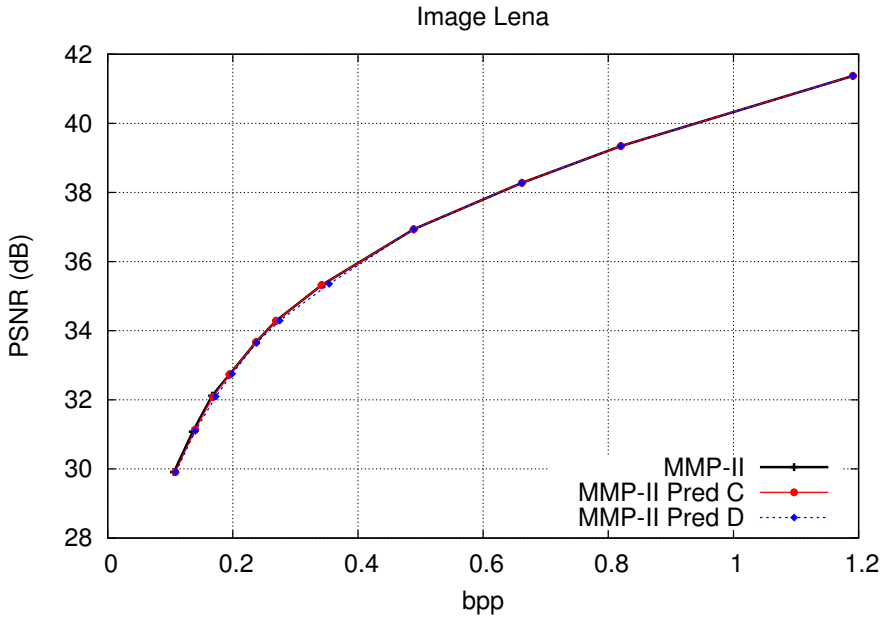


Figure C.36: Effects of using a different number of prediction modes for higher scales of MMP-II, for image Lena.

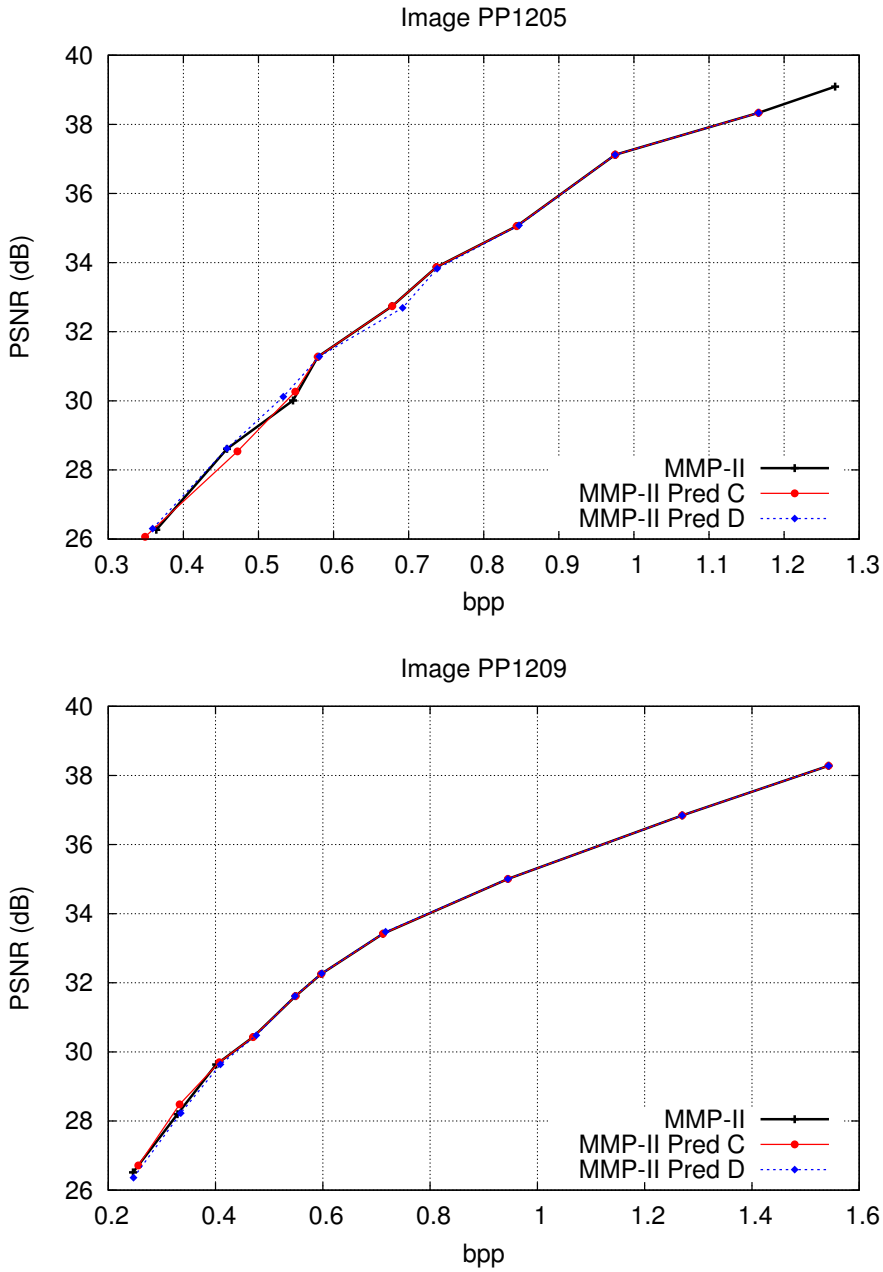


Figure C.37: Effects of using a different number of prediction modes for higher scales of MMP-II, for images PP1205 and PP1209.

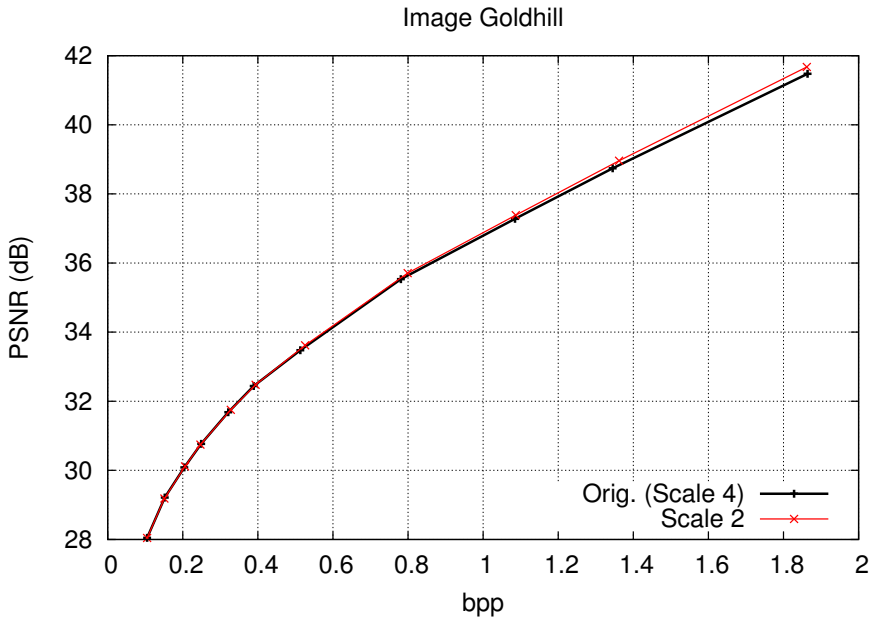


Figure C.38: Effects of changing the minimum block scale used in the prediction stage of MMP-II, for image Goldhill.

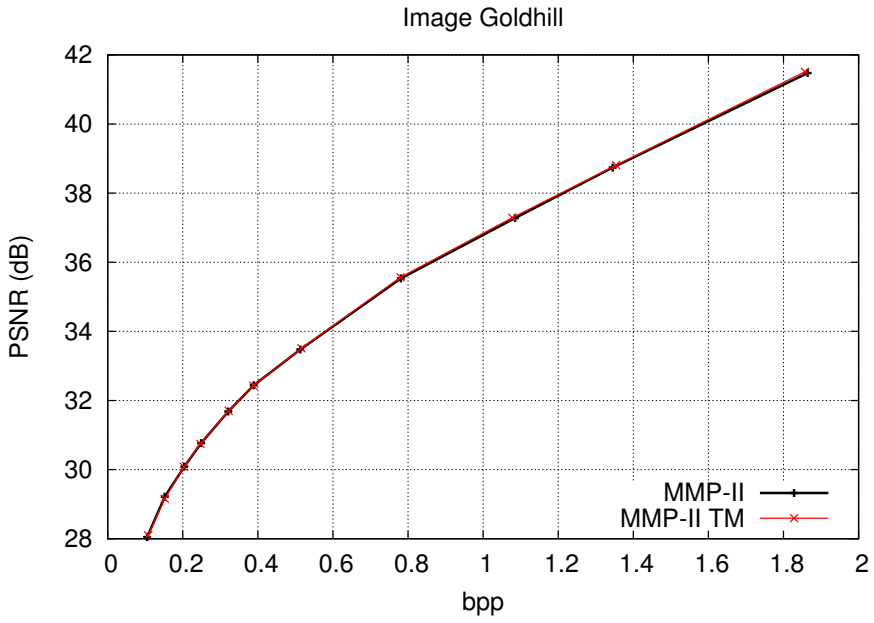


Figure C.39: Effects of using template matching in the prediction stage of MMP-II, for image Goldhill.

(a)

(b)

Figure C.40: Image Lena, encoded with MMP-II at 0.135 bpp: a) no deblocking (31.08 dB); b) MMP deblocking with rectangular kernel [1] (28.94 dB).

(a)

(b)

Figure C.41: Image Lena, encoded with MMP-II at 0.135 bpp: a) MMP deblocking with Gaussian kernel [2] (30.25 dB); b) MMP-II deblocking (31.35 dB).

(a) No deblocking (31.70 dB)

(b) MMP deblocking rectang. [1] (28.76 dB)

(c) MMP deblocking Gaussian [2] (30.26 dB)

(d) MMP-II deblocking ($\alpha = 0.10$ and $s = 32$) (31.91 dB)

Figure C.42: A detail of image Peppers, encoded with MMP-II at 0.16 bpp.

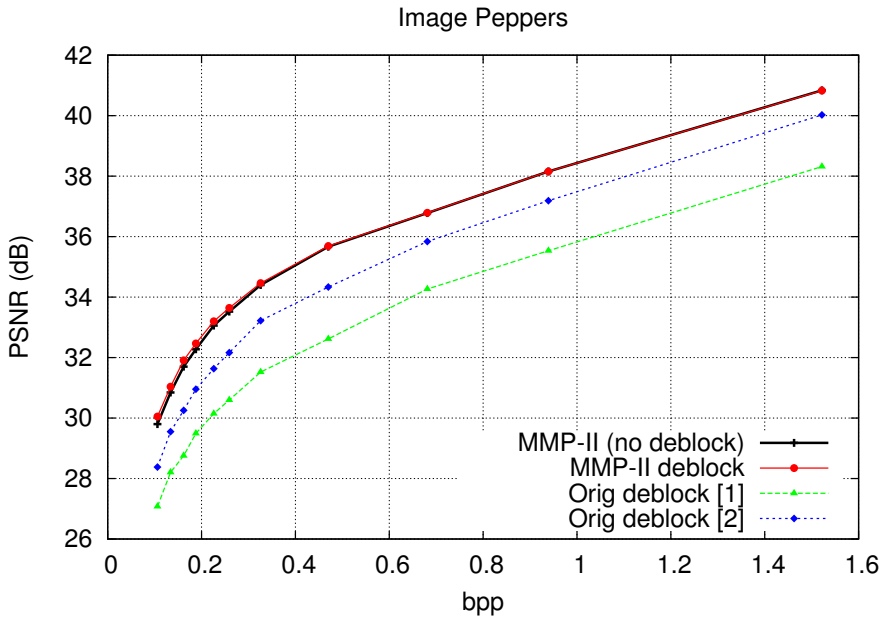


Figure C.43: Objective quality gains for image Peppers (adaptive deblocking filter used with $\alpha = 0.10$ and $s = 32$).

C.7 Complementary results for Chapter 8

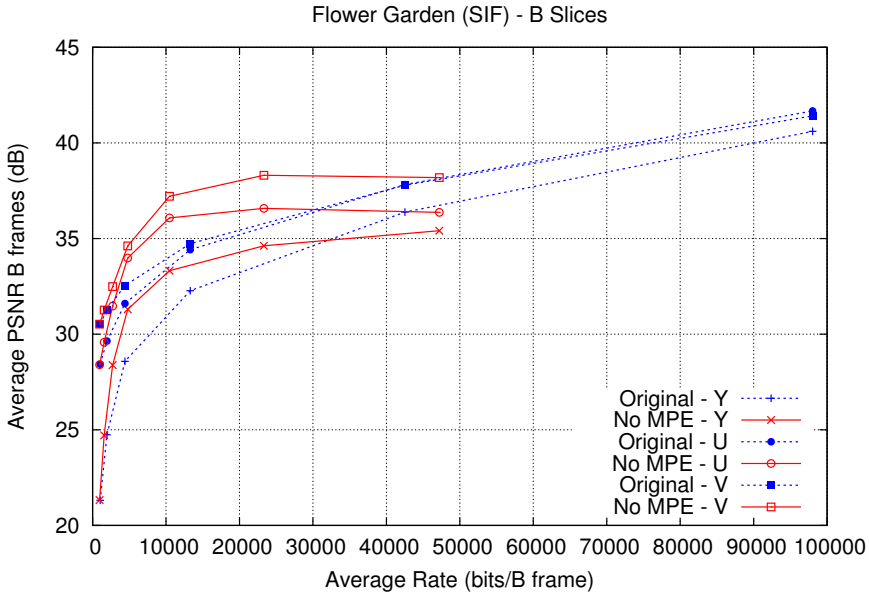


Figure C.44: Experimental results of the original H.264/AVC encoder and the new encoder, that disables the compression of motion-predicted error (for B slices only).

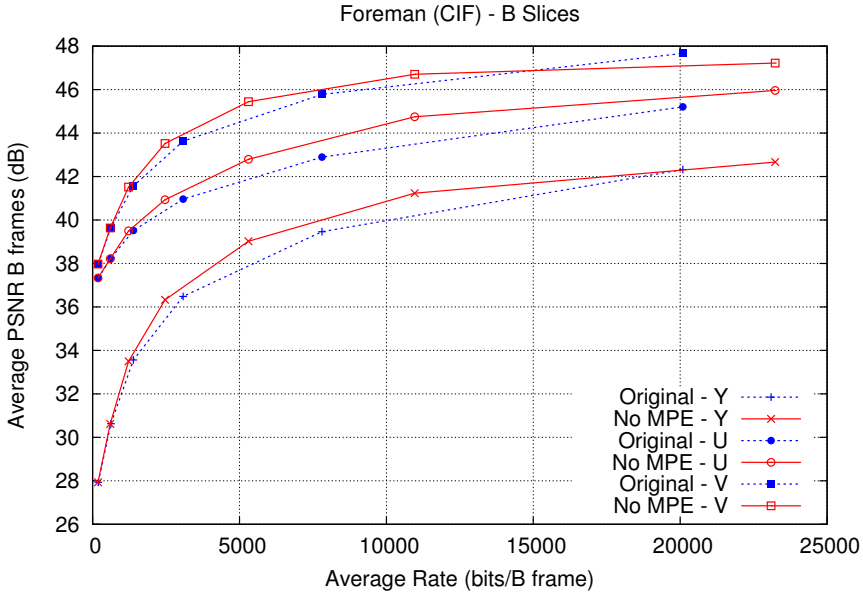


Figure C.45: Experimental results of original H.264/AVC and new encoder, that disables the compression of motion-predicted error (for B slices only), for the main profile.

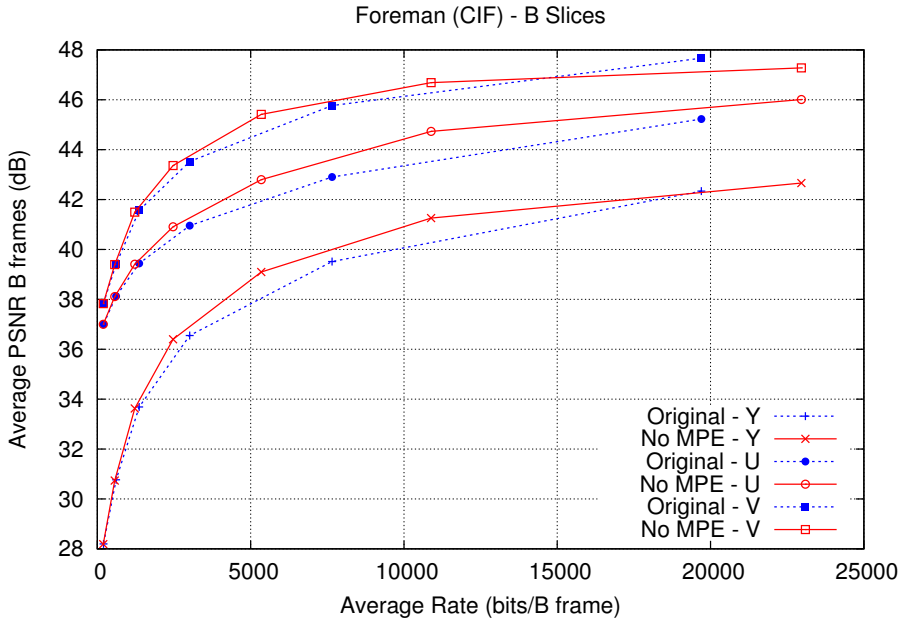


Figure C.46: Experimental results of original H.264/AVC and new encoder, that disables the compression of motion-predicted error (for B slices only), when no bi-predictive coding is used.

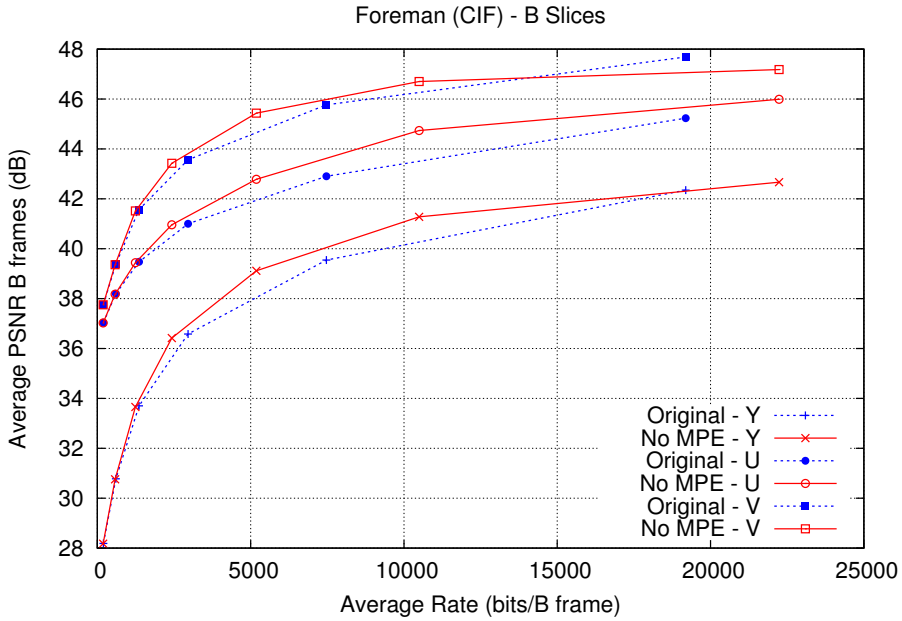


Figure C.47: Experimental results of original H.264/AVC and new encoder, that disables the compression of motion-predicted error (for B slices only), when a 32×32 search window is used.

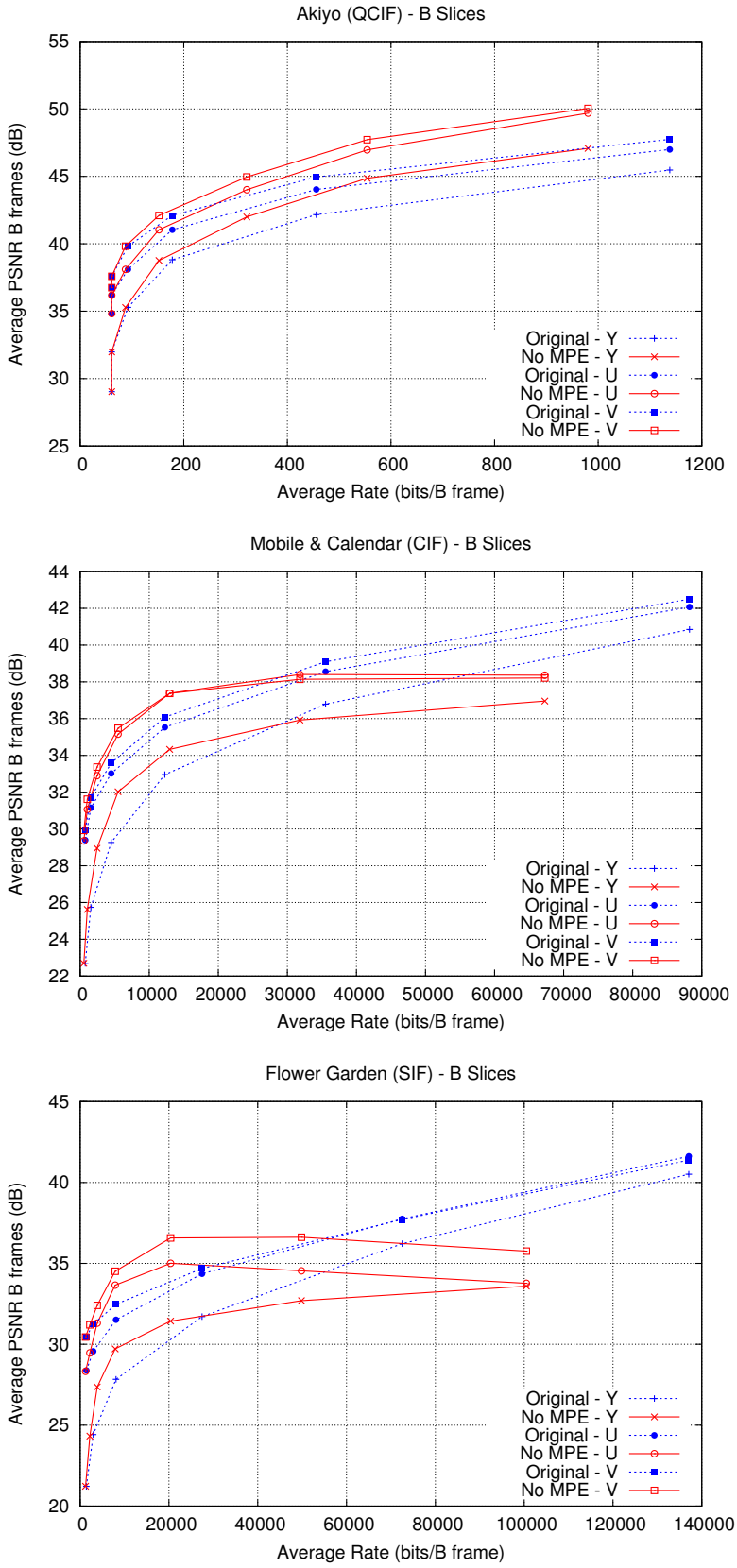


Figure C.48: Experimental results of original H.264/AVC and new encoder, with no compression of motion-predicted error (B slices only), for a *IPBBBP...* slice pattern.

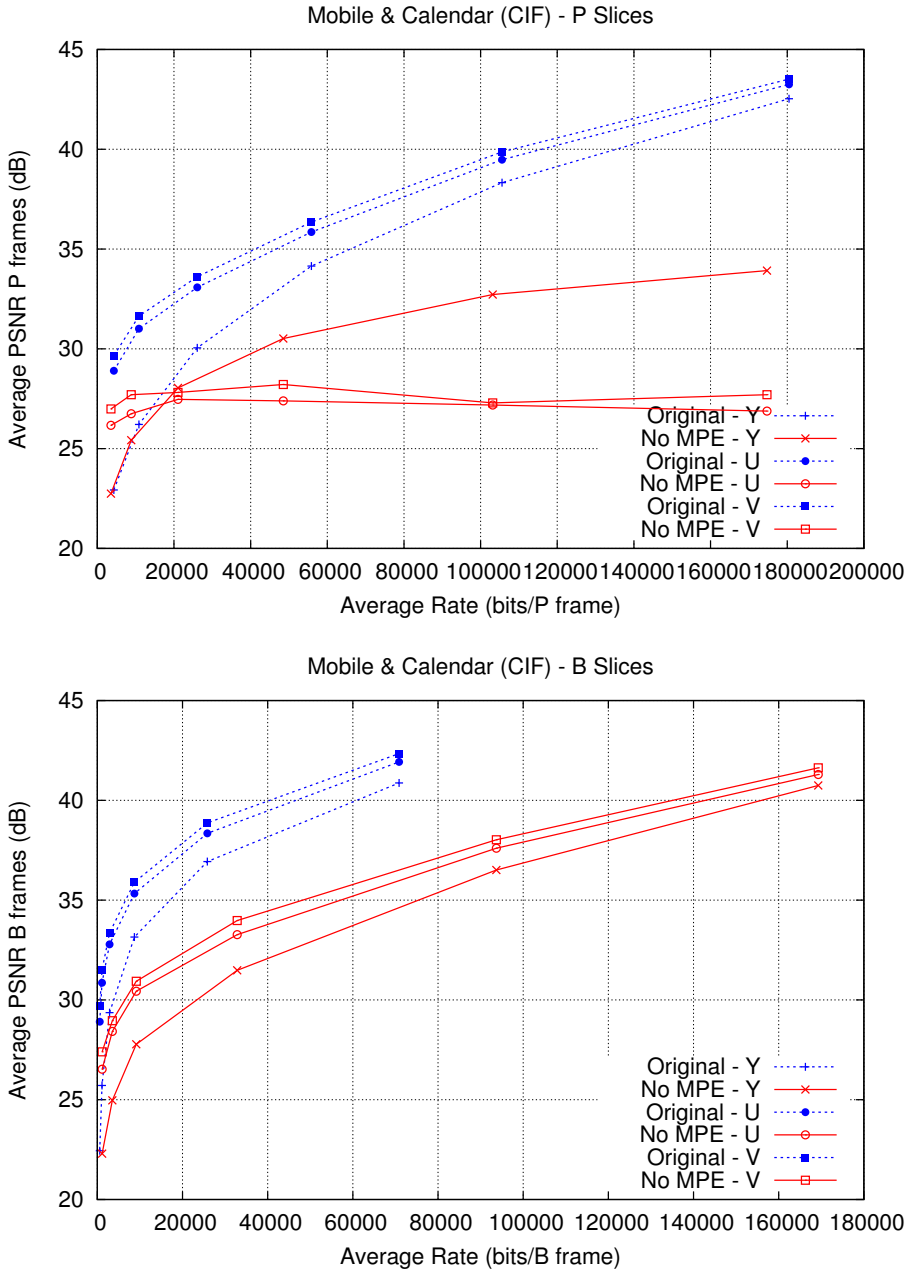


Figure C.49: Experimental results of original H.264/AVC and H.264/AVC No MPE, when the compression of motion-predicted error is disabled for P slices (results for P and B slices, of sequence Mobile & Calendar).

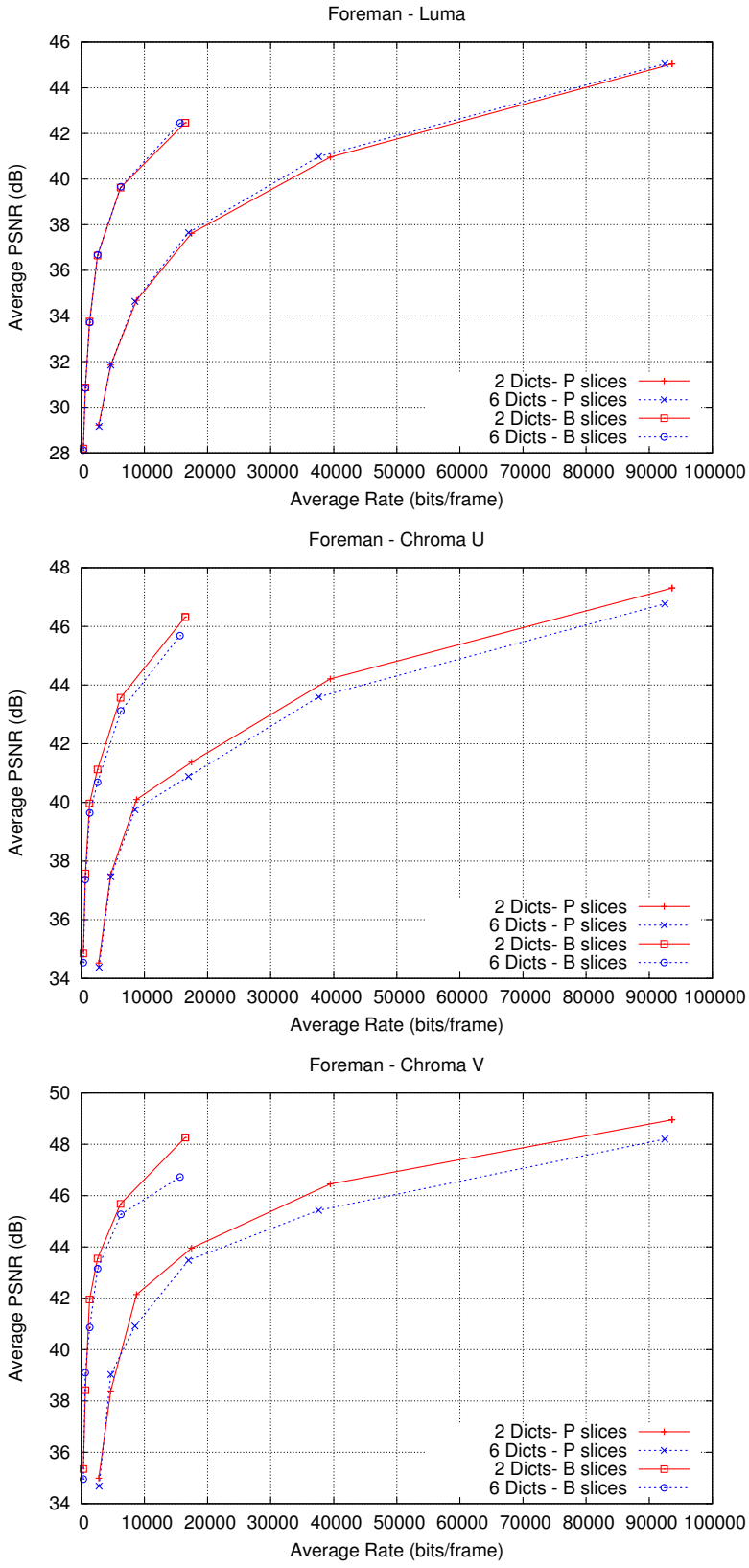


Figure C.50: Comparative results for MMP-Video encoder using two and six independent dictionaries, for sequence Foreman (CIF).

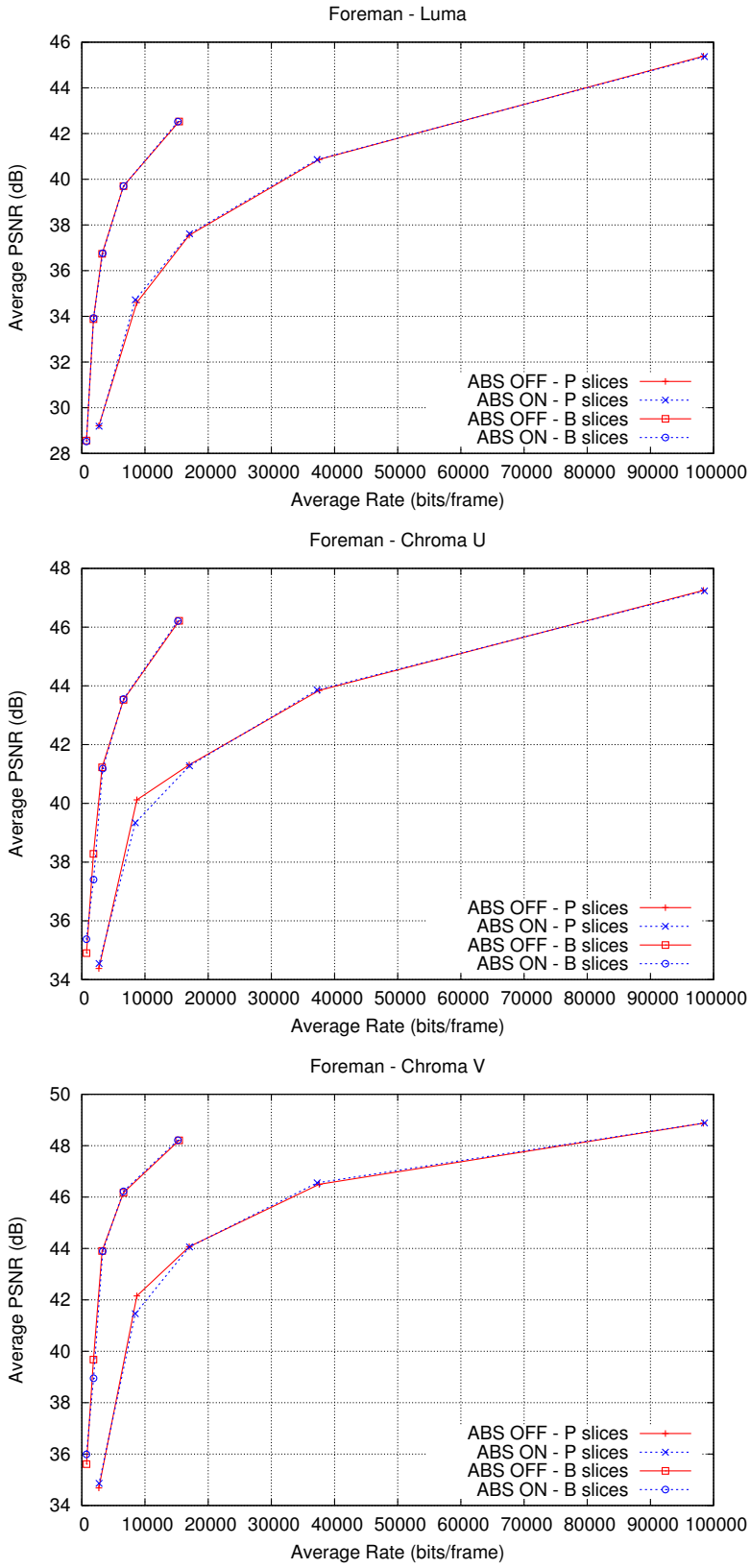


Figure C.51: Comparative results for MMP-Video encoder (2 dictionaries), for sequence Foreman (CIF). In the ABS test, the initial block size for MMP compression corresponds to the MC partitions.

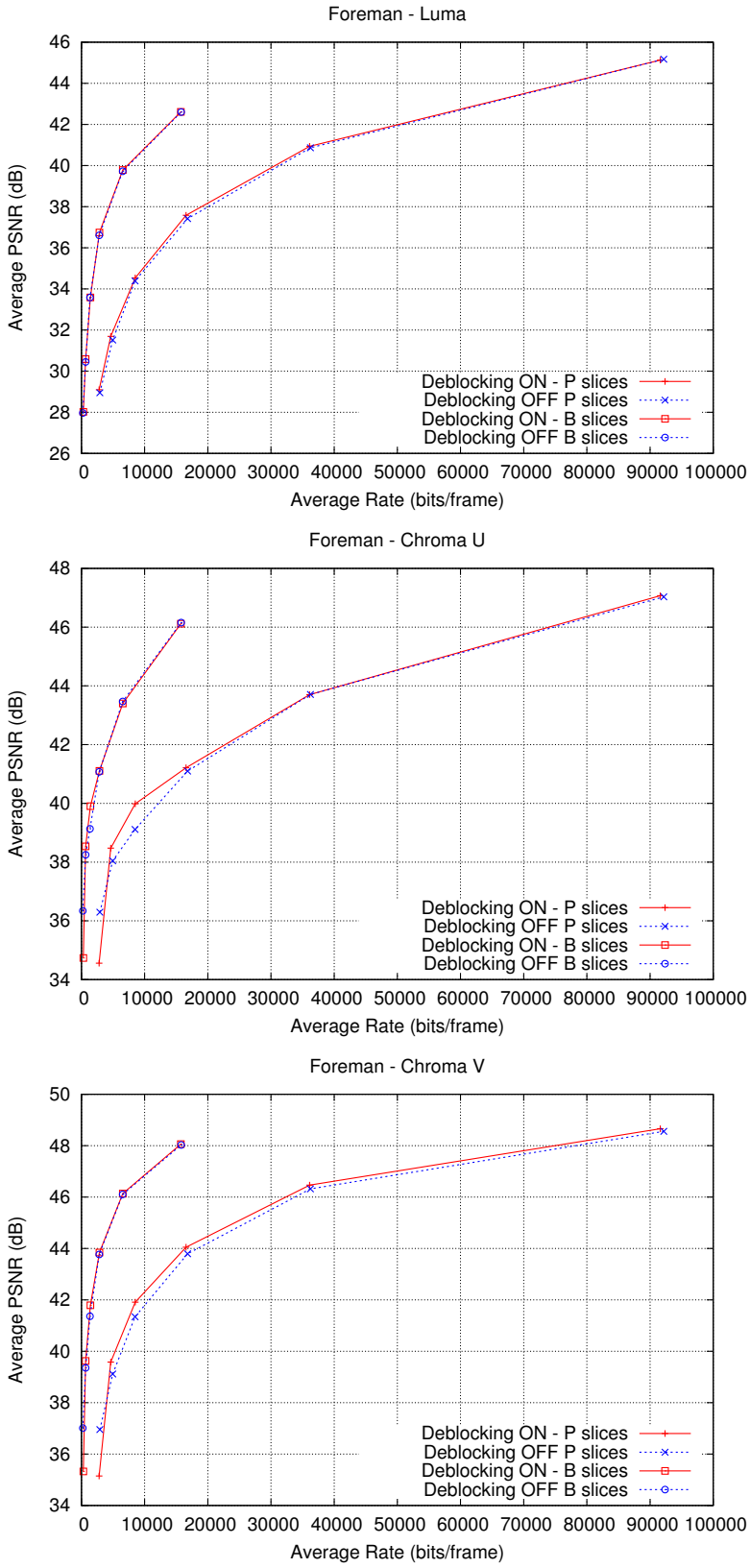


Figure C.52: Comparative results for MMP-Video encoder (1 dictionary), with and without the in-loop deblocking filter, for sequence Foreman (CIF).

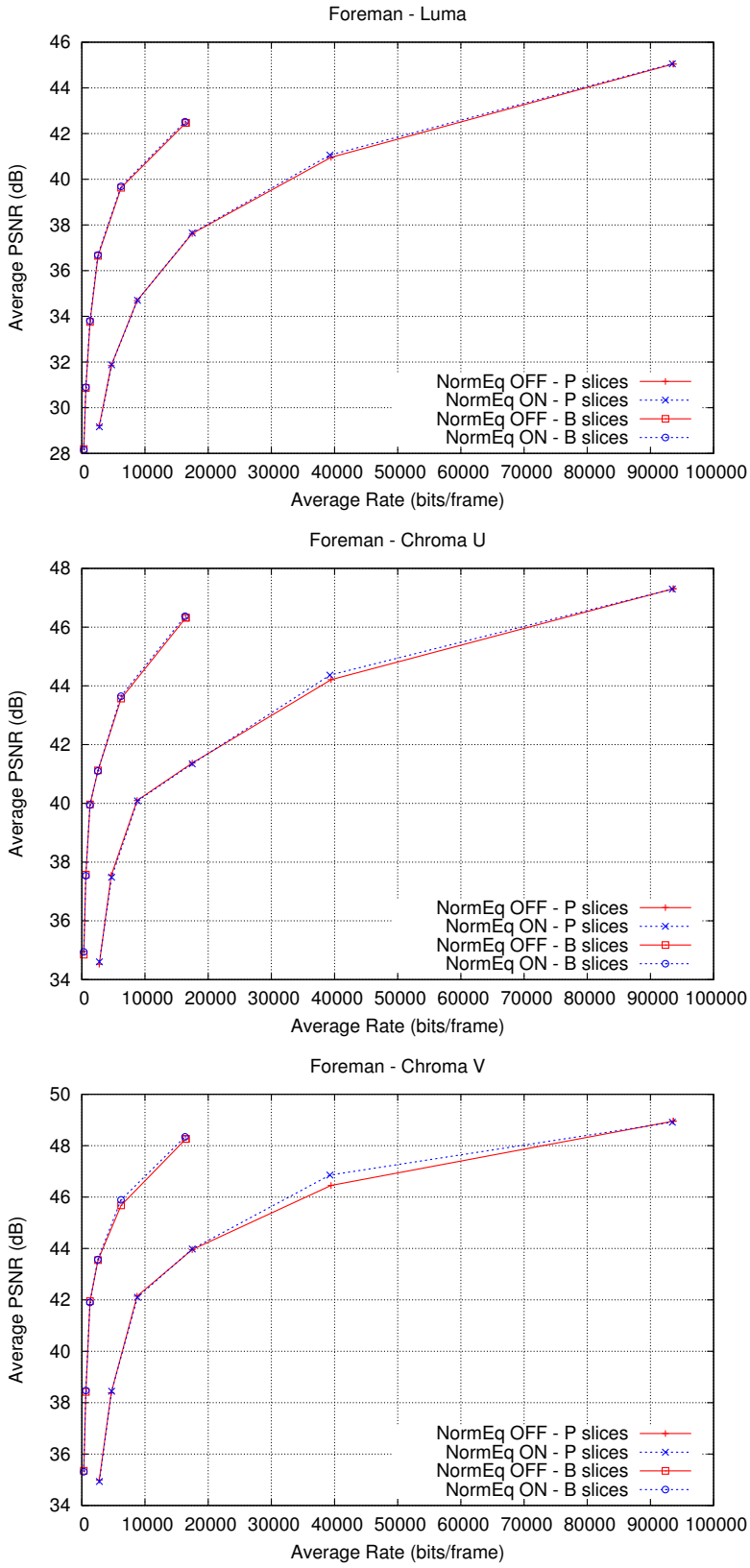


Figure C.53: Comparative results for MMP-Video encoder using two dictionaries, with and without the norm equalisation (L^1), for sequence Foreman (CIF).

Appendix D

Published papers

D.1 Journal papers

J1. N. M. M. Rodrigues, E. A. B. da Silva, M. B. de Carvalho, S. M. M. de Faria and Vitor M. M. Silva, "On Dictionary Adaptation for Recurrent Pattern Image Coding", Image Processing, IEEE Transactions on, vol.17, no.9, pp.1640-1653, Sept. 2008

D.2 Book sections

B1. N. M. M. Rodrigues, E. A. B. da Silva, M. B. de Carvalho, S. M. M. de Faria and Vitor M. M. Silva, "Improving Multiscale Recurrent Pattern Image Coding with Deblocking Filtering", *Third International Conference, ICETE 2006, Setúbal, Portugal, August 7-10, 2006, Selected Papers Series: Communications in Computer and Information Science*, Vol. 9, Filipe, Joaquim; Obaidat, Mohammad S. (Eds.), ISBN: 978-3-540-70759-2, Springer, 2008.

B2. N. M. M. Rodrigues, E. A. B. da Silva, M. B. de Carvalho, S. M. M. de Faria and Vitor M. M. Silva, "H.264/AVC Based Video Coding Using Multiscale Recurrent Patterns: First Results", *VLBV 2005, Lecture Notes on Computer Science*, vol. 3893, pp 107-114, Luigi Atzori, Daniel D. Giusto, Riccardo Leonardi, Fernando Pereira (Eds.), ISBN: 3-540-33578-1, Springer-Verlag, 2006.

D.3 Conference papers

C1. N. M. M. Rodrigues, E. A. B. da Silva, M. B. de Carvalho, S. M. M. de Faria and Vitor M. M. Silva, "On Overriding H.264/AVC B-Slice Predicted Residue Coding", *PCS2007 - Picture Coding Symposium*, Lisbon, Portugal, November, 2007.

C2. N. M. M. Rodrigues, E. A. B. da Silva, M. B. de Carvalho, S. M. M. de Faria and Vitor M. M. Silva, "Multiscale Recurrent Pattern Predictive Image Coding with Template Matching", *ConfTele 2007 - Conference on Telecommunications*, Peniche, Portugal, May 2007.

C3. N. M. M. Rodrigues, E. A. B. da Silva, M. B. de Carvalho, S. M. M. de Faria and Vitor M. M. Silva, "Improving H.264/AVC Inter Compression with Multiscale Recurrent Patterns", *ICIP 2006 - IEEE International Conference on Image Processing*, Atlanta, United States, October 2006.

C4. N. M. M. Rodrigues, E. A. B. da Silva, M. B. de Carvalho, S. M. M. de Faria and Vitor M. M. Silva, "Improving Multiscale Recurrent Pattern Image Coding with Enhanced Dictionary Updating Strategies", *IEEE International Telecommunications Symposium*, Fortaleza, Brazil, September 2006 .

C5. N. M. M. Rodrigues, E. A. B. da Silva, M. B. de Carvalho, S. M. M. de Faria and Vitor M. M. Silva, "Basis Optimisation for Multiscale Recurrent Pattern Coding", *Mathematical Techniques and Problems in Telecommunications*, Leiria, Portugal, September, 2006.

C6. N. M. M. Rodrigues, E. A. B. da Silva, M. B. de Carvalho, S. M. M. de Faria and Vitor M. M. Silva, "An Efficient H.264-based Video Encoder Using Multiscale Recurrent Patterns", *SPIE Optics & Photonics - Applications of Digital Image Processing*, San Diego, United States, August, 2006.

C7. N. M. M. Rodrigues, E. A. B. da Silva, M. B. de Carvalho, S. M. M. de Faria and Vitor M. M. Silva, "Improving Multiscale Recurrent Pattern Image Coding with Deblocking Filtering", *International Conference on Signal Processing and Multimedia Applications*, Setúbal, Portugal, Vol. 1, pp. 118 - 125, August 2006.

C8. N. M. M. Rodrigues, E. A. B. da Silva, M. B. de Carvalho, S. M. M. de Faria, Vitor M. M. Silva and F. Pinagé, "Efficient Dictionary Design for Multiscale Recurrent Patterns Image Coding", *ISCAS 2006 - IEEE International Symposium on Circuits and Systems*, Kos, Greece, May 2006.

C9. N. M. M. Rodrigues, E. A. B. da Silva, M. B. de Carvalho, S. M. M. de Faria and Vitor M. M. Silva, " Universal Image Coding Using Multiscale Recurrent Patterns and Prediction", *ICIP 2005 - IEEE International Conference on Image Processing*, Genova, Italy, Vol. II, pp. 245 - 248, September 2005.

C10. N. M. M. Rodrigues, E. A. B. da Silva, M. B. de Carvalho, S. M. M. de Faria and Vitor M. M. Silva, " H.264/AVC Based Video Coding Using Multiscale Recurrent Patterns: First Results", *International Workshop on Very Low Bitrate Video*, Sardinia, Italy, September 2005.

C11. N. M. M. Rodrigues, E. A. B. da Silva, M. B. de Carvalho, S. M. M. de Faria and Vitor M. M. Silva, " Image Predictive Coding using Multiscale Recurrent Patterns", *ConfTele 2005 - Conference on Telecommunications*, Tomar, Portugal, April 2005.

Bibliography

[1] Murilo Bresciani de Carvalho, *Compression of Multidimensional Signals based on Recurrent Multiscale Patterns*, Ph.D. thesis, COPPE - Universidade Federal do Rio de Janeiro, April 2001, *http://www.lps.ufrj.br/profs/eduardo/teses/murilo-carvalho.ps.gz*.

[2] M. de Carvalho, E. da Silva, and W. Finamore, "Multidimensional signal compression using multiscale recurrent patterns," *Elsevier Signal Processing*, vol. 82, pp. 1559–1580, November 2002.

[3] Jacob Ziv and Abraham Lempel, "A universal algorithm for sequential data compression," *IEEE Transactions on Information Theory*, vol. 23, no. 3, pp. 337–343, 1977.

[4] Jacob Ziv and Abraham Lempel, "Compression of individual sequences via variable-rate coding," *IEEE Transactions on Information Theory*, vol. 24, no. 5, pp. 530–536, 1978.

[5] A. Gersho and R. Gray, *Vector Quantization and Signal Compression*, Springer, 1991.

[6] M. J. Atallah, Y. Genin, and W. Szpankowski, "Pattern matching image compression: Algorithmic and empirical results," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 7, pp. 614–627, 1999.

[7] M. B. de Carvalho and W. A. Finamore, "Lossy lempel-ziv on subband coding of images," *IEEE International Symposium on Information Theory*, June 1999.

[8] C. Constantinescu and J. A. Storer, "Improved techniques for single-pass adaptive vector quantization," *Proceedings of the IEEE*, vol. 82, no. 6, pp. 933–939, June 1994.

[9] M. Effros, P. A. Chou, and R. M. Gray, "One-pass adaptive universal vector quantization," *ICASSP 94 - IEEE International Conference on Acoustics, Speech, and Signal Processing*, April 1994.

[10] Amir Said and William A. Pearlman, "A new fast and efficient image codec based on set partitioning in hierarchical trees," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, pp. 243–250, 1996.

[11] W. Pennebaker and J. Mitchel, *JPEG: Still Image Data Compression Standard*, Van Nostrand Reinhold, 1993.

[12] D. S. Taubman and M.W. Marcelin, *JPEG2000: Image Compression Fundamentals, Standards and Practice*, Kluwer Academic Publishers, 2001.

[13] Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG (ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6), *Draft of Version 4 of H.264/AVC (ITU-T Recommendation H.264 and ISO/IEC 14496-10 (MPEG-4 part 10) Advanced Video Coding)*, March 2005.

[14] "Special issue on the H.264/AVC video coding standard," *Circuits and Systems for Video Technology, IEEE Transactions on,*, vol. 13, no. 7, July 2003.

[15] E. B. L. Filho, N. M.M. Rodrigues, E. A. B. da Silva, S. M. M. de Faria, V. M. M. da Silva, and M. B. de Carvalho, "ECG signal compression based on DC equalization and complexity sorting," *Biomedical Engineering, IEEE Transactions on*, vol. 55, no. 7, pp. 1923–1926, July 2008.

[16] E. B. L. Filho, N. M. M. Rodrigues, E. A. B. da Silva, S. M. M. de Faria, Vitor M. M. Silva, and M. B. de Carvalho, "On ECG signal compression with one-dimensional multiscale recurrent patterns allied to pre-processing techniques," *To appear in IEEE Transactions on Biomedical Engineering*, 2008.

[17] E. B. L. Filho, E. A. B. da Silva, M. B. de Carvalho, W. S. S. Júnior, and J. Koiller, "Electrocardiographic signal compression using multiscale recurrent patterns," *IEEE Transactions on Circuits and Systems I*, vol. 52, no. 12, pp. 2739–2753, December 2005.

[18] M. H. V. Duarte, M. B. de Carvalho, E. A. B. da Silva, C. L. Pagliari, and G. V. Mendonça, "Multiscale recurrent patterns applied to stereo image coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 15, November 2005.

[19] T. Syzmanski J. Storer, "Data compression via textual substitution," *Journal of the ACM*, vol. 29, no. 4, pp. 928–951, 1982.

[20] DEFLATE Compressed Data Format Specification version 1.3, *http://tools.ietf.org/html/rfc1951*, May 1996.

[21] D. Huffman, "A method for the construction of minimum redundancy codes," *Proceedings of the IRE*, vol. 40, no. 8, pp. 1098–1101, 1952.

[22] T. Bell, "Better opm/l text compression," *IEEE Transactions on Communications*, vol. 34, no. 12, pp. 1176–1182, 1986.

[23] R. Brent, "A linear algorithm for data compression," *Australian Computer Journal*, vol. 19, no. 2, pp. 64–68, 1987.

[24] S. Even M. Rodeh, V. Pratt, "Linear algorithm for data compression via string matching," *Journal of the ACM*, vol. 28, no. 1, pp. 16–24, 1981.

[25] Terry A. Welch, "A technique for high performance data compression," *IEEE Computer*, vol. 17, no. 6, pp. 8–19, 1984.

[26] M. Jakobsson, "Compression of character strings by an adaptive dictionary," *BIT Numerical Mathematics*, vol. 4, no. 25, pp. 593–603, 1985.

[27] P. Tischer, "A modified Lempel-Ziv-Welch data compression scheme," *Australian Computer Science Communications*, vol. 9, no. 1, pp. 262–272, 1987.

[28] D. Greene E. Fiala, "Data compression with finite windows," *Communications of the ACM*, vol. 32, no. 4, pp. 490–505, 1989.

[29] Nathanael J. Brittain and Mahmoud R. El-Sakka, "Grayscale true two-dimensional dictionary-based image compression," *J. Vis. Comun. Image Represent.*, vol. 18, no. 1, pp. 35–44, 2007.

[30] M. Alzina, W. Szpankowski, and A. Grama, "2D-pattern matching image and video compression: theory, algorithms, and experiments," *IEEE Transactions on Image Processing*, vol. 11, no. 3, pp. 318–331, March 2002.

[31] Gabriela Dudek, Przemysław Borys, and Zbigniew J. Grzywna, "Lossy dictionary-based image compression method," *Image Vision Comput.*, vol. 25, no. 6, pp. 883–889, 2007.

[32] Paulo S.R. Diniz, Eduardo A.B. da Silva, and Sergio L. Netto, *Digital Signal Processing: System Analysis and Design*, Cambridge University Press, 2002.

[33] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantization design," *IEEE Transactions on Communications*, vol. 28, pp. 84–95, January 1980.

[34] P. Cosman, K. Oehler, E. Riskin, and R. Gray, "Using vector quantization for image processing," *Proc. of the IEEE*, vol. 81, no. 9, pp. 1326–1341, September 1993.

[35] C. Chan and M. Vetterli, "Lossy compression of individual signals based on string matching and one pass codebook design," *ICASSP - IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1995.

[36] M.-L. Guobin Shen; Bing Zeng; Liou, "Adaptive vector quantization with codebook updating based on locality and history," *Image Processing, IEEE Transactions on*, vol. 12, no. 3, pp. 283–295, March 2003.

[37] J. Fowler, "Generalized threshold replenishment: An adaptive vector quantization algorithm for the coding of nonstationary sources," *IEEE Transactions on Image Processing*, October 1998.

[38] M. Wagner and D. Saupe, "Video coding with quadtrees and adaptive vector quantization," *Proceedings EUSIPCO 2000, Tampere, Finland*, September 2000.

[39] ISO/IEC JTC1/SC29/WG1 N1545, "JBIG2 final draft international standard," December 1999.

[40] ITU-T, "Information technology - coded representation of picture and audio information - progressive bi-level image compression, recommendation T.82," March 1993, *http://www.itu.int/rec/T-REC-T.82*.

[41] F. Ono, W. Rucklidge, R. Arps, and C. Constantinescu, "JBIG2 - the ultimate bi-level image coding standard," *IEEE International Conference on Image Processing*, 2000.

[42] W.B. Pennebaker, J.L. Mitchell, G.G. Langdon, and R.B. Arps, "An overview of the basic principles of the q-coder adaptive binary arithmetic coder," *IBM Journal of research and development*, vol. 32, no. 6, pp. 717–726, Nov 1988.

[43] International Telecommunicatiosn Union Recommendation T.6, "Facsimile coding schemes and coding control functions for group 4 facsimile apparatus," 1988.

[44] Yan Ye and Pamela Cosman, "Dictionary design for text image compression with JBIG2," *IEEE Trans. Image Processing*, vol. 10, no. 6, pp. 818–828, 2001.

[45] Yann Le Cun, Léon Bottou, Patrick Haffner, Jeffery Triggs, Bill Riemers, and Luc Vincent, "Overview of the DjVu document compression technology," in *Proceedings of the Symposium on Document Image Understanding Technologies (SDIUT'01)*, Columbia, MD, April 2001, pp. 119–122.

[46] http://djvu.org/.

[47] S. G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *Signal Processing, IEEE Transactions on*, vol. 41, no. 12, pp. 3397–3415, December 1993.

[48] F. Bergeaud and S. Mallat, "Matching pursuit of images," in *ICIP - IEEE Internatinal Conference on Image Processing*, 1995, pp. 53–56.

[49] P. Frossard, P. Vandergheynst, R. Figueras, and I. Ventura, "High flexibility scalable image coding," 2003.

[50] R. Neff and A. Zakhor, "Very low bit-rate video coding based on matching pursuits," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 7, no. 1, pp. 158–171, February 1997.

[51] J.L. Lin, W.L. Hwang, and S.C. Pei, "A joint design of dictionary approximation and maximum atom extraction for fast matching pursuit," in *ICIP - IEEE Internatinal Conference on Image Processing*, 2004, pp. II: 1125–1128.

[52] R. Neff and A. Zakhor, "Matching pursuit video coding - part I: Dictionary approximation," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 12, no. 1, pp. 13–26, January 2002.

[53] R. Neff and A. Zakhor, "Matching pursuit video coding - part II: Operational models for rate and distortion," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 12, no. 1, pp. 27–39, January 2002.

[54] K. Ramchandran and M. Vetterli, "Best wavelet packet basis in a rate-distortion sense," *IEEE Transactions on Image Processing*, vol. 2, no. 2, pp. 160–175, April 1993.

[55] A. Ortega and K. Ramchandran, "Rate-distortion methods for image and video compression," *IEEE Signal Processing Magazine*, pp. 23–50, November 1998.

[56] Ian H. Witten, Radford M. Neal, and John G. Cleary, "Arithmetic coding for data compression," *Commun. ACM*, vol. 30, no. 6, pp. 520–540, 1987.

[57] Anil K. Jain, *Fundamentals of digital image processing*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.

[58] Eddie de Lima Filho, "Compressão de imagens utilizando recorrência de padrões multiescalas com critério de continuidade inter-blocos," M.S. thesis, COPPE - Universidade Federal do Rio de Janeiro, April 2004, *http://www.lps.ufrj.br/profs/eduardo/teses/eddie-filho-msc.pdf*.

[59] Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG (ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6), *Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC), Fidelity Range Extensions documents JVT-L047 (non-integrated form) and JVT-L050 (integrated form)*, July 2004.

[60] D. Marpe, T. Wiegand, and S. Gordon, "H.264/MPEG4-AVC fidelity range extensions: Tools, profiles, performance, and application areas," *IEEE International Conference on Image Processing 2005*, vol. 1, pp. 593–596, September 2005.

[61] Gary J. Sullivan, Pankaj Topiwala, and Ajay Luthra, "The H.264/AVC advanced video coding standard: Overview and introduction to the fidelity range extensions," *SPIE Conference on Applications of Digital Image Processing XXVII*, August 2004.

[62] Khalid Sayood, *Introduction to Data Compression*, Morgan Kaufmann, 2000.

[63] S. Mallat, *A Wavelet Tour of Signal Processing*, Academic Press; 2nd edition, 1999.

[64] Dan E. Dudgeon and Russel M. Mersereau, *Multidimensional Digital Signal Processing*, Prentice Hall, 1984.

[65] M. de Carvalho, D. M. Lima, E. da Silva, and W. Finamore, "Universal multiscale matching pursuits algorithm with reduced blocking effect," *IEEE International Conference on Image Processing*, September 2000.

[66] M. Unser, "Splines: a perfect fit for signal and image processing," *Signal Processing Magazine, IEEE*, vol. 16, no. 6, pp. 22–38, Nov 1999.

[67] F.L. Bookstein, "Principal warps: Thin-plate splines and the decomposition of deformations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 6, pp. 567–585, 1989.

[68] C. de Boor, *Matlab Spline Toolbox User's Guide*, C. de Boor and The MathWorks, Inc, 1990-2007, *http://www.mathworks.com/access/helpdesk/help/pdf_doc/splines/splines.pdf*.

[69] Luc Devroye, *Non-Uniform Random Variate Generation*, Springer-Verlag, 1986.

[70] J. Armando Domínguez-Molina, Graciela González-Farías, and Ramón M. Rodríguez-Dagnino, "A practical procedure to estimate the shape parameter in the generalized gaussian distribution," *http://www.cimat.mx/reportes/enlinea/I-01-18_eng.pdf*.

[71] S. G. Mallat, "A theory for multiresolution signal decomposition: The wavelet representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 7, pp. 674–693, 1989.

[72] E.Y. Lam, "Statistical modelling of the wavelet coefficients with different bases and decomposition levels," *IEE Proc.- Vis. Image Signal Processing*, vol. 151, no. 3, pp. 203–206, June 2004.

[73] Edmund Y. Lam, "Analysis of the dct coefficient distributions for document coding," *IEEE Signal Processing Letters*, vol. 11, no. 2, pp. 97–100, February 2004.

[74] F. Chen, Z. Gao, and J. Villasenor, "Lattice vector quantization of generalized gaussian sources," *IEEE Transactions on Information Theory*, vol. 43, no. 1, pp. 92–103, January 1997.

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

[76] Ian E.G. Richardson, *H.264 and MPEG-4 video compression*, John Wiley & Sons Ltd, 2003.

[77] M. Wien, "Variable block-size transforms for H.264/AVC," *Circuits and Systems for Video Technology, IEEE Transactions on,*, vol. 13, no. 7, pp. 604–613, July 2003.

[78] T. Fischer, "A pyramid vector quantizer," *IEEE Transactions on Information Theory*, vol. IT-32, no. 4, pp. 568–583, July 1986.

[79] Frederico da Silva Pinagé, "Avaliação do desempenho de algoritmos de compressão de imagens usando recorrência de padrões multiescalas," M.S. thesis, COPPE - Universidade Federal do Rio de Janeiro, July 2005, *http://www.lps.ufrj.br/profs/eduardo/teses/frederico-pinage.pdf*.

[80] Alexander M. Mood, Franklin A. Graybill, and Duane C. Boes, *Introduction to the Theory of Statistics*, International 3rd Revised Ed edition; McGraw-Hill Publishing Co., 1974.

[81] Z. Pan, K. Kotani, and T. Ohmi, "Improved fast search method for vector quantization using discrete walsh transform," *ICIP '04 - IEEE International Conference on Image Processing*, vol. 5, pp. 3177–3180, October 2004.

[82] L. Guan and M. Kamel, "Equal-average hyper plane partitioning method for vector quantization of image data," *Pattern Recognition Letters*, vol. 13, pp. 693–699, October 1992.

[83] P. List, A. Joch, J. Lainema, G. Bjntegaard, and M. Karczewicz, "Adaptive deblocking filter," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, no. 7, pp. 614–619, July 2003.

[84] Thiow Tan, Choong Boon, and Yoshinory Suzuki, "Intra prediction by template matching," *IEEE International Conference on Image Processing*, October 2006.

[85] Li-Yi Wei and Marc Levoy, "Fast texture synthesis using tree-structured vector quantization," *Proceeding of SIG-GRAPH 2000*, pp. 479–488, July 2000.

[86] K. Sugimoto, M. Kobayashi, Y. Suzuki, S. Kato, and C. Seng Boon, "Inter frame coding with template matching spatio-temporal prediction," *IEEE International Conference on Image Processing*, October 2004.

[87] R. Caetano, E.A. da Silva, and A.G. Ciancio, "Matching pursuits video coding using generalized bit-planes," *Image Processing. 2002. Proceedings. 2002 International Conference on*, vol. 3, pp. III–677–III–680 vol.3, 2002.

[88] ITU-T & ISO/IEC JTC 1, ITU-T Recommendation H.262 and ISO/IEC 13818-2 (MPEG-2), *Generic Coding of Miving Pictures and Associated Audio Information - Part 2: Video*, 1994.

[89] T. Wedi and H.G. Musmann, "Motion- and aliasing-compensated prediction for hybrid video coding," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, no. 7, pp. 577–586, July 2003.

[90] H.S. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky, "Low-complexity transform and quantization in H.264/AVC," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, no. 7, pp. 598–603, July 2003.

[91] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, no. 7, pp. 620–636, July 2003.

[92] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G.J. Sullivan, "Rate-constrained coder control and comparison of video coding standards," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, no. 7, pp. 688–703, July 2003.

[93] http://iphome.hhi.de/suehring/tml/download/.

[94] J. Ribas-Corbera, P.A. Chou, and S.L. Regunathan, "A generalized hypothetical reference decoder for H.264/AVC," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, no. 7, pp. 674–687, July 2003.

[95] Limin Liu, Yuxin Liu, and Edward J. Delp, "Enhanced intra prediction using context-adaptive linear prediction," *PCS2007 - Picture Coding Symposium*, November 2007.

[96] J. Balle and M. Wien, "Extended texture prediction for H.264/AVC intra coding," *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, vol. 6, pp. VI –93–VI –96, 16 2007-Oct. 19 2007.

[97] Yung-Lyul Lee, Ki-Hun Han, and G.J. Sullivan, "Improved lossless intra coding for H.264/MPEG-4 AVC," *Image Processing, IEEE Transactions on*, vol. 15, no. 9, pp. 2610–2615, Sept. 2006.

[98] Xiang Li, Norbert Oertel, and Andre Kaup, "Gradient intra prediction for coding of computer animated videos," *Multimedia Signal Processing, 2006 IEEE 8th Workshop on*, pp. 50–53, Oct. 2006.

[99] Yu-Nan Pan and Tsung-Han Tsai, "Fast motion estimation and edge information inter-mode decision on H.264 video coding," *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, vol. 2, pp. II –473–II –476, 16 2007-Oct. 19 2007.

[100] Hoi-Ming Wong, O.C. Au, A. Chang, Shu-Kei Yip, and Chi-Wang Ho, "Fast mode decision and motion estimation for H.264 (FMDME)," *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, pp. 4 pp.–, May 2006.

[101] Peng Yin, H.-Y.C. Tourapis, A.M. Tourapis, and J. Boyce, "Fast mode decision and motion estimation for JVT/H.264," *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, vol. 3, pp. III–853–6 vol.2, Sept. 2003.