



Tuukka Toivonen

EFFICIENT METHODS FOR VIDEO CODING AND PROCESSING



FACULTY OF TECHNOLOGY,
DEPARTMENT OF ELECTRICAL AND INFORMATION ENGINEERING,
INFOTECH OULU,
UNIVERSITY OF OULU



ACTA UNIVERSITATIS OULUENSIS
C Technica 290

TUUKKA TOIVONEN

**EFFICIENT METHODS FOR VIDEO
CODING AND PROCESSING**

Academic dissertation to be presented, with the assent of
the Faculty of Technology of the University of Oulu, for
public defence in Auditorium TS101, Linnanmaa, on
January 11th, 2008, at 12 noon

OULUN YLIOPISTO, OULU 2007

Copyright © 2007
Acta Univ. Oul. C 290, 2007

Supervised by
Professor Janne Heikkilä

Reviewed by
Professor Reiner Creutzburg
Professor Jarmo Takala

ISBN 978-951-42-8694-0 (Paperback)
ISBN 978-951-42-8695-7 (PDF)
<http://herkules.oulu.fi/isbn9789514286957/>
ISSN 0355-3213 (Printed)
ISSN 1796-2226 (Online)
<http://herkules.oulu.fi/issn03553213/>

Cover design
Raimo Ahonen

OULU UNIVERSITY PRESS
OULU 2007

Toivonen, Tuukka, Efficient methods for video coding and processing

Faculty of Technology, University of Oulu, P.O.Box 4000, FI-90014 University of Oulu, Finland,
Department of Electrical and Information Engineering, Infotech Oulu, University of Oulu, P.O.Box
4500, FI-90014 University of Oulu, Finland
Acta Univ. Oul. C 290, 2007
Oulu, Finland

Abstract

This thesis presents several novel improvements to video coding algorithms, including block-based motion estimation, quantization selection, and video filtering. Most of the presented improvements are fully compatible with the standards in general use, including MPEG-1, MPEG-2, MPEG-4, H.261, H.263, and H.264.

For quantization selection, new methods are developed based on the rate-distortion theory. The first method obtains locally optimal frame-level quantization parameter considering frame-wise dependencies. The method is applicable to generic optimization problems, including also motion estimation. The second method, aimed at real-time performance, heuristically modulates the quantization parameter in sequential frames improving significantly the rate-distortion performance. It also utilizes multiple reference frames when available, as in H.264. Finally, coding efficiency is improved by introducing a new matching criterion for motion estimation which can estimate the bit rate after transform coding more accurately, leading to better motion vectors.

For fast motion estimation, several improvements on prior methods are proposed. First, fast matching, based on filtering and subsampling, is combined with a state-of-the-art search strategy to create a very quick and high-quality motion estimation method. The successive elimination algorithm (SEA) is also applied to the method and its performance is improved by deriving a new tighter lower bound and increasing it with a small constant, which eliminates a larger part of the candidate motion vectors, degrading quality only insignificantly. As an alternative, the multilevel SEA (MSEA) is applied to the H.264-compatible motion estimation utilizing efficiently the various available block sizes in the standard.

Then, a new method is developed for refining the motion vector obtained from any fast and suboptimal motion estimation method. The resulting algorithm can be easily adjusted to have the desired tradeoff between computational complexity and rate-distortion performance. For refining integer motion vectors into half-pixel resolution, a new very quick but accurate method is developed based on the mathematical properties of bilinear interpolation.

Finally, novel number theoretic transforms are developed which are best suited for two-dimensional image filtering, including image restoration and enhancement, but methods are developed with a view to the use of the transforms also for very reliable motion estimation.

Keywords: block matching, filtering, motion estimation, number theoretic transforms, rate-distortion optimization

To Sylvia

Preface

The work described in this thesis was completed in the Machine Vision Group of the Department of Electrical and Information Engineering at the University of Oulu, during the years 2002–2007.

Professors Janne Heikkilä and Olli Silvén were the supervisors for this thesis. In particular Prof. Janne Heikkilä was a great source of new ideas upon which this thesis is based. I am grateful to professors Jarmo Takala and Reiner Creutzburg for reviewing the manuscript. Their feedback helped to improve the quality of the thesis. I also wish to thank Gordon Roberts for the corrections in the English language. All errors still left are mine.

The work has been supported financially by the Infotech Oulu Graduate School, Technology Agency of Finland (Tekes), Nokia Foundation, and Foundation for the Promotion of Technology (TES).

I wish to thank all my colleagues in the Machine Vision Group, especially Ville, Matti, Markus, Jari, Sami, and Teuvo for allowing me to join very interesting discussions on scientific and other matters.

Finally, I wish to express gratitude to my parents Briitta and Matti, my brother Veli-Matti, and to my uncle's family in Tampere for their generous hospitality: Tapio, Alicja, Alice, and in particular Sylvia. Without her this thesis would not have been possible.

Oulu, December 10, 2007

Tuukka Toivonen

Symbols and abbreviations

a_i	Element of vector \mathbf{a} at position i (zero-based index)
\tilde{a}_i	Element of transformed vector \mathbf{a} at position i (zero-based index)
b	Positive arbitrary integer
d_n	Distortion of coding unit n
f	Current partition
$f(x,y)$	Luma value in current partition at position (x,y)
\hat{f}	Reference partition
f_p	Luma value in coding unit at position p
\hat{f}_p	Luma value in reconstructed coding unit at position p
j	Imaginary unit, $j^2 = -1$
k	Arbitrary positive power of two, $k = 2^b$
(m_x, m_y)	Candidate motion vector
(m_x^*, m_y^*)	Optimum motion vector
q	Quantization parameter (Chapter 3) or modulus (Chapter 5)
q_0	Original quantization parameter value
q_i	A prime factor of modulus q
q_n	Quantization parameter for coding unit n
r	Correlation function
r_n	Bit rate for coding unit n
\bar{x}	Average of variable x
$A_{i,j}$	Element in matrix \mathbf{A} at row i and column j (zero-based indices)
$C(m_x, m_y)$	Matching criterion value at candidate motion vector (m_x, m_y)
C_b	Chroma component b
C_r	Chroma component r
D	Total distortion
$F(u, v)$	Transformed block of $f(x, y)$
Q	Quantization value higher in a hierarchy (Chapter 3) or number of prime factors (Chapter 5)
R	Total bit rate
R_{\max}	Bit rate limit
R_{mv}	Bit rate corresponding to encoding a motion vector
$\widetilde{\text{SAD}}(m_x, m_y)$	Approximated sum of the absolute differences at motion vector (m_x, m_y)
SSD_{AC}	Sum of squared differences of alternating current

W	Transform kernel
X	Partition width
Y	Partition height
Y'	Luma component
\mathbf{f}	Current video frame
$\hat{\mathbf{f}}$	Reference video frame
\mathbf{f}_i	Coding unit number i
\mathbf{h}	One-dimensional filter (a vector)
\mathbf{q}^*	Vector of optimal quantization parameters
\mathbf{A}	Two-dimensional signal or a matrix
\mathbf{F}	Whole video including all frames
\mathbf{H}	Two-dimensional filter
\mathcal{P}	Set of luma and chroma positions p in coding units
\mathcal{Q}	Set of possible quantization parameter values
\mathcal{T}	Transform
λ	Lagrangian multiplier
λ_{mv}	Lagrangian multiplier for motion vector bit rate
AC	Alternating current
Codec	Encoder and decoder pair
CRT	Chinese remainder theorem
DC	Direct current
DCT	Discrete cosine transform
DS	Diamond search
ESEA	Extended successive elimination algorithm
FFT	Fast Fourier transform
FME	Fractional pixel accurate motion estimation
FNT	Fermat number transform
FPP	Fractional pixel positions
FS	Full search
GFNT	Generalized Fermat number transform
HEXBS	Hexagon-based search
HME	Half pixel accurate motion estimation
HMV	Half integer accurate motion vector
HPP	Half pixel position
HVS	Human visual system
IDCT	Inverse discrete cosine transform
IEC	International Electrotechnical Commission

IME	Integer pixel accurate motion estimation
IPP	Integer pixel position
IS	International standard
ISO	International Organization for Standardization
ITU	International Telecommunication Union
IVLC	Inverse variable length coder
JTC	Joint technical committee
MC	Motion compensation
MCV	Matching criterion value
ME	Motion estimation
MPEG	Moving picture experts group
MRC	Mixed radix conversion
MSEA	Multilevel successive elimination algorithm
MV	Motion vector
NTT	Number theoretic transform
PDE	Partial distortion elimination
Pixel	Picture element
PSNR	Peak signal-to-noise power ratio
QCWSD	Quadratic coordinate-wise steepest descent
QME	Quarter pixel accurate motion estimation
QP	Quantization parameter
R-D	Rate-distortion
RGB	Red, green, and blue
RNS	Residue number system
RS	Rotation search
SAD	Sum of absolute differences
SC	Subcommittee
SDS	Small diamond search
SEA	Successive elimination algorithm
SSD	Sum of squared differences
TSS	Three step search
VCEG	Video coding experts group
VLC	Variable length coder
WG	Working group

List of original articles

This dissertation is based on the following articles, which are referred to in the text by their Roman numerals:

- I Toivonen T, Merritt L, Ojansivu V & Heikkilä J (2007) A New Rotation Search for Dependent Rate-Distortion Optimization in Video Coding. Proc. 32nd International Conference on Acoustics, Speech, and Signal Processing (ICASSP) 1: 1165–1168.
- II Toivonen T & Heikkilä J (2006) Reduced Frame Quantization in Video Coding. In: 9th International Workshop VLBV Proceedings, Lecture Notes in Computer Science 3893, Springer-Verlag: 61–67.
- III Toivonen T & Heikkilä J (2006) Motion Vector Refinement in Video Coding Based on Statistical Distribution. Proc. 13th International Conference on Systems, Signals & Image Processing (IWSSIP): 39–42.
- IV Toivonen T & Heikkilä J (2003) A New Rate-Minimizing Matching Criterion and a Fast Algorithm for Block Motion Estimation. Proc. IEEE International Conference on Image Processing (ICIP) 2: 355–358.
- V Toivonen T & Heikkilä J (2004) Fast Full Search Block Motion Estimation for H.264/AVC with Multilevel Successive Elimination Algorithm. Proc. IEEE International Conference on Image Processing (ICIP) 3: 1485–1488.
- VI Toivonen T & Heikkilä J (2006) Improved Unsymmetric-Cross Multi-Hexagon-Grid Search Algorithm for Fast Block Motion Estimation. Proc. IEEE International Conference on Image Processing (ICIP): 2369–2372.
- VII Toivonen T & Heikkilä J (2003) Efficient Method for Half-Pixel Block Motion Estimation Using Block Differentials. In: Visual Content Processing and Representation, 8th International Workshop VLBV Proceedings, Lecture Notes in Computer Science 2849, Springer-Verlag: 225–232.
- VIII Toivonen T & Heikkilä J (2006) Video Filtering with Fermat Number Theoretic Transforms using Residue Number System. IEEE Transactions on Circuits and Systems for Video Technology 16(1): 92–101.
- IX Toivonen T, Heikkilä J & Silvén O (2002) A New Algorithm for Fast Full Search Block Motion Estimation Based on Number Theoretic Transforms. Proc. 9th International Workshop on Systems, Signals and Image Processing (IWSSIP): 90–94.

All Papers from I to IX were completely written and the experiments conducted by the author, although Prof. Heikkilä gave valuable input to the text and structure of the papers and gave comments and guidance for implementing the experiments.

For Papers I, II, and III the idea was solely the author's. Paper IV was based on Prof. Heikkilä's ideas, and the idea for Paper V was a joint effort by the author and Prof. Heikkilä. For Papers VI and VII, the idea was developed again solely by the author. The mathematical derivations for Paper VII were performed by the author.

For Paper I, Mr. Merritt implemented the first version of the steepest descent algorithm for H.264 and added hooks into the encoder to allow changing the frame quantization parameters easily. The author had discussions about efficient search algorithms

with Mr. Ojansivu, who also gave important comments on the structure of the paper along with Prof. Heikkilä.

Paper VIII was again based on the basic ideas from Prof. Heikkilä, but the author had important contributions, most importantly the application of the residue number system (RNS) to the Fermat number transform. The author derived mathematically and implemented the conversion circuit from RNS into normal binary code.

Prof. Heikkilä also obtained the idea of using number theoretic transforms for motion estimation, on which Paper IX was based, although the author noticed that a multiplication-free transform is possible also with the generalized Fermat number transform. Prof. Silvén served as a source of guidance for Paper IX.

Contents

Abstract	
Preface	7
Symbols and abbreviations	9
List of original articles	13
Contents	15
1 Introduction	17
1.1 Video capturing and coding	17
1.2 The contribution of the thesis	19
1.3 Summary of the original papers	20
2 Basics of video coding	23
2.1 Video coding tools	23
2.1.1 Color space transform	26
2.1.2 Motion estimation and compensation	26
2.1.3 Intra prediction	28
2.1.4 Transform coding and quantization	28
2.1.5 Entropy coding	29
2.1.6 Loop filter	30
2.2 Video coding standards	30
3 Video coding rate-distortion optimization	35
3.1 Rate and distortion measures	37
3.2 Benchmarking video encoding methods	40
3.3 Optimal bit allocation	42
3.3.1 Independent coding units	42
3.3.2 Lagrangian relaxation	44
3.3.3 Dependent coding units	46
3.4 Proposed new optimization methods	49
3.4.1 Rotation search	49
3.4.2 Faster real-time optimization algorithm	50
3.5 Summary	52
4 Fast block motion estimation	55
4.1 Fast search strategies	57
4.1.1 Motion vector prediction	57
4.1.2 Fast search patterns	58
4.1.3 Fast end condition	61

4.1.4	Motion vector refinement	63
4.2	Fast calculation of a matching criterion	64
4.2.1	Subsampling the matching criterion	64
4.2.2	Bit depth transforms	67
4.3	Hierarchical search algorithms	68
4.3.1	Sequential methods	69
4.3.2	Successive elimination algorithm	70
4.3.3	Multilevel successive elimination algorithm	72
4.3.4	Partial distortion elimination	73
4.3.5	New algorithms	75
4.4	Fractional pixel motion estimation	77
4.4.1	Hierarchical fractional pixel motion estimation	78
4.4.2	Polynomial interpolation	79
4.4.3	Exact SSD interpolation	80
4.5	Summary	85
5	Number theoretic transforms	87
5.1	The generalized Fermat number transform	87
5.2	Residue number systems	89
5.3	Video filtering	91
5.4	Motion estimation	93
5.5	Summary	95
6	Conclusions	97
	References	98
	Original articles	106

1 Introduction

There would be no modern society as we know it without technological means of communicating with people from a distance. In 1876, Alexander Graham Bell devised an apparatus which was able to transmit instantaneously speech over long distances—the telephone. It was a great success: these days, there are around two billion mobile phone subscribers alone [1], even without counting wire phones. However, it is unnatural to communicate with other people with just spoken words. For example, Mehrabian [2] found out that in some situations more than half of communication content is relayed by visual means—facial expressions and body language.

In 1827, Joseph Nicéphore Niépce obtained the first photograph, but we had to wait until the end of the century, when Louis Lumiere (in 1895) and some other people demonstrated a practical method of recording and playing back moving pictures. Even before that, Paul Gottlieb Nipkow developed in 1884 a television based on a rotating disc which was able to transmit moving pictures over an electrical wire. Nowadays, television has surpassed radio in many aspects. On average, people watch television almost three hours each day [3]. Although this is slightly less than the daily radio listening time [4], television captures the full attention of a viewer.

Taken the great need for personal communications, the importance of visual content, and the popularity of television, why do we still use phones that are based on voice-only transmission? It seems obvious that the major obstacle is the expensiveness of visual communications. In mobile phones, speech transmission requires around 5–12 kilobits per second. Even the lowest quality video requires at least one order of more bandwidth. Television is affordable only because it uses multicasting: the same signal can be sent simultaneously to millions of receivers. This is not possible in personal communications. Nevertheless, we are on the threshold of new technology available to common people which is capable of capturing video with miniature cameras, transmitting it over wire or wireless networks, and showing it on lightweight flat displays. However, all this is possible only with efficient video coding methods, which not only can compress the moving pictures sufficiently to be transmitted over low bandwidth channels, but can do so in real-time and with limited computing power.

1.1 Video capturing and coding

A typical situation in which video coding is required is depicted in Fig. 1. There are two users, both having a portable multimedia terminal containing a color video camera

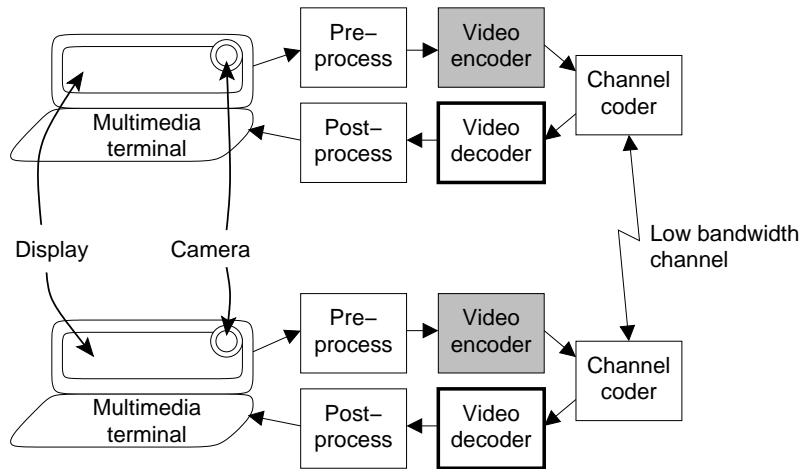


Fig 1. Typical application for video codec.

and a screen capable of displaying video. The situation is symmetric: both terminals need both a video encoder and decoder so that the users can communicate with the video connection. However, before the video captured from the camera is encoded, it is preprocessed. The preprocessing might include image conversion from a Bayer color filter array format into RGB or $Y'C_bC_r$ images, noise removal filtering, detail enhancement filtering, motion deblurring, and other processing that increases the image quality. For preprocessing using linear filtering, a method based on number theoretic transforms (NTTs) is described in Chapter 5.

In the next step, the video is encoded (compressed). The several aspects of encoding are the main topic of this thesis and they are discussed in the forthcoming chapters, and for this reason the rectangle depicting encoding is grayed in the figure. Typical methods used together with common video coding standards are described in Section 2.1. Then, the encoded video sequence is coded for transmission over a low bandwidth channel or for storing it on a form of mass media. The channel coding adds error protection, as necessary, to prevent data corruption during transmission. At the receiver end, the channel coding is removed and possible errors are corrected. Then the video is decoded. The video coding standards, described in Section 2.2, define the operation of video decoder, and therefore the rectangle containing the decoder is strengthened in Fig. 1. The encoding can be implemented in any way, as long as it produces a data stream that the decoder understands and can decode. This makes it possible to improve the encoder algorithms while still being standards-compliant. In the final step, before displaying

the video, the decoded video may be postprocessed to remove known coding artifacts caused by lossy coding methods, such as blockiness and ringing, and to compensate for known display limitations such as nonlinearity of pixel intensity values.

1.2 The contribution of the thesis

This thesis presents several novel improvements to video coding algorithms, including block-based motion estimation, quantization selection, and to video filtering. Most of the improvements presented are fully compatible with standards in general use, including MPEG-1, MPEG-2, MPEG-4, H.261, H.263, and H.264.

For quantization selection, new methods are developed based on rate-distortion theory. The first method obtains a locally optimal frame-level quantization parameter considering frame-wise dependencies. The method is applicable to generic optimization problems, including also motion estimation. The second method, aimed for real-time performance, heuristically modulates the quantization parameter in sequential frames, improving significantly the rate-distortion performance. It also utilizes multiple reference frames when available, as in H.264. Coding efficiency is improved also by introducing a new matching criterion for motion estimation, which can estimate the bit rate of a macroblock partition after transform coding more accurately, leading to better motion vectors.

For fast motion estimation, several improvements are proposed to the previous methods. First, fast matching, based on filtering and subsampling, and the successive elimination algorithm (SEA), are combined with the state-of-the-art unsymmetric-cross multi-hexagon-grid search algorithm to create a very fast and high-quality motion estimation algorithm. The SEA performance is also improved by deriving a new tighter lower bound for the sum of squared differences (SSD) criterion, derived from statistical theory, and increasing the lower bound by a small constant; this eliminates a larger part of the candidate motion vectors, degrading quality only insignificantly. In addition, the Multilevel SEA (MSEA) is applied into the H.264-compatible motion estimation utilizing efficiently the various available partition sizes in the standard. It is shown that the algorithm improves the motion estimation speed significantly, while decreasing only slightly the video coding quality.

A method is then developed for refining the motion vector obtained from any fast and suboptimal motion estimation method. A special refining pattern is obtained experimentally considering the optimal search points. The resulting method can be easily adjusted to have the desired tradeoff between computational complexity and rate-distortion performance. For refining integer motion vectors into half-pixel resolution, a

new very quick but accurate method is developed based on the mathematical properties of bilinear interpolation.

Finally, novel number theoretic transforms are developed which are best suited for two-dimensional image filtering. The transforms are best utilized in the digital circuits which were developed for this thesis. Filtering can be used for image restoration and enhancement, but methods are developed allowing the use of the transforms also for very reliable block motion estimation.

1.3 Summary of the original papers

This thesis consists of eight conference publications and one journal publication. Paper I represents a new efficient optimization method for high-dimensional discrete functions. The method is applied primarily to rate-distortion optimization of H.264 quantization parameters (QPs) among frames in video sequences using Lagrangian relaxation. The algorithm is much more efficient than other nearly globally optimal algorithms. Although the result is guaranteed to be only locally optimal, in practice the results are in most cases the same as in the nearly globally optimal algorithms. The optimization method is also applied successfully in motion estimation, proving its versatility.

Paper II shows a method of allocating bits in some sense optimally among two consecutive video frames so that the quality is maximized. Since the later frame is predicted from the first, more bits should be allocated for the first frame to increase the average quality. For the newest standard H.264, which allows multiple reference frames, the paper also describes a method for reordering two reference frames to further increase quality.

Paper III adds a new stage after any conventional fast motion estimation algorithm. The new stage refines the initial motion vector by also checking the most prospective candidate motion vectors around it for the best match. The additional motion vectors are obtained from a lookup table, which is precalculated from statistical distribution of the matching criterion. The method obtains some of the speed advantage from previous fast motion estimation algorithms but improves the motion estimation accuracy.

Paper IV discusses extensions to the SSD (sum of squared differences) distortion measure. First, a new criterion called SSD_{AC} is represented. It can improve the motion estimate by eliminating the effect of lighting changes, which leads to better correspondence of the criterion with the number of bits required for coding the partition. Second, a new tighter lower bound is derived for the SSD criterion. The new bound can also be used as a lower bound for the new SSD_{AC} measure. A lower bound allows efficient implementation of motion estimation with the SEA and MSEA methods.

Paper V applies the MSEA motion estimation method to an H.264 video encoder. Both MSEA and H.264 motion estimation process the 16×16 pixel macroblocks hierarchically, by subdividing them into smaller partitions. With H.264, the smallest partition size is 4×4 pixels. In the paper, this similarity is exploited and the lower bound, as required by MSEA, is computed hierarchically for all partition sizes, starting from the smallest partitions. The MSEA is also modified to take into account the rate-distortion (R-D) optimized matching criterion, which is employed (instead of plain SAD) by all newer video encoders.

Paper VI uses a layered image presentation to further speed up fast motion estimation algorithms. Particular attention is given to the unsymmetric-cross multi-hexagon-grid search, because it can deliver very good motion estimates, close to the exhaustive search algorithm, but with much less computation. The layered images are used for subsampling the blocks to be matched and for implementing the successive elimination algorithm (SEA). Also the SEA is improved by adding a small constant to the lower bound. This speeds up the algorithm significantly but the loss in quality is very small.

Paper VII presents an efficient method for refining an integer pixel precision motion estimate into half pixel precision. Instead of interpolating image partitions into higher resolution and then matching them as usual, the method interpolates directly the matching criterion at integer pixel locations. Mathematically the two methods are identical to the SSD criterion. For the more common sum of the absolute differences (SAD) criterion, the interpolation gives approximate results, but still the quality remains very good.

Paper VIII develops a new fast convolution method based on the Fermat number theoretic transform (FNT) and the residue number system (RNS). The FNT is also generalized into generalized FNT (GFNT) which allows more flexible word length selection. FNT and GFNT allow one-component real valued transforms without multiplications, which reduces greatly the necessary computations. The multiplications left in the convolution are further simplified by applying the RNS. This appears to be the first time that real valued number theoretic transforms are performed within an RNS. The efficient convolution methods can be used for image preprocessing before the video encoder, or postprocessing after the video has been decoded.

Paper IX presents a new motion estimation algorithm based on number theoretic transforms. Although the paper by Kim *et al.* [5] was published a year earlier, Paper IX has two advantages: first, since the FNT is generalized to use the modulus $q = 2^{24} + 1$ instead of $q = 2^{16} + 1$, the pixel values do not need to be quantized to 4 bits, which attenuates the motion compensation residual. Second, the overlap-save algorithm is applied to allow a larger search region of ± 16 pixels instead of ± 8 , as in [5]. Papers

VIII and IX also note that the GFNT can be performed with only bit shifts, without multiplications.

2 Basics of video coding

2.1 Video coding tools

Video encoders consist of several coding tools. Each tool extracts a part of the video and encodes it using an efficient coding scheme. The extracted and encoded part is sent to the transmission channel. The variance of the residual part is smaller than that of the original video, and can be either given to other coding tools to be compressed further, or, if it is negligible, completely discarded. The residual is the difference between the original video sequence and the decoded video sequence at the receiver. The residual can be discarded when it is very small and the error is invisible, or at least not annoying to human visual system (HVS). Generally the tradeoff between the generated bit rate and the residual variance can be controlled. Since a video encoder has multiple coding tools, an appropriate tradeoff should be chosen for each. This is called rate-distortion optimization, and will be discussed in Chapter 3.

The most commonly used encoders employ hybrid coding [6]. It includes two successive coding tools: temporal prediction of pixel values using motion compensation, followed by spatial transform coding of the residual for removing redundancy within a single frame (Fig. 2). The motion compensation tool generates motion vectors (MVs) using motion estimation; they are entropy coded and transmitted to the decoder. The bit rate required by the MVs depends roughly on the average MV length, and by favoring shorter MVs, the resulting bit rate from the motion compensation can be reduced, at the cost of leaving residual with a larger variance. Since motion estimation is a very important coding tool, and computationally very demanding, it will be discussed in Chapter 4. The residual signal is then transformed with the discrete cosine transform (DCT) or a similar transform and the resulting coefficients are quantized. Due to the quantization, the coefficients can be represented with a small number of bits, but the original video can no longer be restored exactly from the coefficients and MVs. The coefficients are then also entropy coded and added to the bit stream. The residual is discarded. Motion compensation and transform coding are the two most important examples of numerous tools, which are used for reducing the bit rate in video coding. Most video coding standards (see Section 2.2) are very close to the flowchart shown in Fig. 3. The corresponding video decoder is shown in Fig. 4. The parts of the encoder are described next.

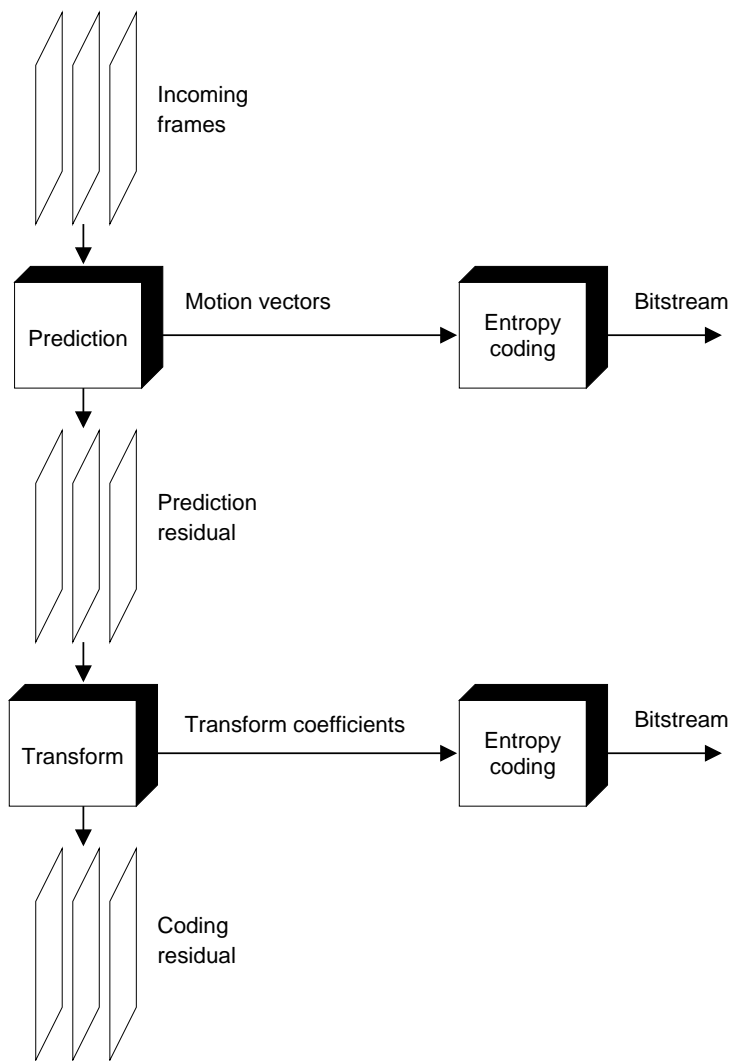


Fig 2. Encoding pipeline.

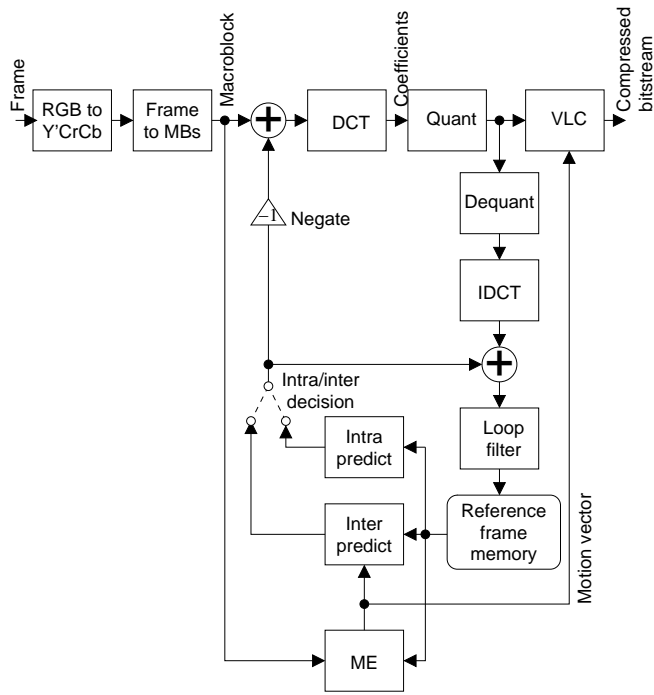


Fig 3. Typical video encoder.

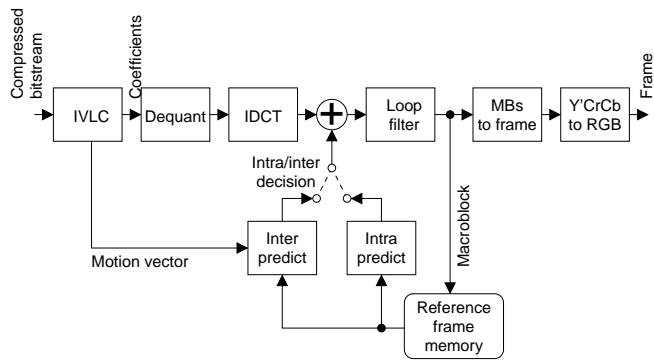


Fig 4. Typical video decoder.

2.1.1 Color space transform

Camera sensors contain photocells which convert light intensity into voltage. To recover colors, there are three different kind of cells, with three different absorption spectrums called red, green, and blue. However, the correlation between the different spectrums (or color channels) is very high. If each of the channels were to be encoded independently, a large part of the information would be encoded three times. Also, since human vision system (HVS) is much more sensitive to details in brightness than in color, the original color space is converted into $Y'C_bC_r$ as follows [7]:

$$\begin{bmatrix} Y' \\ C_r \\ C_b \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix}, \quad (1)$$

where R' , G' , and B' are the non-linear gamma-compressed red, green, and blue color components of pixels. The conversion coefficients are defined in the ITU-R BT.601 standard. Y' is luma and C_b and C_r represent the chroma or color part of the pixel [8]. The color components are usually downsampled by two or four (4:2:2 or 4:2:0 color sampling), and they can be quantized more (leaving a larger residual) than the luma part. Thus, while plain color space transform alone does not reduce information, it allows easily 75% compression of the color information (50% of the total amount of video) and more efficient compression using later coding tools. The amount of data required for representing the chroma part is small compared to the luma part, and often only the luma part is considered when designing encoding methods, although the same methods are used for compressing both luma and chroma.

After the color space transform, the video frames are subdivided into 16×16 -pixel blocks called macroblocks. Macroblocks are encoded relatively independently, although they can be predicted from previously encoded parts of the video. Newer standards allow subdividing macroblocks into smaller rectangular blocks called partitions (following the terminology from the H.264 standard), down to 8×8 pixels in MPEG-4 or 4×4 pixels in the latest H.264 standard [9].

2.1.2 Motion estimation and compensation

Most macroblocks can be inter-predicted using motion compensation from previously encoded (and decoded) reference frames, to allow reconstruction of the images at the decoder side. Fig. 5 shows the principle of motion estimation and compensation. It is assumed that under motion, images of an object move translationally. In other words,

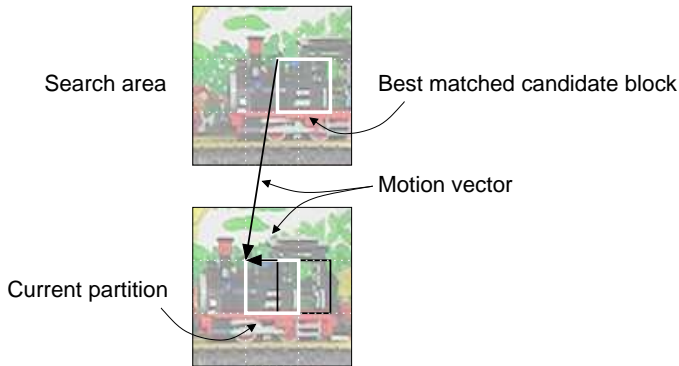


Fig 5. Objective of block motion estimation.

rotation and perspective deformations are ignored. Another assumption is that whole macroblock or partition belongs to a single object. Although this model is inexact, it holds relatively well for small image regions. The motion vector describes the spatial difference between the current partition and the corresponding block in the reference frame, and it is obtained using a motion estimation algorithm. In Chapter 4, most common motion estimation algorithms will be classified.

In newer standards there may be multiple reference frames: H.264 allows at most 16 reference frames, and a motion vector points to one of them. A single partition can be also predicted from two reference frames simultaneously, in which case the two predictions are averaged. Typically the temporally previous and the following frames are used, and the method is called bidirectional prediction. In this case, the frames have to be sent to the decoder out-of-order, so that the decoder has both frames available for bidirectional prediction.

In the first standards, the motion vector accuracy was a single pixel, which means that the translation is assumed to be an integer number of pixels horizontally and vertically. In the newer standards, half-pixel accuracy can be used. If a motion vector points to a half-pixel location, in most standards the predicted pixels are obtained using (bi)linear interpolation from the nearest pixels. In the latest standard H.264, motion vectors have a quarter pixel accuracy. In this case, the luma values at half-pixel positions are obtained using six-tap interpolation from luma values at integer pixel locations and pixel values at quarter pixel accurate locations are obtained using (bi)linear interpolation from the nearest pixels at half or integer pixel locations. The chroma values are always interpolated linearly. Note that in the 4:2:0 subsampled chroma planes, one eighth sample accuracy is needed. Efficient half pixel accurate motion estimation meth-

ods will be discussed in Section 4.4.

In motion compensation, the predicted partition is subtracted from the original partition and the difference is sent to a transform-based coding tool. To reduce the number of bits needed for representing the motion vectors, the motion vectors are predicted from neighboring previously estimated motion vectors, and only the difference from the prediction is transmitted. In the latest standards, the motion vectors are predicted using a median calculated from horizontal and vertical components of motion vectors of the neighboring partitions left of, above, and to the top right of the current partition.

Some standards allow global motion estimation and compensation with more complex motion models than just translation, but in this thesis we will consider only translational block motion estimation with a block size of 16×16 pixels or less.

2.1.3 Intra prediction

In older standards, intra prediction (prediction of sample values in a partition using previously encoded samples in the same frame) was not used, except for transform coefficients, but in the H.264 standard, there are various alternatives for intra prediction. The standard defines four modes for predicting the sample values for the whole macroblock: the pixels can be extrapolated vertically from pixels above the current macroblock, horizontally from the macroblock to the left, as an average of all neighboring pixels, or using a plane prediction function. Alternatively, at highly textured regions, the macroblocks can be subdivided into sixteen 4×4 -pixel partitions, which are predicted from the pixels at the left and above of the partition using nine different modes [9].

The encoder may select any of the prediction modes, but it should select the mode which gives the least rate-distortion cost, as described in Chapter 3. The selected mode number must be transmitted to the decoder. To reduce the number of required bits, also the mode number is predicted from neighboring partitions. Fast methods for estimating the cost and selecting the mode have been one active area of video coding research.

2.1.4 Transform coding and quantization

The residual of a partition, after either intra prediction or motion compensation, is next transform coded. Most older standards subdivide the 16×16 -pixel macroblocks into four 8×8 pixel blocks, and apply a discrete cosine transform (DCT) to each:

$$F(u, v) = \frac{2}{N} C(u) C(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \cos \left[\frac{(2x+1)u\pi}{2N} \right] \cos \left[\frac{(2y+1)v\pi}{2N} \right] \quad (2)$$

$$C(w) = \begin{cases} \frac{1}{\sqrt{2}}, & w = 0 \\ 1, & w \neq 0 \end{cases} \quad (3)$$

where $N = 8$, $f(x, y)$ is the original image block, and $F(u, v)$ is the transformed image block with $0 \leq u, v \leq 7$. DCT is a linear transform and can be written in matrix form. However, since DCT uses a transcendental function \cos , which has to be approximated causing rounding errors, the newest standard H.264 uses instead a DCT-like integer approximation which can be carried out exactly, without rounding errors. Motion compensation is more accurate in H.264 (due to the smaller partition sizes and quarter pixel accuracy), the residual is smaller, and there is less spatial correlation within the residual frame. Therefore H.264 allows also subdividing a macroblock into smaller partitions and using shorter 4×4 -pixel transforms [9].

If the transform coefficients after transform are less correlated than the pixel values or the residual difference before transform, quantization of the coefficients may lead to data compression [10]: just a few coefficients will suffice for representing the original data quite well, and most coefficients can be quantized very roughly or even discarded completely. The amount of quantization is adjustable in all standards, and in fact, it is the main method for controlling the resulting video bit rate. Various rate-distortion optimization methods (see Chapter 3) have been developed for selecting the quantizer and even the actual value of the coefficients.

2.1.5 Entropy coding

The video coding standards use various entropy coding methods. Common to all of them is that they represent often appearing codes and syntax elements using short bit strings, and rare codes using longer bit strings. Entropy coding is an exact compression method: the bit rate is reduced, but no extra distortion is introduced to the video (assuming error-free data transmission).

A preprocessing step for quantized transform coefficients in all standards is zig-zag scanning: since the probability of non-zero elements is highest in the upper-left corner of the transformed block (where u and v in Eq. (2) are near zero), with the probability decreasing to the lower-right corner, the block of coefficients is reordered into a one-dimensional array so that the coefficients at the upper-left corner are first and the coefficients at the lower-right corner are last in the array. This usually orders zeros in the block into long sequential runs, which can be encoded efficiently with run-length coding.

In more recent standards, the run-length codes and coefficients are further com-

pressed using Huffman or arithmetic coding. These coding methods may be used for compressing also other syntax elements. H.264 uses more efficient versions of the coding methods called context-based adaptive variable length coding (CAVLC) and context-based adaptive binary arithmetic coding (CABAC, available in the main profile) [9]. The encoder may use either of the two methods for encoding: CAVLC is less efficient, but requires less computation power. H.264 encodes certain syntax elements, such as motion vectors, using exponential Golomb (exp-Golomb) codes.

2.1.6 Loop filter

Since the video coding standards employ block-based coding tools, there are often discontinuities at the block edges, called blocking artifacts, that are very visible to the HVS. Consider, for example, that the motion vectors of neighboring blocks are different. At high compression ratios, when the residual after motion compensation is encoded roughly, this will almost inevitably lead to visible edges between the blocks. In the H.263 standard extension, the overlapped block motion estimation is presented, which reduces the problem. However, the block transform may also cause blocking artifacts. Consider a smoothly varying flat image region, such as a skyview. Due to the smoothness, only the first coefficient (average luma value of a block) might be preserved in a block, and even it would be quantized strongly. Even a single step difference in the quantized scale might cause largely different average values for the neighboring blocks during decoding, leading to strong blocking artifacts.

The H.264 (and H.263 in its extensions) represent a loop filter to reduce the blocking artifacts. The standard defines a method for a decoder (and encoder) to select the filter strength depending on the expected strength of the artifact at block edges. After the strength is chosen, the pixels at the edges of the blocks are filtered. The filtering is performed before the frame is used in motion compensation: thus, motion compensation uses filtered frames with higher quality. Decoders for older standards excluding loop filter can also apply a filter to reduce blocking artifacts in a postprocessing step, but since the standard forces the motion compensation to use the frames without filtering, this will lead to lower quality.

2.2 Video coding standards

Standards are necessary for people to communicate with each other. As shown in Fig. 1, decoders at both terminals must understand the bitstream which is received over the low bandwidth channel. Standards specify the bitstream syntax and define a hypothetical

decoder. The operation of encoders is not specified, and companies and other parties are free to implement them as they feel best. The operation of decoders is more restricted, but they also have some free design decisions, for example, if postprocessing will be used to improve image quality [11].

The International Telecommunication Union (ITU) is the first of the two major bodies which defines video coding standards. It was originally founded in 1865. ITU is a very large and complex organization. It is divided into several sectors, of which Telecommunication Standardization Sector of ITU (ITU-T), renamed from CCITT in 1993, defines new standards. ITU-T is divided into Study Groups, which are divided into Working Parties. Finally, Working Parties are divided into Questions. The organization changes often, but currently Study Group 16 is called Multimedia Terminals, Systems and Applications and it includes Working Party 3 (Media Coding), which in turn contains Question 6: Video Coding—ITU-T Q.6/SG 16, commonly known as Video Coding Experts Group (VCEG)¹. VCEG was founded in 1997 and it is now responsible for maintaining existing and developing new ITU-T video coding standards, called Recommendations [7, 12, 13, 14]:

- H.120 was the first video coding standard, published in 1984. It includes delta pulse code modulation (DPCM), scalar quantization, and conditional replenishment to decrease the bit rate for still image regions. Version 2, published in 1988, included also frame difference coding, but neither versions are in use today.
- H.261 (published 1991) is the first standard still in common use. It is also the first standard to include a hybrid coding method: first a motion compensation (MC) is performed, and then the residual error is transform coded. The motion vectors and transform coefficients are encoded with a variable length code. The standard is aimed for video conferencing and video phone services over the integrated service digital network (ISDN) with bit rate a multiple of 64 kilobits per second. The newest standards are in many aspects still very similar to H.261: for example, the macroblock size has been kept as 16×16 pixels.
- H.262 (1994) is identical to the ISO/IEC IS 13818-2 (MPEG-2 video). It is designed for television quality video, providing support for interlaced frame formats and much higher bit rates than H.261. H.262 also supports scalability. It includes profiles and levels which allow simple partial implementations of the standard. H.262 is one of the most successful standards and is used in digital video broadcasting (DVB) and Digital Versatile Disks (DVDs).
- H.263 (1996) is based on H.261, but includes several extensions such as half pixel,

¹See <http://www.itu.int/ITU-T/studygroups/com16/sg16-q6.html>.

overlapped block, and bidirectional motion compensation, and a wider variety of available picture sizes. H.263 Version 2 (denoted often as H.263+ and finished in 1998) and H.263 Version 3 (H.263++, 2000) add several new annexes to the original standard: in fact so many that hardly anywhere have all of them been implemented. New features allow, for example, the use of a deblocking filter and transform coefficient prediction in intra blocks.

- H.264 (2003) is also part 10 of ISO/IEC IS 14496 (MPEG-4). It is the latest and tightest video coding standard [6]. H.264 can compress video down to half of the size compared to H.262 at the same quality[15] and is defined as a mandatory video codec in the upcoming video disk standards high definition DVD (HD-DVD) and Blu-ray Disc [16]. The basic structure is still the same as before: images are divided into 16×16 pixel macroblocks, which are optionally motion compensated (inter coding) and transform coded. However, the MC can use a much wider variety of block sizes, from 16×16 pixels down to 4×4 pixels, the transform is an exact 4×4 DCT-like integer transform, the intra prediction is much more advanced than in previous standards, and the variable length coding is much improved.
- H.265 (2008–2010) is the current name for the next standard beyond H.264. Its development has been barely started, so it is too early to guess what the final standard will be like [17], but it should improve the quality (in coding efficiency or otherwise) by a factor of two compared to current standards.

Another two very important entities are the International Electrotechnical Commission (IEC), established in 1906, and the International Organization for Standardization (ISO), founded in 1947. The two organizations are collaborating very closely and have developed their standards together. As with ITU, also ISO/IEC has been divided into numerous smaller groups. The Joint Technical Committee number 1 (JTC 1) deals with information technology issues. Within it, Subcommittee 29 (SC 29) is named for Coding of Audio, Picture, Multimedia and Hypermedia Information, which in turn contains Working Group 11 (WG 11), Coding of Moving Pictures and Audio. The famous ISO/IEC JTC 1/SC 29/WG 11 group, better known as Moving Picture Experts Group (MPEG), was established in 1988.

ISO/IEC and ITU-T collaborate frequently to create common video coding standards. All newer standards from both organizations are very similar, some standards even word for word. In 2002, the common efforts were moved into a newly formed group, the Joint Video Team (JVT). It was established to combine the development efforts of both MPEG and VCEG and to finish the development of H.264/IS 14496-10.

ISO creates documents called International Standards (IS), and for video coding there are

- IS 11172-2: MPEG-1 (1991) includes most of the features of H.261, but also some new ones like half pixel accurate motion vectors and bidirectionally predicted B-frames. On the other hand, the loop filter was removed. The designed operational bit rate is at about 1.5 megabits per second: MPEG's standards are usually aimed at a higher video quality and rate than ITU-T's standards.
- IS 13818-2: MPEG-2 (1994) is very similar to MPEG-1. The same standard is also known as H.262 (see above).
- IS 14496-2: MPEG-4 Version 1 (1999) is a large deviation from previous standards. While earlier standards concentrated mainly in improving coding efficiency, MPEG-4 brings semantic significance into video coding. Pictures may be constructed from several video objects corresponding to real objects such as people, cars, and so on. Each object is encoded independently from other objects, which are then composed to form the final picture. MPEG-4 also supports synthetic video, by allowing inserting still textures and animated meshes into the bitstream. However, since even automatic segmentation of natural images is extremely difficult, these advanced features are quite rarely used. For practical video coding, new features include the possibility to subdivide 16×16 pixel macroblocks down to 8×8 blocks for motion compensation, very long motion vectors, global motion compensation, and quarter pixel accurate motion compensation. MPEG-4 also defines profiles and levels, similarly to MPEG-2. Version 2 was approved in 2000. However, new versions do not add new coding tools into existing profiles, and thus they will be completely backwards compatible.
- IS 14496-10: MPEG-4 Advanced Video Coding (AVC, 2003) is textually the same as the ITU-T H.264 standard. Noteworthy is that H.264 concentrates again on high coding efficiency, leaving out most of the semantic aspects that were found in IS 14496-2.

There are also some proprietary video codecs (encoder and decoder pairs) which use more or less closed algorithms. However, typically these are quite similar to the general block-based hybrid coding, as used in all of the standards.

3 Video coding rate-distortion optimization

As described in the previous chapter, a video coder consists of a set of adjustable coding tools working in conjunction. The operation mode of the tools should be adjusted so that for a given maximum bit rate, the video quality should be maximized. Alternatively, for a given quality the bit rate should be minimized. To do so, we must first specify a method for measuring video bit rate and a quality. Then we can derive an optimization procedure that tries to achieve the objective.

An optimal procedure is of little use, if it requires too much computation in practice. Therefore, a third aspect to consider is the computational complexity [10]. A simple algorithm, which always finds the optimum in finite discrete cases, is the exhaustive search: try all coding parameters and select the best. However, in most cases this is a far too burdensome method and faster algorithms are necessary. Fast but optimal algorithms have been devised, but by giving up slightly on rate-distortion performance, the complexity can be usually drastically reduced.

In this thesis, we consider the case where the desired average bit rate is given. In situations where there is a strict limit on the bit rate, and only a limited buffer available, the average bit rate within the buffer must not exceed the limit [18]. In general, we ignore this risk and assume that the buffer is sufficiently large and a higher level control algorithm takes care of buffer overflows. This is a very typical case, since usually any unused buffer space after coding a part of a video can be used for coding the next part of the video, and the amount of unused buffer space may fluctuate [19]. Risk of buffer overflow can be then minimized by allocating sufficiently large buffer and letting the unused portion of the buffer to fluctuate as much as necessary, depending on how large variation there is on the encoded video instantaneous bit rate.

Fig. 6 represents video coding. The video \mathbf{F} is divided into coding units $[\mathbf{f}_1, \dots, \mathbf{f}_N]$, which could be frames, macroblocks, or something else [20]. For each unit there is a single adjustable parameter or a parameter vector q_n which adjusts the R-D tradeoff for the coding unit. The parameter can obtain only a finite number of possible values: $q_n \in \mathcal{Q}$. For example, in the H.264 standard, the quantization parameter (QP) is an integer between 0 and 51.

The encoder decides the coding parameters q_n hierarchically based on some other coding parameter Q higher in the hierarchy. Usually the coding parameters Q and q_n are discrete scalars: for example, at the top of the hierarchy, Q could be the target bit rate which could be used to decide also the frame rate [21]. A frame-level bit allocation algorithm can then decide the QP for the individual frames. Usually frame-level QPs

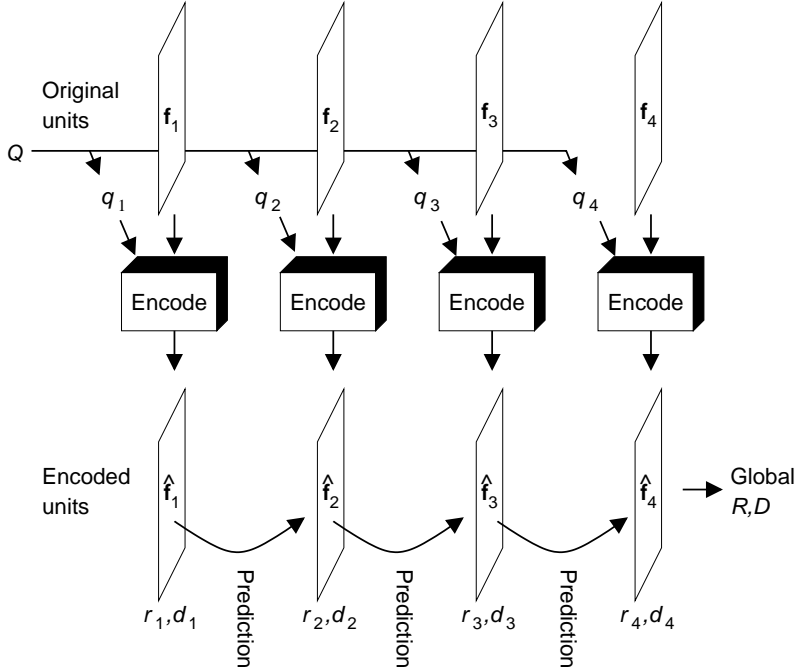


Fig 6. Coding video as a sequence of coding units.

only adjust the macroblock-level coding parameters [22], but they could be used also to decide the frame spatial resolution [21]. The frames are subdivided down to macroblocks, and the macroblock-level QP is then decided for each macroblock [23, 24]. Given the QP for each macroblock, the coding mode, motion vector, and transform coefficients can be then decided.

The bit rate and distortion can be measured individually for each coding unit as (r_n, d_n) , but the actual quantities to optimize are the global bit rate R and distortion D , which are obtained by adding the corresponding quantities from each unit together:

$$R = \sum_{n=1}^N r_n [\mathbf{f}_1, \dots, \mathbf{f}_n, q_1, \dots, q_n] \quad (4)$$

$$D = \sum_{n=1}^N d_n \left[\mathbf{f}_n, \hat{\mathbf{f}}_n \left(\mathbf{f}_n, \hat{\mathbf{f}}_1, \dots, \hat{\mathbf{f}}_{n-1}, q_1, \dots, q_n \right) \right]. \quad (5)$$

Variable $\hat{\mathbf{f}}_n$ denotes the reconstructed coding unit after it has been encoded and decoded to approximate the original unit. Each coding unit may be predicted from the

previously encoded units: for example, motion compensation predicts pixels using previously transmitted frames. Since, at the lowest level in the hierarchy, the coding units are transmitted to the decoder as coding parameters, the decoder may also directly use previously transmitted parameters for prediction. For example, the motion vector of a partition is usually predicted as a median of three previously transmitted motion vectors. Thus, $\hat{\mathbf{f}}_n$ does not only depend on \mathbf{f}_n and q_n , but potentially also on all of the previously coded units $\hat{\mathbf{f}}_1, \dots, \hat{\mathbf{f}}_{n-1}$ and their coding parameters q_1, \dots, q_{n-1} . Occasionally, we may ignore this dependency and implement a locally optimal optimization procedure; however, a globally optimal procedure must take this dependency into consideration.

In this thesis, we assume that both bit rate and distortion are additive. For bit rate this always holds in practice, and it is also true for the most common distortion measures, including sum of absolute differences (SAD) and sum of squared differences (SSD). The methods for measuring bit rate and distortion are discussed in the next section.

3.1 Rate and distortion measures

The bit rate denotes the number of bits used for transmitting a certain amount of video source. A typical unit is bits per second, but in this thesis we prefer to use bits per pixel, because the bit rate is then less sensitive to frame rate and image size. For example, a video with a frame size of 352×288 pixels and 25 frames per second contains 2 534 400 pixels per second. At the typical rate of one megabit per second, there would on average be 0.39 bits available per pixel after encoding. In this section, we assume that the original uncompressed video data are in $Y'CbCr$ color space with 4:2:0-sampling: the two chroma planes are downsampled by two (see Subsection 2.1.1). A pixel contains 12 bits: 8 bits for luma and 4 bits for chroma, on average. Thus, in this case, the video would need to be compressed 30-fold.

The calculation of the distortion of a coding unit is a much more complex subject. First, in general we need to have also the original video sequence data corresponding to the coding unit for which the distortion is measured, in addition to the reconstructed data from a decoder. Then, there are several fundamentally different methods for measuring the distortion. One of the most common and simplest measures is the sum of the absolute differences (SAD) between the distorted and the original coding unit [11]:

$$SAD(\mathbf{f}, \hat{\mathbf{f}}) = \sum_{p \in \mathcal{P}} |\hat{f}_p - f_p| \quad (6)$$

where f_p is a luma or chroma scalar value at location p . The set \mathcal{P} contains the positions of all values which contribute to the distortion. Since the human eye is relatively insensitive to color information, the distortion is often measured using only the luma

component.

Another simple and popular distortion measure is the sum of squared differences (SSD) [11, 25]:

$$\text{SSD}(\mathbf{f}, \hat{\mathbf{f}}) = \sum_{p \in \mathcal{P}} (\hat{f}_p - f_p)^2. \quad (7)$$

SSD differs from SAD in that it gives more weight for large differences. SSD requires a multiplication per difference, and is therefore computationally more burdensome than SAD. SSD also requires a larger numerical range, because while the differences for 8-bit values are in $0 \dots 255$, the squares are in $0 \dots 65025$. However, SSD is mathematically easier to handle as there is no absolute value, and it also corresponds directly to the mean squared error (MSE) [11],

$$\text{MSE}(\mathbf{f}, \hat{\mathbf{f}}) = \frac{1}{|\mathcal{P}|} \text{SSD}(\mathbf{f}, \hat{\mathbf{f}}) \quad (8)$$

where $|\mathcal{P}|$ is the number of scalar values in \mathbf{f} , and to the peak signal-to-noise power ratio (PSNR) [11], which is the most commonly used measure of distortion in benchmarks:

$$\text{PSNR}(\mathbf{f}, \hat{\mathbf{f}}) = 10 \times \log_{10} \frac{255^2}{\text{MSE}(\mathbf{f}, \hat{\mathbf{f}})} \text{ dB}. \quad (9)$$

For larger distortion, PSNR is smaller. Neither MSE nor PSNR are strictly additive, but the latter is often used as if it were: many video encoders report average PSNR by summing the PSNR from different frames and dividing by the number of frames [26]. Although the resulting average PSNR is often near the real PSNR, the difference should be kept in mind and average PSNRs must not be compared directly to real PSNRs.

There are also many other distortion measures. For example, in the minmax optimization [27, 28, 29], the criterion to minimize is the maximum distortion:

$$\text{MAX}(\mathbf{f}, \hat{\mathbf{f}}) = \max \left\{ \left| \hat{f}_p - f_p \right| \mid p \in \mathcal{P} \right\}. \quad (10)$$

However, the maximum distortion is not additive and Eq. (5) does not hold. The minimization of the maximum error requires a different optimization procedure to the minimization of an additive criterion [29]. In particular, the Lagrangian relaxation method presented in Subsection 3.3.2 cannot be used in minmax optimization. Since we assume additive distortion metric in this thesis, we do not consider minmax measure further.

The ultimate truth that we seek is how do people perceive the image quality. Thus, one would tend to arrange a subjective testing event, where the distorted and possibly

original images are shown to people who are asked to judge the image quality. The average of the answers is called the mean opinion score (MOS), and it should be the best possible quality judgement [30]. However, the exact test situation arrangement makes a significant contribution to the results. Although there are standards giving test situations, there are multiple different variants of them, such as single stimulus continuous quality evaluation (SSCQE), double stimulus continuous quality scale (DSCQS) [31], and double stimulus impairment scale (DSIS), and no general agreement which is the best set up. Most importantly, however, as the methods need people for assessment, they are very slow and expensive to conduct, and totally unsuitable for real-time R-D optimization.

Many objective image and video quality assessment methods have been developed that try to match human judgement [23]. These can be classified into full reference, reduced reference, and no reference methods, depending on whether the original undistorted image or video is available, only some features are calculated from it, or none at all. These methods try to model the human vision system (HVS) at some level. Most older methods are based on a bottom-up approach, and they try to model HVS at the early processing stages, which includes effects such as HVS frequency response, texture masking, motion strength, center of fovea on an image, and the nonlinear relationship between luminance and perceived brightness. These methods try to estimate the visibility of the error signal — the difference between the original and distorted image. A newer class of methods use a top-bottom approach, taking into account that the function of the human brain is to recognize objects, and thus structural information in images is very important. The methods, such as the structural similarity index method (SSIM), try to extract structural information from images that are independent of luminance and contrast and then compare the distortion of this information [32].

However, in benchmarks including many of the objective HVS-based quality metrics, the methods have not shown a statistically significant improvement over simple PSNR and MSE-based metrics (including also SSD), under strict testing conditions and different image distortion environments [33, 30, 34]. On the other hand, it has been reported that MSE-based measures give useful results when images with the same type of degradation are compared. As MSE and SSD are easy to evaluate and mathematically tractable, and the distorted images which we compare are always distorted due to the same coding tools used in video encoders, in this work we use the MSE and SSD, along with their close relative SAD for computational reasons. For the algorithms presented in this thesis, the HVS could be easily taken into account by weighting the SSD

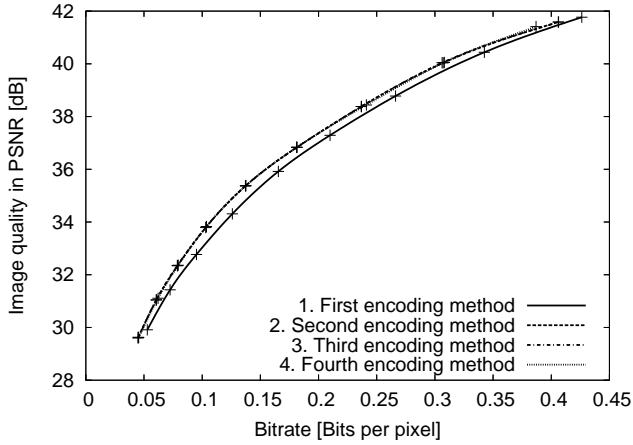


Fig 7. Example of four R-D curves.

measure subjectively as follows [20, 35, 36, 32]:

$$\text{WSSD}(\mathbf{f}, \hat{\mathbf{f}}) = \frac{|\mathcal{P}|}{\sum_{p \in \mathcal{P}} w_p} \sum_{p \in \mathcal{P}} w_p (\hat{f}_p - f_p)^2 \quad (11)$$

where w_p is a weight for a luma or chroma value at location p . For example, distortions in image regions containing a complex texture at high contrast are less perceivable than distortions in flat regions, and the distortions in these locations could be weighted with a smaller weight [23].

3.2 Benchmarking video encoding methods

When a coding control parameter goes through all possible values, for example, when the QP is swept from the minimum to the maximum, the produced rate and distortion pairs for a coded unit produce the operational R-D curve, as shown in Fig. 7. In a standards-defined coding environment, video encoding can be considered to be the selection of coding parameters. To compare different methods for selecting the parameters, we can encode a video sequence using the methods, subject to various rate constraints, and then plot the resulting R-D curves, as in Fig. 7. The most efficient method gives a curve which is closest to the upper-left corner of the graph. Although this is a widely used benchmarking method [37], it has several flaws:

1. It is difficult to say exactly how much one given method is better than an other,

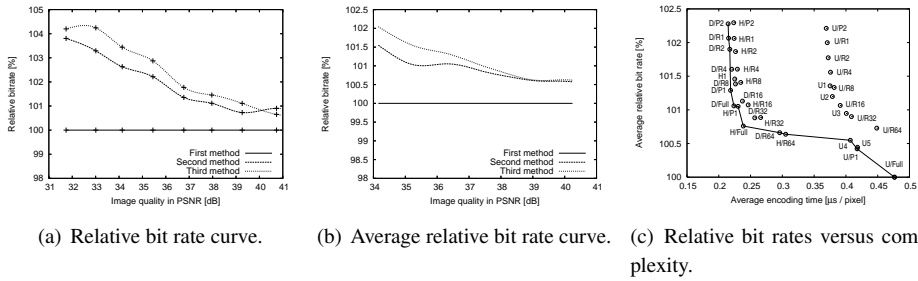


Fig 8. Relative bit rate curves.

because there is no simple numerical measure.

2. The differences in the curves may be quite small, so by simply looking at a graphical representation it may be difficult to judge the methods at all.
3. Different video sequences produce different R-D curves, which makes it difficult to find the typical efficiency of a method.
4. Computational complexity is not considered and it is not visible from the R-D curve.

Bjontegaard suggested that the area between the curves would be integrated [38] and the average PSNR or bit rate change would be calculated from the area. However, Bjontegaard does not propose how the results from several sequences could be combined or how the computational complexity would be represented together with the coding efficiency.

In this thesis, the experiments have been mainly conducted by encoding several video sequences with an H.264 encoder. The sequences are mostly standard sequences used widely in the video coding community, including Carphone, Claire, Container, Grandma, Miss America, Mother and Daughter, News, Salesman, Silent, Suzie, Trevor, Foreman, Mobile and Calendar, Munchener Hall, Paris, Stefan, Tempete, and Tourists in San Francisco. Nine different QP values were used between 20 and 36. The benchmarking method includes the following steps:

1. The R-D points corresponding to the QP values are generated by running the encoder for each video sequence and the computational complexity (for example, the encoding time or arithmetic operation count) is saved. This data can be directly used for plotting R-D curves similar to Fig. 7.
2. One of the tested methods is chosen to be the baseline method, to which all the other methods are compared. This is usually the original unmodified encoder.
3. A cubic spline interpolating the R-D points is constructed corresponding to each of the tested methods, giving the approximate bit rate for any PSNR value. Then, for

each R-D point of the baseline method, the corresponding bit rate of a tested method at the same PSNR is obtained by evaluating the spline at that PSNR value. The relative bit rate change is calculated by dividing the bit rate of the tested method by the rate of the baseline method. When the results are plotted, a relative bit rate graph similar to Fig. 8a is obtained for each video sequence. The relative bit rate of the baseline method is always 100%.

4. A cubic spline is constructed through all the (PSNR, relative bit rate) points and averaged over the video sequences. A typical resulting curve is plotted in Fig. 8b, which gives a very reliable and intuitive picture of the efficiency of the different encoding methods at different picture quality levels.
5. When there are dozens of different methods, a graphical comparison is not possible in practice. In this case, the averaged relative bit rate curves are integrated over the PSNR range of interest. This gives a single value, relative bit rate averaged over the different sequences and quality levels, for each encoding method. Based on this value, the methods can be easily sorted by efficiency.
6. When also the computational complexity is of interest, the average encoding performance is calculated for each method by simply averaging the performance obtained for each R-D point over all of the video sequences. The complexity of the encoding methods can be plotted into a (computational complexity, average relative bit rate) graph, as shown in Fig. 8c. A solid line is drawn through the hull of the set of methods. The methods residing at the line are the best in efficiency for some maximum amount of computation.

These steps for benchmarking are used in Papers II, V, and in particular, Paper III.

There is no single method for measuring the performance: one could count simple arithmetic operations, but this would ignore the implementation details, such as different execution time of different kinds of operations and number of memory accesses. Or, one could measure the execution time of an actual implementation of an algorithm, but the acquired result will be strongly dependent on implementation details and underlying hardware. In this thesis, we use both measuring methods, depending on whether the research is more theoretical or practical.

3.3 Optimal bit allocation

3.3.1 Independent coding units

Very often there is a strict limit R_{\max} on the average bit rate, which may not be exceeded. In this case, video coding is a constrained discrete optimization problem: find

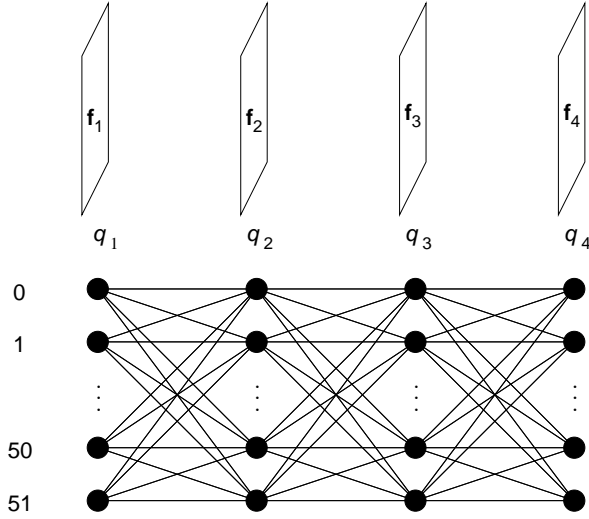


Fig 9. Trellis of all possible quantization choices.

the optimum coding parameter vector $\mathbf{q}^* = [q_1^* \ \cdots \ q_N^*]$ that minimizes the coded video distortion [20]:

$$\mathbf{q}^* = \arg \min_{\mathbf{q}} D(\mathbf{F}, \mathbf{q}) \quad (12)$$

$$\text{subject to } R(\mathbf{F}, \mathbf{q}) \leq R_{\max}. \quad (13)$$

Put in other words, the problem is to find optimal allocation of the available bits among the coding units.

The problem can be formulated as a shortest path search through a trellis [20, 21]. Consider Fig. 9, where there are four frames and a QP between 0 and 51 must be chosen to each. The simplest, but computationally most burdensome, method for solving the problem is to try all possible coding parameter vectors \mathbf{q} , reject those which exceed the bit rate constraint, and select the one with the least distortion [20]. However, this would require going over $|\mathcal{Q}|^N$ different combinations, something which is impossible but for a very small number N of coding units. Clearly, more efficient algorithms are needed.

Dynamic programming can be also used to find the optimal path through the trellis. After each coding unit, all existing paths leading there are pruned as long as it still maintains optimality. For example, paths that would exceed the rate constraint or paths that have a higher rate and distortion than some other paths, are pruned. This maintains optimality when the coding units are encoded independently and saves computation by discarding possible paths through the trellis as soon as possible [20]. The rate-

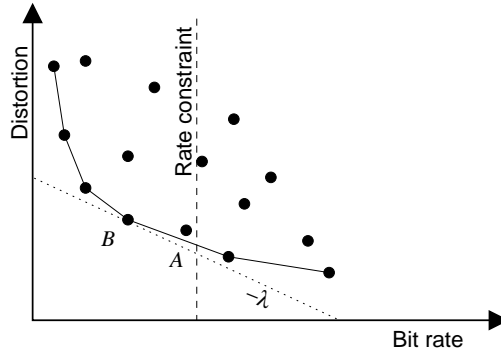


Fig 10. Lagrangian relaxation.

distortion pairs for different QPs for the frames need to be gathered only once for the whole sequence, and they can be used when evaluating the cost of the different paths. This limits the maximum number of rate-distortion pairs to be gathered to $N|\mathcal{Q}|$ in the independent case and saves a huge amount of computation, since calculating the rate and distortion typically requires actually encoding the coding unit [20, 35]. The search for the shortest path may still require much computation, and by giving up the requirement of a strictly optimal solution, more paths can be pruned and computation saved [21].

3.3.2 Lagrangian relaxation

A classical approach for solving continuous constrained optimization problems is the Lagrangian relaxation. Everett [39] introduced this method also to discrete optimization, and nowadays it is the most widely used technique in video coding [25, 20, 40]. The theorem states that Eqs. (12)–(13) are equivalent to

$$\mathbf{q}^* = \arg \min_{\mathbf{q}} [D(\mathbf{F}, \mathbf{q}) + \lambda R(\mathbf{F}, \mathbf{q})] \quad (14)$$

when $R(\mathbf{F}, \mathbf{q}) = R_{\max}$. The Lagrangian multiplier λ selects the tradeoff between the distortion and rate. The obtained video quality is optimal for the obtained rate. However, there is a problem: when the bit rate is the known constraint, it is not straightforward to know which value of λ corresponds to it giving the desired bit rate (or slightly less). It is also possible that for the optimal \mathbf{q} there is no λ at all, and the optimization algorithm would return a suboptimal solution.

The graphical interpretation of optimization with the Lagrangian technique is given

in Fig. 10. The points represent different achievable rate-distortion points that are obtained by different coding parameter vectors. The optimal point—the point with the least distortion of the points with a rate less than the constraint—would be point *A*. However, Lagrangian optimization can return only points that lie on the convex hull of the points, that is, some point that lies on the solid line in the figure. Therefore, the optimal parameter vector in the Lagrangian sense is point *B*. The Lagrangian multiplier is set to a negative of the slope at some location on the convex hull of the rate-distortion curve [41]. One can imagine Lagrangian optimization so that a line with slope $-\lambda$ is drawn through every possible point and the point producing the bottommost line is chosen, as it gives the least Lagrangian cost in Eq. (14). If there are many points in the rate-distortion graph, the point obtained with Lagrangian optimization is very close to the global optimum. However, if the graph is very sparse, it may be necessary to instead use dynamic programming or some other optimization technique.

The real value of Lagrangian optimization is attained in the independent case when Eqs. (4) and (5) are substituted into Eq. (14) giving

$$\mathbf{q}^* = \arg \min_{\mathbf{q}} \left[\sum_{n=1}^N d_n \left(\mathbf{f}_n, \hat{\mathbf{f}}_n \left(\hat{\mathbf{f}}_1, \dots, \hat{\mathbf{f}}_{n-1}, \mathbf{f}_n, q_1, \dots, q_n \right) \right) + \lambda \sum_{n=1}^N r_n \left(\hat{\mathbf{f}}_1, \dots, \hat{\mathbf{f}}_n, q_1, \dots, q_n \right) \right] \quad (15)$$

$$= \arg \min_{\mathbf{q}} \left[\sum_{n=1}^N d_n \left(\mathbf{f}_n, \hat{\mathbf{f}}_n \left(\mathbf{f}_n, q_n \right) \right) + \lambda \sum_{n=1}^N r_n \left(\hat{\mathbf{f}}_n, q_n \right) \right] \quad (16)$$

$$= \left\{ \arg \min_{q_n} \left[d_n \left(\mathbf{f}_n, \hat{\mathbf{f}}_n \left(\mathbf{f}_n, q_n \right) \right) + \lambda r_n \left(\hat{\mathbf{f}}_n, q_n \right) \right] \middle| n = 1, \dots, N \right\}. \quad (17)$$

Now, instead of optimizing all components of \mathbf{q} jointly, each component q_n can be optimized one-by-one [25]. For example, if an exhaustive search is used, the complexity is dropped, in many cases dramatically, from $|\mathcal{Q}|^N$ down to $N|\mathcal{Q}|$. However, since the set of possible rate-distortion points for each single coding unit is much less than for all of the coding units together, the possibility of the optimal point not being on the convex hull increases, and the resulting distortion is more likely not to be exactly optimal. Nevertheless, in many cases the result is still good enough, and for the reduced computation, definitely worth it.

The result in Eqs. (15)–(17) can be explained accessibly by noting that at the optimal solution, the chosen point at the rate-distortion curve of each coding unit is at the equal slope of the curve. If this would not be the case, bits could be moved from coding one unit into coding another unit giving smaller Lagrangian cost and a better result [20].

Lagrangian cost can be used together with a dynamic programming solution by choosing a suitable λ and setting the penalty to paths in a trellis to the Lagrangian cost instead of only to the distortion [24]. In this case, pruning by the maximum bit rate is not necessary, which may simplify the algorithm in some cases. This is a commonly used approach in the dependent coding case discussed in the next subsection.

The remaining problem is the choice of λ , when only the maximum rate (or distortion) constraint is given. The rate $R(\lambda)$ as a function of λ is monotonic and usually quite regular. Thus, the equation $R(\lambda) = R_{\max}$ can be solved approximately quite easily using, for example, the bisection algorithm [42, 36] or some other fast convex search algorithm [20, 24, 27]. This would require running the optimization algorithm multiple times for evaluating $R(\lambda)$. Another possibility is to estimate the correct value of λ based on previously encoded coding units [20, 42, 26], where the desired bit rate can be calculated by a rate control algorithm based on free buffer space [11, 42]. One possibility is to simply specify λ as a fixed value instead of R or D , when there is no strict constraint on rate or distortion, as, for example, in some video storage applications. Wiegand and Girod [41, 11] fixed QP for frames (as it has been conventionally selected heuristically without using λ) and derived experimentally a formula giving the corresponding value for λ .

Traditionally, rate control algorithms have kept the video bit rate constant by adjusting the QP in a video coder. When Lagrangian optimization methods were introduced, the value for λ was derived from the current QP value by using an experimental formula. Then, many other coding parameters are derived from the approximate value of λ . This is the most common implementation in video encoders of today; however, if the rate control algorithm would adjust λ directly, and all coding parameters, including QP, would be derived directly from it, the encoder would be conceptually more straightforward and there would be less approximations [42].

3.3.3 *Dependent coding units*

When there is dependency between the coding units and it is taken into account, the optimization problem becomes much harder. An example is presented: all modern video coders use previous frames for predicting the current frame with motion compensation. In a P-frame, a single previous frame is used for prediction, in B-frames, two previously encoded frames are used for prediction. H.264 allows long-term frame memory and the compensation can be done using frames up to 16 frames back. Another example is the prediction of a motion vector for a macroblock using the median motion vector calculated from three neighboring macroblocks, or the prediction of DCT-coefficients, as in

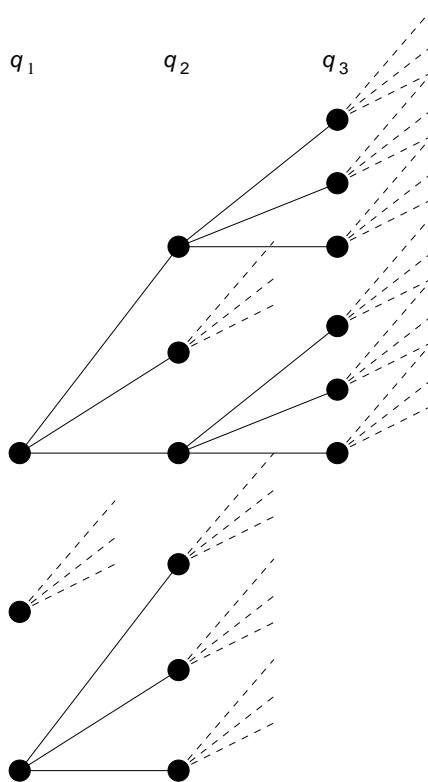


Fig 11. Dependent trellis.

H.263, or prediction of pixel values of intra-coded blocks from neighboring blocks, as in H.264.

The optimal solution also to the dependent case can be found by formulating a trellis and finding the optimal path using dynamic programming [20, 26]. However, unlike in the independent case, in general there are no common rate-distortion points along different paths of the trellis, as shown in Fig. 11. The figure shows only a few branches for clarity. To reduce computation to a practical load, one possibility is to use some optimization algorithm for an independent case and either ignore the dependencies or to take them into account through some heuristic methods [11, 19]. However, better results are obtained by taking the dependencies directly into account in the problem formulation. In this case, dynamic programming needs to prune a very large part of the trellis branches to avoid or at least limit the exponential growth of the trellis [20, 42, 22]. Ramchandran *et al.* [35] introduced and verified experimentally the monotonicity

principle, which states that if a coding unit is encoded with lesser distortion (or lesser Lagrangian cost), that will also reduce the total distortion (or Lagrangian cost).

In some cases there is a limited dependency. For example, MPEG-2 video has been typically divided into a group of pictures (GOP) structure having 15 frames of the following type: IBBPBBPBBPBBPBB. The intra frame (I) repeats after every 15 frames, breaking the frame prediction dependency. Furthermore, each predicted frame (P) is predicted only from other P or I frames, not from bidirectionally predicted B-frames, so the exponential prediction depth contains only 5 frames. Ramchandran *et al.* [35] show how this special case can be handled efficiently using dynamic programming.

In [43] the motion compensation is modeled using matrix operations, which allows formulating a quadratic program for the Lagrangian cost of a video as a function of transform coefficients. The motion estimation is performed using only the original uncompressed frames, which makes them fixed during the R-D optimization. The optimal solution for the quadratic program can be then calculated using well established algorithms [44]. The problem with the approach is that due to the required strong assumptions the optimization of the parameters for many other coding tools (such as intra-block coding modes and selected partition sizes) are difficult to integrate into the solving algorithm, and an approximate solution must be used for them.

In dynamic programming approaches, some assumptions must be made on dependencies of the video encoder, to stop fully exponential growth of computation. This makes dynamic programming solutions dependent on the specific video coding method. It is also difficult to deduce the average or the worst case complexity of a dynamic programming algorithm. In [45], an optimization method called the quadratic coordinate-wise non-backtracking steepest descent (QCWSD) was presented. It is shown that QCWSD gives a globally optimal solution if cross-over, cross-over ordering, and reachability conditions hold. Although these conditions do not hold in general for real video sequences, the result given by QCWSD is still very near the optimal. QCWSD works as follows:

1. Set the initial quantization vector to the maximum value (for example, with H.264 $\mathbf{q} = \begin{bmatrix} 51 & \dots & 51 \end{bmatrix}$).
2. Calculate slope $\Delta D/\Delta R$ with respect to each q_n . This is implemented by:
 - a) Loop over $n = 1, \dots, N$
 - b) Encode the video to obtain the rate and distortion for parameter vector \mathbf{q}' which is equal to \mathbf{q} , but with parameter q_n decreased by one.
3. Select the QP $q_{n_{\max}}$ for which $-\Delta D/\Delta R$ is largest; that is, the parameter which gives the greatest distortion reduction for the least increase in bit rate, and which does not

exceed the given maximum bit rate.

4. Repeat from step 2 until the maximum bit rate has been reached.

The worst case complexity (number of times the sequence needs to be encoded) of QCWSD is $|\mathcal{Q}|N^2$ (where we assume unlimited dependency). Although not exponential, the computation load is still too large in most practical situations, especially with real-time encoders. In [46], the authors suggest that step 2 of the QCWSD algorithm is modified to update the slope only for that QP which has changed. However, as we show in Paper I, this reduces the resulting quality significantly.

3.4 Proposed new optimization methods

3.4.1 *Rotation search*

In Paper I, we devise a fast optimization algorithm called a rotation search (RS) for functions taking multidimensional discrete parameters. The optimization algorithm can be used to optimize any discrete parameter vector and as such it can be used for R-D optimization of QPs in a video coding system using the constraintless Lagrangian cost function 14. The encoder can be handled as a black box; that is, any kind of assumptions on the encoder internal tools need not to be made. Therefore, the method is suitable for optimization of any past or future video encoder with any kind of dependencies, and it guarantees finding the local optimum. In our experiments we used an H.264 video encoder. In most cases in our experiments the local optimum was very near the global optimum. Although [36] gives also many optimization algorithms, the RS is designed specifically to find the local minimum with very small number of function evaluations. Furthermore, RS does not require function derivatives, which would need to be approximated by evaluating the function around the derivation point: for N dimensional signal it would take at least N function evaluations to obtain simple first-order derivatives, which is already very burdensome and would not change the QPs at all.

The algorithm rotates cyclically over the N dimensions, and for each dimension, the direction (forward or backward) towards the local optimum (that is, the sign of the partial derivative) is guessed based on the direction at the last round for the same dimension. A step into this direction is then taken blindly and the function value is evaluated. If the guess was incorrect, a step back and then in to the opposite direction is made. If also this direction leads away from the local optimum, a step back is again made to the original point and the function is at the local optimum at this dimension. When a local optimum in respect to all dimensions has been found, the search terminates. A great amount of computation is saved, in addition to avoiding the computation of par-

tial derivatives and guessing the direction of local optimum, also by guessing the search initial position. A good initial position is important also because otherwise the algorithm might stuck on some local optimum far from the global optimum [36]. In R-D optimization, the initial position is obtained by assuming that the QPs in all frames are equal. This assumptions converts the N dimensional R-D function into one dimensional and allows the use of the efficient golden section search.

The experimental results are shown in Paper I. With 100 frame video sequences, the RS algorithm is around two orders faster than QCWSD while giving in practice equivalent R-D performance. While the computational complexity of QCWSD is quadratic in the number of frames (as the name implies), the RS is nearly linear, as observed from the experiments. Thus, with longer video sequences, the RS has even greater advantage in speed. Nevertheless, the RS is still not sufficiently fast for real-time encoding, because the video needs to be encoded multiple times. It can still be used for off-line video coding, when there are no real-time requirements, and for benchmarking the degree of suboptimality of other video coding algorithms. The computational performance of the RS could be improved significantly by employing models predicting the rate and distortion of frames from the given QP values, without actually encoding the frames. Some R-D models have been presented in literature [26, 23, 22], however, in general they do not consider sufficiently framewise dependencies, in particular in the framework of the newest H.264 standard. Quality-wise, the RS even slightly surpasses the near-optimal QCWSD algorithm in some cases. This can be explained that the conditions assumed by the QCWSD do not exactly hold.

3.4.2 *Faster real-time optimization algorithm*

From experiments it has been observed that those coding units, which are used for predicting many other units directly or indirectly via other coding units, should be coded with smaller distortion than those units which are not used for prediction [47, 48]. For example, B-frames were not used for predicting other frames before H.264 standard. Thus, in earlier standards they should be quantized more strongly than P- and I-frames [49]. Similarly, as I-frames are used indirectly for predicting all the remaining frames until the next intra frame, they should be coded with the highest quality [48]. This is easily understandable, because the bits are used most profitably when they are spent for improving the quality of key frames which improve maximally the quality of other frames. When the frames are more similar to each other, the prediction works better, and the bits should be allocated more unequally to the key frames.

In real-time encoders, the decision about the strength of the quantization is made

commonly by adding or subtracting an ad hoc fixed offset to or from the QP used in the P-frames [48, 49]. This method avoids using the time consuming trellis search or some other optimal algorithm. However, the resulting video quality at the obtained bit rate is significantly worse than what would be obtained by using a near-optimal algorithm. Hence, in Paper II, a new fast method is developed for choosing the offset automatically: every other frame is assumed to be a “keyframe”, and for them the QP is kept at its original value. For the remaining frames, the QP is increased by an offset, which is selected by minimizing the Lagrangian cost of the frame. Thus, the method assumes a short dependency, where each frame depends only on the previous frame. Although this is an idealization, in practice we still obtain significantly better video quality than when using an ad hoc offset.

The original QP is obtained from the rate control algorithm, or it can be set by the user, as in our experiments. The Lagrangian cost to optimize is $D(\mathbf{f}, q) + \lambda R(\mathbf{f}, q)$, where \mathbf{f} is the frame and q is the QP to optimize. Since we optimize only one frame (with one QP value) at once, the search for the optimal q can be made with any fast convex search, as in [23]. Furthermore, the optimal value is predicted using the previously found value. In our algorithm, we simply increase or decrease the predicted QP until a local minimum is found. Since the Lagrangian cost function is convex, the local optimum is also a global optimum. In practice, the optimal QP is higher than the original parameter, and therefore we call the optimized frames “reduced quality frames”. The Lagrangian multiplier is approximated using the function $\lambda = 2^{\alpha q_0 + \beta}$ where q_0 is the original QP used for the non-reduced quality frames, and α and β are obtained off-line using a least squares fit.

In our work, the method is applied only to P-frames, but it could be also used to optimize the QP offset used in B-frames. However, this was not done to remain compatible with lower profiles which do not contain B-frames and to simplify experiments: we used only P-frames in the encoded video sequences. The experimental results show significantly improved rate-distortion behavior of the encoder. Depending on the target rate, a 2–5% reduction is obtained in the bit rate at the same encoded video quality as in the original unmodified video encoder.

In the work described in Paper II, there is also a novel application for the new features of the H.264 standard, which allow long term frame memory and reordering of reference frames: in the usual case with P-frames, each frame is predicted from the previous P-frame. However, since in this case a non-reduced quality frame is predicted from the previous frame, which is a reduced quality frame, the prediction will not work very well. This situation can be remedied by using at least two previous frames for reference. Furthermore, on many occasions the result is improved even further, when

the two previous reference frames are reordered, so that the frame giving a better prediction is first in the list. In our work, the decision as to whether to reorder the two frames in the reference frame list is done using the same principle as when finding the optimal QP offset: the Lagrangian cost for both cases is computed by encoding a frame in both cases for obtaining the rate and distortion, and the alternative with smaller cost is selected.

The computation necessary for the method is significantly lower than in most other R-D optimization methods presented in this chapter for several reasons: first, the algorithm uses a greedy approach, and optimizes the quantization of one frame at once. This allows optimizing just one QP with a fast convex search, in which the initial position is predicted, which typically requires trying just a few different QPs values. Secondly, only every second frame is optimized, which approximately halves the computation load. Even with these features, the algorithm is still significantly slower than other heuristic algorithms used in traditional video encoders. Thus, a fast version of the algorithm was introduced, in which the optimization algorithm is run only very rarely (in our experiments, even only once in 250 frames proved to deliver quite good results). The QP offset found by the algorithm is then applied as such to every other frame until the algorithm is run next time, after tens of frames. With this modification, the algorithm computational complexity can be dropped to be only 3% more than in traditional encoders, while the bit rate is still reduced by around 3–5% with the same image quality.

3.5 Summary

In this chapter, we first noted that R-D optimization of video encoding requires exact metrics for bit rate and video quality, which form the objective function to optimize. For the bit rate, the metric is easy to form, although it can be expressed in several different units. For video quality, the ultimate metric, in principle, would correspond to the judgement of human viewers, but even if human viewers were available, there is no single generally accepted procedure for quality assessment. Furthermore, in real-time video coding, the assessment must be automatic. Several different quality measurement methods have been presented, but none of them have been as successful as simple sum of squared differences (SSD), the related peak signal-to-noise power ratio (PSNR), or sum of absolute differences (SAD), which are simple, mathematically tractable, and computationally relatively efficient. Thus, in the experiments, we have used these three measures for video quality.

We then introduced several optimal R-D optimization algorithms for an independent case, where the rate and distortion of each coding unit depends only on the QP applied.

We also discussed the more complex dependent case, in which case each coding unit is predicted using previously encoded coding units. Many optimization algorithms are based on finding the shortest path through a trellis. However, the complexity of these algorithms depends on the assumptions which are made to prune the trellis branches, and the complexity may in the worst case be exponential in the number of frames. Thus, the steepest descent family of optimization algorithms have been developed, which achieve practically optimal results in quadratic time.

Faster algorithms are nevertheless required in many cases, and we introduced the rotation search based on a Lagrangian cost function, which finds the locally optimal coding parameters in a minimal number of steps. Rotation search is a generic search algorithms and it handles the video encoder as a black box. This makes it independent of the encoding algorithm, unlike many other R-D optimization algorithms. The rotation search does not need function derivatives, which makes it most attractive for discrete high-dimensional optimization problems. Although the computational complexity of the rotation search is approximately linear in the problem size and it is significantly faster than the other general R-D optimization algorithms tested in our benchmarks, it is still not capable working in real-time.

For real-time encoding, we presented a fast greedy optimization method for frame-level QP values. In the method a Lagrangian-based optimization algorithm is at most applied to half of the frames. The scheme takes advantage of the fact that high-quality frames are good predictors for the following frames. The following frames can be quantized strongly, using only a small amount of bits for them, while still maintaining good video quality due to the prediction. The method gives the best advantage with the H.264 standard, since in that case several previous frames can be used for prediction and high-quality frames can be predicted from other high-quality frames instead of the previous lower quality frame. The algorithm is significant because it takes into account frame-wise dependencies while still being reasonably fast.

Several fast R-D optimization algorithms were introduced in this chapter, and all of the algorithms require the gathering of several R-D points, which in practice means that the encoding process must be run several times. This increases much the computation load compared to any method that could select a proper coding mode without the R-D data. One efficient method of decreasing the load is to model the encoding process and to predict approximately the rate and distortion of a coding unit when given the QP, without actually performing the encoding. Several different R-D models have been developed [26, 23, 19, 41, 22], some of them are more accurate requiring more computation, and some faster and less accurate. When the assumed model is simple enough, the optimal solution for coding parameters could be derived analytically [27],

avoiding searching completely. Although the models could improve the speed of any R-D optimization method, they are beyond the scope of this thesis.

4 Fast block motion estimation

Motion estimation and compensation are two of the most important parts of a typical video encoder, as discussed in Subsection 2.1.2. Motion compensation gives a large reduction in bit rate, because the residual is typically very small. On the other hand, motion estimation requires a large part of an encoder computation. With the latest standards, such as H.264, the macroblocks have many alternative modes. Only motion vectors for the chosen mode are sent to a decoder, but the encoder may need to perform motion estimation in all modes to be able to make the mode decision. Thus, fast motion estimation algorithms are becoming increasingly more important.

The objective of motion estimation for a given coding standard is quite clear, taking into consideration the general idea of rate-distortion optimization from Chapter 3. It is different from motion estimation in computer vision applications: in video encoding we do not need to estimate the real motion from a video sequence, but instead we try to improve the encoder rate-distortion behavior. Thus, in video coding the objective function to optimize is the rate-distortion behavior of the video encoder—but it is not clear how to find the optimum or near-optimum with an affordable amount of computation [50, 51, 52, 53]. However, in practice the calculated motion vectors correspond very accurately to the object motions in the video, as shown in Fig. 5.

Typically all new encoders apply Lagrangian relaxation, given in Eq. (14), into macroblock encoding and motion estimation. For any chosen motion vector (MV), the corresponding Lagrangian cost could be obtained by encoding and decoding the partition using the MV. This would give the number of required bits and the distortion between the encoded and the original partition [53]. However, evaluating this matching criterion would be extremely slow. In practice, all encoders evaluate an approximative criterion instead of the exact Lagrangian cost. Most encoders optimize a function similar to Eq. (14), but with the coding distortion D replaced with the distortion between the current partition and the reference partition, and the bit rate R replaced with the bit rate required for encoding the motion vector only. The distortion term then gives also the approximate number of bits required for coding the transform coefficients [54]. The chosen distortion measure is usually the sum of the absolute differences instead of a mean squared error for efficiency reasons; occasionally also the sum of the absolute Hadamard transformed differences (SATD) is employed to give a more accurate estimate of the rate term [49].

Motion estimation is usually performed in two steps. First an integer accurate MV is searched, and then it is refined into half (and possibly quarter) pixel accuracy with a

local search. In this thesis, we mainly focus on integer pixel accurate motion estimation, in Sections 4.1 and 4.3, because it is usually computationally more demanding. For example, if the MV range is 16 pixels, there are $33 \times 33 = 1089$ different candidate integer MVs, but only eight half pixel MVs around the best integer MV. Nevertheless, we also consider a fast half and quarter pixel search in Section 4.4.

Mathematically, the motion estimation can be represented as follows:

$$(m_x^*, m_y^*) = \arg \min_{(m_x, m_y)} C(m_x, m_y) \quad (18)$$

$$C(m_x, m_y; \lambda_{mv}) = \text{SAD}(m_x, m_y) + \lambda_{mv} R_{mv}(m_x, m_y) \quad (19)$$

$$\text{SAD}(m_x, m_y) = \sum_{y=0}^{Y-1} \sum_{x=0}^{X-1} \left| \hat{f}^j(m_x + x, m_y + y) - f(x, y) \right| \quad (20)$$

where (m_x^*, m_y^*) is the optimum MV in the sense that it minimizes Eq. (19), $R_{mv}(m_x, m_y)$ is the number of bits required for encoding MV (m_x, m_y) , taking into consideration the MV predictor, and $\text{SAD}(m_x, m_y)$ is the sum of the absolute differences between the current partition luma values $f(x, y)$ and the reference partition at MV location (m_x, m_y) luma values $\hat{f}^j(x + m_x, y + m_y)$.

The MV rate R_{mv} can be obtained efficiently from a look-up table, and it is the SAD term in Eq. (19) which consumes by far the greatest part of the computation. There are basically three different levels at which the motion estimation performance can be optimized. At the highest level, the motion estimation may search only a small subspace of all candidate motion vectors. This has been the most successful approach, and it is discussed in Section 4.1. Another approach is to approximate the SAD function defined in Eq. (20) by reducing the number of absolute differences, that is, subsampling the partitions, as discussed in Subsection 4.2.1. At the lowest level, the absolute difference in Eq. (20) can be computed approximately. This approach will be discussed in Subsection 4.2.2. In general, the motion estimation performance can and should be optimized at all three levels. A few algorithms utilize directly several levels: one class includes the fast correlation based methods. A fast motion estimation based on correlation computation via number theoretic transforms was developed for this thesis and it will be discussed in Section 5.4. Hierarchical motion estimation methods utilize several different motion estimation algorithms, either sequentially or recursively. These methods are discussed in Section 4.3.

4.1 Fast search strategies

Fast search strategies selectively check only a minor part of all candidate MVs. In the literature, three different methods have been presented for designing a fast search strategy: MV prediction, fast search patterns, and halfway-stopping. The methods complement each other and should be used together in an efficient motion estimation strategy.

4.1.1 Motion vector prediction

The first method tries to predict the motion vector based on previously computed MVs. An important feature of the prediction is that it does not require evaluation of the matching criterion in Eq. (19) and is therefore computationally a very fast part of the whole motion estimation algorithm. The prediction process returns one or several [55, 56] motion vectors, which are called predicted MVs. Prediction was first used in block motion estimation in [57, 58], where it was formulated quite generically: the predicted MV was a weighted sum of previously estimated MVs in the same frame (spatial prediction, [57, 55]) or a weighted sum of MVs in the previous frame (temporal prediction, [58]).

However, in practical implementation, a much smaller set of MVs is usually used for prediction. In [57], the authors used spatial prediction for the first frame by first estimating MV for one quarter (or for one half, as in [59]) of the partitions without prediction, and then predicting the MV for the rest of the partitions from the nearest partition for which ME was performed in the first step. For later frames, temporal prediction was used by obtaining the predictor from the co-located partition in the previous frame. Many other early papers simply took the previous estimated MV as a predictor [60, 61], but soon it was noticed that the MV median of the MVs in the three nearest partitions, left, above, and above right, is a more robust predictor [62]. This scheme is also used in newer video encoding standards to predict MV at the decoder side, allowing reduction of the MV bit rate [63] (see Subsection 2.1.2).

More complex temporal prediction schemes were developed later. If the object, into which the collocated partition belongs, is accelerating, a better predictor may be computed by extrapolating the MVs from the collocated partitions temporally in the two previous frames [66]. On the other hand, if the current partition is near the front edge of a fast moving object, the collocated partition in the previous frame might belong to a still background. This problem can be taken into account by projecting the partitions from the previous frames into the current frame by their MVs, and calculating which partitions correspond to the current partition [67].

The predicted MV or MVs may be also complicated functions of the previous MVs

[67]. For example, Tourapis formulates the predictor MV as a parameterized function of some fixed set of previously computed MVs and optimizes the free parameters by minimizing the entropy of the difference between the estimated MVs and their predictors [65]. The most common MVs used for calculating the predicted MVs include the MV of the partition at the top left of the current partition, the same MVs as in the median prediction, and the MV of the colocated partition in the previous frame [68, 65]. Also the zero motion vector should be used as a predictor [61], as background is very often motionless.

Newer standards, such as MPEG-4 and H.264, allow a subdivision of larger partitions into smaller ones. In this case, the encoder would first estimate the motion for larger partitions and then subdivide them into smaller partitions. The motion estimation for the smaller partitions could utilize the corresponding larger partition MVs in the prediction process [66].

Although the predicted MVs are often not optimal, they do give a very good starting point for the subsequent search with very little computation. This allows using highly local and efficient motion estimation patterns in the following steps.

4.1.2 Fast search patterns

The core of a fast search strategy applies a sequence of carefully designed search patterns at the current search position. The pattern specifies the candidate MVs around the current position, in which the matching criterion is evaluated and compared to the best previous match. If a new best match so far is found during a round, the center position of the pattern in the next round can be moved at it. The search pattern is typically fixed for simplicity, but it might also change adaptively for each partition [69] or even after each round [70].

Most fast search patterns require a single search center position, from which the search begins. Without prediction, the search could be performed starting from the zero MV, but this would be inefficient. One of the predictors can be chosen as the initial position by evaluating the matching criterion in Eq. (19) at all predicted MVs and choosing the MV giving the best result [56]. In this case, the initial search pattern can be thought to consist of all the predictors.

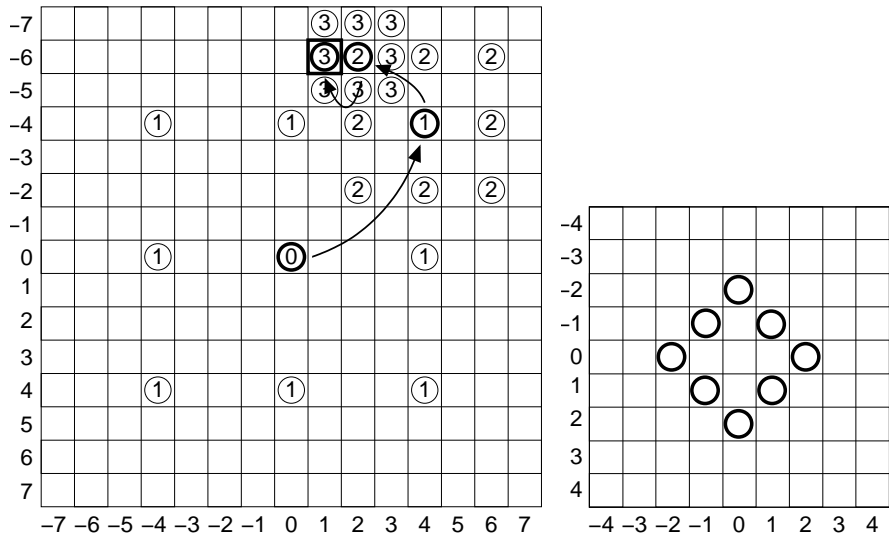
The most straightforward approach is to pick a search area size, often so that the maximum motion vector length horizontally or vertically will not exceed 16 pixels (due to limitations in older video coding standards), and then to simply evaluate the matching criterion at every candidate MV position in the search area and choose the MV giving the best match. This is called a full search (FS). In modern standards, the MV range is

often unrestricted, and the FS will actually check only a small part of all possible MVs. In addition, the FS is usually too slow in practice, or if the search range is very small, the resulting MVs are not adequate. Thus, numerous fast search patterns have been developed. The simplest patterns subsample the search space regularly [71] or limit the search area adaptively based on, for example, MV lengths of neighboring partitions [60, 69].

Some other early search patterns include the 2-D logarithmic search [72], the three step search (TSS), and the cross search algorithm [73]. These algorithms have been designed to find a global optimum of the matching criterion with a minimum number of rounds, as long as the matching criterion function monotonically increases from the minimum [72]. These algorithms differ mainly in that they give a slightly different balance between computation and quality.

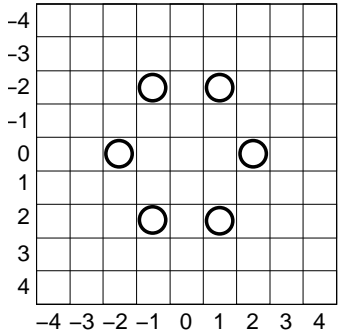
TSS, shown in Fig. 12a, is explained here as a typical example of the first generation motion search patterns. In the beginning, the criterion is evaluated at the center of the search, denoted with a zero enclosed in a circle in the figure. In each of the subsequent three rounds, eight points are checked. In round one, the candidate MVs at positions $(\pm 4, \pm 4)$, $(0, \pm 4)$, and $(\pm 4, 0)$ are checked. The best of the checked positions is chosen as the center for the pattern in the next round. It is similar to the first pattern, but half of the size and thus the checked points are $(\pm 2, \pm 2)$, $(0, \pm 2)$, and $(\pm 2, 0)$ around the new center. The center is again moved to the best match. In the final round, the nearest eight points around the center are checked. The final best position is then returned as the motion vector. In total, the criterion needs to be evaluated at 25 positions. The FS with the same MV range would need to check 225 points, and compared to this, TSS saves 89% of the computation. It is easy to generalize TSS for a larger search range, in which case the savings would be even more. However, TSS as well as most of the other early patterns often fail miserably, because if the matching criterion is not monotonic, even the first step could be taken in a wrong direction, and the end result would be very far from the optimum.

Some research was made to improve the early algorithms so that they would not so easily lead into completely wrong direction. For example, in [74] the TSS is modified so that if the two best matches in one round are nearly equal, the search avoids converging too rapidly into the better one. It was noticed [75, 76] that most video sequences have very high probability for the optimum MV to be near the original search center. This is most true for video conferencing applications or head-and-shoulders sequences with little motion, but especially with advanced MV prediction schemes, holds also for videos with strong motions. The first patterns developed based on this principle are the new three step search [75] and the four step search algorithms [77]. Motion vectors also

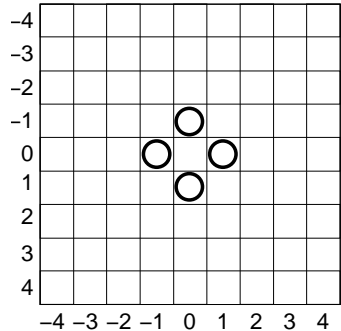


(a) Three step search example.

(b) Large diamond search pattern.



(c) Hexagon-based search pattern.



(d) Small diamond search pattern.

Fig 12. Various search patterns.

tend to be more often horizontal or vertical than diagonal, which leads to using a cross (or plus) shaped pattern [76] in the first round.

Newer developments have used even more localized patterns, such as the diamond search (DS, Fig. 12b) [78, 79], the hexagon-based search (HEXBS, Fig. 12c) [80], the block-based gradient descent search (3×3 block pattern) [62], or even the small diamond search (SDS, Fig. 12d) [81]. These algorithms successively apply the same pattern, moving the search center after each round to the new best position, stopping typically only when a local minimum is found. Alternatively, Tourapis has suggested using zonal patterns, where the search center does not move, but instead the pattern size increases after each round, and is either circular [61, 82] or diamond shaped [83]. Tourapis suggests also applying the zonal patterns to multiple predictor MVs, calling it a radar technique. In extreme cases, in each round only one new position is checked, as in [68] and in the rotation search (RS) presented in Paper I. Being highly localized, these search patterns rely on efficient MV predictors and on the motion vector field being not too chaotic so that the predictors are accurate.

When it is desirable to obtain the highest possible quality motion compensation, the local search patterns fail even with the most sophisticated predictors. In this case, there are no other alternatives to applying larger patterns that are less localized. The FS remains one possibility. Other more esoteric algorithms are, for example, the genetic motion search algorithm [84], which is based on principles from genetic algorithms. A significant new search strategy accepted into the H.264 reference encoder, which is nearly equally good to the FS but much faster, is called unsymmetric-cross multi-hexagon-grid search (UMHexagonS) [85]. This algorithm applies a series of both local and larger patterns. Although the applied patterns require many matching criterion evaluations, the algorithm can be significantly speeded up by applying subsampling techniques, as discussed in Section 4.3 and in Paper VI.

4.1.3 Fast end condition

The first block motion estimation algorithms, such as the TSS and the 2-D logarithmic search [72], terminated after a fixed number of rounds. However, sometimes during the search it is possible to guess that there will not be significantly better matches (MVs with lower criterion values from Eq. (19)) than the best match found so far. If this condition can be detected automatically, it would save all of the remaining computation without degrading the motion estimation result. On the other hand, if the condition is detected incorrectly, the MV will be less accurate, degrading the rate-distortion performance of the encoder. Thus, research has been carried out to find end conditions which

save as much computation as possible without degrading quality too much.

The most obvious and common end condition is to stop when a local minimum is found: when the matching criterion has a better value at an MV than at any other MV around it, the search terminates. The local minimum end condition is generally used by all new local gradient descent type search patterns, such as the DS [79], the HEXBS, [80], the block-based gradient descent search [62], and the SDS [81]. The disadvantage is that there are often local minimums which are much worse than the global minimum match and the search may terminate prematurely.

Another type of end condition is the use of thresholds: if a match better than the given threshold is found, the search stops. A rarely used alternative is to stop when the improvement of the new best MV is less than a given threshold compared to the best previous match [62]. The first methods simply had a fixed threshold obtained experimentally [64, 56, 61, 82].

Different fast end conditions can be used simultaneously, or combined. For example, [56] and [82] employed another looser threshold, which terminated the search only if the corresponding MV was also a local minimum, or terminated the search only after the first set of the search patterns, correspondingly. There are also wider possibilities in controlling the search patterns adaptively depending on thresholds [65].

Typically thresholds are compared to the matching criterion value, which is an approximation of the Lagrangian cost; that is, the combined function of the rate and distortion corresponding to the MV. However, computational complexity is an important aspect too. In [81], a Lagrangian cost corresponding to a combination of distortion and computational complexity is calculated, and the search is terminated when the cost is too high compared to the cost in the previous round. As the distortion generally increases when the search positions moves further away from the optimum, the threshold and the Lagrangian multiplier provide a control method for the tradeoff between computational complexity and quality in the motion estimation.

However, the problem in the previously mentioned thresholding methods is the selection of suitable thresholds. The threshold should be close to the matching criterion at the best match, so that the best match falls below the threshold, but not at other candidate MVs to avoid a premature end. Similar methods can be used for predicting the criterion value at the best match as for predicting the MVs. Thus, in [69], the threshold is set to the median of the matching criterion of the three neighboring previously processed partitions. In [66], the threshold was set to the minimum of four neighboring partitions (plus a small constant), efficiently avoiding too high threshold values. More complex predictor functions for the threshold were considered in [65].

In general, fast end conditions were left out of the scope of this research. The

methods presented in Papers IV, V, and IX are fast full search algorithms, and fast end conditions would have evidently decreased the quality more or less. In Paper VI, we use a local minimum to terminate the HEXBS step and to terminate whole search in the SDS step. In the original UMHexagonS, there is a threshold used in the beginning of the search to skip whole motion estimation. This thresholding was not included in our experiments: instead it was assumed that the same thresholding would be done for all algorithms before applying the actual motion estimation used in the experiments.

4.1.4 Motion vector refinement

The motion estimation method in Paper III can be considered to be a special search strategy. However, unlike any other known search strategy, the search pattern has been designed based on experimentally obtained error statistics between the optimal motion vector and the motion vector returned from other fast motion estimation algorithms. Any fast motion estimation algorithm can be used in the first step. In the second step, the statistically obtained pattern is employed. An important feature of the algorithm is that it can be adjusted and a suitable tradeoff between computation and MV precision can be easily chosen.

The statistics are first collected offline by obtaining MV estimates for each partition using both the fast algorithm and the “optimal” FS strategy with full matching (that is, not approximating the SAD). Both algorithms aim to minimize the rate-distortion cost, with the FS succeeding significantly better. Thus, for every partition, two MV estimates are obtained. The estimate MV error of the fast algorithm is the difference between the two MVs and the loss of rate-distortion performance is the difference in the Lagrangian cost. For smooth areas, the Lagrangian cost difference could be small, even if the MV difference would be large. On the other hand, even a small error in MV may give significantly worse rate-distortion performance at highly detailed image regions. In any case, the Lagrangian cost for the “optimal” MV is always smaller or equal to the MV obtained with the fast algorithm.

The increase in Lagrangian cost is accumulated separately for each possible error MV. The possible error MVs are then sorted in decreasing error with the error MV associated with the largest average increase in the cost first. As the result, a sorted list of new candidate MVs to check around the final MV obtained with the fast algorithm is obtained. The first candidate MVs in the list are the most probable to give the largest reduction in the Lagrangian cost, when they are checked for the best MV in addition to the estimate from the fast algorithm. In this second step, the full (not approximated) SAD-based Lagrangian cost, as in Eq. (19), is used. The algorithm used during encod-

ing the video contains the following steps:

1. Obtain the initial motion estimate using any fast motion estimation algorithm, employing a fast search strategy and/or fast matching.
2. Check N additional MVs around the initial motion estimate which are most likely to improve the resulting coding performance the most.

Since the number of additional MVs to check can be freely adjusted, the algorithm is easily tuned between fast execution speed or high quality. When the quality is set to very high, corresponding to the FS, a 29% reduction in encoding time can be obtained compared to the FS. In our work, we tested several different fast motion estimation algorithms for the first step. It depends on the desired tradeoff between computation and quality which should be chosen, but SDS followed by a 32-point refinement was one algorithm giving good results. The results are shown in Paper III.

4.2 Fast calculation of a matching criterion

4.2.1 Subsampling the matching criterion

Since Eq. (19) can be considered to be an approximated matching criterion, it makes sense to ask whether the complexity could be further reduced without increasing the rate or distortion too much. A straightforward way is to subsample the matching criterion approximating it, for example [59], with

$$\text{SAD}(m_x, m_y) \approx \widetilde{\text{SAD}}(m_x, m_y) \quad (21)$$

$$= 4 \sum_{y=0}^{Y/2-1} \sum_{x=0}^{X/2-1} \left| \widehat{f}(2x + m_x, 2y + m_y) - f(2x, 2y) \right|. \quad (22)$$

While the original formula required accumulation of $X \times Y$ absolute differences, the subsampled approximation requires only one quarter of them. How well does the new criterion approximate the original? In this example, a regular 1:4 subsampling pattern, called a quarter pattern, is used (Fig. 13a). In natural images, neighboring pixels have high correlation, that is, $f(x, y) \approx f(x + 1, y) \approx f(x, y + 1)$. This is especially true at flat areas, and the higher the correlation, the better approximation the subsampled criterion gives. It is important to notice that only the matching scan is subsampled, not the images themselves. Thus, the number of candidate motion vectors does not change. Subsampling the search space (reducing the number of candidate MVs) was discussed in Section 4.1. In [86], a recursive formula is defined that allows regular subsampling

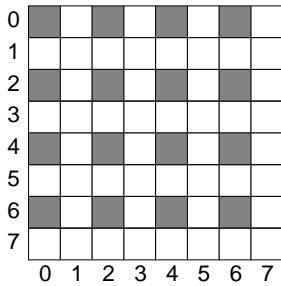
by an arbitrary factor² which includes Eq. (21) as a special case, and in [87] the Sobol pseudorandom sequence is applied to select sampled pixels, which also allows an arbitrary subsampling factor.

However, when there are sharp transitions, such as edges, the approximation is less accurate. Consider images $\hat{f}(x, y) = f(x, y) = 0, x \leq 2, 255$ otherwise. In this case, $\widetilde{\text{SAD}}(0, 0) = \widetilde{\text{SAD}}(1, 0) = 0$ although for SAD the two motion vector candidates have very different values. Different subsampling patterns may give better accuracy in these cases, because they can be designed to sample points at straight edges in all directions. Thus, various subsampling patterns have been developed which are reported to work better than the quarter pattern, while having the same number of sampled points, including the hexagonal pattern (Fig. 13b), the queen-4 pattern (Fig. 13c), and the Yu's pattern [88]. Alternatively, a denser sampling may be used, as in the quincunx pattern (Fig. 13d, [88]), at the cost of twice the amount of operations, or subsampling can be used in a hierarchical fashion, in which a coarse MV is first obtained using a very low sampling factor (such as 1:64 in [89] or 1:32 and 1:8 in [90], or even 1:256 in [91]) which is then refined on the following levels with denser sampling patterns (see Section 4.3 for more discussion about hierarchical algorithms).

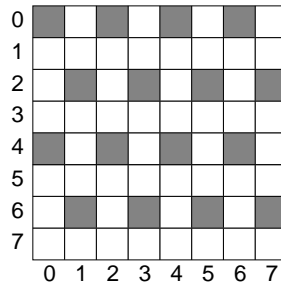
General purpose parallel computer architectures (such as typical modern microprocessors) have often performance loss if non-consecutive pixels are accessed from memory. In these cases, the pixels in the frame memory can be rearranged so that the sampled pixels are consecutive in memory [88], but it requires some extra overhead. Alternatively, subsampling patterns have been designed to contain consecutively sampled pixels such as the P1 (Fig. 13e) and P2 (Fig. 13f) patterns [92].

Instead of using fixed subsampling patterns, the use of adaptive patterns has been suggested. Because the matching errors due to subsampling are caused by sharp edges and transitions, those locations should be sampled with higher resolution. Usually an adaptive algorithm selects the pixels to use in matching before the motion estimation for a partition begins, based on the contents of the current partition. One of the first adaptive subsampling methods was developed in [93], in which the current partitions were first subsampled by 1:9 using a regular pattern, and then more pixels were selected based on thresholding inter-pixel differences: pixels with sufficiently different value from a neighboring pixel were chosen to be used in the matching process. Newer methods include [94], where the authors apply high-pass, sobel, and morphological gradient filters to extract edges and create a mask which disables computation for part of the matching process in dedicated hardware, saving power. In [95], the authors extract

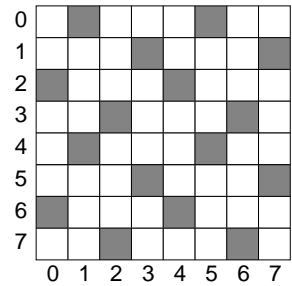
²The recursive formula should be $D^N = \begin{bmatrix} 4D^{N/2} + 0U^{N/2} & 4D^{N/2} + 2U^{N/2} \\ 4D^{N/2} + 3U^{N/2} & 4D^{N/2} + 1U^{N/2} \end{bmatrix}$ in [86].



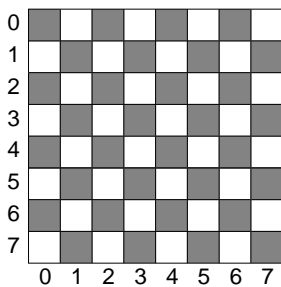
(a) Quarter pattern, 1:4 subsampling.



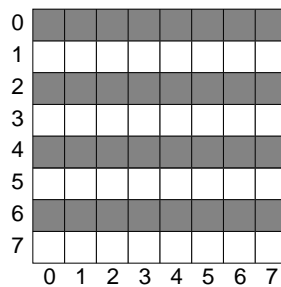
(b) Hexagonal pattern, 1:4 subsampling.



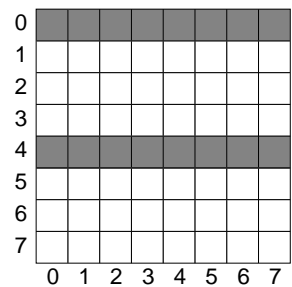
(c) Queen-4 pattern, 1:4 subsampling.



(d) Quincunx pattern, 1:2 subsampling.



(e) P1 pattern for 8×8 partition, 1:2 subsampling.



(f) P2 pattern for 8×8 partition, 1:4 subsampling.

Fig 13. Various matching scan subsampling patterns.

edges by converting the current partition into one dimension using a Hilbert curve scan and then recursively extracting the strongest edges until the desired number of points are selected for the matching scan. Unfortunately, arbitrary subsampling patterns are difficult to implement efficiently. Alternatively, in [96], the authors suggest subdividing the 16×16 pixel partitions into sixteen 4×4 subpartitions. For each subpartition, a suitable subsampling pattern is chosen from 29 predefined patterns based on the current partition content.

Even with clever subsampling patterns, the rate-distortion performance is diminished compared to exact SAD computation due to aliasing effects. The performance can be significantly improved by filtering the images and computing the subsampled SAD from the filtered images. The filter should be designed in conjunction with the subsampling pattern. For example, with the quarter pattern, typically a 2×2 averaging filter is used for fast calculation [97, 98], although a longer filter would give better re-

sults (as in [89], where an averaging 5×5 -tap filter is employed, or in [99] with a 3- or 5-tap separable filter $\{\frac{1}{4} - \frac{a}{2}, \frac{1}{4}, a, \frac{1}{4}, \frac{1}{4} - \frac{a}{2}\}$). With the P1 and P2 patterns, it is plausible that the filter impulse response should be vertical. However, proper filters have been little examined in cases other than with the regular quarter pattern. In any case, filtering the images before motion estimation does require additional computation, and in particular, more memory and memory bandwidth.

Integral projection techniques compute vertical, horizontal or area integrals from both the current and candidate partition, and compare the integrals [100]. These methods can be considered to be subsampling methods with an area filter before the subsampling. In [55], both vertical and horizontal integrals are computed from the whole search area (instead of from each candidate partition). The integrals are normalized and compared both to the horizontal and vertical integrals of the current block. The method is very fast, as only a couple of arithmetic operations are required for each search position, but it is also quite inaccurate.

The successive elimination algorithm (SEA) [101] can be also considered to be a 1:256 (subsampling both vertically and horizontally by the partition size 1:16) subsampling algorithm with a 16×16 -tap averaging filter, although the hierarchical structure and mathematical properties of the averaging guarantee equal rate-distortion performance compared to exact SAD computation. SEA will be discussed in Subsection 4.3.2.

4.2.2 Bit depth transforms

Computing Eq. (20) with partition size 16×16 pixels ($X = 16, Y = 16$) requires accumulating 256 absolute differences. In the previous subsection, methods for reducing the number of absolute differences were discussed. In this subsection, methods for reducing the computation necessary for calculating one absolute difference are reviewed. Almost always images consist of 8-bit pixel values. The image bit depth can be reduced for motion estimation without significantly reducing the rate-distortion performance of the video encoder. The simplest way is to simply discard n least significant bits, and calculate the SAD using only the $8 - n$ most significant bits. With dedicated hardware, this can give significant savings; for example, when $n = 4$, 57% of transistors can be saved, and the resulting motion estimator is 34% faster [102]. The number of discarded bits can also be changed adaptively, with a different number of truncated bits for each macroblock. Although hardware simplification with this method is difficult, it can still lead to power consumption savings due to decreased switching activity in digital circuits. In [103], 70% of power is saved by adaptively changing n , with an average $\bar{n} = 5.7$. The

selection of n is based on a quantization parameter value set by the rate control.

The matching would be most efficient with only one bit, as the absolute difference could be computed with one exclusive or operation. Simply by taking the most significant bit, the quality degradation would be intolerable. However, by using more clever mapping of the images into one bit depth, the quality can be significantly improved. As was noted in Subsection 4.2.1, image edges are important in motion estimation. In [104], the images are first filtered with a 17×17 -tap band-pass filter to detect edges and remove noise, and then the original frame pixel values are thresholded into binary using the corresponding pixel values from the filtered image as thresholds. However, by basing motion estimation solely on one bit image depth, the resulting motion estimate quality will be inevitably significantly worse than with the full eight bits. A few researches [105, 106] have proposed using two bits to represent the original pixel values, where three thresholds are used for quantization. In [105], the absolute differences are computed with two bits while in [106] the error criterion is the sum of two one-bit criteria allowing the use of the efficient exclusive or operation.

Even with two bits, the result tends to be quite poor. In [107], a single bit transform is used, but the two best MVs are chosen. The exact SAD is then computed at both positions and the final best MV is selected. This improves the matching result using a hierarchical search with two different matching criteria. Another, more complex, hierarchical scheme with two different levels is presented in [108], in which the four most significant bits of the pixel values are used on the first level. The result is then refined by applying the FS with a reduced search range, and a full eight bit SAD around the best match from the first level. The full SAD calculation is hastened by taking the four bit SAD into account from the first level. Several other proposed hierarchical search algorithms will be discussed in the next section.

4.3 Hierarchical search algorithms

The term hierarchical search algorithm is used here in a wider sense than usual. Here, all motion estimation algorithms are called hierarchical that can be considered to contain multiple matching criteria, of which some are fast but approximate, and select candidate MVs for one or more later levels, which use slower but more accurate matching criteria. The advantage of a hierarchical algorithm is that a large part of the candidate motion vectors can be rejected based on fast and inaccurate matching criterion, and later levels in the hierarchy with slower matching criteria need to check only a few candidate motion vectors, making ideally the overall algorithm faster, but more accurate than would be possible by using any one matching criterion.

There are two different major types of hierarchical algorithms: sequential and recursive. In the sequential algorithms, one level of hierarchy is finished and motion vectors selected for later levels completely before entering the next level. With recursive algorithms, when a candidate motion vector is visited and not rejected, the next level is applied immediately to the MV before checking the next candidate MV. The sequential methods, discussed in Subsection 4.3.1, allow ease in choosing a different search strategy (Section 4.1) for each level. On the other hand, they require bookkeeping to mark all candidate vectors which are selected for later levels. In the recursive methods (discussed in Subsections 4.3.2–4.3.5), the obtained results from later levels could be used to aid the prior levels to reject a larger part of the candidate motion vectors, such as in the successive elimination algorithm (SEA), and the matching criterion value from earlier levels could be used as a part of the matching criterion at later levels, as in the partial distortion elimination (PDE) algorithm.

4.3.1 Sequential methods

A hierarchical search is an old concept. The most basic way is to first calculate an image pyramid consisting of subsampled images of the original frames at various (typically at 2–4) resolutions [97, 99]. A search strategy is then applied sequentially on each level, starting from the best candidate MV obtained on the previous level. Often the search pattern is simply a subsampled full search, with 3×3 candidate MVs matched at each level. The levels are gone through sequentially and the search range is halved after each level. In this case, both the matching criterion and the search space are subsampled, more on the coarsest levels.

The disadvantage of the basic implementation of the hierarchical search comes from two features: first, because the search space is strongly subsampled on the first levels, the matching criterion minimum could fall in the middle of the checking points, and especially with highly textured blocks with many local minima, another local minimum far from the best MV could be chosen. Secondly, because the matching criterion is also subsampled to only a few pixels (in some papers, to 2×2) the probability of good (at the coarse scale) but false matches increases.

There are several different modifications used to fight against these problems. In [98], the initial hierarchy level is chosen based on image texture complexity, avoiding too large subsampling ratios at complex regions of images. A very popular method to avoid being trapped to a local minimum is to select several MVs to be used on the later levels of the hierarchy instead of just one. In [97], the number of MVs propagated to later levels can be varied, leading to an adjustable tradeoff between computation and

quality. In [90], three levels of hierarchy are used. Different combinations of subsampling for both matching and searching, and for the number of selected candidate MVs on the first two levels are tried. Two good computation-distortion tradeoff alternatives turn out to have 1:64 subsampling with 12 candidate MVs on the first level, then 1:4 subsampling with 2 candidate MVs, or alternatively 1:256 subsampling with 10 candidate MVs on the first level and 1:64 subsampling with 2 candidate MVs on the second.

The problem of too small a number of pixels can be battled against by using larger blocks [97, 55]. However, since the video coding standards define the partition size, the block size on the final level should be equal to the partition size in a standard. Several partitions can still be combined into one larger block on the previous levels. The obtained motion vector from these levels is then used as the initial MV for all of the partitions on the later levels. A slightly different alternative is to use a larger block around the current partition and estimate MV for only part of the partitions on coarse levels; on the last level the block size used for ME equals the partition size and the initial MVs for partitions skipped on the previous levels are interpolated from neighboring partitions [89].

Obviously an easy way to avoid problems is to have only a few hierarchy levels without too excessive subsampling ratios. The famous paper by Liu and Zaccarin [59] has only two hierarchy levels. In the first level, the matching criterion is subsampled by 1:4 using quarter patterns in different phases. The search space is not subsampled at all, and the full search is used. Four candidate MVs are chosen, which are then matched at the last level calculating an exact SAD for each. A similar scheme, but with three levels of hierarchy, is presented in [86]. On the first level, both search space (using 1:4 subsampling) and matching (the best number of selected pixels was found to be 48) are subsampled and the six best MVs are selected. Then, on the second level, still the same subsampled matching criterion is used, but the search pattern is modified to include nine MVs around the best six MVs, and a new set of the six best MVs is chosen. Finally, on the last level, the best MV is chosen among the six candidates using full matching.

4.3.2 Successive elimination algorithm

The recursive hierarchical methods have the advantage that a prior level may select an arbitrary number of motion vectors, but nevertheless, there is no overhead for keeping track of the chosen MVs. A remarkable but simple two-level recursive algorithm called the successive elimination algorithm (SEA) was presented by Li and Salari [101]. The current and reference frames are first filtered with a 16×16 filter consisting of only ones: the filtered current partition and search area contain effectively sum norms of the

current and candidate partitions. The first level of the hierarchy subsamples the SAD matching criterion for a 16×16 partition by 1:256 down to just a single pixel using the filtered frames. If the subsampled matching criterion is sufficiently small, the next level in the hierarchy is immediately entered and the usual non-subsampled SAD of the partitions is calculated and the best MV and the SAD so far are updated, if a new best match is found.

A significant feature of the SEA is that on the first level of the hierarchy, the best SAD of the last level in the hierarchy is used for comparison when deciding whether to descend to the following hierarchy level. It can be proved [101] that the filtered and subsampled matching criterion is a lower bound for the SAD on the last level, and therefore, the algorithm is lossless: the resulting MV would be the same if the first hierarchy level were to be removed completely. Nevertheless, as the last level of the hierarchy is not entered at many candidate MVs, the algorithm is much faster than without the first hierarchy level.

Improvements to SEA modify the first level by adding more computation to it to tighten the lower bound and to decrease the number of candidate MVs visited at the last level. For example, the subsampling factor can be changed to 1:4, 1:16, and 1:64 so that instead of computing the absolute difference of one 16×16 -sized block sum norm pair, the sum of the absolute differences of 64 2×2 , sixteen 4×4 , or four 8×8 -pixel block sums, respectively, are computed [109, 110]. In [111], sixteen 1×16 and 16×1 block norms are used in addition to the 16×16 block norm as in the original SEA. While denser subsampling increases computation on the first hierarchy level, it “eliminates” more candidate MVs and consequently less candidate MVs have to be checked on the last level to guarantee a lossless result.

It is important to understand that if a good MV with a small SAD is found early in the search, the SEA increases the elimination percentage and decreases the overall computation. Thus, although an arbitrary search pattern is possible with SEA, nevertheless the most likely candidate MVs should be tested first [112]. A spiral search around a predicted MV is often employed [110].

Although the original SEA was developed for the SAD matching criterion, it is easy to extend to the rate-distortion criterion in Eq. (19) [113] and to SSD [111, 114, 112]. In [115], the SEA is extended for one-bit matching criterion. Huang *et al.* [116] present a sequential version of SEA. Although it requires more bookkeeping of the chosen MVs, the algorithm is more regular than standard SEA, and therefore may be more suitable for hardware implementation. As the SEA is a hierarchical fast matching method, it can be used with any fast search strategy. For example, in [112], the search space is subsampled by 1:2.

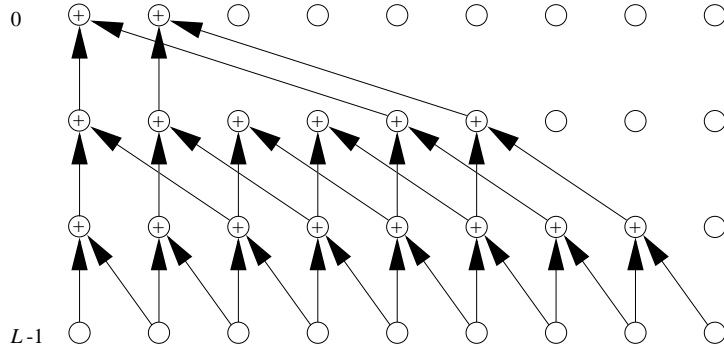


Fig 14. Computing layered image representation.

The partition sum norms can be calculated efficiently using either a sliding window method [100, 101, 117], norm pyramid computation [118, 119, 120, 117], or an integral transform method [121]. The sliding window method requires the least memory, but it is inefficient with the most fast search strategies. Norm pyramid computation (Fig. 14) is an efficient method, and it also allows efficient random access to candidate partition norms allowing the use of any search strategy, but it requires extra memory for storing the norm pyramid. Integral transform requires also extra memory, and a few arithmetic operations for obtaining a partition norm, but it can be used for computing the norm for an arbitrary sized partition, which is useful with the newer standards that include several different partition sizes.

4.3.3 Multilevel successive elimination algorithm

The multilevel SEA (MSEA, or BSPA for the block sum pyramid algorithm) [118, 119] is a recursive hierarchical matching method, which extends the two SEA hierarchies to several hierarchy levels. Since on the first level of the SEA less computation is required with a larger subsampling factor, but less of the candidate MVs get eliminated than with a smaller subsampling factor, on the first level of the MSEA the largest possible subsampling factor (1:256 for a 16×16 -pixel partition) is used, but there are additional hierarchy levels which employ a smaller subsampling factors (1:64, 1:16, and 1:4). The exact SAD is calculated only if the elimination fails at all prior levels.

Since several different filtered images of the partitions have to be available simultaneously, the norm pyramid computation (Fig. 14, see the previous subsection) is the most useful method for efficiently generating the filtered partitions from which the

block norms are obtained.

Improvements into MSEA are presented in [122, 123], where the level of elimination is estimated for some of the candidate MVs; the subsampled matching criterion computation for all prior levels can be then avoided. The papers also employ PDE-type techniques (described in the next subsection) for aborting matching in the middle of a hierarchy level. This can be considered as subdividing MSEA hierarchy levels into finer subhierarchy levels.

4.3.4 Partial distortion elimination

Partial distortion elimination (PDE) methods [112] can be also considered to be recursive hierarchical matching algorithms. In Eq. (20), there is a summation of $X \times Y$ terms. The sum of the n first terms can be considered to be the matching criterion on n :th level of hierarchy. On the last level of the hierarchy, $n = X \times Y$, and the matching criterion is the complete SAD. This hierarchical algorithm has two very useful features: first, the matching criterion for level n is the matching criterion for level $n - 1$ plus some additional terms, making it very efficient for calculating the matching criterion. Second, the matching criterion for any level n is a lower bound for the matching criterion on level $n + 1$. As the SEA, the basic PDE is also a lossless motion estimation algorithm: if the partial sum on any level is greater than the best match on the final level so far, the candidate MV can be immediately rejected. Both the PDE and the SEA can be considered to subsample the matching criterion, but while the SEA uses filtered versions of the images for computing the subsampled criterion, PDE uses the original unfiltered images. In a practical implementation, instead of $X \times Y$ levels of hierarchy, the number of levels is reduced by summing typically 16 (one line of pixels in a partition) absolute differences whenever entering the next level. Similarly to the SEA, a good match should be found early to reduce computation for the later candidate MVs. A spiral search pattern is commonly used also with the PDE.

The PDE can be combined with the SEA, so that the lower bound from the first level of the SEA is the initial value of the SAD calculation on the last level [114, 124]. This algorithm is called extended SEA (ESEA). Since the sum accumulation then does not begin from zero as usual, but from the lower bound, the accumulation value after any number of added terms is higher and the summation can terminate sooner, based on the SAD value of the best previous match. The ESEA has been applied to the MSEA in [123].

Proposed improvements to PDE for decreasing computation, while maintaining the same quality, modify the order in which the sum terms are accumulated. The accumu-

lation should sum the largest values first, to obtain maximally large partial sums and to reject a candidate MVs as early as possible in the hierarchy. Several ways have been proposed for estimating the average magnitude of a matching error (one SAD term) at each pixel location in the current partition. In [125, 126, 127, 128], it is shown that with some assumptions, the matching error is proportional to the gradient magnitude of the corresponding location in the current partition. Alternatively, in [129] the matching error is assumed to be proportional to the corresponding pixel deviation from the current partition mean value. And finally, in [128], the matching criterion calculation is actually performed at zero MV and the magnitude of absolute differences are assumed to be similar at other candidate MV locations as well.

When the expected matching error magnitude is available for each pixel in the current partition, the pixels may be sorted and the matching performed in this order for all of the candidate MVs [128]. However, the sorting and pixel order bookkeeping causes significant overhead. In [129], instead of individual pixels, the expected matching errors are estimated for groups of consecutive 4, 8, or 16 pixels. The overhead is reduced by the corresponding amount, and in particular, the algorithm is much more suitable for hardware architectures which can calculate the SAD of several consecutive pixels in parallel. The overhead can be further reduced by using larger blocks of pixels which are sorted as one unit. In [125], the 16×16 -pixel partitions are subdivided into four 8×8 -pixel or sixteen 4×4 -pixel subblocks, which are sorted. Furthermore, the matching scan order for the subblocks is chosen from four possible choices (top-to-bottom, bottom-to-top, left-to-right, and right-to-left), so that the expected largest absolute differences are scanned early. Instead of varying scanning order, in [126] the rows or columns of the subblocks are sorted.

In the previously described methods, the sum terms in the matching scan are sorted based only on the current partition or on the current partition and the candidate partition at zero MV, to avoid excessive overhead. Since the optimal order of the matching scan in reality varies depending on the candidate MV, in [110] the sorting is performed on a candidate MV basis, as follows: the 16×16 -pixel partition is subdivided into four 8×8 subblocks. For each subblock, the SAD is calculated from eight pixels. Then the four subblocks are sorted, with the subblock having the largest partial SAD first. Finally, the remaining 56 absolute differences are computed and summed from each subblock in order using the PDE technique.

In [87], the authors observe that if the terms in the matching scan are summed in a pseudo-random sequence, instead of top-to-bottom as in Eq. (20), the summation can be aborted on average earlier, saving computation. This is an alternative to the sorting-based approach, avoiding the overhead from sorting. Pseudo-random or a dithering

sequence can be also used for matching the individual sorted subblocks [127].

Although PDE is a lossless motion estimation algorithm, since it minimizes Eq. (20) or (19), it is not optimal in the sense of minimizing the coded video distortion subject to rate. Therefore, the property of lossless is not very important, and it is a valid question to ask whether similar but much faster algorithms compared to PDE could be derived by giving up the exact lossless property. In the normalized partial distortion search (NPDS) [130, 91], and probabilistic partial-distance algorithms [131], this is done by estimating the final fully computed SAD based on a partial sum and comparing this estimate (instead of directly the partial sum) to the smallest previous SAD. The estimate can be calculated simply with normalization [130], but this may eliminate even good matches if the first few partial sums happen to be quite large. Better results are obtained using an adjustable conservative function [91], or using a probability distribution model for the final SAD based on the partial SAD [131].

4.3.5 New algorithms

Most motion estimation algorithms have been designed to minimize Eq. (19). However, as discussed in this thesis, to obtain (locally) optimal MV, the distortion in the equation should be the distortion between the final encoded partition and the partition in the original uncompressed video, and the bit rate should correspond to the number of bits used for encoding the whole partition, not just the MV as in Eq. (19). In particular, the SAD matching criterion ignores completely the number of bits required for the transform coefficients. To obtain the exact cost, the partition would need to be encoded, which is too slow in practice.

A novel matching criterion called SSD_{AC} is presented in Paper IV, which separates the constant level (DC, direct current) part of the matching criterion from the rest of the criterion. The DC part corresponds to the first transform coefficient and the AC (alternating current) part to the rest coefficients. By weighting the DC part using a different weight than the AC part, a more accurate estimate of the required bits after the transform coding can be obtained. Since the algorithm aims for high quality, it uses the FS strategy which is sped up using lossless SEA and MSEA algorithms. With a constant bit rate of 0.13 bits per pixel, a 0.12 dB improvement in PSNR is obtained. The improvement depends strongly on the video content: with custom videos with varying background lighting, up to a 1.9 dB improvement was obtained. This can be explained by when the DC level is ignored, it partially eliminates the varying lighting from the block matching results, and the motion estimation can find MVs which better compensate video details.

A peculiarity in the SSD_{AC} criterion is that if the lower bound for the SSD [112] is modified for the SSD_{AC} , the bound is always less than the best previous match, and no candidate MVs will be eliminated, leading to no speed increase. Thus, in Paper IV also a new tighter lower bound is derived for the SSD. Although the new lower bound requires more operations to be calculated, it can eliminate more candidate MVs, leading to overall speed improvement for both SSD and SSD_{AC} criteria. It is also the only way to apply SEA for SSD_{AC} . In the experiments, the conventional lower bound for SSD eliminated 64% of candidate MVs, while the new bound eliminated 74%. For SSD_{AC} , the new criterion eliminated 51% of candidate MVs. In the paper, in addition to the SEA, also MSEA is modified to use the new, more accurate matching criterion.

The MSEA algorithm and the latest H.264 standard have a similarity: both subdivide macroblocks into smaller partitions. In MSEA, a large partition can be subdivided into smaller blocks to obtain a tighter lower bound and to eliminate more candidate MVs. In H.264, larger partitions can be subdivided into smaller partitions to reduce the residual after motion compensation. Paper V represents a modification of MSEA, using it efficiently for H.264-style motion estimation. In the paper, only block norms for 2×2 and 4×4 -pixel blocks (for 1:4 and 1:16 subsampling) are computed and stored into the norm pyramid. The smallest partition size in H.264 is 4×4 pixels. In the presented algorithm, the motion estimation is performed for each of the sixteen 4×4 -pixel partitions in the 16×16 -pixel macroblock simultaneously using the usual MSEA algorithm. For each candidate MV and for each of the small partitions, either the exact SAD has to be computed, or the partition is eliminated based on a lower bound computed from either of the two subsampled hierarchy levels.

Since the larger partitions overlap completely with the smallest partitions, the SADs and lower bounds from the small partitions can be summed up hierarchically to form lower bounds, or even exact SADs for the larger partitions, from 4×8 and 8×4 -pixel partitions up to a whole 16×16 -pixel macroblock. The exact SAD value for a larger block is obtained when the exact SAD is calculated also for each of the overlapping smaller partitions. With the exact SAD, the rate-distortion optimized criterion in Eq. (19) is easily obtained by adding the cost of the MV to the SAD. Otherwise, a lower bound for the rate-distortion criterion is obtained, and the bound is compared to the best previous match for the partition. If the elimination fails, the algorithm computes the missing exact SADs for the smaller partitions and performs the comparison again. In this manner, the motion estimation is performed simultaneously for all of the partition sizes in a macroblock, but the SAD has to be computed from image data only for the smallest partitions. For larger partitions, the SAD or lower bound is obtained from smaller partitions with a very small overhead. With the algorithm, 60%-70% of

computation time is reduced compared to the original FS implementation used in the reference video encoder. Although a great saving, it is much less than the reported 97% [119] decrease with MSEA. This is due to the implementation overhead forced by the complex macroblock structure in the H.264 standard.

The unsymmetric-cross multi-hexagon-grid search (UMHexagonS) [85] was developed to be a very accurate motion estimation search strategy. It attains this objective, losing only negligibly in rate-distortion performance for the FS strategy. However, although much faster than the FS, it is still also much slower than many other fast search strategies, such as the HEXBS (see Subsection 4.1.2). In Paper VI, the computational complexity of the UMHexagonS is improved using several of the hierarchical matching techniques discussed in this section. Although theoretically slower, the SEA is sometimes preferable over the MSEA because it is simpler to implement, works with less memory, and is usable with smaller partition sizes. Thus, the SEA is used to improve the UMHexagonS execution speed. The SEA candidate MV elimination percentage is further improved in the paper by adding a small constant to the lower bound, causing the SEA to eliminate more candidate MVs. When the elimination fails, the SAD is computed, but from filtered and subsampled images to decrease computation. The test results in Paper VI show that the algorithm maintains the high accuracy of UMHexagonS, while reducing the number of operations by up to 95%. It achieves higher rate-distortion performance than DS and HEXBS while having many fewer operations. The practical difference in computation is likely to be smaller, since parallel architectures are more difficult to employ, but this depends heavily on the chosen computer architecture.

4.4 Fractional pixel motion estimation

Only the H.261 standard defines plain integer pixel accurate motion compensation, as discussed in Subsection 2.1.2 and Section 2.2. All newer standards incorporate at least half-pixel accurate motion compensation, and the latest H.264 standard adopts quarter pixel accurate motion compensation. Since it would be computationally unbearable to perform a full search at half or quarter pixel resolution, in practice fast methods are a necessity. We can use all of the principles in this chapter to optimize the search. Nevertheless, the situation is sufficiently different that the implementations of half and quarter pixel accurate ME (HME and QME) will benefit if the algorithms are rethought. Because the pixel values at fractional pixel positions (FPPs) are obtained by interpolating them from integer pixel positions (IPPs), often simply bilinearly, high frequencies and noise are suppressed at FPPs, and the matching criterion may look similar to Fig. 15b. Furthermore, as the interpolation formula is known, in some cases the matching crite-

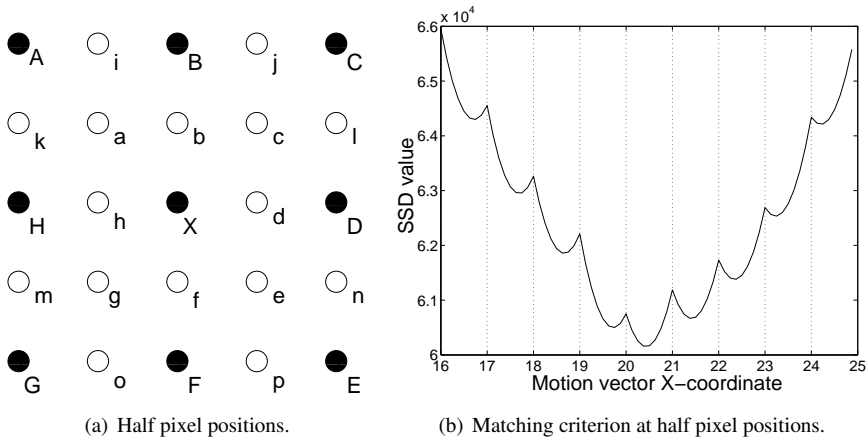


Fig 15. Half pixel accurate motion estimation.

rion value (MCV, Eq. (19)) at some FPP can be estimated efficiently from the surrounding MCVs at IPPs. In the best case, a fast fractional accuracy ME (FME) algorithm can save computation even from that required for image interpolation [135, 136, 137, 138], but still the largest part is required for computing the MCVs.

In [139], the SEA is used for HME. The algorithm uses the full search strategy, searching all IPPs and half pixel positions (HPPs) without using a fast search strategy. The novelty in the paper is the calculation of the lower bounds corresponding to HPPs from the candidate partition norms at IPPs. As bilinear interpolation is assumed, the algorithm is not compatible with H.264. It is also quite slow since the FS is employed, despite the SEA.

4.4.1 Hierarchical fractional pixel motion estimation

Almost all fast FME algorithms perform ME hierarchically, integer pixel accurate ME (IME) at the original resolution, and then local FME around the best integer MV. For HME, the most common method is called a two-step strategy [135], which is used in many reference video encoders. It refines the integer MV (denoted with X in Fig. 15a) into half pixel accuracy (denoted with lower case letters) by calculating the MCVs at the eight nearest HPPs around the best integer MV [140]. The method can be easily extended into QME by performing first IME and HME, and then refining the result further into quarter pixel accuracy by checking the eight nearest quarter pixel positions around

the best HPP. This method resembles very much the TSS presented in Subsection 4.1.2, but the results are better because the matching criterion function at the higher resolution is smoother and the algorithm is less likely to be stuck to a local optimum.

Lee *et al.* [140] represent a fast version of the two-step algorithm: after IME, the MCV is calculated for every second of the 8 half pixel accurate MVs (HMs) around the best integer MV (at positions b, d, f, and h or alternatively a, c, e, and g in Fig. 15). Then a fifth of the eight HMs is checked between the best two of the four first HMs. Thus, the number of checking points in HME is reduced from 8 down to 5, by 38%.

4.4.2 Polynomial interpolation

After the IME has been performed but before the FME, the MCV has been usually already computed at the chosen best integer MV and at some nearby integer MVs. This can be used for guiding the HME. Since the matching criterion function is quite smooth [141], it is possible to estimate its values using polynomial interpolation from nearby positions (instead of calculating the MCVs using the interpolated image data). In [142, 141] a paraboloid with 5 free parameters is fitted to the MCVs at the best integer MV and at the four nearest integer MVs (denoted as X, B, D, F, and H in Fig. 15a). The best HMV among the eight HMs a–h can be then either estimated analytically from the paraboloid parameters [142], or the paraboloid values can be directly evaluated at the eight positions and the best candidate MV giving the smallest value chosen [141]. Since the result is often inaccurate, the MCVs at the chosen and two neighboring position are evaluated using the interpolated image and the best of the three chosen as the final best HMV. The computation required for computing the paraboloid parameters and evaluating its value is quite small, and the total reduction of computation is roughly 62.5%.

Since the four MCVs at the integer MVs surrounding the best integer MV are not always available (for example, at frame corners and edges, or with certain fast IME algorithms), Liu and Orintara [143] proposed a dual parabola model, which requires only three known MCVs at the integer MVs around the best integer MV. In addition to the predicted HMV, the MCV for a few (at most four) extra HMs are also checked, saving on average 70% of the computation. When three MCVs were calculated during the HME, a new set of parabola parameters can be calculated from them for QME.

A paraboloid is used also in [138] for estimating the matching criterion values, but the best position is then searched for using either the DS or TSS at quarter pixel accuracy, using the paraboloid for obtaining approximate MCVs. The exact MCV is evaluated at the chosen position and compared to the approximate value obtained from

the paraboloid. If the difference exceeds a threshold, the paraboloid model is judged to have failed, and a conventional FME is made.

Since the paraboloid model is too inaccurate to be used alone for FME without actually calculating MCVs at least in some positions, a more accurate model would be preferable. Senda [135] noticed that a relatively good estimate for SADs at HPPs is obtained by interpolating MCVs simply linearly from the two nearest MCVs in Eq. (19) at IPPs, and multiplying with a factor slightly less than one (see Fig. 15b):

$$\text{SAD}\left(m_x + \frac{1}{2}, m_y\right) \approx \frac{1}{2}\psi_h [\text{SAD}(m_x, m_y) + \text{SAD}(m_x + 1, m_y)], \quad (23)$$

$$\text{SAD}\left(m_x, m_y + \frac{1}{2}\right) \approx \frac{1}{2}\psi_v [\text{SAD}(m_x, m_y) + \text{SAD}(m_x, m_y + 1)], \text{ and } (24)$$

$$\text{SAD}\left(m_x + \frac{1}{2}, m_y + \frac{1}{2}\right) \approx \frac{1}{2}\psi_{hv} [\text{SAD}(m_x, m_y) + \text{SAD}(m_x + 1, m_y + 1)], (25)$$

for horizontal, vertical, and diagonal HPP, respectively, where (m_x, m_y) is some integer MV. With relations in Eqs. (23)–(25) the SADs can be estimated at positions a–h in Fig. 15a. The matching criterion does not need to be evaluated at the HPPs at all, and HME can be performed almost without any computation, assuming that the MCVs at IPPs have been calculated and saved during IME. The experimentally obtained constants are given by Senda and they depend on the bit rate. For example, at 0.39 bits per pixel (MPEG-2), $\psi_h = \psi_v = 14/16$ and $\psi_{hv} = 13/16$. However, quality is decreased in some cases significantly, as the constants are not adapted to video content.

4.4.3 Exact SSD interpolation

In later papers [136, 137], Senda derives exact SSD at HPPs in the linear interpolation case using SSDs at the two nearest IPPs. The SSD matching criterion at HPP is a sum as follows:

$$\text{SSD}\left(m_x + \frac{1}{2}, m_y\right) = \sum_{y=0}^{Y-1} \sum_{x=0}^{X-1} \text{SD}_{x,y}\left(m_x + \frac{1}{2}, m_y\right). \quad (26)$$

SD denotes a squared difference term in the sum:

$$\text{SD}_{x,y}\left(m_x + \frac{1}{2}, m_y\right) = \left[\hat{f}\left(x + m_x + \frac{1}{2}, y + m_y\right) - f(x, y)\right]^2. \quad (27)$$

The integer MV (m_x, m_y) points to an IPP, and the image at HPP is obtained via linear interpolation

$$\hat{f}\left(x + m_x + \frac{1}{2}, y + m_y\right) = \frac{1}{2}\hat{f}(x + m_x, y + m_y) \quad (28)$$

$$+\frac{1}{2}\widehat{f}'(x+m_x+1,y+m_y).$$

Now, substituting Eq. (28) into (27), and considering one sum term SD of the SSD gives

$$\text{SD}_{0,0}\left(m_x+\frac{1}{2},m_y\right)=\left[\frac{1}{2}\widehat{f}'(m_x,m_y)+\frac{1}{2}\widehat{f}'(m_x+1,m_y)-f\right]^2 \quad (29)$$

$$=\left[\frac{1}{\sqrt{2}}f-\frac{1}{\sqrt{2}}\widehat{f}'(m_x,m_y)\right]^2 \quad (30)$$

$$+\left[\frac{1}{\sqrt{2}}f-\frac{1}{\sqrt{2}}\widehat{f}'(m_x+1,m_y)\right]^2 \\ -\left[\frac{1}{2}\widehat{f}'(m_x,m_y)-\frac{1}{2}\widehat{f}'(m_x+1,m_y)\right]^2,$$

which can be proved by expanding both sides and comparing. Substituting this result back into Eq. (26) gives

$$\text{SSD}\left(m_x+\frac{1}{2},m_y\right)=\frac{1}{2}\sum_{y=0}^{Y-1}\sum_{x=0}^{X-1}\left[f(x,y)-\widehat{f}'(x+m_x,y+m_y)\right]^2 \quad (31)$$

$$+\frac{1}{2}\sum_{y=0}^{Y-1}\sum_{x=0}^{X-1}\left[f(x,y)-\widehat{f}'(x+m_x+1,y+m_y)\right]^2$$

$$-\frac{1}{4}\sum_{y=0}^{Y-1}\sum_{x=0}^{X-1}\left[\widehat{f}'(x+m_x,y+m_y)-\widehat{f}'(x+m_x+1,y+m_y)\right]^2$$

$$=\frac{1}{2}\text{SSD}(m_x,m_y)+\frac{1}{2}\text{SSD}(m_x+1,m_y) \quad (32)$$

$$-\frac{1}{4}\mathcal{H}_{\widehat{f}'}(m_x,m_y)$$

where $\mathcal{H}_{\widehat{f}'}(m_x,m_y)$ is the sum of horizontal squared differentials calculated from the reference image. For horizontal HPP, the horizontal differential is used

$$\mathcal{H}_{\widehat{f}'}(m_x,m_y)=\sum_{y=0}^{Y-1}\sum_{x=0}^{X-1}\left[\widehat{f}'(x+m_x,y+m_y)-\widehat{f}'(x+m_x+1,y+m_y)\right]^2. \quad (33)$$

For vertical HPP, the SSD is obtained similarly, the end result being

$$\text{SSD}\left(m_x,m_y+\frac{1}{2}\right)=\frac{1}{2}\text{SSD}(m_x,m_y)+\frac{1}{2}\text{SSD}(m_x,m_y+1) \quad (34)$$

$$-\frac{1}{4}\mathcal{V}_{\widehat{f}'}(m_x,m_y)$$

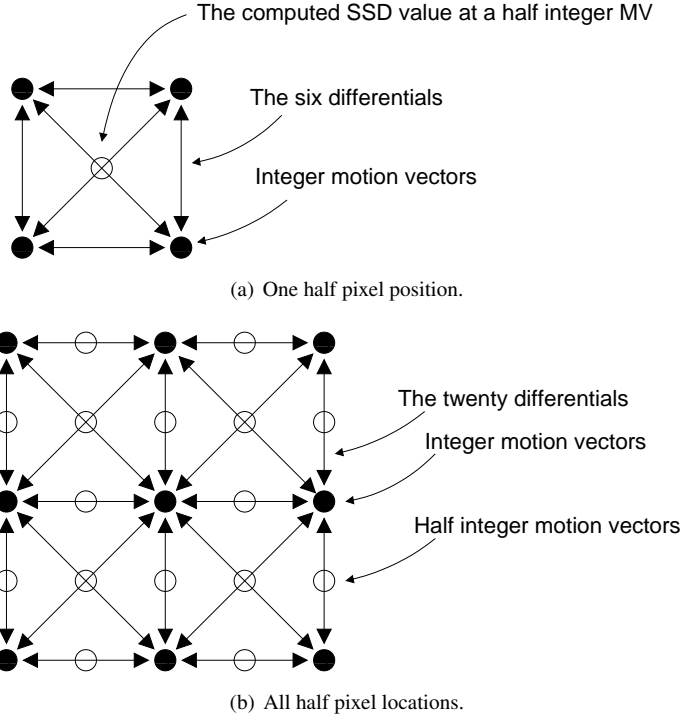


Fig 16. Fast half pixel motion estimation with differentials.

where the sum of vertical differentials is

$$\mathcal{V}_{\hat{f}}(m_x, m_y) = \sum_{y=0}^{Y-1} \sum_{x=0}^{X-1} \left[\hat{f}'(x + m_x, y + m_y) - \hat{f}'(x + m_x, y + m_y + 1) \right]^2. \quad (35)$$

The horizontal and vertical differentials are depicted in Fig. 16. Thus, the SSD at a HPP equals the average of SSDs at the two nearest IPP minus a weighted differential from the reference block. Using Eqs. (32) and (34), the MCV can be calculated at horizontal and vertical HPPs b, d, f, and h in Fig. 15a.

However, Senda did not give formulas for calculating MCVs at diagonal HPPs a, c, e, and g. These can be derived in a similar fashion as for horizontal and vertical HPPs, but the equations are much more complex. In this case,

$$\text{SSD} \left(m_x + \frac{1}{2}, m_y + \frac{1}{2} \right) = \sum_{y=0}^{Y-1} \sum_{x=0}^{X-1} \left[\hat{f}' \left(x + m_x + \frac{1}{2}, y + m_y + \frac{1}{2} \right) - f(x, y) \right]^2 \quad (36)$$

where

$$\begin{aligned}
\widehat{f}'\left(x+m_x+\frac{1}{2},y+m_y+\frac{1}{2}\right) &= \frac{1}{4}\widehat{f}'(x+m_x,y+m_y) \\
&+ \frac{1}{4}\widehat{f}'(x+m_x+1,y+m_y) \\
&+ \frac{1}{4}\widehat{f}'(x+m_x,y+m_y+1) \\
&+ \frac{1}{4}\widehat{f}'(x+m_x+1,y+m_y+1).
\end{aligned} \tag{37}$$

Substituting Eq. (37) into (36) gives

$$\text{SD}_{0,0}\left(m_x+\frac{1}{2},m_y+\frac{1}{2}\right) \tag{38}$$

$$= \left[\frac{1}{4}\widehat{f}'(m_x,m_y)+\frac{1}{4}\widehat{f}'(m_x+1,m_y)\right. \tag{39}$$

$$\left.+\frac{1}{4}\widehat{f}'(m_x,m_y+1)+\frac{1}{4}\widehat{f}'(m_x+1,m_y+1)-f\right]^2$$

$$= \left[\frac{1}{2}f-\frac{1}{2}\widehat{f}'(m_x,m_y)\right]^2+\left[\frac{1}{2}f-\frac{1}{2}\widehat{f}'(m_x+1,m_y)\right]^2 \tag{40}$$

$$+\left[\frac{1}{2}f-\frac{1}{2}\widehat{f}'(m_x,m_y+1)\right]^2+\left[\frac{1}{2}f-\frac{1}{2}\widehat{f}'(m_x+1,m_y+1)\right]^2$$

$$-\left[\frac{1}{4}\widehat{f}'(m_x,m_y)-\frac{1}{4}\widehat{f}'(m_x+1,m_y)\right]^2-\left[\frac{1}{4}\widehat{f}'(m_x,m_y)-\frac{1}{4}\widehat{f}'(m_x,m_y+1)\right]^2$$

$$-\left[\frac{1}{4}\widehat{f}'(m_x,m_y)-\frac{1}{4}\widehat{f}'(m_x+1,m_y+1)\right]^2-\left[\frac{1}{4}\widehat{f}'(m_x+1,m_y)-\frac{1}{4}\widehat{f}'(m_x,m_y+1)\right]^2$$

$$-\left[\frac{1}{4}\widehat{f}'(m_x+1,m_y)-\frac{1}{4}\widehat{f}'(m_x+1,m_y+1)\right]^2$$

$$-\left[\frac{1}{4}\widehat{f}'(m_x,m_y+1)-\frac{1}{4}\widehat{f}'(m_x+1,m_y+1)\right]^2$$

for a squared difference. As before, this can be proved easily by expanding Eqs. (39) and (40) and comparing. Substituting this result back into Eq. (36) gives a formula for calculating the SSD MCV for a diagonal HPP as

$$\text{SSD}\left(m_x+\frac{1}{2},m_y+\frac{1}{2}\right) = \frac{1}{4}\text{SSD}(m_x,m_y)+\frac{1}{4}\text{SSD}(m_x+1,m_y) \tag{41}$$

$$+\frac{1}{4}\text{SSD}(m_x,m_y+1)+\frac{1}{4}\text{SSD}(m_x+1,m_y+1)$$

$$-\frac{1}{16}\mathcal{H}_{\widehat{f}'}(m_x,m_y)-\frac{1}{16}\mathcal{V}_{\widehat{f}'}(m_x,m_y)$$

$$-\frac{1}{16}\mathcal{N}_{\widehat{f}'}(m_x,m_y)-\frac{1}{16}\mathcal{S}_{\widehat{f}'}(m_x,m_y)$$

$$-\frac{1}{16}\mathcal{V}_{\hat{f}}(m_x+1, m_y) - \frac{1}{16}\mathcal{H}_{\hat{f}}(m_x, m_y+1)$$

where $\mathcal{N}_{\hat{f}}$ is the sum of diagonal squared differentials in the northwest-southeast direction calculated from the reference image as

$$\mathcal{N}_{\hat{f}}(m_x, m_y) = \sum_{y=0}^{Y-1} \sum_{x=0}^{X-1} \left[\hat{f}(x+m_x, y+m_y) - \hat{f}(x+m_x+1, y+m_y+1) \right]^2 \quad (42)$$

and $\mathcal{S}_{\hat{f}}$ is the sum of diagonal squared differentials in the southwest-northeast direction

$$\mathcal{S}_{\hat{f}}(m_x, m_y) = \sum_{y=0}^{Y-1} \sum_{x=0}^{X-1} \left[\hat{f}(x+m_x+1, y+m_y) - \hat{f}(x+m_x, y+m_y+1) \right]^2. \quad (43)$$

For calculating the SSD for one HMV, six differentials are needed, shown in Fig. 16a as arrows. The result in Eq. (41) without intermediate phases was given for the first time in Paper VII. Together with Eqs. (32) and (34) the SSD can be calculated for all HPPs a–h in Fig. 15a using the twenty differentials shown in Fig. 16b. The SSD can be calculated also in HPPs i–p with the twenty differentials, without any significant extra computation.

The computation of one differential in Eq. (33), (35), (42), or (43) requires $X \times Y$ differences, squares, and accumulations, and it is therefore computationally equally fast to compute directly the SSD at some MV. If the nine nearest HMVs are checked by computing the SSDs using differentials, the direct calculation of nine SSDs is replaced by the calculation of twenty differentials. However, the image interpolation into half pixel resolution is completely avoided, saving computation and precious memory. More importantly, the differentials can be calculated efficiently using a sliding window method, as described in Paper VII, or approximated very accurately using nearby differentials in the same or even in different directions. For example,

$$\mathcal{H}_{\hat{f}}(m_x, m_y) \approx \mathcal{H}_{\hat{f}}(m_x+1, m_y) \approx \mathcal{H}_{\hat{f}}(m_x, m_y+1) \approx \mathcal{H}_{\hat{f}}(m_x+1, m_y+1)$$

and similarly for $\mathcal{V}_{\hat{f}}$, $\mathcal{N}_{\hat{f}}$, and $\mathcal{S}_{\hat{f}}$ differentials. In this way, only four differentials need to be calculated instead of twenty. Furthermore, in the experiments it was noticed that a good approximation for diagonal differentials is obtained with $\mathcal{N}_{\hat{f}} \approx \mathcal{S}_{\hat{f}} \approx \max\{\mathcal{H}_{\hat{f}}, \mathcal{V}_{\hat{f}}\}$. There are also numerous other methods for estimating the differentials more efficiently than directly evaluating the SSD for HPPs.

As mentioned in Section 3.1, commonly SAD is used in ME instead of SSD. Although results in Eqs. (32), (34), and (41) are exact only for SSD, they can be used for approximating efficiently and relatively accurately also SAD MCV by replacing

the squares in the differentials with absolute values, as shown in Paper VII. The same methods can be used for approximating SAD differentials as SSD differentials, and as the result computation can be decreased by 44%–94% in the refinement of integer MVs into HPPs, compared to the traditional hierarchical refinement which computes SAD directly at eight HPPs. The loss in video quality will be only 0.04–0.21 dB PSNR at a fixed bit rate, which is much less than in other fast HME methods that the new method was compared against.

4.5 Summary

In this chapter widely different fast block motion estimation algorithms were reviewed and classified, and new improved algorithms were presented. The motion estimation algorithms can be classified into fast search strategies, which compute the matching criterion at only a few candidate MV locations, and to fast matching algorithms, which approximate the matching criterion. We also discussed fast half and quarter pixel accurate motion estimation methods. Integer pixel accurate methods can be directly extended into half pixel accuracy, but improved results can be obtained by designing new algorithms for fractional pixel accuracy.

This work provides contribution for both integer and fractional pixel accurate motion estimation. The proposed methods in Papers IV and V (and in Paper IX, to be discussed in Section 5.4) are full search algorithms: with these methods, the quality remains at the same level as with any full search algorithm, but the methods obtain a significant reduction in computation compared to the other algorithms. In Paper IV, a new tighter bound is derived for the SSD matching criterion, which allows a larger portion of checking points to be skipped than with previously presented bounds. The paper also proposes the SSD_{AC} matching criterion, which can minimize the impact of lighting changes in video, providing more accurate motion vectors (MVs). Paper V applies the MSEA to the H.264 video coding standard, taking into account the peculiar hierarchical motion compensation structure in the standard.

Paper I presents a general optimization algorithm called rotation search, which is applicable also to motion estimation as a fast search strategy. This algorithm is explained in detail in Subsection 3.4.1. For motion estimation, it is extremely fast but loses in quality to other search strategies. On the other hand, the UMHexagonS is one of the best search strategies in the produced image quality, but somewhat slow. In Paper VI, the UMHexagonS is sped up by applying filtering, subsampling, and elimination by bounding. The resulting algorithm contains 95% less computation while the image quality remains essentially at the same level. In Paper III we devise a novel motion es-

timate refinement method. It is unique in that it is applicable to any fast search strategy: the method takes as an input the MV obtained from any fast but inaccurate method, and refines it by checking the given number of statistically optimal MVs around the original MV.

In Paper VII an efficient method is presented for fractional pixel accurate motion estimation. The paper notes that the SSD matching criterion can be interpolated exactly for half pixel accuracy using image differentials. The paper presents efficient methods for obtaining the differentials and proposes also a high-quality approximation for the more common SAD matching criterion. The presented method is very accurate but still very fast compared to traditional methods. A negative aspect in it is that it is not directly applicable to the H.264 standard, which employs a higher order half pixel interpolation. However, even with the H.264, the method could be used for quarter pixel accurate motion estimation.

It should be noted that most papers describing new motion estimation algorithms actually mix several different methods, particularly from Subsections 4.1.1, 4.1.2, and 4.1.3, and present them as one algorithm, which is then compared to other similarly composed algorithms. This should be avoided, since the combination of different algorithms is huge. Instead each particular method should be compared against other similar methods. For example, MV prediction methods should be compared to other MV prediction methods, and not to some algorithm composed mainly of a fast search pattern.

5 Number theoretic transforms

The field of mathematics called number theory is not very well known in the signal processing community compared to many other fields. Number theory is best known for its use in error correction and cryptography [144], but it has applications also in coding theory, communications, physics, digital information [144], bilinear and other transforms, and solving partial differential equations [145]. For us, the applications of interest are digital filtering, image processing, and video coding.

Convolution and correlation are used as parts of algorithms for enhancing visual quality, removing distortions due to noise, motion blurring, defocus, and distortions due to imaging systems. Spectacularly, number theory is an important tool for deriving efficient convolution algorithms. There are several ways of employing number theory for fast convolution. One method is to derive the structure (data flow) of fast algorithms from number theory [144]. In the author's prior work, Winograd number theoretic transform algorithms were developed [117], but they are not particularly suitable either for software (due to modulo arithmetic) nor for hardware (due to the complex algorithm structure) implementation. Number theory can be used also for designing new arithmetic, instead of employing the standard floating or fixed point arithmetic operations. Since in this chapter primarily hardware implementation is considered, the emphasis is put on developing efficient arithmetic. The well-known radix-2 fast Fourier transform (FFT) algorithm is used for transforming image data.

The residue number system (RNS) [145] is employed to allow efficient and exact carry-free multiplication, addition, and subtraction operations, as described in Section 5.2. Furthermore, the Fermat number transform (FNT) is used to remove the need for a complex-valued FFT. Since the FNT places strict restrictions to allowed RNSes, it is described first in Section 5.1. These two methods are used to implement efficient video filtering (Section 5.3) and motion estimation (Section 5.4) algorithms.

5.1 The generalized Fermat number transform

A cyclic convolution \mathbf{c} between two one-dimensional discrete signals \mathbf{a} and \mathbf{h} can be defined as

$$c_i = \sum_{n=0}^{N-1} a_{i-n} h_n \quad (44)$$

where a_i , h_i , and c_i are some elements in \mathbf{a} , \mathbf{h} , and \mathbf{c} , respectively. The indices are calculated cyclically with period N : $a_i = a_{i+\alpha N}$ for any integer α . For simplicity, the index i is usually between 0 and $N - 1$ and the signals are said to be of length N . The convolution can be easily extended into two dimensions as follows:

$$C_{i,j} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} A_{i-n,j-m} H_{n,m} \quad (45)$$

where $A_{i,j}$, $H_{i,j}$, and $C_{i,j}$ are elements in two-dimensional signals \mathbf{A} , \mathbf{H} , and \mathbf{C} . In linear image filtering, the overlap-save or overlap-add algorithms [146] are used: \mathbf{A} is a small block of an image and \mathbf{H} is a filter. The problem in the direct computation of convolution is that it requires a huge number of operations [147]. For computing the NM elements in \mathbf{C} , N^2M^2 multiplications and accumulations are necessary. However, it is well known that the convolution of two signals corresponds to elementwise multiplication in the Fourier transform domain [146]. Thus, only NM multiplications are necessary for the actual convolution. The signals have to be first transformed into Fourier domain and after the multiplications inverse transformed. Nevertheless, as there are fast transform algorithms, the total number of operations is much less than with a direct convolution as in Eqs. (44) and (45):

$$\mathbf{c} = \mathcal{F}^{-1} [\mathcal{F}(\mathbf{a}) \cdot \mathcal{F}(\mathbf{h})] \quad (46)$$

where

$$\mathcal{F}(\mathbf{a}) = [\tilde{a}_0, \dots, \tilde{a}_{N-1}] \text{ and} \quad (47)$$

$$\tilde{a}_i = \sum_{n=0}^{N-1} a_n W^{ni}, i = 0, \dots, N-1. \quad (48)$$

\mathcal{F} denotes the transform and \tilde{a}_i are the transformed coefficients of \mathbf{a} . In the Fourier transform, the transform kernel $W = e^{-j2\pi/N}$, $j^2 = -1$. Convolution using a FFT is widely used in practice, but it has a number of shortcomings. In image processing, the signals are real-valued, but the Fourier transform operates in the complex domain. This causes extra overhead in the implementation. Furthermore, while pixel values are integers, typically at most 255, in the Fourier transform the transform kernel contains transcendental functions which need infinite precision to be presented exactly. In practice, these need to be approximated using floating or fixed point values causing rounding errors and requiring difficult analysis on the algorithm accuracy.

Pollard [148] noticed that the convolution property is not unique to the Fourier transform. He defined similar transforms over finite fields, noticing that the resulting transforms still have the convolution property. The number theoretic transforms are

still defined using Eq. (48), but all arithmetic operations are defined modulo q . The transform kernel is an element in the finite field with property $W^N = 1 \pmod{q}$. There are strict restrictions, depending on the transform length N , for choosing the modulus q and the kernel W which are discussed in [117] and in Paper VIII. However, in many cases, they can be chosen from several different possibilities and should be chosen to simplify the implementation. The modulus q must be odd, which rules out the easiest-to-implement moduli 2^b , with b a positive integer.

In the Fermat number transform (FNT), the modulus is a prime of form $q = 2^k + 1$, $k = 2^b$. FNT is well-known and widely used [146, 147, 144, 149, 150, 5]. However, the generalized Fermat number transform (GFNT) [151], discussed in Paper VIII, is much less known, although it is still very easy to implement. In the GFNT, the modulus is of the form $q = 2^b + 1$ and may be non-prime, which allows such moduli as $2^{24} + 1$ and $2^{32} + 1$. Furthermore, in the GFNT, the transform kernel W powers can be often decomposed into a sum of simple terms, and multiplications by them can be implemented with a few bit shifts and additions, without general multiplications. In practice, the GFNT is carried out using some modified FFT algorithm. The input signal and the filter into GFNT consists of pure integers. The modulus q must be sufficiently large so that the resulting signal from the convolution after the inverse transform has no values larger than q : thus, the minimum value of q is constrained by the dynamic range of the result.

Although GFNT allows rounding error free transforms to be implemented without general multiplications and with integer and single-component values, the downside is that the algorithm requires modulo arithmetic. This is often a significant obstacle in software implementation, and typically specially-designed hardware is used for the implementation. Paper VIII and [149] discusses the implementation of efficient operations in hardware.

5.2 Residue number systems

In a residue number system (RNS) [145], every integer value and operation in a ring modulo $q = q_1 \times \dots \times q_Q$ is transformed from one value or operation into Q values or operations in subrings modulo q_1, \dots, q_Q , where q_i are relatively prime. The advantage for the RNS is that while the word length without RNS would be at least $\lceil \log_2 q \rceil$ bits, the word length $\lceil \log_2 q_i \rceil$ of the operations in the RNS is significantly less: the sum of the word lengths in the RNS is only slightly larger than the original word length. Although the number of operations is Q times more, this is generally more than justified since addition, subtraction, and multiplication operations with a shorter word length are typically faster, the shorter word length reduces power consumption, and the system

requires less chip area in an integrated circuit implementation. Other advantages, although not evaluated in this thesis, are that RNS reduces the complexity of integrated circuit testing and allows skewed circuit clocking for removing power-drawing glitches. RNS can be also used to implement circuits with fault tolerance. [145]

RNSes have also disadvantages that has made them unpopular. To interface with other systems using common binary arithmetic, the values need to be converted back to normal binary code (NBC), which is usually a complex operation. General division is also more difficult within a RNS than with the NBC, since rounding (and bit shifting to right, as in NBC) cannot be easily performed. As a consequence, also fixed point arithmetic is cumbersome, unless sufficiently long word length is used to avoid the need for division after multiplications. In this work the RNS values are converted back to NBC immediately after the processing has been done, using specially designed very efficient conversion circuit.

Conversion of a value from the original ring to the subrings is obtained simply by calculating the residues of the values divided by the moduli. When the original values are sufficiently small (for example, 8-bit pixel values) and the moduli are sufficiently large (more than the maximum pixel value), the initial modulo calculation can be completely avoided. The computation in the subrings proceeds in the same manner as it would without RNS, in parallel, but the moduli are smaller. The most difficult part is the conversion of the result vector back to the original ring. There are several approaches to perform the conversion, as discussed in Paper VIII, but the most common methods are the Chinese remainder theorem (CRT) and the mixed radix conversion (MRC).

For example, let us compute $x + y = 456 + 123$ using a RNS with $q_1 = 2^4 + 1$ and $q_2 = 2^8 + 1$. Since the moduli are relatively prime and their product is larger than the final result, this is a valid RNS. In the RNS, $x_1 = x \bmod q_1 = 14$ and $x_2 = x \bmod q_2 = 199$. Thus, x is represented as a vector of two independent values, $[14, 199]$. Similarly, y is represented as the vector $[123, 123]$. Adding these vectors together modulo the corresponding modulus, we get the sum $[z_1, z_2] = [14 + 123 \bmod q_1, 199 + 123 \bmod q_2] = [1, 65]$. Finally the result has to be converted back to the original ring modulo $q_1 q_2 = 4369$. With the CRT, this is made as follows:

$$z = w_1 z_1 + w_2 z_2 \bmod q \quad (49)$$

where $w_1 = q_2 (q_2^{-1} \bmod q_1) = 2313$ and $w_2 = q_1 (q_1^{-1} \bmod q_2) = 2057$. Now the values can be plugged in and we get

$$z = 2313 \times 1 + 2057 \times 65 \bmod 4369 \quad (50)$$

$$= 579 \quad (51)$$

which is the correct result.

In Paper VIII, we present in detail an integrated digital circuit based on MRC for converting values from the two ring RNS based on moduli $2^8 + 1$ and $2^{16} + 1$ back into the original ring. As described in the next section, this RNS is particularly interesting for practical use. The gate count for the conversion circuit is only 261 gates, but it saves a large amount of gates from the arithmetic operations, most importantly from multiplications. Since this is the first time that a specialized conversion circuit for this RNS is presented, its gate count cannot be compared to similar existing circuits. However, a multiplier modulo $2^8 + 1$ requires 828 gates and multiplier modulo $2^{16} + 1$ requires 2741 gates, as shown in Paper VIII. A multiplier in a ring modulo $2^{24} + 1$, a comparable dynamic range to that in the RNS, would require 5777 gates, or 61% more than the two smaller multipliers combined in the RNS.

5.3 Video filtering

A very common operation in image and video restoration and enhancement is linear filtering of images with a convolution mask. The overlap-save algorithm [146, 150] is usually used when the image is relatively large compared to the filter. In this case, each image is divided into overlapping blocks. According to the convolution theorem, the whole frame and the filter mask (zero-padded to the frame size) could be transformed, multiplied, and inverse transformed to obtain the convolution between them. However, using the overlap-save algorithm, much shorter transforms can be used and computation is saved, as described next.

Let the filter mask be of equal length H both horizontally and vertically. The original image is divided into square blocks, which are each N samples wide and tall. $N \geq H$ so that the convolution between the mask and the image block can be computed. N should be also an integer power of 2 so that simple fast transform algorithms can be used, but otherwise N can be chosen freely to minimize computation.

First the convolution mask is zero padded to size N in both dimensions, as shown in Fig. 17, and both the mask and each image block is transformed one by one. The transformed mask is multiplied with each of the transformed image blocks, and the result is inverse transformed. This gives $N - H + 1$ samples of valid linear convolution vertically and horizontally, as shown in Fig. 17. Therefore, the blocks in the original image must be at most $N - H + 1$ samples apart so that for each sample in the original image a corresponding convoluted sample can be obtained. Thus, the blocks must overlap by at least $H - 1$ samples, which gives the algorithm name.

Let us assume that the filter mask is fixed and it can be transformed before entering

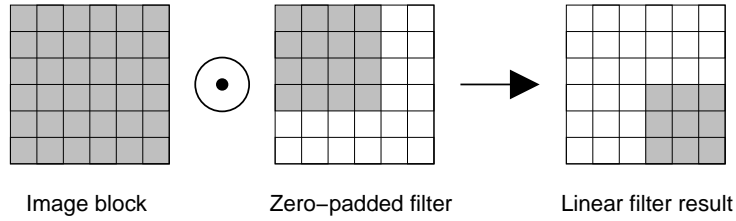


Fig 17. The overlap-save algorithm. In this example $N = 6$ and $H = 4$.

the overlap-save algorithm. In this case, only a single transform per image block is necessary. If the transform is performed using the radix-2 fast transform algorithm, there are about $\frac{3}{2}N \log_2 N$ operations in a single one-dimensional N -point transform [146]. If rows and columns are transformed separately, there are $2N \times \frac{3}{2}N \log_2 N$ operations for both transform and inverse transform. Additionally, when the blocks are multiplied together in the transform domain, N^2 multiplications are necessary. Thus, for each block $N^2(1 + 6 \log_2 N)$ operations are required, or $N^2(1 + 6 \log_2 N) / (N - H + 1)^2$ on average per convoluted sample.

As an example, if the filter mask size $H = 16$, we can find that the optimum block length is $N = 173$, or 128, if an integer power of 2 is required. With the 128-point radix-2 transform, on average about 55 operations are required for each pixel. If the convolution were to be performed in the spatial domain, $2H^2 = 512$ operations would be necessary per pixel. As can be seen in Fig. 18, for small image block sizes the operation count is very high, but decreases rapidly and obtains the minimum for 173 samples wide blocks. Thus, great savings can be obtained using the overlap-save algorithm and fast transforms. However, it should be emphasized that the operation count is only the first step in comparing different implementation alternatives. Also other aspects, such as the required memory and hardware complexity should be taken into account.

Although number theoretic transforms do not quantitatively reduce the number of operations in video filtering compared to fast Fourier transforms, they do decrease the complexity of each operation. Shakaff *et al.* [150] presented a video filtering method based on Fermat number transforms and described a software-based implementation using $q = 2^{16} + 1$. However, this modulus does not allow a sufficiently large dynamic range for many applications, and the next Fermat number $q = 2^{32} + 1$ would be excessively large. In Paper VIII, the word length restriction is relaxed by using GFNTs, and the performance is further improved by introducing the use of RNS with them.

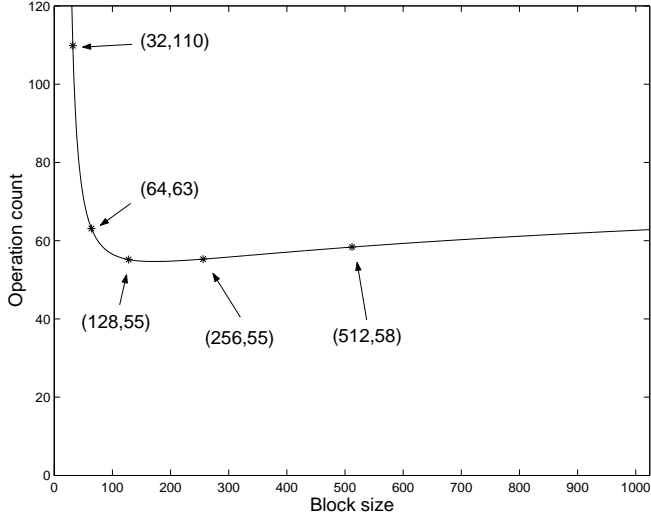


Fig 18. Operations in the overlap-save algorithm, $H = 16$.

5.4 Motion estimation

Efficient convolution algorithms can be also used for block motion estimation in video encoders, as in Fig. 5. Instead of SAD (Eq. (20)), the sum of squared differences (SSD) matching criterion must be used. It can be expanded into three terms in Eqs. (53)–(55) [152]:

$$\text{SSD}(m_x, m_y) = \sum_{y=0}^{Y-1} \sum_{x=0}^{X-1} [\hat{f}(m_x + x, m_y + y) - f(x, y)]^2 \quad (52)$$

$$= \underbrace{\sum_{y=0}^{Y-1} \sum_{x=0}^{X-1} f(x, y)^2}_{S_1} \quad (53)$$

$$+ \underbrace{\sum_{y=0}^{Y-1} \sum_{x=0}^{X-1} \hat{f}(m_x + x, m_y + y)^2}_{S_2(m_x, m_y)} \quad (54)$$

$$- 2 \underbrace{\sum_{y=0}^{Y-1} \sum_{x=0}^{X-1} f(x, y) \hat{f}(m_x + x, m_y + y)}_{\text{correlation } r(m_x, m_y)}. \quad (55)$$

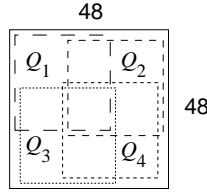


Fig 19. Four overlapping 32×32 pixel blocks.

Of these terms, S_1 in Eq. (53), is a constant and does not need to be computed, since we are interested only in finding m_x and m_y where the SSD function has the minimum, not the actual minimum value. For computing the sum norm S_2 in Eq. (54), there are several efficient methods. Finally, the correlation r in Eq. (55) can be computed using any fast convolution algorithm. In this section the partition size $X \times Y$ is assumed to be 16×16 pixels, as in most standards.

In [152], the authors apply a FIR filter decomposition method, reducing computation in motion estimation by 77% compared to directly evaluating the SSD matching criterion at each search point. However, the number of operations can be reduced further by computing the correlation via number theoretic transforms. In [5], the FNT is used for this purpose. However, due to the FNT with modulus $2^{16} + 1$, the authors have to quantize the pixel values down to 4 bits, which degrades the motion estimation accuracy. Furthermore, the MV search range is limited to ± 8 pixels due to the 32-point transform length.

In Paper IX, these limitations are removed by employing the GFNT with modulus $2^{24} + 1$ and by subdividing the search area into four overlapping blocks, each 32×32 pixels, as shown in Fig. 19. Each of the blocks is then transformed. The current partition is zero-padded to 32×32 pixels and flipped cyclically around the origin, as shown in Fig. 20. All of the blocks are transformed with the GFNT, the transformed current partition is multiplied with each of the four quarters, and the result is inverse transformed. This gives four blocks of 16×16 elements of linear correlation, which are stitched together to form 32×32 elements of correlation. The result is then used for motion estimation with the SSD criterion according to Eqs. (53)–(55).

Any of the three different sum norm computation methods described in Subsection 4.3.2 can be used: the sliding window method, the norm pyramid method, or an integral transform. In our experiments, we used the sliding window method as it requires the least extra memory buffers. The sum norms are computed from squared pixel values of the reference frame.

When the full search is used with an MV range of ± 15 pixels horizontally and ver-

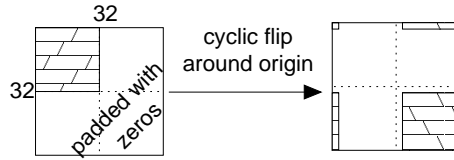


Fig 20. Cyclical flip around origin.

tically, and the partition size is 16×16 pixels, and the SSD matching criterion is evaluated at each candidate MV, there are around $31^2 \times 16^2 \times 3 = 738048$ operations, including differences, squares, and accumulations (additions). When the MV is computed using the method described in Paper IX, based on the GFNT, only 52896 operations are necessary, providing a reduction of 93%. However, since the modulo arithmetic has to be used, the software implementation (as in the paper) achieves a smaller reduction 83% in actual time. If a special-purpose hardware were to be used, it can be expected that the reduction achieved in practice would approach the theoretical reduction in operations. However, most importantly, the obtained image quality and rate-distortion performance are equivalent to the performance obtained by using the full search with a full matching scan.

5.5 Summary

In this chapter, the usage of number theoretic transforms for video filtering and for block motion estimation was discussed. In particular, it was emphasized that the generalized Fermat number transform suits linear video filtering very well when implemented with digital circuits. In addition, the residue number system was applied successfully to generalized Fermat number transforms, for the first time to the author knowledge. An efficient conversion circuit for converting a number from the residue number system back to normal binary code was presented in Paper VIII.

It was shown in Section 5.4 and in Paper IX how to use the generalized Fermat number transform for motion estimation. The transform is used for computing the correlation between the search area and the current image block, and the correlation is then used for finding the minimum of the SSD matching criterion. The algorithm checks every point within the search region and is equivalent in motion vector accuracy to full search methods. Nevertheless, the theoretical complexity is lower due to the employed fast transform algorithm.

A shortcoming with the number theoretic techniques is that they are relatively in-

flexible: while it is easy to adopt the fast Fourier transform (FFT) for various transform lengths, designing new NTTs usually takes more effort. Thus, number theoretic techniques appear to be most useful in specific problems with fixed requirements. When this restriction can be allowed, NTTs are very attractive.

6 Conclusions

Video processing and encoding are important topics in communications, which in itself is one of the basic infrastructures in modern society. In this thesis, the background and previous work on the topics were discussed and reviewed to give the reader an overview of them beyond the breadth of the papers in the appendices.

From the novelty point of view, in this thesis several new improvements were presented. Common to all the improvements was that they are applicable with commonly used video coding standards. This is possible because the standards define bit stream syntax and a method for decoding the video, but not the actual encoding method. This makes it possible to tune the encoders for improved performance.

There are generally three different aspects of video encoding to consider: video quality, the generated bit rate, and computational complexity. One issue can be easily improved at the cost of making the encoder performance in one or two of the other aspects worse. Video encoders consist of several algorithms working in conjunction, and typically a suitable tradeoff in each of the three aspects can be chosen for each algorithm. To balance all of the algorithms for optimal or near optimal conglomerate, various methods were presented in Chapter 3. In that chapter, the algorithms in video encoders were considered mostly as black boxes, with a functionality to adjust between rate and distortion. The advantage of this approach is that the optimization methods are very general. They are typically applicable to any video encoder, and to some degree even to other coding purposes, as long as the distortion metric is defined meaningfully. With the rate-distortion optimization methods, video coding can be optimized from the highest level. As an original work, two new rate-distortion optimization methods were presented, one very nearly optimal but slow, and one heuristic but considerably faster method.

One of the most important single algorithms in a video encoder is block matching for motion estimation. The purpose of motion estimation and compensation is to remove temporal redundancy from video sequences. Motion estimation generates motion vectors, that represent the movements of image blocks between successive frames. Motion estimation is conceptually quite simple, and it allows a very large reduction in the video bit rate. However, it is difficult to implement efficiently and often requires considerably large part of the computation in video encoders. The main classes of motion estimation algorithms were presented in Chapter 4, and several novel algorithms were presented.

In the last part of the thesis, in Chapter 5, an efficient method for video filtering was

presented, based on number theoretic transforms. Filtering and convolution can be used for improving image quality, and number theoretic transforms allow efficient convolution of images with arbitrary filters, requiring a minimal number of multiplications. The generalized Fermat number theoretic transform was extended to use a residue number system, improving the efficiency of the arithmetic operations, in particular the remnant multiplications. Finally, a new motion estimation algorithm based on number theoretic transforms was presented, proving that the applications of the transforms are wider than just plain filtering.

References

1. Brightman I (2005) Technology, Media and Telecommunications Trends: Predictions, 2005—A Focus on the Mobile and Wireless Sector. Goliath Business News [cited 25 Aug 2007] Available from: http://goliath.ecnext.com/coms2/gi_0199-3711093/TMT-Trends-Predictions-2005-A.html.
2. Mehrabian A (1981) Silent Messages: Implicit Communication of Emotions and Attitudes. Belmont, CA: Wadsworth.
3. Finnpanel Oy (2003) Press release. [cited 25 Aug 2007] Available from: http://www.finnpanel.fi/tulokset/tiedotteet/tv_150103_2.html.
4. Finnpanel Oy (2005) Press release. [cited 25 Aug 2007] Available from: http://www.finnpanel.fi/tulokset/tiedotteet/radio_160905.html.
5. Kim NH & Song S (2001) Motion Estimation by Fermat Number Transform. Visual Communications and Image Processing, Proceedings of SPIE 4310: 762–765.
6. Ostermann J, Bormans J, List P, Marpe D, Narroschke M, Pereira F, Stockhammer T & Wedi T (2004) Video Coding with H.264/AVC: Tools, Performance, and Complexity. IEEE Circuits and Systems Magazine 1: 7–28.
7. Yao W, Ostermann J & Zhang YQ (2002): Video Processing and Communications. Upper Saddle River, New Jersey: Prentice Hall.
8. Poynton C (2000) YUV and Luminance Considered Harmful: A Plea for Precise Terminology in Video. Available from http://www.poynton.com/PDFs/YUV_and_luminance_harmful.pdf.
9. Richardson I (2003) H.264 / MPEG-4 Part 10 Tutorials. [cited 9 May 2003] Available from: <http://www.vcodex.com/h264.html>.
10. Shi Y & Sun H (2000) Image and Video Compression for Multimedia Engineering: Fundamentals, Algorithms, and Standards. CRC Press LLC.
11. Sullivan G & Wiegand T (1998) Rate-Distortion Optimization for Video Compression. IEEE Signal Processing Magazine November: 74–90.
12. Till Halbach (2002) Some Important International Standards and Organizations in Visual Telecommunications 1/3. NORSIGNalet 2: 9–13.
13. Till Halbach (2002) Some Important International Standards and Organizations in Visual Telecommunications 2/3. NORSIGNalet 3: 23–30.
14. Till Halbach (2003) Some Important International Standards and Organizations in Visual Telecommunications 3/3. NORSIGNalet 1: 12–19.
15. Wiegand T, Schwarz H, Joch A, Kossentini F & Sullivan G (2003) Rate-Constrained Coder Control and Comparison of Video Coding Standards. IEEE Transactions on Circuits and Systems for Video Technology 13(7): 688–703.
16. Wikipedia (2007) H.264/MPEG-4 AVC. [cited 25 Aug 2007] Available from: <http://en.wikipedia.org/wiki/H.264>.
17. Sullivan G & Wiegand T (2003) Liaison statement to ISO/IEC JTC1/SC29/WG11 (MPEG). ITU-T Video Coding Experts Group (ITU-T SG16 Q.6) document VCEG-T08, San Diego.
18. Ding W & Liu B (1996) Rate Control of MPEG Video Coding and Recording by Rate-Quantization Modeling. IEEE Transactions on Circuits and Systems for Video Technology 6(1): 12–20.
19. Ribas-Corbera J & Lei SM (2000) A Frame-Layer Bit Allocation for H.263+. IEEE Transactions on Circuits and Systems for Video Technology 10(7): 1154–1158.
20. Ortega A & Ramchandran K (1998) Rate-Distortion Methods for Image and Video Com-

- pression. *IEEE Signal Processing Magazine* November: 23–50.
21. Reed E & Lim J (2002) Optimal Multidimensional Bit-Rate Control for Video Communication. *IEEE Transactions on Image Processing* 11(8): 873–885.
 22. Kim WK, Kim SD & Kim J (2002) Bit Allocation for Interframe Video Coding Systems. *ETRI Journal* 24(4): 280–289.
 23. Tao B, Dickinson B & Peterson H (2000) Adaptive Model-Driven Bit Allocation for MPEG Video Coding. *IEEE Transactions on Circuits and Systems for Video Technology* 10(1): 147–157.
 24. Yang Y & Hemami S (2000) Generalized Rate-Distortion Optimization for Motion-Compensated Video Coders. *IEEE Transactions on Circuits and Systems for Video Technology* 10(6): 942–955.
 25. Shoham Y & Gersho A (1988) Efficient Bit Allocation for an Arbitrary Set of Quantizers. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 36(9): 1445–1453.
 26. Lin LJ & Ortega A (1998) Bit-Rate Control Using Piecewise Approximated Rate-Distortion Characteristics. *IEEE Transactions on Image Processing* 8(4): 446–459.
 27. Uz K, Shapiro J & Czigler M (1993) Optimal Bit Allocation in the Presence of Quantizer Feedback. *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* 5: 385–388.
 28. Chen MJ, Chen LG, Chiueh TD & Lee YP (1995) New Block-Matching Criterion for Motion Estimation and Its Implementation. *IEEE Transactions on Circuits and Systems for Video Technology* 5(3): 231–236.
 29. Schuster G, Melnikov G & Katsaggelos A (1999) A Review of the Minimum Maximum Criterion for Optimal Bit Allocation Among Dependent Quantizers. *IEEE Transactions on Multimedia* 1(1): 3–17.
 30. Ivkovic G & Sankar R (2004) Algorithm for Image Quality Assessment. *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* 3: 713–716.
 31. Narita N & Sugiura Y (1997) On an Absolute Evaluation Method of the Quality of Television Sequences. *IEEE Transactions on Broadcasting* 43(1): 26–35.
 32. Wang Z, Bovik A, Sheikh H & Simoncelli E (2004) Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing* 13(4): 600–612.
 33. Wang Z, Bovik A & Lu L (2002) Why Is Image Quality Assessment So Difficult? *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* 4: 3313–3316.
 34. Carlsson P (2004) Transform Coefficient Thresholding and Lagrangian Optimization for H.264 Video Coding. M.S. thesis, Linköpings Universitet, Sweden.
 35. Ramchandran K, Ortega A & Vetterli M (1994) Bit Allocation for Dependent Quantization with Applications to Multiresolution and MPEG video coders. *IEEE Transactions on Image Processing* 3(5): 533–545.
 36. Lin LJ (1997) Video Bit-Rate Control with Spline Approximated Rate-Distortion Characteristics. Ph.D. thesis, University of Southern California, USA.
 37. Sullivan G & Bjontegaard G (2001) Recommended Simulation Common Conditions for H.26L Coding Efficiency Experiments on Low-Resolution Progressive-Scan Source Material. ITU-T Study Group 16 document VCEG-N81, Santa Barbara, CA, USA.
 38. Bjontegaard G (2001) Calculation of Average PSNR Differences Between RD-Curves. ITU-T Study Group 16 document VCEG-M33, Austin, Texas, USA.
 39. Everett H (1963) Generalized Lagrange Multiplier Method for Solving Problems of Optimum Allocation of Resources. *Operations Research* 11 (3): 399–417.
 40. Schwarz H & Wiegand T (2001) An Improved MPEG-4 Coder Using Lagrangian Coder Control. Video Coding Experts Group (VCEG) of ITU-T (ITU-T SG16 Q.6) document

- VCEG-M49, Austin, Texas, USA.
41. Wiegand T & Girod B (2001) Lagrange Multiplier Selection in Hybrid Video Coder Control. Proc. IEEE International Conference on Image Processing (ICIP) 3: 542–545.
 42. Wiegand T, Lightstone M, Mukherjee D & Campbell T (1996) Rate-Distortion Optimized Mode Selection for Very Low Bit Rate Video Coding and the Emerging H.263 Standard. IEEE Transactions on Circuits and Systems for Video Technology 6(2): 182–190.
 43. Schumitsch B, Schwarz H & Wiegand T (2005) Optimization of Transform Coefficient Selection and Motion Vector Estimation Considering Inter-Picture Dependencies in Hybrid Video Coding. Proc. SPIE 5685: 327–334.
 44. Boyd S & Vandenberghe L (2004) Convex Optimization. United Kingdom, Cambridge University Press.
 45. Sermadevi Y & Hemami S (2004) Efficient Bit Allocation for Dependent Video Coding. Proc. Data Compression Conference (DCC): 232–241.
 46. Sermadevi Y, Chen J, Hemami S & Berger T (2005) When is Bit Allocation for Predictive Video Coding Easy? Proc. Data Compression Conference (DCC): 289–298.
 47. Pan F, Li Z, Lim K & Feng G (2003) A Study of MPEG-4 Rate Control Scheme and Its Improvements. IEEE Transactions on Circuits and Systems for Video Technology 13(5): 440–446.
 48. Schwarz H, Marpe D & Wiegand T (2005) Hierarchical B pictures. Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG (ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6) document JVT-P014, Poznan, PL.
 49. Flierl M & Girod B (2003) Generalized B Pictures and the Draft H.264/AVC Video-Compression Standard. IEEE Transactions on Circuits and Systems for Video Technology 13(7): 587–597.
 50. Dufaux F & Moscheni F (1995) Motion Estimation Techniques for Digital TV. Proceedings of the IEEE 83(6): 858–876.
 51. Schuster G & Katsaggelos A (1997) A Theory for the Optimal Bit Allocation Between Displacement Vector Field and Displaced Frame Difference. IEEE Journal of Selected Areas in Communications 15(9): 1739–1751.
 52. Chen M & Willson A (1998) Rate-Distortion Optimal Motion Estimation Algorithms for Motion-Compensated Transform Video Coding. IEEE Transactions on Circuits and Systems for Video Technology 8(2): 147–158.
 53. Hoang D, Long P & Vitter J (1998) Efficient Cost Measures for Motion Estimation at Low Bit Rates. IEEE Transactions on Circuits and Systems for Video Technology 8(4): 488–500.
 54. Sangi P (2004) Selection of the Lagrange Multiplier for Block-Based Motion Estimation Criteria. IEEE International Conference on Acoustics, Speech, and Signal Processing, Montreal, Canada, 3: 325–328.
 55. Sauer K & Schwarz B (1996) Efficient Block Motion Estimation Using Integral Projections. IEEE Transactions on Circuits and Systems for Video Technology 6(5): 513–518.
 56. Chalidabhongse J & Kuo C (1997) Fast Motion Vector Estimation Using Multiresolution-Spatio-Temporal Correlations. IEEE Transactions on Circuits and Systems for Video Technology 7(3): 477–487.
 57. Zafar S, Zhang Y & Baras J (1991) Predictive Block-Matching Motion Estimation for TV Coding—Part I: Inter-Block Prediction. IEEE Transactions on Broadcasting 37(3): 97–101.
 58. Zhang Y and Zafar S (1991) Predictive Block-Matching Motion Estimation for TV Coding—Part II: Inter-Frame Prediction. IEEE Transactions on Broadcasting 37(3): 102–105.
 59. Liu B & Zaccarin A (1993) New Fast Algorithms for the Estimation of Block Motion Vectors. IEEE Transactions on Circuits and Systems for Video Technology 3 (2): 148–157.
 60. Lin CH, Wu JL & Tung YS (1997) DSRA: A Block Matching Algorithm for Near-Real-

- Time Video Encoding. *IEEE Transactions on Consumer Electronics* 43(2): 112–122.
61. Tourapis A, Au O & Liou M (1999) Fast Motion Estimation using Circular Zonal Search. *Proc. Visual Communications and Image Processing* 3653: 1496–1504.
 62. Liu LK & Feig E (1996) Block-Based Gradient Descent Search Algorithm for Block Motion Estimation in Video Coding. *IEEE Transactions on Circuits and Systems for Video Technology* 6(4): 419–422.
 63. Kossentini F & Lee YW (1997) Computation-Constrained Fast MPEG-2 Encoding. *IEEE Signal Processing Letters* 4(8): 224–226.
 64. Chung W, Kossentini F & Smith M (1995) Rate-Distortion Constrained Statistical Motion Estimation for Video Coding. *Proc. International Conference on Image Processing* 3: 184–187.
 65. Tourapis A, Au O, Liou M (2002) Highly Efficient Predictive Zonal Algorithms for Fast Block-Matching Motion Estimation. *IEEE Transactions on Circuits and Systems for Video Technology* 12(10): 934–947.
 66. Pascalis P, Pezzoni L, Mian G, Bagni D (2003) Fast Motion Estimation with Size-Based Predictors Selection Hexagon Search in H.264/AVC Encoding. *Proc. European Signal Processing Conference (EUSIPCO)*: 273–276.
 67. Ismaeil I, Docef A, Kossentini F & Ward R (1999) Efficient Motion Estimation Using Spatial and Temporal Motion Vector Prediction. *Proc. International Conference on Image Processing* 1: 70–74.
 68. Chen O (2000) Motion Estimation Using a One-Dimensional Gradient Descent Search. *IEEE Transactions on Circuits and Systems for Video Technology* 10(4): 608–616.
 69. Alkanhal M, Turaga D & Chen T (1999) Correlation Based Search Algorithms for Motion Estimation. *Proc. Picture Coding Symposium*, April 21–23, Portland, Oregon.
 70. Huang SY, Cho CY & Wang JS (2005) Adaptive Fast Block-Matching Algorithm by Switching Search Patterns for Sequences With Wide-Range Motion Content. *IEEE Transactions on Circuits and Systems for Video Technology* 15(11): 1373–1384.
 71. Francis J & de Jager G (1997) Subsampling the Search Space for Fast Block Motion Estimation. *Proc. Communications and Signal Processing*: 53–58.
 72. Jain J & Jain A (1981) Displacement Measurement and Its Applications in Interframe Image Coding. *IEEE Transactions on Communications* 29(12): 1799–1808.
 73. Ghanbari M (1990) The Cross-Search Algorithm for Motion Estimation. *IEEE Transactions on Communications* 38(7): 950–953.
 74. Lee LW, Wang JF, Lee JY & Shie JD (1993) Dynamic Search-Window Adjustment and Interlaced Search for Block-Matching Algorithm. *IEEE Transactions on Circuits and Systems for Video Technology* 3(1): 85–87.
 75. Li R, Zeng B & Liou M (1994) New Three-Step Search Algorithm for Block Motion Estimation. *IEEE Transactions on Circuits and Systems for Video Technology* 4(4): 438–442.
 76. Cheung CH & Po LM (2002) A Novel Cross-Diamond Search Algorithm for Fast Block Motion Estimation. *IEEE Transactions on Circuits and Systems for Video Technology* 12(12): 1168–1177.
 77. Po L & Ma W (1996) A Novel Four-Step Search Algorithm for Fast Block Motion Estimation. *IEEE Transactions on Circuits and Systems for Video Technology* 6(3): 313–317.
 78. Zhu S & Ma KK (2000) A New Diamond Search Algorithm for Fast Block-Matching Motion Estimation. *IEEE Transactions on Image Processing* 9(2): 287–290.
 79. Tham J, Ranganath S, Ranganath M, & Kassim A (1998) A Novel Unrestricted Center-Biased Diamond Search Algorithm for Block Motion Estimation. *IEEE Transactions on Circuits and Systems for Video Technology* 8(4): 369–377.
 80. Zhu C, Lin X & Chau LP (2002) Hexagon-Based Search Pattern for Fast Block Motion

- Estimation. *IEEE Transactions on Circuits and Systems for Video Technology* 12(5): 349–355.
81. Gallant M, Côté G & Kossentini F (1999) An Efficient Computation-Constrained Block-Based Motion Estimation Algorithm for Low Bit Rate Video Coding. *IEEE Transactions on Image Processing* 8(12): 1816–1823.
 82. Tourapis A & Au O (1999) Fast Motion Estimation Using Modified Circular Zonal Search. *Proc. IEEE International Symposium on Circuits and Systems* 4: 231–234.
 83. Tourapis A, Au O, Liou M, Shen G, & Ahmad I (2000) Optimizing the MPEG-4 Encoder—Advanced Diamond Zonal Search. *Proc. IEEE International Symposium on Circuits and Systems* 3: 674–677.
 84. Chow K & Liou M (1993) Genetic Motion Search Algorithm for Video Compression. *IEEE Transactions on Circuits and Systems for Video Technology* 3(6): 440–445.
 85. Chen Z, Zhou P, He Y & Chen Y (2002) Fast Integer Pel and Fractional Pel Motion Estimation for JVT. ISO/IEC JTC1/SC29/WG11 and ITU- T SG16 document JVT-F017, Awaji Island, JP.
 86. Cheng FH & Sun SN (1999) New Fast and Efficient Two-Step Search Algorithm for Block Motion Estimation. *IEEE Transactions on Circuits and Systems for Video Technology* 9(7): 977–983.
 87. Quaglia D & Montrucchio B (2001) Sobol Partial Distortion Algorithm for Fast Full Search in Block Motion Estimation. *Proc. 6th Eurographics Workshop on Multimedia*.
 88. Wang CN, Yang SW, Liu CM & Chiang T (2004) Hierarchical N-Queen Decimation Lattice and Hardware Architecture for Motion Estimation. *IEEE Transactions on Circuits and Systems for Video Technology* 14 (4): 429–440.
 89. Gupta G & Chakrabarti C (1995) Architectures for Hierarchical and Other Block Matching Algorithms. *IEEE Transactions on Circuits and Systems for Video Technology* 5 (6): 477–489.
 90. Cheung CK & Po LM (1997) Hierarchical Block Motion Estimation Algorithm using Partial Distortion Measure. *International Conference on Image Processing* 3: 606–609.
 91. Cheung CH & Po LM (2003) Adjustable Partial Distortion Search Algorithm for Fast Block Motion Estimation. *IEEE Transactions on Circuits and Systems for Video Technology* 13(1): 100–110.
 92. Moschetti F, Kunt M & Debes E (2003) Statistical Adaptive Block-Matching Motion Estimation. *IEEE Transactions on Circuits and Systems for Video Technology* 14(4): 417–431.
 93. Chan YL & Siu WC (1996) New Adaptive Pixel Decimation for Block Motion Vector Estimation. *IEEE Transactions on Circuits and Systems for Video Technology* 6(1): 113–118.
 94. Cheng HW & Dung LR (2004) Vario-Power ME Architecture Using Content-Based Sub-sample Algorithm. *IEEE Transactions on Consumer Electronics* 50(1): 349–354.
 95. Wang Y, Wang Y & Kuroda H (2000) Globally Adaptive Pixel-Decimation Algorithm for Block-Motion Estimation. *IEEE Transactions on Circuits and Systems for Video Technology* 10(6): 1006–1011.
 96. Chan YL, Hui WL & Siu WC (1997) Block Motion Vector Estimation Using Pattern Based Pixel Decimation. *Proc. IEEE International Symposium on Circuits and Systems* 2: 1153–1156.
 97. Nam KM, Kim JS, Park RH & Shim YS (1995) Fast Hierarchical Motion Vector Estimation Algorithm Using Mean Pyramid. *IEEE Transactions on Circuits and Systems for Video Technology* 5(4): 344–351.
 98. Han TH & Hwang SH (1998) Novel Hierarchical-Search Block Matching Algorithm and VLSI Architecture Considering the Spatial Complexity of the Macroblock. *IEEE Transac-*

- tions on Consumer Electronics 44(2): 337–342.
99. Zan J, Ahmad M, Swamy M (2004) Multiplicationless Burt and Adelsons Pyramids for Motion Estimation. *IEEE Transactions on Circuits and Systems for Video Technology* 14 (1): 136–141.
 100. Kim JS & Park RH (1992) Fast Feature-Based Block Matching Algorithm Using Integral Projections. *IEEE Journal on Selected Areas in Communications* 10(5): 968–971.
 101. Li W & Salari E (1995) Successive Elimination Algorithm for Motion Estimation. *IEEE Transactions on Image Processing* 4(1): 105–107.
 102. Baek Y, Oh HS & Lee HK (1996) Efficient Block-Matching Criterion for Motion Estimation and Its VLSI Implementation. *IEEE Transactions on Consumer Electronics* 42(4): 885–892.
 103. He ZL, Chen KK, Tsui CY & Liou M (1997) Low Power Motion Estimation Design Using Adaptive Pixel Truncation. *Proc. International Symposium on Low Power Electronics and Design*: 167–172.
 104. Natarajan B, Bhaskaran V & Konstantinides K (1997) Low-Complexity Block-Based Motion Estimation via One-Bit Transforms. *IEEE Transactions on Circuits and Systems for Video Technology* 7(4): 702–706.
 105. Lee S & Chae SI (1998) New Motion Estimation Algorithm and Its Block-Matching Criteria Using Low-Resolution Quantization. *Proc. International Technical Conference on Circuits/Systems, Computers and Communications*: 175–182.
 106. Ertürk A & Ertürk S (2005) Two-Bit Transform for Binary Block Motion Estimation. *IEEE Transactions on Circuits and Systems for Video Technology* 15(7): 938–946.
 107. Wong P & Au O (1999) Modified One-Bit Transform for Motion Estimation. *IEEE Transactions on Circuits and Systems for Video Technology* 9(7): 1020–1024.
 108. Agha S, Dwyer V & Chouliaras V (2005) Motion Estimation with Low Resolution Distortion Metric. *Electronics Letters* 41(12): 693–694.
 109. Oh TM, Kim YR, Hong WG & Ko SJ (2000) Fast Full Search Motion Estimation Algorithm Using the Sum of Partial Norms. *Proc. International Conference on Consumer Electronics*: 236–237.
 110. Jung SM, Shin SC, Baik H & Park MS (2000) Nobel Successive Elimination Algorithms for the Estimation of Motion Vectors. *Proc. International Symposium on Multimedia Software Engineering*: 332–335.
 111. Lin YC & Tai SC (1997) Fast Full-Search Block-Matching Algorithm for Motion-Compensated Video Compression. *IEEE Transactions on Communications* 45(5): 527–531.
 112. Wang HS & Mersereau R (1999) Fast Algorithms for the Estimation of Motion Vectors. *IEEE Transactions on Image Processing* 8(3): 435–438.
 113. Coban M & Mersereau R (1997) Computationally Efficient Exhaustive Search Algorithm for Rate-Constrained Motion Estimation. *Proc. International Conference on Image Processing 1*: 101–104.
 114. Brüning M & Menser B (1999) Efficient Motion Estimation Using Successive Approximation of the Error Measure. *Proc. IEEE Int. Symposium on Intelligent Signal Processing and Communication Systems*: 537–540.
 115. Wang YK & Tu GF (2000) Successive Elimination Algorithm for Binary Block Matching Motion Estimation. *Electronics Letters* 41(12): 2007–2008.
 116. Huang YW, Chien SY, Hsieh BY & Chen LG (2004) Global Elimination Algorithm and Architecture Design for Fast Block Matching Motion Estimation. *IEEE Transactions on Circuits and Systems for Video Technology* 14(6): 898–907.
 117. Toivonen T (2002) Number Theoretic Transform -Based Block Motion Estimation. M.S. thesis, University of Oulu, Finland.

118. Lee CH & Chen LH (1997) Fast Motion Estimation Algorithm Based on the Block Sum Pyramid. *IEEE Transactions on Image Processing* 6(11): 1587–1591.
119. Gao X, Duanmu C & Zou C (2000) Multilevel Successive Elimination Algorithm for Block Matching Motion Estimation. *IEEE Transactions on Image Processing* 9(3): 501–504.
120. Chen YS, Hung YP, Fuh CS (2001) Fast Block Matching Algorithm Based on the Winner-Update Strategy. *IEEE Transactions on Image Processing* 10(8): 1212–1222.
121. Nguyen V & Tan YP (2004) Fast Block-Based Motion Estimation Using Integral Frames. *IEEE Signal Processing Letters* 11(9): 744–747.
122. Jung SM, Shin SC, Baik H, Park MS (2000) Advanced Multilevel Successive Elimination Algorithms for Motion Estimation in Video Coding. *Lecture Notes in Computer Science* 1983: 121–147.
123. Ahn TG, Moon YH & Kim JH (2004) Fast Full-Search Motion Estimation Based on Multilevel Successive Elimination Algorithm. *IEEE Transactions on Circuits and Systems for Video Technology* 14(11): 1265–1269.
124. Brünig M & Niehsen W (2001) Fast Full-Search Block Matching. *IEEE Transactions on Circuits and Systems for Video Technology* 11 (2): 241–247.
125. Kim JN & Choi TS (1999) Adaptive Matching Scan Algorithm Based on Gradient Magnitude For Fast Full Search in Motion Estimation. *IEEE Transactions on Consumer Electronics* 45(3): 762–772.
126. Kim JN & Choi TS (2000) Fast Full-Search Motion Estimation Algorithm Using Representative Pixels and Adaptive Matching Scan. *IEEE Transactions on Circuits and Systems for Video Technology* 10(7): 1040–1048.
127. Kim JN, Byun SC & Ahn BH (2001) Fast Full Search Motion Estimation Algorithm Using Various Matching Scans in Video Coding. *IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews* 31(4): 540–548.
128. Montrucchio B & Quaglia D (2005) New Sorting-Based Lossless Motion Estimation Algorithms and a Partial Distortion Elimination Performance Analysis. *IEEE Transactions on Circuits and Systems for Video Technology* 15(2): 210–220.
129. Hui KC, Siu WC & Chan YL (2005) New Adaptive Partial Distortion Search Using Clustered Pixel Matching Error Characteristic. *IEEE Transactions on Image Processing* 14(5): 597–607.
130. Cheung CK & Po LM (2000) Normalized Partial Distortion Search Algorithm for Block Motion Estimation. *IEEE Transactions on Circuits and Systems for Video Technology* 10(3): 417–422.
131. Lengwehasatit K & Ortega A (2001) Probabilistic Partial-Distance Fast Matching Algorithms for Motion Estimation. *IEEE Transactions on Circuits and Systems for Video Technology* 11(2): 139–152.
132. Aimar L *et al.* (2006) x264. [cited 22 Aug 2006] Available from: <http://www.videolan.org/developers/x264.html>.
133. Noguchi Y, Furukawa J & Kiya H (1999) Fast Full Search Block Matching Algorithm for MPEG-4 Video. *Proc. International Conference on Image Processing* 1: 61–65.
134. Yang M, Chui H & Tang K (2004) Efficient Tree Structured Motion Estimation Using Successive Elimination. *IEE Proceedings on Vision, Image, and Signal Processing* 151(5): 369–377.
135. Senda Y, Harasaki H & Yano M (1995) Simplified Motion Estimation Using an Approximation for the MPEG-2 Real-Time Encoder. *IEEE International Conference on Acoustics, Speech, and Signal Processing* 4: 2273–2276.
136. Senda Y, Harasaki H & Yano M (1996) Theoretical Background and Improvement of a Simplified Half-Pel Motion Estimation. *Proc. International Conference on Image Processing* 3:

- 263–266.
137. Senda Y (2000) Approximate Criteria for the MPEG-2 Motion Estimation. *IEEE Transactions on Circuits and Systems for Video Technology* 10(3): 490–497.
 138. Hill P, Chiew T & Bull D (2006) Interpolation Free Sub-Pixel Motion Estimation for H.264. *Proc. IEEE International Conference on Image Processing (ICIP)*: 1337–1340.
 139. Bando Y, Takamura S & Yashima Y (2004) Fast Full Search Algorithm for Fractional Pel Accuracy Motion Estimation. *Proc. Picture Coding Symposium (PCS)*, San Francisco, CA, USA.
 140. Lee KH, Choi JH, Lee BK & Kim DG (2000) Fast Two-Step Half-Pixel Accuracy Motion Vector Prediction. *Electronics Letters* 36(7): 625–627.
 141. Du C, He Y & Zheng J (2004) PPHPS: A Parabolic Prediction-Based, Fast Half-Pixel Search Algorithm for Very Low Bit-Rate Moving-Picture Coding. *IEEE Transactions on Circuits and Systems for Video Technology* 13(6): 512–518.
 142. Chen Z, Du C, Wang J & He Y (2002) PPFPS—A Paraboloid Prediction Based Fractional Pixel Search Strategy for H.26L. *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)* 3: 9–12.
 143. Liu Y & Orintara S (2004) Fractional-Pel Motion Refinement Based on Hierarchical Adjustable Dual-Parabola Model. *Proc. IEEE International Symposium on Communications and Information Technologies (ISCIT)* 2: 752–755.
 144. Jullien G & Dimitrov V (1996) 2-Dimensional Transforms Using Number Theoretic Techniques. *Control and Dynamic Systems 75: Computer Techniques and Algorithms in Digital Signal Processing*, Academic Press: 155–210.
 145. Jenkins W, Jullien G & Dimitrov V (1999) *Residue Arithmetic with Applications in Digital Signal Processing*.
 146. Nussbaumer H (1982) *Fast Fourier Transform and Convolution Algorithms*. Springer-Verlag.
 147. Jullien G (1991) *Number Theoretic Techniques in Digital Signal Processing*. *Advances in Electronics and Electron Physics* 80, Academic Press: 69–163.
 148. Pollard J (1971) The Fast Fourier Transform in a Finite Field. *Mathematics of Computation* 25: 365–374.
 149. Alfredsson L (1996) *VLSI Architectures and Arithmetic Operations with Application to the Fermat Number Transform*. Ph.D. dissertation, Linköping University, Linköping, Sweden.
 150. Shakaff A, Pajayakrit A & Holt A (1988) Practical Implementations of Block-Mode Image Filters Using the Fermat Number Transform on a Microprocessor-Based System. *IEE Proceedings* 135 (4): 141–154.
 151. Agarwal R & Burrus C (1974) Fast Convolution Using Fermat Number Transform with Applications to Digital Filtering. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 22(2): 87–97.
 152. Naito Y, Miyazaki T & Kuroda I (1996) A Fast Full-Search Motion Estimation Method for Programmable Processors with a Multiply-Accumulator. *IEEE International Conference on Acoustics, Speech, and Signal Processing* 6: 3221–3224.

Original articles

- I Toivonen T, Merritt L, Ojansivu V & Heikkilä J (2007) A New Rotation Search for Dependent Rate-Distortion Optimization in Video Coding. Proc. 32nd International Conference on Acoustics, Speech, and Signal Processing (ICASSP) 1: 1165–1168.
- II Toivonen T & Heikkilä J (2006) Reduced Frame Quantization in Video Coding. In: 9th International Workshop VLBV Proceedings, Lecture Notes in Computer Science 3893, Springer-Verlag: 61–67.
- III Toivonen T & Heikkilä J (2006) Motion Vector Refinement in Video Coding Based on Statistical Distribution. Proc. 13th International Conference on Systems, Signals & Image Processing (IWSSIP): 39–42.
- IV Toivonen T & Heikkilä J (2003) A New Rate-Minimizing Matching Criterion and a Fast Algorithm for Block Motion Estimation. Proc. IEEE International Conference on Image Processing (ICIP) 2: 355–358.
- V Toivonen T & Heikkilä J (2004) Fast Full Search Block Motion Estimation for H.264/AVC with Multilevel Successive Elimination Algorithm. Proc. IEEE International Conference on Image Processing (ICIP) 3: 1485–1488.
- VI Toivonen T & Heikkilä J (2006) Improved Unsymmetric-Cross Multi-Hexagon-Grid Search Algorithm for Fast Block Motion Estimation. Proc. IEEE International Conference on Image Processing (ICIP): 2369–2372.
- VII Toivonen T & Heikkilä J (2003) Efficient Method for Half-Pixel Block Motion Estimation Using Block Differentials. In: Visual Content Processing and Representation, 8th International Workshop VLBV Proceedings, Lecture Notes in Computer Science 2849, Springer-Verlag: 225–232.
- VIII Toivonen T & Heikkilä J (2006) Video Filtering with Fermat Number Theoretic Transforms using Residue Number System. IEEE Transactions on Circuits and Systems for Video Technology 16(1): 92–101.
- IX Toivonen T, Heikkilä J & Silvén O (2002) A New Algorithm for Fast Full Search Block Motion Estimation Based on Number Theoretic Transforms. Proc. 9th International Workshop on Systems, Signals and Image Processing (IWSSIP): 90–94.

Reprinted with permission from IEEE (Papers I, IV, V, VI, and VIII) and Springer-Verlag (II and VII).

Original publications are not included in the electronic version of the dissertation.

ACTA UNIVERSITATIS OULUENSIS
SERIES C TECHNICA

273. Oiva, Annukka (2007) Strategiakeskeinen kyvykkyyden johtaminen ja organisaation strateginen valmius. Kahden johtamismallin testaus
274. Jokinen, Hanna (2007) Screening and cleaning of pulp—a study to the parameters affecting separation
275. Sarja, Tiina (2007) Measurement, nature and removal of stickies in deinked pulp
276. Tóth, Géza (2007) Computer modeling supported fabrication processes for electronics applications
277. Näsi, Jari (2007) Intensified use of process measurements in hydrometallurgical zinc production processes
278. Turtinen, Markus (2007) Learning and recognizing texture characteristics using local binary patterns
279. Sarpola, Arja (2007) The hydrolysis of aluminium, a mass spectrometric study
280. Keski-Säntti, Jarmo (2007) Neural networks in the production optimization of a kraft pulp bleach plant
281. Hamada, Atef Saad (2007) Manufacturing, mechanical properties and corrosion behaviour of high-Mn TWIP steels
282. Rahtu, Esa (2007) A multiscale framework for affine invariant pattern recognition and registration
283. Kröger, Virpi (2007) Poisoning of automotive exhaust gas catalyst components. The role of phosphorus in the poisoning phenomena
284. Codreanu, Marian (2007) Multidimensional adaptive radio links for broadband communications
285. Tiikkaja, Esa (2007) Konenäköä soveltavan kuituanalysaattorin ja virtauskenttäfraktionaattorin mittausten yhteydet kuumahierteen paperitekniisiin ominaisuuksiin. Kokeellinen tutkimus
286. Taparugssanagorn, Attaphongse (2007) Evaluation of MIMO radio channel characteristics from TDM-switched MIMO channel sounding
287. Elsilä, Ulla (2007) Knowledge discovery method for deriving conditional probabilities from large datasets
288. Perkkiö, Miia (2007) *Utilitas* restauroinnissa. Historiallisen rakennuksen käyttötarkoituksen muutos ja funktionaalinen integriteetti

Book orders:
OULU UNIVERSITY PRESS
P.O. Box 8200, FI-90014
University of Oulu, Finland

Distributed by
OULU UNIVERSITY LIBRARY
P.O. Box 7500, FI-90014
University of Oulu, Finland

S E R I E S E D I T O R S

A
SCIENTIAE RERUM NATURALIUM
Professor Mikko Siponen

B
HUMANIORA
Professor Harri Mantila

C
TECHNICA
Professor Juha Kostamovaara

D
MEDICA
Professor Olli Vuolteenaho

E
SCIENTIAE RERUM SOCIALIUM
Senior Assistant Timo Latomaa

E
SCRIPTA ACADEMICA
Communications Officer Elna Stjerna

G
OECONOMICA
Senior Lecturer Seppo Eriksson

EDITOR IN CHIEF
Professor Olli Vuolteenaho

EDITORIAL SECRETARY
Publications Editor Kirsti Nurkkala

ISBN 978-951-42-8694-0 (Paperback)

ISBN 978-951-42-8695-7 (PDF)

ISSN 0355-3213 (Print)

ISSN 1796-2226 (Online)

