Helsinki University of Technology, Electronic Circuit Design Laboratory
Report 41, Espoo 2005

Analog Parallel Processor Solutions for Video Encoding

Lauri Koskinen

Dissertation for the degree of Doctor of Science in Technology to be presented with due permission of the Department of Electrical and Communications Engineering for public examination and debate in Auditorium S4 at Helsinki University of Technology (Espoo, Finland) on the 16th of December, 2005, at 12 o'clock.

Helsinki University of Technology
Department of Electrical and Communications Engineering
Electronic Circuit Design Laboratory

Teknillinen korkeakoulu
Sähkö- ja tietoliikennetekniikan osasto
Piiritekniikan laboratorio

Distribution:
Helsinki University of Technology
Department of Electrical and Communications Engineering
Electronic Circuit Design Laboratory
P.O.Box 3000
FIN-02015 HUT
Finland
Tel. +358 9 4515322
Fax: +358 9 4512269

ISBN 951-22-7958-4
ISSN 1455-8440

Otamedia Oy
Espoo 2005

Abstract

This thesis deals with Cellular Nonlinear Network (CNN) analog parallel processor networks and their implementations in current video coding standards. The target applications are low-power video encoders within 3^{rd} generation mobile terminals. The video codecs of such mobile terminals are defined by either the MPEG-4/H.263 or H.264 video standard. All of these standards are based on the block-based hybrid approach. As block-based motion estimation (ME) is responsible for most of the power consumption of such hybrid video encoders, this thesis deals mostly with low-power ME implementations.

Low-power solutions are introduced at both the algorithmic and hardware levels. On the algorithmic level, the introduced implementations are derived from a segmentation algorithm, which has previously been partly realized. The first introduced algorithm reduces the computational complexity of ME within an object-based MPEG-4 encoder. The use of this algorithm enables a 60% drop in the power consumption of Full Search ME. The second algorithm calculates a near-optimal block-size partition for H.264 motion estimation. With this algorithm, the use of computationally complex Lagrange optimization in H.264 ME is not required. The third algorithm reduces the shape bit-rate of an object-based MPEG-4 encoder.

On the hardware level a CNN-type ME architecture is introduced. The architecture includes connections and circuitry to fully realize block-based ME. The analog ME implemented with this architecture is capable of lower power than comparable digital realizations. A 9x9 test chip has also been realized. Additionally implemented is a digital predictive ME realization that takes advantage of the introduced partition algorithm. Although the IC layout of the ME algorithm was drawn, the design was verified as an FPGA.

Keywords: Motion Estimation, Cellular Neural/Nonlinear Networks, MPEG-4, H.264, Parallel Processing

Preface

The research for this thesis was carried out during 2000-2005 at the Electronic Circuit Design Laboratory (ECDL) of the Helsinki University of Technology. The research was funded by the Academy of Finland.

Could it be that, with the exception of a few bad knees, nothing in my life has really changed since I wrote my previous preface? Or is it that I am an unimaginative person? Whatever the reason, I only have to offer you, dear reader, an updated version of my Licentiate's thesis preface:

The thing that still keeps me at the ECDL lab is the un-job-like atmosphere. Although during the course of this research I have realized that too much independence may not always be a blessing. Anyway, all that the ECDL atmosphere is not is summed up with a few verses from the DK's "Well Paid Scientist":

> You're a well paid scientist. You only talk in facts.
> You know you're always right. 'Cause you know how to prove it. Step by step.
> A PhD to show you're smart. With textbook formulas.
> But you're used up. Just like a factory hand.
> Something is wrong here. You won't find in on a shelf.
> You're well paid. You're well trained. You're tied to a rack.

First, I would like to thank Professor Timo D. Hämäläinen and Dr. Piotr Dudek for reviewing my thesis. For the atmosphere at ECDL, the thanks first and foremost goes to Professor Kari Halonen for his flexibility. For the atmosphere at the lab, my thanks go out to all my co-workers but especially: Esis, Helena, co-supervisor Mika, Pikkis, Saska, and Väiski. For all my roommates, past and present, thanks for putting up with my music selection. Thanks to Saku for his contribution to this thesis. Last but not least a special thank you goes to Asko for being a good friend and enlightening me to the different aspects of boring music. Finally, for making this thesis possible and for offering a very interesting research topic, huge thanks go to my supervisor and friend Apa. Take it easy!

For making my life fun outside the office: The stiffs of Helsingin Hämäys and the extraordinary musicians of FINTHOR (keep on rockin'). Especially Eläke-Mikko, Strakile, and Flamin' Chili Ass Teemu. Special thanks to Jykä for fixing up my car(s). Huge thanks also goes out two of my oldest friends: Antti and G (lay off the booze!).

This thesis is dedicated to my mother and father without whom none of this would have been possible.

Thanks to Jenny and Antti Wihuri foundation, Ulla Tuominen foundation, Tekniikan edistämissäätiö, and the Finnish Society of Electronics Engineers for support.

Outi, in the final half year in the making of this thesis, I have, more than ever, seen how important you are in my life.

Espoo, 2005.

Lauri Koskinen

Contents

Symbols and Abbreviations

//	Division with Round-to-Nearest
2.5G	2.5 Generation Cellular Wireless Network
3G	3rd Generation Cellular Wireless Network
3GPP	Third Generation Partnership Project
4:2:0	Picture Format.
4:2:2	Picture Format.
4:4:4	Picture Format.
α	Switching Dynamics Parameter
β	MOS Transconductance Parameter
γ	MOS Transistor Body Factor
λ	Lagrange Parameter
$\lceil\,\rceil$	Rounding with Truncation Towards ∞
$\lfloor\,\rfloor$	Rounding with Truncation Towards Zero
μ	Carrier Mobility
$\otimes$	Scalar Multiplication
ρ	Macroblock Motion Ratio
σ	Standard Deviation
σ^2	Variance
τ	On-Time Ratio
θ	Rotation Angle
ϖ	Number of Operations Ratio
$\tilde{s}_{m,n}$	Approximate Random Symbol
A	CNN Feedforward Template
a	4x4 DCT approximation coefficient

A_β	Transconductance Proportionality Constant
A_{V_T}	Threshold Voltage Proportionality Constant
B	CNN Feedback Template
b	4x4 DCT approximation coefficient
b_m	m:th Bit
B_w, B_f	Number of Macroblocks
BW	Bandwidth
$c(k), c(l)$	DCT Constant
C_{dg}	MOS Transistor Drain-Gate Overlap Capacitance
C_g	MOS Transistor Gate Capacitance
$C_{i,j}, C_{k,l}$	CNN Cell
C_{ox}	Oxide Capacitance
D	Distortion
d_x, d_y	Translation Vector
E	Constant Matrix
e	Prediction Signal
f	Clock Frequency
g_m	MOS Transistor Transconductance
I	Current
i	Variable
I_{DC}	Direct Current
I_i	Coding Option
I_{RMS}	RMS Current
I_{VDD}	Power Supply Current
I_y	Luminance
J	Lagrangian Cost Function
j	Variable
K	Exp-Golomb Constant
L	MOS Transistor Gate Length
L_α, L_1, L_2	MVFAST Motion Activity Thresholds
M	Block Width Parameter

MV_x, MV_y	Motion Vector
N	Block Height Parameter
N_r	CNN Cell Neighborhood
P	Power Consumption
p	Distortion Measure Parameter
q	Quantization Coefficient
Q_{inj}	Injected Charge of MOS Switch
q_{v2}	H.264 Quantization Vector
R	Rate
r	Search Area
s	Scaling Ratio
T, T_c, T_r	Transform Matrices
$T1$	High Frequency Transform Coefficients with the Value ± 1.
th	Segmentation threshold
U	Inverse Transform Matrix
$u_{i,j}$	CNN Cell Input
V	Voltage
V_{dd}	Operating Voltage
V_g	MOS Transistor Gate Voltage
V_T	MOS Transistor Threshold Voltage
W	MOS Transistor Gate Width
X	2D Matrix
x	Horizontal Size Parameter
$x0, y0$	Macroblock Top-Left Pixel
$x_{i,j}$	CNN Cell State
x_k, x_{kl}	Random Symbol
Y	Transformed 2D Matrix
y	Vertical Size Parameter
$y_{i,j}$	CNN Cell Output
y_k, y_{kl}	Transformed Symbol
z	CNN Bias Template

A/D	Analog to Digital Conversion
AGU	Address Generation Unit
AME	Analog Motion Estimation
B (B-frame)	Bi-Predictive Frame
BAB	Binary Alpha Plane
CABAC	Context-Based Adaptive Arithmetic Encoding
CAE	Context-Based Arithmetic Encoding
CAVLC	Context-Based Adaptive VLC
CCF	Cross-Correlation Function
CIF	Common Intermediate Format, 352X288 pixels
CMOS	Complementary Metal-Oxide-Semiconductor
CNN	Cellular Nonlinear Network
CNNUM	CNN Universal Machine
COD	Not Coded Bit
CPU	Central Processing Unit
D/A	Digital to Analog Conversion
DCT	Discrete Cosine Transform
DFF	Delay Flip-Flop
DivX, WM9, XviD	Commercial Audio-Video Codecs
DNL	Differential Non-Linearity
DSP	Digital Signal Processor
DTCNN	Discrete-Time CNN
DVD	Digital Video Disc
ECDL	Electronic Circuit Design Laboratory
EDGE	Enhanced Data Rates for GSM Evolution
ENOB	Effective Number of Bits
fps	Frames per Second
FRext	H.264 Fidelity Range Extension Profile
FS	Full Search
FSRCNN	Full Signal Range CNN
GPRS	General Packet Radio Service

GSM	Global System For Mobile Telecommunications
H.261,H.262,H.263,H.264 (H.26L)	ITU-T Video Standards
HD	High Definition
I (I-frame)	Intra Frame
IC	Integrated Chip
IEC	International Electrotechnical Commission
INL	Integral Non-Linearity
ISO	International Organization for Standardization
ITU-T	International Telecommunication Union Standardization Sector
LSB	Least Significant Bit
MB	Macroblock, 16x16 pixels
MBMS	Multimedia Broadcast/Multicast Service
MC	Motion Compensation
MCM	Multichip Module
ME	Motion Estimation
MIMD	Multiple Instruction, Multiple Data
MMS	Multimedia Messaging Service
MOS	Metal-Oxide-Semiconductor
MPEG	Motion Pictures Experts Group of ISO/IEC
MPEG-1,MPEG-2,MPEG-4	ISO/IEC Video Standards
MSB	Most Significant Bit
MV	Motion Vector
MVFAST	Motion Vector Adaptive Search Technique
NTSC	American Color Encoding System
P (P-frame)	Inter Frame
PAL	European Color Encoding System
PCS	Packet-Switched Conversational Service
PE	Processing Element
PMVFAST	Predictive Motion Vector Adaptive Search Technique
PPS	Packet-Switched Streaming Service
PSNR	Peak Signal-to-Noise Ratio

QCIF	Quarter Common Intermediate Format, 176X144 pixels
QP, Q_p	Quantization Parameter
R/D	Rate-Distortion.
RGB	Red, Green, Blue. Primary colors
RLC	Run-Length Coding
RMS	Root Mean Square
SAD	Sum of Absolute Differences
SAR	Successive Approximation Register
SD	Standard Definition
SI	Switched-Current
SIMD	Single Instruction, Multiple Data
SIP	System-in-Package
SOC	System-on-Chip
SOP	System-on-Package
SRAM	Static Random Access Memory
SSD	Sum of Squared Differences
SVGA	Super VGA, 800X600 pixels
TSS	Three Step Search
UMTS	Universal Mobile Telecommunications System
VGA	Video Graphics Array, 640X480 pixels
VLC	Variable Length Coding
VLIW	Very Long Instruction Word
VO	MPEG-4 Video Object
VOP	MPEG-4 Video Object Plane
YUV, YCbCr	Color Models

Chapter 1

Introduction

1.1 Motivation

Current mobile phones incorporate megapixel-sized cameras. Although these camera sizes would allow the capture of near-HD (High Definition) video, phones incorporating such cameras encode video with a very small picture size. For example the Nokia 6630 3G mobile phone, which contains a 1280 x 960 pixel camera, encodes only QCIF-sized (144 x 176 pixels) video.

There are two main reasons for this small frame size: the power consumption of the video encoding and available memory. Of these, power consumption is dominant. In the example phone, the video encoding consumes roughly 2.5 times more power than a 3G phone call. In the near future, technology scaling will enable the integration of more memory into the mobile terminals, thus solving the encoding memory dilemma. However, any reduction in power consumption due to technology scaling will be offset by the need to enlarge the frame size of the video. To achieve acceptable playback quality with larger displays, such as Standard Definition (SD) television, the frame size would have to be in the range of VGA (Video Graphics Array, 640 x 480 pixels) or SVGA (Super VGA, 800 x 600 pixels). This would also enable the storage of the video on a remote server, thus effectively realizing a video camera with a memory size limited only such a remote server.

In addition to the small frame size, future video standards will achieve better compression with higher-complexity encoding. An example of this can be seen in the latest standard H.264, which is significantly more complex than previous standards. Thus, with conventional digital implementations in the near future, higher-complexity video encoders, and the demand for larger frame sizes, will offset any gains in power consumption due to technology scaling. Such critical demands can only be fulfilled by novel hardware architectures.

1.1.1 Video Coding Requirements in Mobile Terminals

Video coding is used in a wide range of different applications. On the high end of the application range is the broadcast or storage of standard-definition (SD) or high-definition (HD) television. The broadcast medium could be a cable or satellite network and the storage medium a digital video disc (DVD). The low end consists of video telephony applications sent over low bit-rate channels. Depending on the target application, the requirements for a video encoder differ vastly.

For video encoders in mobile terminals, the theoretical data bandwidth upper limits are 384 kbit/s for 2.5G (GSM-GPRS, GSM-EDGE) systems and 2 Mbit/s for 3G (UMTS) systems. The current frame sizes used for streaming are QCIF (Quarter Common Intermediate Format, 176X144 pixels) or CIF (352X288 pixels). The current data bandwidth already enables the transmission of VGA-sized video. The theoretical data bandwidth limit of 3G networks enables even larger frame-sizes.

Three major service categories for video transmission from mobile terminals exist [1]:

1) Base station to mobile video packet-switched streaming services (PPS) [2] or multimedia broadcast/multicast services (MBMS) [3].

2) Multimedia messaging service (MMS) containing video [4].

3) Video telephony and conferencing with packet-switched conversational services (PCS) [5].

Of these, 1) does not require any encoding in the mobile terminal; in 2), the coded video is stored locally before transmission. Thus, in both of these cases, the encoding does not have any real-time constraints.

For the conversational services of 3), the maximum end-to-end delay cannot be more than 250ms [6]. This delay includes encoding, transmission, and decoding. In encoding, the delay constraint prohibits the use of future reference frames, as this would induce extra delay. The delay constraint also places demand on the error-resiliency tools on the encoder. Error-resilient video coding is outside the scope of this thesis.

Video coding algorithms are characterized by very high computational complexity. The high computational complexity combined with real-time and low-power requirements is difficult to realize. Block-based motion estimation (ME) is the main source of compression from the first video coding standard MPEG-1 up to the current standards MPEG-4 and H.264. As the other compression techniques of these standards can be efficiently implemented with conventional digital realizations, and as motion estimation is also the main source of an encoder's power consumption (up to 80%), it is reasonable to concentrate the novel architecture research on motion estimation.

From the criteria above the objectives of this research can now be collected together:

1. Real-time operation with 25-30 fps (PAL/NTSC).
2. NTSC SD frame size (640x480).
3. Lower power consumption than comparable digital realizations.
4. Maximum end-to-end delay of 250ms.

1.1.2 Related Work

Several full conventional audio-video codecs have been presented. In [7], a 29 mW CIF (288 x 352 pixels) 15fps MPEG-4 Simple Profile Level 1 (SP@L1) video codec is presented. The codec uses an external frame buffer, dedicated module architectures and a fast search motion estimation algorithm. In [8], a 120mW VGA 30fps MPEG-4 SP@L4a is presented. The codec uses an eDRAM frame buffer, dedicated module architectures, and a fast search motion estimation algorithm. The codec also decodes H.264 Baseline profile Level 1.2 (BP@L1.2) At the present H.264 is implemented only in HD applications, such as [9], and videoconferencing applications, such as [10], neither of which are low-power applications.

In recent years, few separate ME realizations have been presented. Current ME realizations are embedded within full audio-video codecs such as the ones described above. Comparing the presented work to such implementations is difficult due to the fact that little specific information (such as power consumption) on the ME part is available. In [11], a 0.4mW (QCIF@15fps, 0.85MHz) / 2.5 mW (CIF@30fps, 6.75MHz) motion estimation IC using a gradient search algorithm is presented. The IC has a 1.0V power supply and is implemented with 0.18μm technology. As the design does not incorporate frame memory the stated power consumption figures do not include the data transfer between the frame memory and local search area memories. In [11], it is also estimated that the power consumption of a Full Search QCIF@15fps ME IC would be in the range of 20mW. In [12], a 16.2mW QCIF@15fps motion estimation IC using pixel wordlength truncation is presented. The IC has a 20MHz clock frequency (the operating voltage is not stated) and is implemented with 0.18 μm technology. The design incorporates frame memories whose portion of the power consumption is 11.7mW. In [13], a 1920x1080 HDTV@30fps ME core is presented. The ME core is designed for a power supply of 1.0V and a clock frequency of 81MHz and is implemented with 0.13μm technology. The estimated power consumption is 65mW without frame memories.

For H.264 [14] presents a 720x480 SVGA@30fps systolic array design that incorporates Full Search for the seven different block sizes of H.264. With 0.35 μm technology and a clock frequency of 67 MHz (the operating voltage is not stated) the design has a simulated power consumption of 737 mW without frame memories. In [15], a QCIF@15fps implementation using variable block-size Full Search is presented. With 0.13 μm technology, a clock frequency of 6.7 MHz, and an operating voltage of 1.2V the design has a simulated power consumption of 9.1 mW without frame memories. Both of these variable block-size ME implementations operate by computing the distortion measure values for the smallest block size and then combining these values to form the distortion measures for the larger block sizes. Also, neither of these designs comments on the choice of the optimal block size.

1.1.2.1 Cellular Nonlinear Networks in Video Coding Applications

On the algorithmic level, the Cellular Nonlinear Network (CNN) paradigm is extremely well suited to video coding applications due to its parallel nature, that enables assigning a single processor to each pixel in a frame. If the output of a CNN is to be used as the input to a video codec, the inaccuracies inherent in analog computation must be taken into account. The peak signal-to-noise ratio (PSNR) (Eq. 2.18) of an Intra coded image should be over 30 dB. The accuracy of analog computation is usually only 6-7 bits with realistic transistor sizes [16]. A small amount of analog noise can easily deteriorate the PSNR of an image to under 30 dB, giving possible commercial CNN video applications an inferior starting point in comparison to digital video applications, especially in large picture size applications. For low bit-rate applications with a small picture size, such as the upcoming mobile video applications, the criterion above is less rigid. Some noise is added due to the quality of the current mobile lens and sensor array combinations, which can mask a part of the noise added by the analog inaccuracies. Even in this case, the effect of the inaccuracies inherent in analog CNN computation cannot be overly stressed.

To avoid the analog inaccuracies, the CNN can be used as an auxiliary processor, as is suggested in this thesis (Fig. 1.1). Then the original digital frame is stored in a non-volatile memory and referenced with the output of the CNN computation (i.e. segmentation or motion estimation). In this case, as the original frame is used as the input to a video codec, the analog inaccuracies do not degrade the video compression.

Several CNN motion estimation approaches have been suggested but they tend to be algorithmic implementations ([17], [18]) or they are not block-based ([19], [20]). In all these CNN ME cases the effect of the analog inaccuracies on the motion estimation result has not been taken into account. Digital CNN, such as [16], eliminates the inaccuracy problem but has the disadvantage of a large silicon area compared to extremely efficient existing VLSI ME implementations.

When discussing motion estimation implementations (and ME in general) it is important to differentiate between block-based motion estimation and computer vision (e.g. visual motion). The former is an integral part of all current video standards. The concept of block-based ME is to find the best match between two or more blocks of data. On the other hand, computer vision [21] applications try to find the true motion or optical flow in a video sequence. This is achieved with a bank of spatio-temporal filters with different spatial and temporal tunings. Due to the required length of these filters, the use of optical flow is not feasible in video coding applications. CNN is extremely well suited for these computer vision applications [22] but, as is stated above, such applications are not very well suited for block-base ME.

For the segmentation used only in MPEG-4 Core profile applications, CNN is extremely well suited [23], [17], but the advancement of the Core profile has been slowed by the absence of segmentation algorithms that are reliable with various types of video content.

1.1.2.2 Motion Estimation with Analog Circuits

Block-based analog motion estimation has been considered previously in [24]. Also, in [25], another analog ME realization has been designed simultaneously and independently of this work. In both of these designs, although the computation, memory, and data transfer are analog, the architectures resemble conventional digital ME processors (shown in Fig. 2.7). This is in contrast to this work which is an inter-connected analog parallel processor architecture. In both [24] and [25], only the picture quality results are presented which, without presenting the effect on bit-rate, is not fully meaningful and makes comparison difficult.

Additionally, the effect of error in the ME distortion measure has been studied in [26] and [27].

1.2 Research Contribution

The basis for the work was the CNN segmentation chip [28] developed at the ECDL by Professor Ari Paasio. The chip realizes the black-and-white part of the segmentation algorithm introduced in [29]. This thesis researches the feasibility of using the designed chip or similar implementations in current video coding standards. The target of this research is low-power block-based motion estimation. Methods of achieving the low power are introduced on both the algorithmic and hardware design levels.

The basic principle of the research is portrayed in Fig. 1.1 where two different methods are shown. The first (analog ME) portrays a stand-alone analog motion estimator. This approach is the objective of the hardware design, where a CNN-type cell for analog ME is developed. The effect of the unidealities inherent in analog computa-

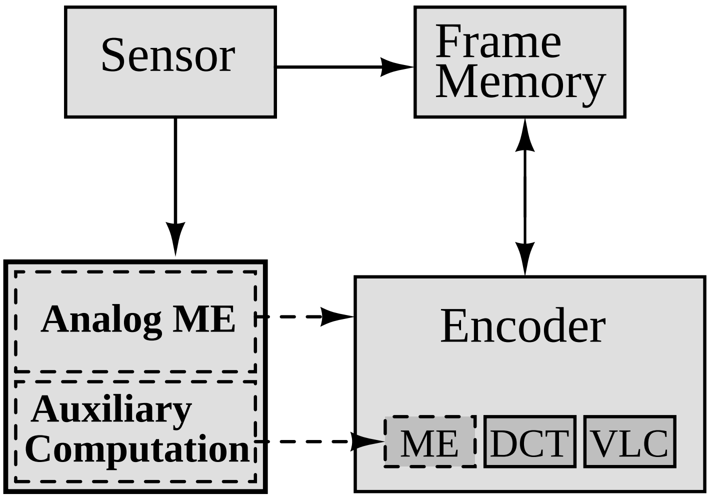


Figure 1.1 Basic principle of the suggested approach.

tion on the motion estimation result are also researched. The second method (auxiliary computation) portrays a method of sending additional data to the conventional ME engine. This data enables lower computational complexity and thus lower power consumption for the ME engine. This approach is the objective of the algorithmic design where the segmentation algorithm of [29] is examined and two different low-power motion estimation implementations are developed. Both of these implementations transmit auxiliary information to a conventional motion estimation realization. This auxiliary information enables a reduction in the computational complexity of conventional motion estimation engines.

The material published in [30] through [37] is included within this thesis. The author bears the main responsibility for the research included within these publications, with the exception of [33], where the author was responsible for mainly the algorithmic-level design. The A/D- and D/A-converters used in [35] and [43] were designed by Prof. Ari Paasio. The layout of the converters was designed by Asko Kananen M.Sc. Mr. Kananen also designed the layout of a single cell in [35] and [43]. The quadratic circuits used in [36] were originally designed by Dr. Mika Laiho and modified by the author. Publications [38] through [44] are also relevant to this thesis. The author again bears the main responsibility for the research included within these publications, with the exception of [40] where the motivation behind the research was partly due to [39]. The measurement results in Chapter 7 are based on the measurement results of [45]. These measurements were conducted by Mr. Kananen.

1.3 Organization of the Thesis

The thesis is divided into eight chapters. Chapters 2 and 3 describe the basic theory of the thesis. Chapter 2 describes the main features of current video standards. Only the features within the parts of the standards (MPEG-4 Simple and Core profiles, H.264 Baseline profile) that are relevant to this work are described. Also described in Chapter 2 is the compression theory behind these standards. Although this thesis deals solely with motion estimation, the other compression methods (DCT, VLC) are also reviewed since the analysis of the results is not meaningful without an understanding of all of the used compression methods. Reviewing the compression theory is also relevant since the MPEG-4 and H.264 encoders used in Chapters 5 through 8 are fully designed in the course of this research as opposed to using publicly available codecs such as [46]. Chapter 3 describes briefly the basic CNN theory.

Chapters 4 [30] and 5 [31], [32] depict novel methods of combining CNN algorithms with block-based motion estimation. These chapters introduce algorithms that reduce the motion estimation computational complexity of MPEG-4 and H.264, respectively. Chapter 5 also describes a digital motion estimation implementation [33] that utilizes the algorithm presented in the same chapter. A layout of the digital implementation is also drawn. As the chip has not been realized, test measurements have been carried out on an FPGA.

Chapter 6 [34] introduces CNN templates which reduce the bit-rate of the shape information of an MPEG-4 Core profile encoder.

Chapter 7 [35], [36], [37] introduces a novel 3^{rd}-neighborhood connection for CNN or CNN-type arrays. Also introduced is an account of how analog block-based motion estimation can be realized with this connection and simple logic. Although the analog ME is introduced here in context with MPEG-4, the AME can be utilized with all codecs that incorporate ME. Further analyzed are the different error sources in an analog motion estimation implementation and their effect on the rate-distortion of video coders using such motion estimation. An analog motion estimation test chip partly based on these results is also introduced.

Chapter 2

Video Coding Standards Overview

2.1 Generalized Video Coding

All of the major video coding standards (H.261 [47], MPEG-1 [48], MPEG-2 [49] (H.262), MPEG-4 [50] (H.263 [51]), H.264 [52]) and commercial video codecs such as XviD [53], DivX [54], and Windows Media Video 9 [55] are based on block-based hybrid video coding [56], as shown in Fig. 2.1. The coding is a hybrid of exploiting spatial redundancy with transform coding and temporal redundancy with inter-frame prediction. Other coding architectures, such as the block-based motion-compensated wavelet codec MC-EZBC [57], exist, but have not become common.

2.1.1 Basic Video Coder

The input video frame is first partitioned into 16x16 or smaller samples (macroblocks) which are then coded. In intra mode, the macroblocks are coded without reference to other frames. The intra coding is achieved by, first, transform coding (Chap. 2.2.2), then quantizing the transformed coefficients (Chap. 2.2.3) and entropy coding (Chap. 2.2.1) of the quantized symbols. In the inter mode, the prediction error (Chap. 2.2.4) between the current block and a reference block corresponding to the best predictor in another frame is coded with the intra coding techniques. With each inter coded block, a displacement vector (motion vector) is also transmitted. The motion vector refers to the position of the reference block. Depending on the standard, the intra/inter decision can be at the macroblock level. The common terms used in context with the intra/inter decision are I-frame (intra), P-frame (inter, one MV) and B-frame (inter, bi-predictive ME).

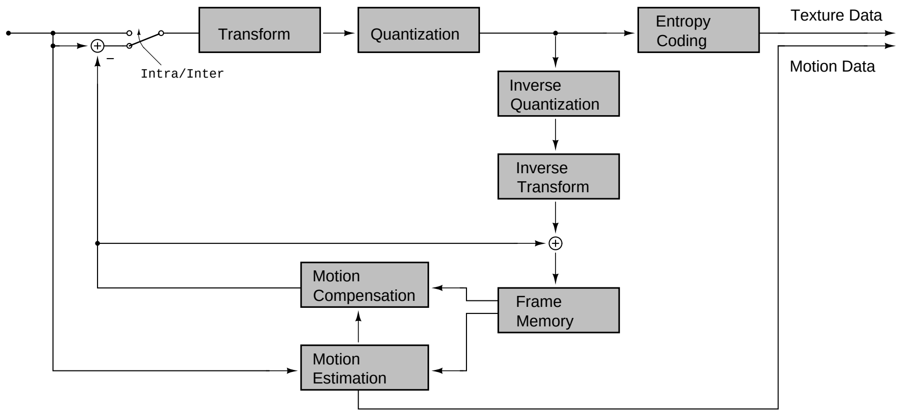


Figure 2.1 Block diagram of a generalized hybrid transform motion prediction encoder

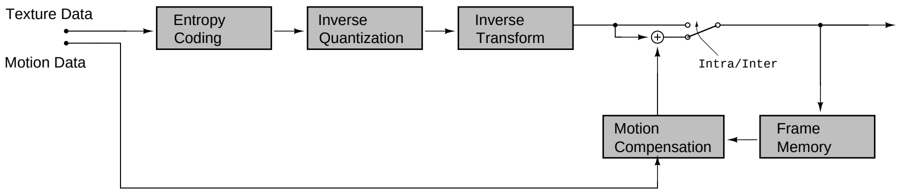


Figure 2.2 Block diagram of a generalized hybrid transform motion prediction decoder

The advantage of this particular hybrid approach is that motion estimation, which can take up to 80% of computational power [58], does not have to be performed at the decoder. A generalized decoder is shown in Fig. 2.2.

2.2 Video Compression Theory

As the human visual system perceives brightness and color information separately, current digital video standards use the YUV color model which includes a luminance component (Y) and two color difference or chrominance components (Cb or U) and (Cr or V). In this model, Y represents brightness (gray-scale information) and Cb and Cr represent the extent to which the color deviates from gray towards blue and red, respectively. An additional advantage of decoupling the color information is that the human visual system is less sensitive to changes in color information than changes in luminance [59], meaning that the color information can be represented with a smaller bandwidth.

A digital image is represented by a two-dimensional frame of picture elements (pixels or pels). The most-used picture format is the 4:2:0 format [60], where both chrominance components are subsampled by 2 in the vertical and horizontal directions.

Real video sequences exhibit redundancy both spatially and temporally. Spatial redundancy is most evident in smooth surfaces with low spatial frequencies. In such areas, the correlation of adjacent parts of the image is strong. Temporally the redundancy is most evident when no change happens in the video sequence. In current video standards, the redundancy is taken advantage of with three basic coding methods: entropy coding (Variable Length Coding), transform coding, and motion estimation/compensation. Entropy coding and transform coding reduce the spatial redundancy, while motion estimation/compensation reduces the temporal redundancy. Compression is also achieved by taking advantage of the perceptual characteristics of human vision. Thus, the loss introduced by compression may not be detectable to the viewer.

2.2.1 Variable Length Coding

Entropy coders, such Huffman coders [61], arithmetic coders [62], and Golomb coders [63] are commonly called Variable Length Coders (VLC). VLC coders try to minimize the average length of a symbol group. This is achieved by mapping the group of symbols to codewords. The length of each codeword is inversely dependent on the probability of the symbol the codeword represents. A codeword table is achieved by coding the symbol with the highest probability with the smallest number of bits, the symbol with the second lowest probability with the second lowest number of bits, and so on. None of the shorter codewords may form the prefix of a longer codeword. The efficiency of the coding is bound to the accuracy of the probability table. Below is a short summary of various VLC methods used in current video standards.

2.2.1.1 Run-Length Coding

In practice, the VLC is achieved with run-length coding (RLC), which maps a block of coefficients into a vector of three value format symbols. Many different scans to map the block into a vector exist [64]p.159. The three value format is such that the first value represents the actual level (LEVEL) of the current coefficient, the second value represents the number of zero-level coefficients (RUN) before the current coefficient and the third value indicates the last non-zero level coefficient (LAST). The RLC tables exist for the most common of these last-run-level combinations. An example of such a RLC table is shown in Table 2.1 [50]. The advantage of combined scanning and RLC is that the blocks produced by hybrid predictive transform coding frequently contain a large number of zeros that are situated in the higher spatial frequencies. The scan maps such blocks into vectors containing long runs of zeros at the end of the vector. Such vectors can be coded efficiently with RLC.

VLC CODE	LAST	RUN	LEVEL
10s	0	0	1
1111s	0	0	3
010101s	0	0	6
0010111s	0	0	9
00011111s	0	0	10
0111s	1	0	1
000011001s	0	11	1
00000000101s	1	0	6
001111s	1	1	1

Table 2.1 Part of the intra VLC table for luminance and chrominance (s = sign bit) [50].

2.2.1.2 Context-Based Adaptive VLC

Context-Based Adaptive VLC (CAVLC) [65], [66] adapts the RLC tables according to neighboring blocks and the number of high-frequency coefficients having the value ± 1 (T1). The number of T1s and non-zero coefficients are first coded with one of four RLC tables. The levels of the non-zero coefficients are then coded with seven different RLC tables. The RLC tables are designed for different coefficient value levels. The first-used table is chosen according to the number of coefficients and high frequency T1s. The tables are then adaptively incremented according to the current coefficient.

2.2.1.3 Exponential Golomb Coding

Exponential Golomb (Exp-Golomb) [67] codes take on the form [K zeros][1][K-bit DATA] where K=0,1,2,... and DATA is a binary representation of an unsigned integer. An example of Exp-Golomb codes is shown in Table 2.2 [52]. Exp-Golomb codes are decodable with output data = 2^K+ int(DATA) -1, where int(DATA) is the integer corresponding to the binary string.

Input Data	Exp-Golomb Code
0	1
1	010
2	011
3	00100
4	00101
5	00110

Table 2.2 Example of Exp-Golomb codes [52].

2.2.1.4 Context-Based Arithmetic Encoding

Context-Based Arithmetic Encoding (CAE) [68] is used to encode binary symbols. CAE takes advantage of the correlation of previously coded symbols. In CAE, a context number of a pixel is computed according to the neighborhood of the pixel in either the same frame or the reference frame. This context number is then used to drive an arithmetic encoder [62] so that the probability of the symbol is computed according to the context number.

In the arithmetic coding process, the original input probability interval [0,1) is divided according to the input symbol's corresponding probability. The new probability interval is again divided according to the next input symbol's corresponding probability. A single output symbol is received after the whole data set has been inputed.

2.2.1.5 Context-Based Adaptive Arithmetic Encoding

Context-Based Adaptive Arithmetic Encoding (CABAC) [69] uses a two-stage VLC encoder. The input symbols are first coded with a basic VLC-type encoder. The individual bins (i.e. the individual bits of the VLC encoded symbol) are further encoded with a binary arithmetic coder. The arithmetic encoder adaptively uses several context models that store the following symbol's probability.

2.2.2 Transform Coding

In transform coding, the used transform maps the image into another domain, such as the frequency domain. The advantage of transform coding is that the transform decorrelates the image, and therefore the image can be represented with a smaller number of bits [70]. Due to its compression efficiency and efficient hardware realization [64] the Discrete Cosine Transform (DCT) has been chosen as the transform of almost all video coding implementations. The block size is 8x8 due to the trade-off between memory requirements and compression efficiency [59]p.380, [72].

2.2.2.1 Discrete Cosine Transform

The NxM 2-D DCT is expressed as

$$y_{kl} = \frac{c(k)c(l)}{4} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} x_{ij} \cos\left(\frac{(2i+1)k\pi}{2M}\right) \cos\left(\frac{(2j+1)l\pi}{2N}\right) \tag{2.1}$$

where k = 0,1,...,M, l = 0,1,...,N and

$$c(k) = c(l) = \begin{cases} \frac{1}{\sqrt{2}}\; if\, k,l = 0 \\ 1\;\; otherwise \end{cases} . \tag{2.2}$$

After the transformation the energy of the image is packed into the upper left hand corner of the block. The DC-coefficient is the upper left hand coefficient and the AC-coefficients span the rest of the block. The waveforms of the 8x8 DCT basis functions

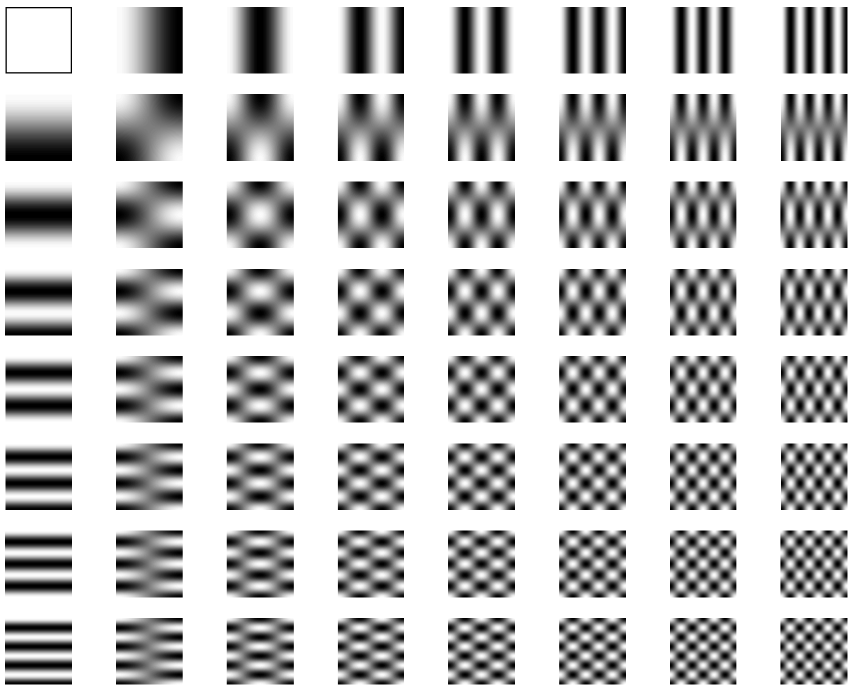


Figure 2.3 The 8X8 2-D DCT basis functions

are shown in Fig. 2.3. The 2-D NxM inverse DCT (IDCT) is defined as

$$x_{ij} = \sum_{k=0}^{N-1}\sum_{l=0}^{M-1} \frac{c(k)c(l)}{4} y_{kl} \cos\left(\frac{(2i+1)k\pi}{2N}\right) \cos\left(\frac{(2j+1)l\pi}{2M}\right). \tag{2.3}$$

2.2.2.2 Compression Capability and Transform Length

If video sources were strictly stationary processes, the decorrelation capability, and thus compression of the DCT, would depend linearly on the length of the DCT [59]p.380. As video sources are not strictly stationary [73]p.979, a smaller transform length can locally achieve greater compression. Also, the subjective effects (artifacts) described in Chap. 2.2.5 can vary for transforms of unequal size, making a smaller transform less visually annoying.

2.2.2.3 4x4 DCT Approximation

To reduce the computational complexity of the 8x8 DCT in [74], a 4x4 integer approximation of the 4x4 DCT has been introduced. The approximation reduces the 32-bit arithmetic required by the DCT to 16-bit arithmetic. The 4x4 DCT approximation is

$$Y = T_f X T_f^t \otimes E \tag{2.4}$$

where

$$T_f = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}, \tag{2.5}$$

$$E = \begin{bmatrix} a^2 & \frac{ab}{2} & a^2 & \frac{ab}{2} \\ \frac{ab}{2} & \frac{b^2}{4} & \frac{ab}{2} & \frac{b^2}{4} \\ a^2 & \frac{ab}{2} & a^2 & \frac{ab}{2} \\ \frac{ab}{2} & \frac{b^2}{4} & \frac{ab}{2} & \frac{b^2}{4} \end{bmatrix}, \tag{2.6}$$

$a = \frac{1}{2}$, $b = \sqrt{\frac{2}{5}}$, and $\otimes$ refers to scalar multiplication. The correction term E can be incorporated into the quantization process (Chap. 2.2.3). The PSNR penalty for the approximation is less than 0.01 dB [74].

H.264 (Chap. 2.5) also introduces an 8x8 approximation similar to the 4x4 approximation, but the use of this transform is only standardized for the Fidelity Range Extension (FRext) profile (Chap. 2.3.1) [105]. The computational complexity of FRext is beyond the capabilities of mobile encoders and thus beyond the scope of this thesis.

2.2.3 Quantization

Transform coding in itself is lossless if the transformed coefficients are decoded without error. To achieve lossy coding the transformed coefficient are quantized. Generic quantization reduces entropy by mapping several input levels into a single output level, and is expressed as [64]p.81

$$z_{kl} = round\left(\frac{y_{kl}}{q_{kl}}\right) = \left\lfloor \frac{y_{kl} \pm \lfloor \frac{q_{kl}}{2} \rfloor}{q_{kl}} \right\rfloor,\ k,l = 0,1,\ldots,N,M \tag{2.7}$$

where NxM is the block size and $\pm$ depends on the sign of y_{kl}. The quantization coefficient q_{kl} can be constant for the whole transformed block or a matrix of the same size as the transformed block. Quantization matrices can be used to take advantage of the sensitivity of the human visual system in the quantization operation [64].

2.2.4 Motion Estimation

The Intra coding techniques of 2.2.1 and 2.2.2 can be improved by predictive coding. In basic predictive coders, a prediction of the current pixel is made from previously coded information. Intra-field prediction uses pixels in the horizontal direction from the same line and/or from previous lines. In the static areas of consecutive frames, the temporal correlation could be predicted by taking the predictor from the same pixel position of one or more previous frames, but this model would be inefficient for real scenes with motion.

The motion in a video sequence can be defined with the following model

$$\begin{pmatrix} x \\ y \end{pmatrix} \longmapsto \begin{pmatrix} s\cdot\cos\theta & -s\cdot\sin\theta \\ s\cdot\sin\theta & s\cdot\cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} d_x \\ d_y \end{pmatrix} \tag{2.8}$$

where s defines the scaling ratio, θ the rotation angle, and (dx,dy) the translation vector. In addition to the motion, the illumination can be spatially and temporally changing and occlusion can hide and uncover areas. A prediction algorithm that would cover all of these possibilities would be computationally exhaustive. Also, as the motion parameters have to be sent as an overhead to the decoder, there is a trade-off between motion model complexity and motion parameter overhead. As the coding techniques of Chaps. 2.2.1 and 2.2.2 enable very efficient coding of the prediction error the prediction algorithm can be simplified by the following assumptions:

- The motion is constant within an NxM block
- The motion is translational
- The illumination is spatially and temporally uniform

With these restrictions block-based motion estimation (ME) can be used. Not fulfilling these restrictions leads to degraded prediction results.

2.2.4.1 Block-Based Motion Estimation

Block-based motion estimation is a search scheme that tries to find the best predictor for an NxM block of the current frame. The predictor is searched within a predefined area of one or more previous or future frames. The predictor with the minimum distortion D is chosen as predictor

$$(MV_x, MV_y) = min_{(dx,dy)\subset r} D(dx,dy) \tag{2.9}$$

where (dx,dy) is the current predictor candidate, *r* the search area, and the position of the best predictor is described with a motion vector (MV). Fig. 2.4 describes the basic principle of block-based ME. The temporal prediction signal can then be expressed as

$$e(x,y,t) = I_y(x,y,t) - I_y(x-dx, y-dy, t-n) \tag{2.10}$$

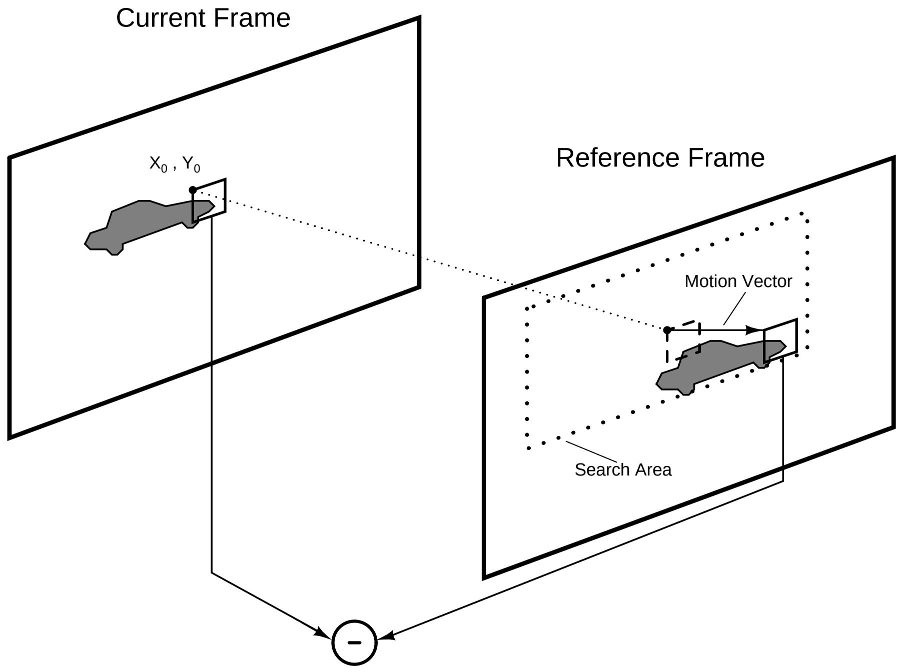


Figure 2.4 Block-based motion estimation.

where (x,y) is the size of the block, I$_y$(t) is the current frame, I$_y$(t-n) is the reference frame and a positive n points to a past frame and a negative n points to a future frame. The operation of Eq. 2.10 is commonly referred to as Motion Compensation (MC), although the terms “estimation” and “compensation” are often interchanged.

In motion estimation, multiple options are available for the distortion measure and the search scheme. In most implementations, the search is only carried out on the luminance information and the luminance MV is then used also for the chrominance. Separate searches on chrominance are only made when the objective is very high quality video.

Integer MVs restrict the motion to the spatial sampling intervals of the image. However, due to the spatial aliasing caused by real motion, a better prediction result can possibly be achieved by using fractional length MVs. In [75], the coding advantages of 1/2-pixel-, 1/4-pixel- and 1/8-pixel-accuracy MVs are investigated. Also investigated is the effect of three different interpolation filters used in creating the inter-pixel positions. It was found that 1/8-pixel accuracy over 1/4-pixel accuracy does not give significant results and that Wiener filtering gives results superior to Sinc and bilinear filtering.

Predicting the MV over several frames gives additional gain [76].

2.2.4.2 Distortion Measure

The distortion measure (also called “matching criterion”, “distance criterion”, “disparity measure”, and “cost function”) indicates the best match block. In predicting real motion, the problem would be to find the maximum cross-correlation

$$CCF(dx,dy) = \frac{\sum_{i=x}^{x+N-1}\sum_{j=y}^{y+M-1} cur(i,j)ref(i+dx,j+dy)}{\sqrt{\sum_{i=x}^{x+N-1}\sum_{j=y}^{y+M-1} cur^2(i,j)}\sqrt{\sum_{i=x}^{x+N-1}\sum_{j=y}^{y+M-1} ref^2(i+dx,j+dy)}} \tag{2.11}$$

between the reference block and the current block [58]p.30, [56], where NxM is the block size, and (x,y) the top-left pixel of the block. In [56], it was found that, when block size is small and when the translation is not pure, the accuracy of the correlation function is poor. This is due to the fact that the restrictions on motion estimation (page 16) are rarely fully fulfilled, making the optimal distortion measure input-data dependent. This means that different types of video (texture, real motion, etc.) have different optimal distortion measures. From here on, in this thesis, the term “sub-optimal distortion measure” refers to a sub-optimal result with a wide range of varying input data.

As the implementation of the cross-correlation function suffers from computational complexity the most-used distortion measure in practical implementations is

$$D_N(dx,dy) = \sum_{i=x0,j=y0}^{x0+M-1,y0+N-1} |ref(i+dx,j+dy) - cur(i,j)|^p . \tag{2.12}$$

The most common values for p are p=1 (Sum of Absolute Differences, SAD) and p=2 (Sum of Squared Differences, SSD). Although the SSD gives the better result, in digital implementations, the SAD is more common, due to its more efficient implementation. A large number of other distance measures have been proposed in the literature, of which a non-exhaustive list is presented below. Many of the references combine more than one method for computational complexity reduction. The distance measures can be grouped.

In distance criterion estimation, the calculation of the distance criterion is modified in a way that reduces the amount of computation needed. In [77], the distance criterion computation is stopped when the value is above a threshold. This is commonly referred to as early-thresholding. The thresholding can also be achieved pixel-wise [78]. In [79], the distance criterion is only partially computed. In [80], the pixels included in the computation are included with a reduced word-length. In [81], only the maximum value of the difference block is used for the motion vector choice.

In block modification, the reference and/or current blocks are modified in a way that reduces the amount of computation needed. In [79], the blocks are subsampled so that only a part of the pixels are taken into account. In [82], the blocks are filtered and compared with the original to achieve 1-bit pixels.

In feature matching the original 2-D data is transformed into a 1-D projection. When the projection data has a higher entropy than the original data the computation load is reduced. In [83], an NxN block is reduced into two Nx1 vectors by computing the 1-D mean in the horizontal and vertical directions. These two vectors are then used for the distance criterion computation. In [84], the DC component of the block and a reduced word-length phase indicator are used as the feature.

In hierarchical (multi-resolution) estimation the distance criterion computation is carried out at several levels. In [85], the search is carried out at three levels, where the upper level is subsampled by averaging from the lower level. The coarse MV at the higher level is then used as a starting point for the lower-level search. In [86], the subsampling is carried out with a lattice that takes edges into account.

2.2.4.3 Search Algorithm

The optimum solution to the problem of finding the block with the minimum distortion would be an exhaustive search of the search area [-r,r-1]. Such a search algorithm is called the Full Search (FS). The search area does not need to be square; furthermore a rectangular search area can exploit the fact that the magnitude of real-motion vectors is larger in the horizontal direction [87].

With an [-r,r-1] search area and an NxM frame size, the distortion measure of the FS algorithm needs to be computed 2r·2r·N·M times. To lessen the computational burden, and speed-up the motion estimation, the search area can be subsampled. Such fast search algorithms assume that the distortion measure increases monotonically away from the point of the minimum distortion. When this assumption does not hold, as is often the case with real video, the fast search algorithm may be trapped in a local minimum. A popular search area subsampling algorithm is Three Step Search (TSS) [88], where the distance measure is calculated in three stages. In the first stage, the distortion measure is computed at the point (0,0) and at eight points along the perimeter of the search area [-7,7]. The subsequent search stages then compute eight additional distortion measure points from the perimeter of the search areas, [-2,2] and [-1,1], respectively. The best match of the previous stage is always used as the center of the subsequent stage. TSS drops the number of search steps to 25, but has the disadvantage of having the maximum motion vector range of ± 10.

Gradient-based methods remove the motion vector length restriction to take advantage of the center-based motion vector distribution of natural video sequences. In [89], an unrestricted diamond search pattern is introduced. The first stage of the search is composed of a nine point ± 2 range diamond. This diamond search is repeated until the best match is at the search center. The diamond search patterns overlap so that the subsequent diamonds need only five or three distortion-measure computations. When the best match of the first stage is found, the search is once more refined with a four point ± 1 range diamond. The advantage of this method is that the motion vector length is not restricted, but the computational complexity can become large if the start of the search is far from the best match.

When using fractional-pixel accuracy, the most common method is to first make an integer accuracy search and refine the integer search result with a sub-pixel accuracy search [58].

2.2.4.4 Predictive Motion Estimation

The spatial-temporal correlation of natural video sequences that is taken advantage of in transform coding (Chap2.2.2) can also be taken advantage of in motion estimation. The motion vector field commonly exhibits the same correlation. Thus already-coded MVs can be used as starting points for the current search. An additional advantage of predictive ME is that, as already coded MVs are used as the center of the current search, the variance of the MV field is reduced. This is an advantage when the MVs are differentially coded (Chap. 2).

In [90], two previously coded MVs from the same frame, and two MVs from the previous frame, are used in predicting the current MV. The distortion measure from these predictors is computed and the MV with the lowest distortion is then chosen as the starting point for the search. The search algorithm can then be thought of as a small refinement of the starting point by computing additional distortion points around the candidate. In [91], the median of three MVs in the current frame and the collocated block in the previous frame are used in predicting the current MV. The predictors are also used to determine an early search termination threshold and a threshold that controls the size of the search range.

2.2.4.5 Motion Vector Adaptive Search Technique (MVFAST)

As an example of a predictive gradient-based algorithm Motion Vector Adaptive Search Technique (MVFAST) [92] is examined. In the MPEG-4 optimized reference software [93], MVFAST and Predictive MVFAST (PMVFAST) [94] are used instead of Full Search. MVFAST is also used in verifying the results of Chap. 4.

In MVFAST, three earlier MVs of blocks arranged according to Fig. 2.9 are first analyzed. The city-block length, defined by $l_i = |x_i| + |y_i|,\ i = \{1,2,3\}$, is first computed for the three MVs. The city-block lengths are then compared and the maximum value of these three is chosen as the reference motion vector threshold L_α. By defining two additional thresholds L_1 and L_2 the Motion Activity of the current block is

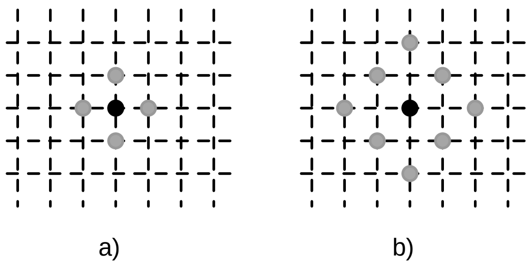


Figure 2.5 Gradient-search patterns. a) Small diamond b) Large diamond

defined by

$$MotionActivity = \begin{cases} Low, & L_\alpha \leq L_1 \\ Medium. & L_1 < L_\alpha \leq L_2 \\ High & L_\alpha > L_2 \end{cases} . \qquad (2.13)$$

For generic sequences MPEG-4 suggests L_1=1 and L_2=2. The found motion activity has an affect on the search center and search strategy of the search.

The search center for MVFAST is chosen as (0,0) if the motion activity is low or medium, otherwise the search center is the location pointed by the MV of the three candidate MVs that gives the minimum SAD. If the motion activity is low or high, a small diamond search is chosen as the search strategy, otherwise a large diamond search is chosen.

In *small diamond search,* all the SADs of the points indicated by Fig. 2.5a are searched. If the center point yields the minimum SAD, the search stops, otherwise the point of the minimum SAD is chosen as the center point for a new small diamond search.

In *large diamond search,* the SADs of the points indicated by Fig. 2.5b are searched. If the center point does not yield the minimum SAD, the point of the minimum SAD is chosen as the new large diamond search center. If the minimum SAD indicated by the large diamond search is at the center a small diamond search is executed with the center point at the point indicated by the minimum SAD of the large diamond search.

The thresholds L_1 and L_2 introduce a possibility of modifying the search strategy, depending on the motion of video sequence. The range for L_1 and L_2 is from -1 to the search range. If the sequence has low motion (i.e. the absolute values of the MVs are small), the optimum values for L_1 and L_2 are high. High values for L_1 and L_2 have the effect of raising the proportional part of low motion activity. Low values for L_1 and L_2 correspondingly raise the proportional part of the high motion activity. A large difference between L_1 and L_2 raises the proportional part of the medium motion activity.

2.2.4.6 Rate-Distortion Optimized Motion Estimation

Rate-Distortion (R/D) Optimized Motion Estimation takes into account the actual cost of coding the block. In addition to the distortion measure, R/D optimization takes into account the overhead needed for the residual data. In this way, the number of bits in the bitstream needed to code the prediction is minimized.

The most widely accepted approach to R/D optimization are Lagrangian techniques. For the symbol group s_1,s_2,...,s_N that can be coded with one of the coding options I_1,I_2,...,I_N the Lagrangian cost function can be computed with [95]

$$J\left(s_k, I_k \mid \lambda\right) = D(s_k, I_k) + \lambda \cdot R(s_k, I_k), \tag{2.14}$$

where D is the distortion, R the rate, and $\lambda \geq 0$ is the Lagrange parameter. The minimum of $J\left(s_k, I_k \mid \lambda\right)$ gives the optimum coding result.

2.2.4.7 Transform Domain Motion Estimation

To reduce computational complexity the motion estimation can be made in the transform domain. In [96], pseudophases are used to indicate the shift of DCT transformed signals. The amount of shift indicates the movement. The Discrete Sine Transform (DST) is needed in computing the pseudophases; this increases the computational complexity. In [97], the ME is made in the Hadamard transform domain.

$$y_{kl} = \frac{1}{N} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} x_{ij} (-1)^{\sum_m^{n-1} (b_m(i) b_m(k) + b_m(j) b_m(l))}, \tag{2.15}$$

where x is the input, y the output, and $b_m(i)$ is the m:th bit of the binary representation of i. The advantage of the Hadamard transform is that its coefficients are ± 1. In [98], the Hadamard transform is used in Lagrange optimization. The distortion term in Eq. 2.14 is replaced by the SAD of the Hadamard transformed block. The Hadamard transform approximates the frequency characteristics of Eq. 2.6 and subsequently reflects the expected reconstruction quality, rather than the amount of prediction error.

2.2.4.8 Other Motion Estimation Techniques

In bi-predictive ME, two motion vectors are transmitted for each block [99]. Traditionally, one of these MVs has pointed to a future frame and the other to a past frame; the reference block is an average of the blocks pointed to the MVs. This has advantages in cases of smooth motion, where bi-predictive ME is, in effect, interpolation between the future and past frames. In H.264 (Chap. 2.5), the concept of bi-predictive ME has been generalized. Any of the motion estimation methods of Chap2.2.4 can be used in deriving the reference blocks.

In global ME, a single MV is predicted for the entire frame [100]. Global ME is most efficient in cases of camera motion, such as pan. In some profiles of MPEG-4 (Chap. 2.4), global motion parameters are transmitted for the segmented background.

Several video standards allow the MV to point partly or fully outside the reference frame. This is advantageous in cases where an object has partly moved outside the frame. In such cases, the pixels outside the frame are interpolated from the frame using standardized methods.

2.2.4.9 Comments on the Translational Block-Based Motion Estimation Model

A full search algorithm should take into account all the points (and sub-pixel points, if applicable) within a search area. When comparing such an FS algorithm to a search area subsampling ME algorithm, to make the comparison valid, the search area of the fast algorithm needs to be the same as the search area of the FS algorithm. Then, for a given block, exploring all the possibilities will always lead to a lower (or equal) estimation result than exploring just a few possibilities. For example, many gradient-based methods do not restrict the search area; this can lead to a better estimation result being found outside the search area of the FS.

A restriction to the statement above is the unideality of the matching criterion. As an ideal matching criterion is input-data dependent, conventional matching criteria are almost always unideal. Thus, with an unideal matching criterion, an inferior motion estimation result can have a superior rate-distortion. For example, small savings in texture bits can lead to a high number of MV bits if the amplitude of the MV is large and does not correlate with neighboring MVs. The closer the matching criterion reflects the actual coding cost, the better the motion estimation algorithm using this matching criterion performs in terms of rate-distortion.

Even if the matching criterion is optimal and a full search is conducted, a superior result can be achieved by taking into account the global effect of the motion vector. As motion vectors are coded differentially (Chap. 2.4, Chap. 2.5), motion vector field smoothness has a significant impact on the total bit-rate. For example, sequences with a homogeneous background will result in chaotic motion vector fields.

2.2.4.10 Motion Estimator Hardware Design Aspects

A low-power silicon VLSI motion estimator is a trade-off between optimizing the power consumption of the search engine (including the distortion computation and choice of MV) and the power consumption of the memory accesses. In CMOS cir-

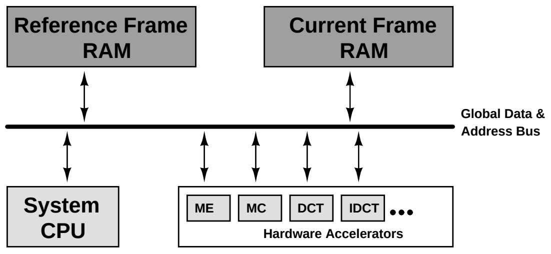


Figure 2.6 Generic architecture of a VLSI video encoder.

cuits, the largest source of power consumption is the dynamic power consumption

$$P_d = \alpha_P \cdot C_L \cdot V_{dd}^2 \cdot f, \tag{2.16}$$

where α_P is the switching activity of the circuit, V_{dd} the power supply voltage and f the clock frequency of the circuit. With 0.13μm and smaller technologies, the leakage power consumption and short-circuit power consumption are also issues, although these cannot be influenced when designing with a vendor library, as is the usual case. Most important in minimizing the power consumption of the memory is minimizing the data bandwidth of the memory [58], as the power consumption of memory is defined by

$$P_{mem} = P_{static} + (BW_r + BW_w) \cdot C_L \cdot V_{dd,}^2 \tag{2.17}$$

where BW_r and BW_w are the bandwidths of the read and write operations, respectively, and P_{static} is the static power consumption of the memory.

Full Search motion estimation consists of a large number of simple arithmetic operations (i.e. $|ref(i+dx, j+dy) - cur(i, j)|$ for SAD, Eq. 2.12) and a large amount of regular data accesses. Due to this large amount of data, the architectures of VLSI video encoders and hardware accelerators resemble closely the generic architectures shown in Fig. 2.6 and Fig. 2.7, respectively. In such architectures, the distortion computation of FS can typically be implemented using regular 1-D or 2-D systolic arrays [101].

With search area subsampling and predictive ME algorithms, both the number of arithmetic operations and memory accesses are reduced with the cost of irregular memory access, low hardware utilization, and sequential procedures with data dependence that cannot be parallelized. This is especially true with ME algorithms where the data used in the next arithmetic operation is not known before the result of the current arithmetic operation is clear. Irregular memory access complicates the design of the memory in the encoder especially the design of the Address Generation Unit (AGU). Thus, the benefits of search-area subsampling algorithms at the algorithmic level are usually larger than those from the architectural level.

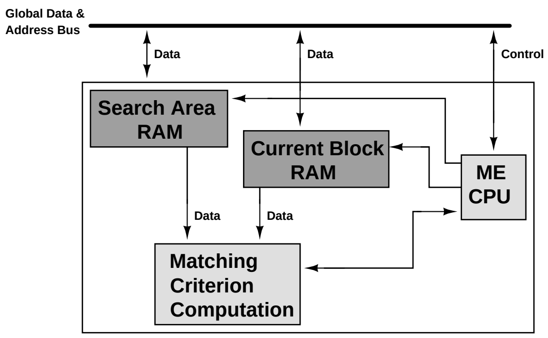


Figure 2.7 Generic architecture of a VLSI motion estimation accelerator.

2.2.5 Quality Measures

The distortion of an image is usually expressed with the peak signal-to-noise ratio (PSNR). For 8-bit images, PSNR is defined by

$$PSNR = 10\log\left[\frac{255^2}{\frac{1}{N\cdot M}\sum_{m=1}^{N}\sum_{m=1}^{M}(s_{m,n}-\tilde{s}_{m,n})^2}\right] \tag{2.18}$$

where s is the original image and $\tilde{s}$ is the coded image.

PSNR does not fully correlate with the perceived subjective quality of the video. The correlation is high if the coding errors are evenly spread within each frame and the PSNR does not significantly change within the time span of a few frames. This means that locally concentrated errors are more visible than the PSNR value of the frame would suggest, and that significant PSNR drops (˜10dB) can be undetected if they last only one or two fames.

Several measures for subjective quality have been developed but none have become wide-spread. PSNR is usually complemented by a panel rating of the subjective quality. In such cases, the most common terms expressing subjective quality include:

Blurriness	i.e. the blurring of detail equivalent to a 2D low-pass effect.
Ringing	distortions near edges equivalent to a 2D high-pass effect.
Blocking	i.e. a blocky appearance in parts of the frame due to block-based coding.
Mosquitoing	error in moving edges - these can appear blurry and/or flickering due to the combined effect of motion estimation and quantization.

2.3 Scope of Video Coding Standards

In the major video coding standards, only the decoder and the bitstream syntax are defined by the standard. The scope of the standards is shown within the generalized video coding and transfer process in shown Fig. 2.8.

The video coding and transfer process consists of optional pre-processing, the encoding process, the transfer or storage medium, the decoding process and the optional post-processing. The pre-processing might consist of picture quality enhancement techniques or spatial redundancy reduction. Possible post-processing includes format conversion or error recovery.

When only the decoder and bitstream are standardized every conformant decoder will produce the same result when decoding an error-free conformant bitstream. The result of this is that the options left in decoder design are limited. In encoder design, the only restriction is that the encoder has to produce a valid bitstream and a conformant encoder is not a guarantee of picture coding quality. The encoder can then be designed flexibly by emphasizing, depending on the application, compression efficiency, computational power, or other issues.

2.3.1 Profiles and Levels

Different applications typically require different functionalities (compression efficiency, compression delay, error resiliency) and different levels of complexity. To manage the large number of coding tools included in the standards, and to make the standards more attractive to varying types of applications, the standards are split into profiles that are further split into levels. The concept of profiles and levels is employed to define a set of conformance points, each targeting a specific class of applications. A profile defines the set of coding tools that the decoder compliant to that profile must support. A level places constraints on certain key parameters of the bitstream, such as the picture resolution and bit-rate.

The third generation partnership project (3GPP) has included MPEG-4 Simple Profile Level 0b, H.263 Profile 3 Level 45 and H.264 (AVC) Baseline Profile Level 1b into its multimedia specifications [102], so only these standards will be dealt in detail in this thesis.

2.4 MPEG-4

MPEG-4 standardizes coding methods for many types of audiovisual data. MPEG-4 incorporates standardized ways to represent various media objects. These objects include aural, visual or audiovisual content which can be natural or synthetic in origin.

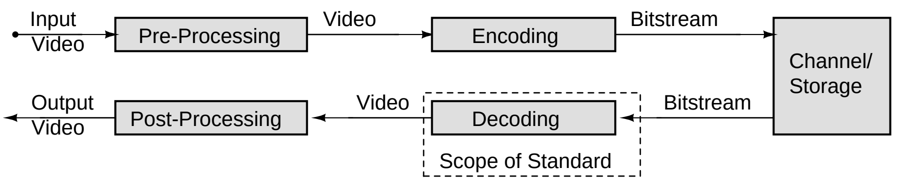


Figure 2.8 The scope of most video coding standards is outlined: The standards define the syntax of the bitstream and the decoding process of this bitstream.

The spatial coding in MPEG-4 uses the 8x8 DCT (Chap. 2.2.2), two different scalar quantization methods, and VLC coding with improved MPEG-2 RLC (Chap. 2.2.1) tables. The first quantization method is an encoder-defined weighted 8x8 quantization matrix that is sent to the decoder. The number of possible different weighted quantization matrices depends on the used profile. The second quantization method is assigning the same quantization coefficient for each AC coefficient in the macroblock. For intra frames, the actual quantization coefficient is computed with the nonlinear scale

$$quant(i,j) = sign(y(i,j)) \cdot \frac{\left(16 \cdot |y(i,j)| // w(i,j)\right)}{Q_p} \tag{2.19}$$

where Q_p is the quantization parameter ranging from 1 to 31, w(i,j) is the optional quantization matrix and // signifies division with round-to-nearest. If the weighted matrix is not used, w(i,j)=16 for all i,j.

With DCT transformed blocks of real video, the amplitude of DC coefficient is typically a magnitude higher than the amplitude of the AC coefficients. To take this into account, the DC coefficient is divided according to Table 2.3. For inter frames, both the DC and AC coefficients are divided with Eq. 2.19 except that $2Q_p$ is used in the denominator. Three different scans exist for the VLC coding.

	Value of Q_p			
	1-4	5-8	9-24	25-31
Luminance	8	$2Q_p$	Q_p+8	$2Q_p$-16
Chrominance	8	(Q_p+13)/2		Q_p-6

Table 2.3 MPEG-4 DC coefficient division

Motion estimation in MPEG-4 is based on 16x16 or 8x8 blocks; the accuracy of the motion vectors can be up to quarter-pixel, depending on the profile and level. Motion vectors are allowed to point outside the reference frame and are encoded differentially with

$$\begin{aligned} P_{x,y} &= Median(MV1_{x,y}, MV2_{x,y}, MV3_{x,y}) \\ MVD_{x,y} &= MV_{x,v} - P_{x,y} \end{aligned} \tag{2.20}$$

where the positions of MV1, MV2, MV3 are as shown in Fig. 2.9 and the labels

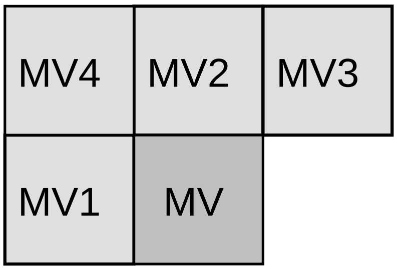


Figure 2.9 Motion vector predictors

x,y correspond to each other. Bi-directive ME (B-frames) allows two motion vectors where one MV points to the previous frame and one to the future frame. B-frames cannot be used as reference frames. MPEG-4 also has the possibility of using skip modes for each macroblock. For a skipped macroblock, no motion vector or transform coefficients are sent, meaning the decoder copies the macroblock from the same position in the previous frame. In the syntax, a skip mode is indicated with a single bit (sometimes referred as the COD bit or the "not coded" bit), that reduces the indication of the motion vector and texture data to this single bit.

Of the various profiles had been defined for MPEG-4 Visual [103], for concise expression, only the following are reviewed in this thesis: Simple, Advanced Simple, Advanced Real-Time Simple, Simple Scalable and Core. The Simple profile incorporates the tools needed for error resilient coding of rectangular-shaped video and it includes all features mentioned above, with the exception of the weighted quantization matrices and quarter-pixel accuracy motion vectors. The Simple Scalable adds temporal and spatial scalability of video objects and is used for applications offering multiple levels of quantity over multiple bit-rates. The Advanced Simple adds weighted quantization matrices, quarter-pixel accuracy motion vectors, B-frames, global motion com-

Visual Profile	Level	Typical visual session size	Max. number of objects	Max. bit-rate (kbit/s)
Simple	L0	QCIF	1	64
Simple	L3	CIF	4	384
Advanced Simple	L1	QCIF	4	64
Simple Scalable	L1	CIF	4	128
Adv. Real-Time Simple	L1	QCIF	4	64
Adv. Real-Time Simple	L4	CIF	16	2000
Core	L1	QCIF	4	384
Core	L2	CIF	16	2000

Table 2.4 The main attributes of MPEG-4 visual profiles suited for mobile applications (Simple profile levels L1 and L2, Advanced Simple levels L2 and L3, Advanced Real-Time Simple levels L2 and L3, and Simple Scalable level L2 not shown).

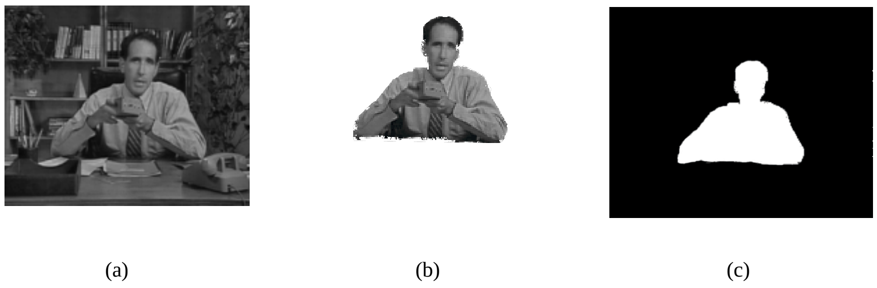


Figure 2.10 Segmentation example: a) Original frame b) Segmented object c) Binary alpha plane.

pensation and special tools for the efficient coding of interlaced video. The Advanced Real-Time Simple profile provides advanced error-resilient coding by defining the use of a back channel [104]. The Core profile adds coding of arbitrary shaped video objects for content-based applications. The main feature attributes of these profiles are collected in Table 2.4.

2.4.1 MPEG-4 Core Profile

If the content-based functionalities of MPEG-4 are to be used, the encoder also has to include shape segmentation. The main advantage of the content-based approach is that the bandwidth can be allocated to the video objects that are most important to the visual quality. In Fig. 2.10a and b, the person in the foreground is segmented from the background. In this case, the bandwidth and computational power of the encoder can be dedicated to the person while the background can be sent only once.

In the Core profile, the Video Object (VO) defines the entity to be coded. The VO can be a segmented natural object or a synthetic one. As a video sequence quantized in time is composed of frames, a VO consists of Video Object Planes (VOP). Without the content-based functionalities, a VOP consists of the whole frame. The frame in Fig 2.10 would be coded as two VOPs: one, the foreground person shown in white in Fig 2.10c, and, the other, the background shown in black in Fig 2.10c. A static background in MPEG-4 is referred to as a *sprite*. MPEG-4 has standardized tools for the modification of a sprite with 8 global motion parameters. The pixels belonging to the VOs are defined with the binary alpha plane, which is composed of Binary Alpha Blocks (BAB). An example of binary alpha plane is shown in Fig. 2.10c. The BABs can be either transparent (not included in the object), opaque (included in the object) or border-BABs. A BAB can be coded in several ways: transparent, opaque, intra-

CAE (Chap. 2.2.1.4) and shape motion compensation with or without inter-CAE. The subsampling of shape is also allowed.

2.4.2 H.263

Baseline profile H.263 [51] is the starting point for the MPEG-4 natural video coding methods, which means that a compliant MPEG-4 decoder is able to decode a valid Baseline H.263 bitstream. The baseline profile provides the minimal capability that all decoders must support. H.263 has eight annexes (annexes A-G) that add additional coding options, for example 8x8 block-size motion estimation. H.263+ and H.263++ are later versions of H.263, which add several coding options, most of which are included in H.264 (Chap. 2.5).

2.5 H.264

ITU-T H.264 / MPEG-4 Part 10 Advanced Video Coding (H.264/AVC) [52] is the latest international video coding standard; this currently offers the best ratio between video quality and bit-rate (rate-distortion). The block diagram of an H.264 encoder is shown in Fig. 2.11.

Intra blocks in H.264 are predicted in a way similar to motion estimation prediction. The predictor candidates are computed with 4 16x16 modes or 9 4x4 modes. The modes correspond to the spatial direction of the predictor candidate. Transform coding in H.264 is achieved with the 4x4 DCT approximation (Chap. 2.2.2.3). In certain cases, all the luminance DC coefficients of the 4x4 transformed blocks are further transformed with the 4x4 Hadamard transform [59]

$$H_{4x4} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}. \tag{2.21}$$

For chrominance, the 2x2 Hadamard transform is used. To remove the “dead zone” effect of previous MPEG quantization H.264 uses a quantization parameter QP$_{H.264}$ ranging from 1 to 52. The dead zone effect arises from the fact that the quantization step sizes are relatively larger when the quantization step is small (QP=1 $\rightarrow$ QP=2), as opposed to the relatively small step sizes when the quantization step is large (QP=31 $\rightarrow$ QP=32). The quantization step Q$_{step}$ is chosen from the vector $q_v = [0.625\ 0.6875\ 0.8125\ 0.875\ 1\ 1.125\ 1.25]$ according to QP$_{H.264}$. First, q_{v2} is gen-

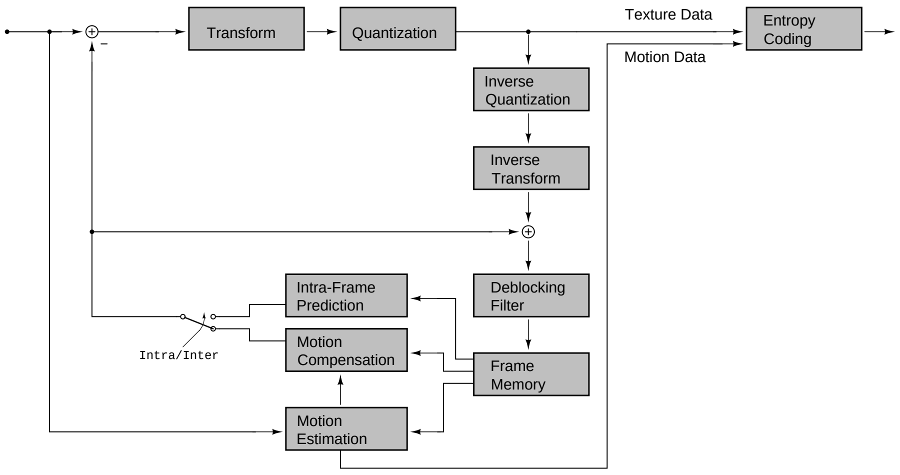


Figure 2.11 Block diagram of an H.264 encoder.

erated with

$$q_{v2} = [0.625\ 0.6875\ 0.8125\ 0.875\ 1\ 1.125\ 1.25] \cdot 2^{\left\lfloor \frac{QP_{H.264}}{6} \right\rfloor}, \tag{2.22}$$

where $\lfloor\ \rfloor$ is rounding with truncation towards zero. Second, Q_{step} is chosen from q_{v2} with

$$Q_{step} = q_{v2}\left(rem\left(\frac{QP_{H.264}}{6}\right)+1\right), \tag{2.23}$$

where *rem* is the remainder after division. From Eq. 2.22 and Eq. 2.23 it can be seen that the quantization step doubles for every increment of 6 in $QP_{H.264}$. $QP_{H.264}$ can be changed for each macroblock separately.

Depending on the profile entropy coding is achieved with CAVLC (Chap. 2.2.1.2) for the transform coefficients and Exp-Golomb (Chap. 2.2.1.3) for other data elements or with CABAC (Chap. 2.2.1.5) and Exp-Golomb coding. CABAC can increase compression efficiency by roughly 10% relative to the CAVLC mode [105], although CABAC is significantly more computationally complex. Due to this computational complexity CABAC will not be used in mobile architectures in the near future.

Motion estimation in H.264 uses seven different block sizes; these are shown in Fig. 2.12 and the accuracy of the motion vectors can be up to quarter-pixel. The maximum number of used past or future reference frames is 15; the MVs can be weighted. The motion vector prediction is computed as shown in Fig. 2.9 and Eq. 2.20, except that the size of the current block or sub-block determines the neighboring blocks. This means that the neighboring blocks can reside within the same macroblock as the current block. Also, if MV3 of Fig. 2.9 is unavailable, MV4 is used instead. Bi-directional prediction is generalized so that both reference blocks can be in the past or

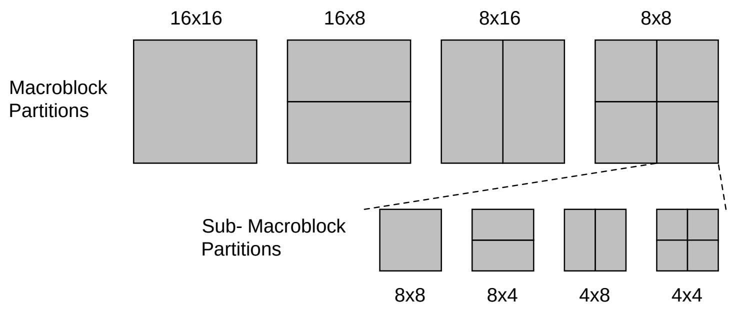


Figure 2.12 H.264 motion estimation partitions

future and both motion vectors are weighted. Unlike previous standards, bi-predicted blocks can also be used as reference. H.264 skip modes are similar to MPEG-4, except that the motion vector used is $P_{x,y}$ from Eq. 2.20, as opposed to the zero motion vector. Bi-predicted blocks also have two direct modes where the motion vector of the respective block is derived with scaling from spatial neighbors or temporally co-located blocks.

As is shown in Fig. 2.11, H.264 has an in-loop adaptive de-blocking filter. De-blocking filters have been used to reduce the block border gradient, which accounts for the blocking effect (Chap. 2.2.5) in the post-processing operations shown in Fig. 2.8. The de-blocking filter affects up to three pixels on either side of the border of 4x4 sub-blocks to reduce the gradient between adjacent blocks. The filter also reduces the prediction residual. The strength of the filter depends upon the compression mode of a macroblock (Intra or Inter), the quantization parameter, motion vector, and the pixel values of the original image. The encoder can also adjust the overall strength of the filter.

H.264 has three original profiles: Baseline, Main and Extended. In addition to these, four Fidelity Range Extension (FRext) [105] profiles are also defined. Of these, Baseline is used in mobile communication. From the basic coding tools the Baseline profile excludes bi-directional prediction. Among other features, the Extended profile includes advanced error resiliency tools, which might make it attractive for mobile use. H.264 defines 16 levels tied mainly to the picture size and frame rate. Level 1b was added in the FRext amendment, primarily to address the expressed needs of some 3G wireless environments. A sample of different levels is shown in Table 2.5.

The H.264 FRext amendment also adds the possibility of sending alpha channel information [106], which could open the possibility of using segmentation implementations (Chap. 3.3) in H.264 encoders. However, the FRext profiles are too computationally intensive for mobile applications.

Level	Typical frame size	Typical frame rate	Max. compressed bit-rate (kbps)	Max. number of reference frames
1	QCIF	15	64	4
1b	QCIF	15	128	4
1.1	QCIF	30	192	9
	CIF	7.5	192	2
1.2	CIF	15	384	6
1.3	CIF	30	768	6
2	CIF	30	2000	6

Table 2.5 The main attributes of H.264 profiles suited for mobile applications (Levels 2.1 through 5.1 not shown).

This page is intentionally left blank.

Chapter 3

Analog Parallel Processor Theory

3.1 Parallel Signal Processing

Various media-processing operations, especially video processing, are defined by a large demand for computational power. The operations in video encoding can be divided into a small number of groups:

- Stream-oriented operations, such as VLC (Chap. 2.2.1) and syntax generation.
- Macroblock-local and macroblock neighborhood operations, such as DCT / IDCT (Chap. 2.2.2) and filtering.
- Regional operations, which are composed only of motion estimation (Chap. 2.2.4).

Combining these operations results in a widely varying number of arithmetic and transfer instructions and data widths, making the mapping of the whole video coding operation onto a single processor platform difficult.

The computational power demand of video encoding is mainly made up of a large number of relatively simple operations of DCT, and, especially, ME, that are performed on substantial amounts of data. Also, the processing usually demands little global data reuse. The substantial amount of data stems from the large number of pixels in video frames.

Therefore, as opposed to general-purpose processors, which mainly operate serially, the most efficient type of processors for video operations are the ones that parallelize the data operations. This parallelization can be grouped into three main categories:

Instruction-Level Parallelism Superscalar processors, such as Very Long Instruction Word (VLIW) processors, execute multiple instructions from a single instruction stream simultaneously. The instruction-level parallelism of video applications is exploited in many VLIW DSPs, such as [107].

Thread-Level Parallelism Multiple Instruction Multiple Data (MIMD) processors exploit data parallelism by executing multiple instruction streams simultaneously [108]. These separate instruction streams may be, for example, executed by separate processor cores within a single architecture [109].

Data-Level Parallelism Single Instruction Multiple Data (SIMD) processors, such as vector processors [110], store the data in vector register files. The same instruction is then executed on these arrays of data.

Many processor architectures combine different forms of parallelism. In all parallelization cases, the most significant issue is the intelligent and efficient division of the computation instructions to the processors.

3.1.1 Array Processing

In array processing, multiple processors are arranged into an array and are usually interconnected. Many examples of array processors exist; these can either be either analog [111] or digital [109]. As the processor array can be designed to have the same size and structure as the video frame (i.e. one processor per pixel and interconnections enabling specific data transfer), a processor array is a prime candidate for video applications.

The Cellular Nonlinear Network (CNN) [112] paradigm is an analog parallel processing theory. The hardware CNN realizations, such as the CNN Universal Machine (CNNUM) [113], are theoretically capable of very high computational speeds combined with a low power consumption [114]. However, the implementation of a general purpose CNN is usually difficult and inefficient [115], leading to realizations that are no longer CNN in the strict sense.

3.2 Cellular Neural/Nonlinear Networks

The Cellular Neural/Nonlinear Network [112] achieves data parallelism with a parallel array of locally connected processors. The processors are customarily referred to as cells. If the connections (weights) between the cells are defined by a learning operation, the network is called a cellular neural network. Otherwise, the term “cellular nonlinear network” is used.

3.2.1 CNN Definitions

The cells are usually connected rectangularly as a two-dimensional grid. The interactions within the grid are defined by the cell’s neighborhood and the number of connections from each cell. The neighborhood of the cell defines the distance of the furthermost connections. Examples of 1- and 2-neighborhoods are shown in Fig. 3.1. A rectangular neighborhood is not the only option, but the neighborhood can also be, for example, hexagonal.

The cell need not be connected to all of the cells within the current neighborhood. Fig. 3.2 shows the most common 4-connected and 8-connected 1-neighborhoods, but other connection patters also exist. Also, the connectivity pattern can vary with each neighborhood. Depending on the application the cells on the edge of the array can require specific conditions which have to be implemented with specific border cells. An example of such connections is shown in Fig. 3.2 where the border cells are marked with dashed lines.

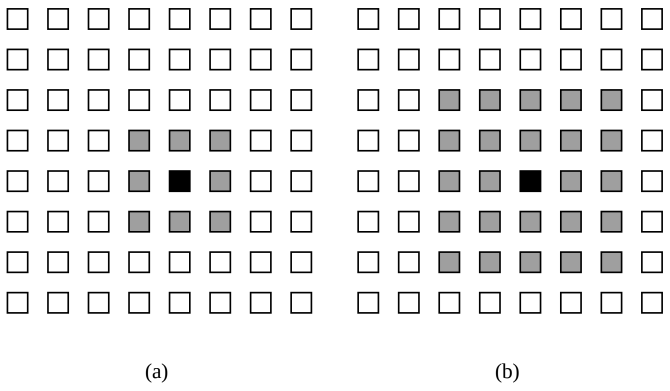


Figure 3.1 a) 1-neighborhood, b) 2-neighborhood. The gray cells depict the cells that are connected to the black cell.

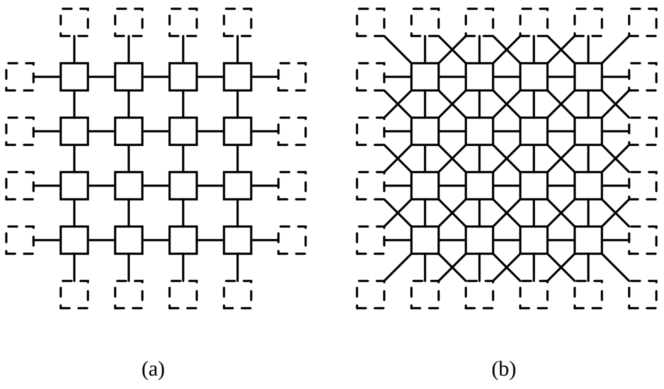


Figure 3.2 a) 4-connected 1-neighborhood, b) 8-connected 1-neighborhood.

3.2.2 Continuous-Time CNN

The state of a cell depends on the inputs and outputs of the cells connected to it. In the original CNN model [112], this state can be expressed with

$$\frac{dx_{i,j}}{dt} = -x_{i,j} + \sum_{C_{k,l} \in N_r(i,j)} A(i,j;k,l) y_{k,l} + \sum_{C_{k,l} \in N_r(i,j)} B(i,j;k,l) u_{k,l} + z \tag{3.1}$$

where x$_{i,j}$ is the cell state, u$_{i,j}$ is the cell input and y$_{i,j}$ is the cell output. A and B are the feedback and feed-forward coefficients, respectively. These coefficients represent the weights in the connections between the cell C$_{k,l}$ and the cell C$_{i,j}$. N$_r(i,j)$ is the neighborhood of the cell C$_{i,j}$ and z sets the operating point of the cell. The modification of the coefficient weights A, B and z achieve the different operations of the network. The output of the cell is a piecewise linear sigmoid that is shown in Fig. 3.3. The output is obtained with the equation

$$y_{i,j} = \frac{1}{2}\left(\left|x_{i,j}+1\right| - \left|x_{i,j}-1\right|\right). \tag{3.2}$$

If the feedback and feedforward coefficients are space invariant the coefficients have the same values for each cell in a network. In the case of a 1-neighborhood, the coefficients are expressed as

$$A = \begin{bmatrix} A_{-1,-1} & A_{0,-1} & A_{1,-1} \\ A_{-1,0} & A_{0,0} & A_{0,1} \\ A_{-1,1} & A_{0,1} & A_{1,1} \end{bmatrix} \qquad B = \begin{bmatrix} B_{-1,-1} & B_{0,-1} & B_{1,-1} \\ B_{-1,0} & B_{0,0} & B_{0,1} \\ B_{-1,1} & B_{0,1} & B_{1,1} \end{bmatrix} \tag{3.3}$$

so the coefficients can be expressed with only 19 terms, which include the term z from Eq. 3.1. In such a case, the coefficients A and B are called “cloning templates” and

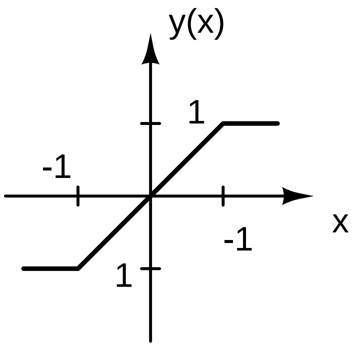


Figure 3.3 CNN cell output function.

the term z is called the bias template. A and B can also be polynomial functions [116].

3.2.3 Extensions to the original CNN model

An extension to the functionalities of CNN is the CNN Universal Machine (CNNUM) [113]. In addition to the connections of a regular CNN cell, a CNNUM cell contains program registers, switch configuration registers, boolean logic, control logic and analog and digital memory. A CNNUM cell enables the use of several templates and programmability in the network.

In Full Signal Range CNN (FSRCNN) [117], the cell state is limited between the range [-1,1], making the cell state and the cell output equivalent, with the result that the whole state swing can be reserved for this range. Also the $-x_{i,j}$ term in Eq. 3.1 can be realized with [$A_{0,0}$ - 1].

In positive range high gain CNN [118], the output function of Eq. 3.2 is changed to

$$y_{i,j} = \begin{cases} 0, \; x < -\varepsilon \\ 1, \; x > \varepsilon \end{cases} \quad 0 < \varepsilon \ll 1 \tag{3.4}$$

making the cell input and output binary. The templates in positive range high gain CNN remain continuous. Binary cell outputs do not permit gray-scale operations, but the analog design is simplified and reliability is increased.

A Discrete-Time CNN (DTCNN) [119] has continuous inputs and connection weights and a bipolar output. A DTCNN can be realized digitally or with an analog discrete time implementation.

3.2.4 Digital Emulated CNN

The advantages of a digital CNN architecture realization are increased processing accuracy and digital data storage. However, as the cell size increases, the largest possible

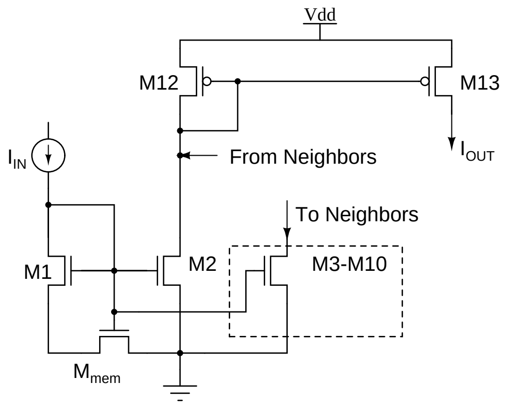


Figure 3.4 B-template realization: Linear non-propagative cell.

network size decreases. Also the computational speed decreases and power consumption increases. In [122], a digital emulated CNNUM is introduced. In [16], a gray-scale multiplier-free digital CNN for spatial filtering is introduced.

3.2.5 Example of a CNN Cell Realization

In practice, the A- and B-templates of Eq. 3.3 can be realized with current mirrors. Figure 3.4 shows the linear non-propagative cell [115], which realizes the B-template. M_{mem} operates as the analog memory of the cell, the current mirror of M1 and M3-M10 conducts the input current to the neighboring cells. The mirror ratio of M1/M2 realizes the coefficient $B_{0,0}$ and the ratios of M1/M3 through M1/M10 realize the other coefficients $B_{-1,-1}$ through $B_{1,1}$. The drain of M12 acts as the summing node for the output currents of the neighboring cells.

Fig. 3.5 shows the realization of a linear resistive grid [120] that can realize an A-template [121]. The coefficients $A_{-1,-1}$ through $A_{1,1}$ are realized with the mirror ratios of M1/M2 and M4/M6-M13. The input of the linear resistive grid is usually the output of the B-template realization. Both of these circuits require unipolar input currents; with bipolar currents, extra circuitry is required [120].

The implementation of a CNN or CNNUM for a specific task can be very inefficient. A more efficient method in realizing hardware with the same functionality can often be achieved by dividing the specific processing tasks into separate categories implemented with dedicated hardware arrays [115]. The resulting processor arrays no longer necessarily realize Eq. 3.1, because, for many tasks, there exist more suitable hardware realizations.

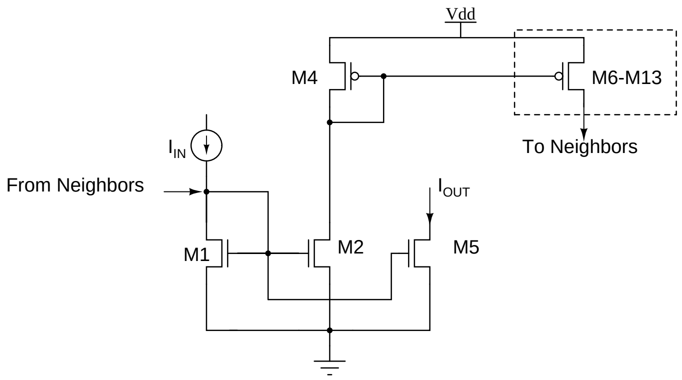


Figure 3.5 A-template realization: Resistive network cell.

3.3 CNN Application Example: Shape Segmentation

As an example of CNN applications, a shape segmentation algorithm [29] is reviewed. This algorithm, outlined in Fig. 3.6, together with the realization of the algorithm's' B/W part [28], is the basis for the algorithms introduced in Chap. 4 and Chap. 5. The segmentation algorithm extracts the shapes from the luminance component (I in Fig. 3.6) of the video sequence. The names of the various steps of the algorithm are from [29]. With modifications, this algorithm is suitable for MPEG-4 Core profile (Chap. 2.4.1) encoders.

The first step of the algorithm (*Edge-Enhancing Low-Pass Filtering*) is a filter that preserves the high-frequency contours of the frame while filtering out high-frequency noise. The contours are detected with a first order gradient operator. The output of this step (L) is formed by adding high-pass and low-pass filtered filtered versions of the frame. The high pass filter is applied to only the parts of the frame indicated by the gradient operator. In the original algorithm, two types of motion are defined: motion between frames I_i and I_{i-3} and motion between successive frames. The motion between frames I_i and I_{i-3} is used in extracting the object borders. The motion between the successive frames is used in defining regions that are to be updated every frame, i.e. intra coded. In Fig. 3.6, *Motion Detection* (MD) finds the motion between frames I_i and I_{i-3} and *Remarkable-Feature Extraction* (RF) finds the motion between two consecutive frames. In both steps, the order of the input frames can be changed. An externally controlled threshold (th) regulates the found *Motion* and *Remarkable-Features*. The threshold controls how many pixels are included in the output area. The *Remarkable-Features* and the low-pass filtered frame are combined in the step *Intra-Frame Segmentation* (S). This step performs the motion analysis of the frame and thus indicates the areas for intra coding. Finally, *Intra-Frame Segmentation* combines the *Motion Detection* and *Intra-Frame Segmentation* results to produce the areas

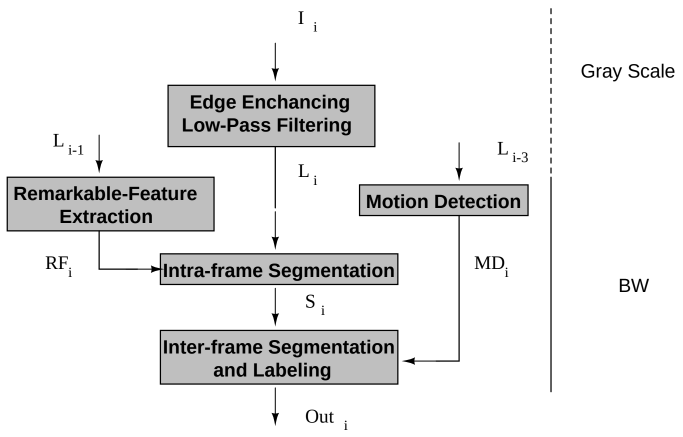

I i
Edge Enchancing Low-Pass Filtering
L i-1
Remarkable-Feature Extraction
L i
L i-3
Motion Detection
RF i
Intra-frame Segmentation
MD i
S i
Inter-frame Segmentation and Labeling
Out i
Gray Scale
BW

Figure 3.6 The CNN segmentation algorithm of [29]. Also indicated are the gray-scale and BW parts of the algorithm.

for inter coding and the final segmented object. Each part of the algorithm has several steps with multiple A- and B-templates. The *Remarkable-Features* and *Intra-Frame Segmentation* parts of the algorithm are used in Chap. 4 and Chap. 5 and are therefore described more closely below. For the exact steps and specific templates of the various parts of the algorithm the reader should see [29].

Remarkable-Feature Extraction, shown in Fig. 3.7a, first calculates the difference between frames I_i and I_{i-1}. The absolute value of the difference frame is then thresholded with the threshold *th* into a binary image. The threshold controls the number of pixels that are included in the binary image. A too-small *th* can lead to under-segmentation[1]. This binary image contains several small shapes. These small shapes are not homogeneous but contain holes. First the holes have to be filled with the Hole Filler template. Then the several small shapes are combined with erosion and dilation operations so that the number of shapes is decreased. The erosion shrinks and dilation enlarges the binary shape. The erosion and dilation operations simplify the frame. Finally, the simplified frame is compared with the output of the hole filler template. This is achieved with the Restoration template. The effect of this step is the removal of shapes deemed too small. All of the templates used in *Remarkable-Feature Extraction*, are originally presented in [123].

The original function of *Intra-Frame Segmentation*, shown in Fig. 3.7b, is to indicate which parts of the frame should be intra coded. This could be achieved by marking these parts as separate objects in MPEG-4 Core profile (Chap. 2.4.1) or designating them into independent slice groups of H.264 (Chap. 2.5). The first step

[1]In [29] *th* was set to a constant value of 8 for all sequences. This is due to the fact that the input video sequences were video conferencing (i.e. “Head & Shoulders”) type of material. For other types of video sequences *th* has to be varied. As the optimal *th* is for other types of sequences is not discussed in [29] *th* was set subjectively for the sequences used in this thesis.

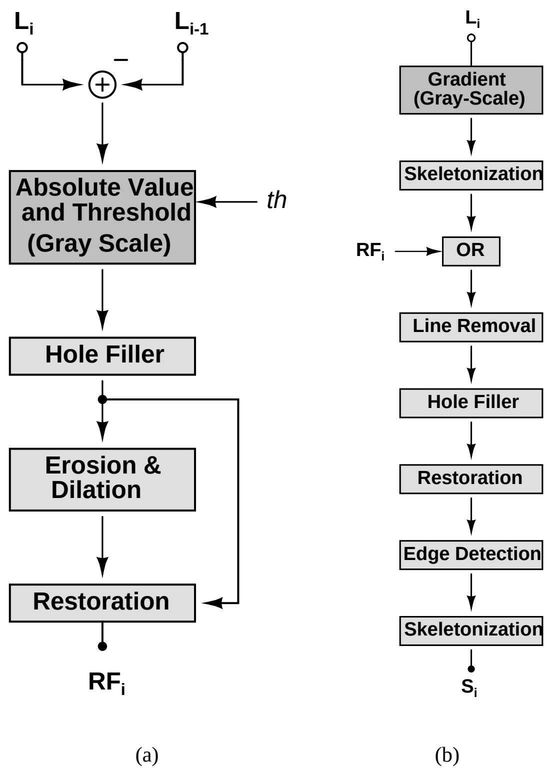


Figure 3.7 Detailed parts of the segmentation algorithm of [29]. a) *Remarkable-Features* b) *Intra-Frame segmentation*

of *Intra-Frame Segmentation* is gray-scale edge detection with a first order derivative operator. The binary lines indicated by the gradient operator are then thinned with the Skeletonization template [124]. This skeleton is then combined with the *Remarkable-Feature* result with an OR-operation. Lines that do not belong to the *Remarkable-Features* are removed with the Insignificant Line Removal template [29]. Small objects are then merged with larger neighbors with the Hole Filler and Restoration templates. Finally, the edges of the remaining objects are detected with a binary edge detection template [125] and simplified with the Skeletonization template.

3.3.1 Implementation of the Shape Segmentation Algorithm with Dedicated Hardware

The segmentation algorithm of [29] can be implemented more efficiently by dividing the tasks into gray-scale and black-and-white operations. A QCIF resolution B/W chip realized in [28] is capable of achieving the B/W tasks of the segmentation algorithm and is able to segment up to 1000 frames/s. With lower frame rates, the processor array part of the chip can be turned off when it is idle, lowering the power consumption to 0.46 mW@30fps [115]. The core size of the B/W processor is 3.2 X 2.6 mm^2, and the whole processor approximately 15 mm^2, when implemented with 0.25μm technology.

The rest of the necessary circuitry that needs to be implemented in order to realize the considered algorithm is reported in [115]. A 64×16 test chip of the gray-scale part of the algorithm is realized in [126] with 0.18μm technology. The size of the array was 950μm X 450μm. From the measurement results it can be calculated that the power consumption of the gray-scale part is below 5 mW with CIF@30fps. This power consumption figure also includes the absolute value and thresholding operations of *Remarkable-Features* and *Motion Detection.*

Chapter 4

Motion Estimation Computational Complexity Reduction

4.1 Description of Test Environment

4.1.1 Description of Video Sequences

Four QCIF-sized video sequences [127] have been used in this thesis. The sequences Carphone, Foreman and Silent Voice depict possible video telephony situations. The sequence News depicts a typical news cast with two news anchors and a picture behind them. The first frames of each sequence is shown in Fig. 4.1. The content of each sequence is described in Table 4.1 [127], [87]. *Simple Motion* refers to slow motion of the moving objects. *Complex motion* refers to faster motion and/or occlusion and appearing objects. The input and output frame-rates are 30fps for all the video sequences in this thesis. All sequences were 100 frames in length.

4.1.2 Implemented Encoders

As the purpose of this research was to study various motion estimation methods and their effect on picture quality and bit-rate, only the various parts of the general encoder shown in Fig. 2.1 (ME, MC, DCT, IDCT Q, Q^{-1}, and VLC) were implemented in Matlab. An additional reason for this was that the presented methods have little effect on the other parts of a full encoder such as rate-control and the CPU. The main consequence of this approach is that the syntax generation was left outside the scope

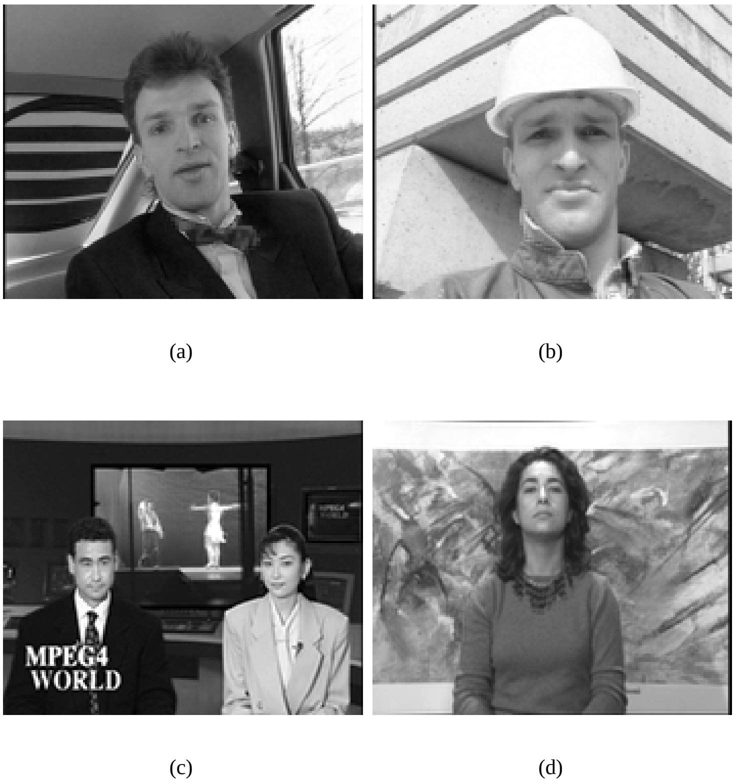


Figure 4.1 First frames of video sequences used in this thesis. a) Carphone b) Foreman c) News d) Silent Voice

of this thesis. Thus, only the payload bits (i.e. bits output from the VLC of each respective standard) are used in the results. The quantization parameter Q_p was used as the variable to control the number of these output bits. The coding sequence was IPPP... and Q_p was constant for the whole sequence. The output bits are referred to as “Coded Bits” in the results. As the transform and VLC are specific to each standard, the “Coded Bits” are interchangeable only within boundaries of the each respective standard. The “Coded Bits” can be proportioned to results in literature with a multiplication factor of 0.3[1]. Most results in literature include the syntax bits, which has to be taken into account in possible comparisons. The above applies to all results in this thesis.

In all simulations, only integer motion vectors were used and MVs were not allowed to point outside frame boundaries. In all H.264 simulations, the deblocking filter was turned off and no macroblock-level decisions were made.

[1]Sequence length = 100, frame rate = 30 fps.

	Spatial Detail	Simple Motion	Complex Motion	Camera Pan
Carphone	Low/Medium	X	X	-
Foreman	Medium/High	X	-	X
News	Low	X	X	-
Silent Voice	Low	X	-	-

Table 4.1 Description of test sequences [127], [87].

4.2 Application Target

The MPEG-4 Core profile (Chap. 2.4.1) enables object-based coding. A Core profile encoder may also have to be able to carry out frame-based Simple profile coding. Frame-based coding is needed when, for instance, the object-based functionalities are not used or the target decoder has only Simple profile capabilities.

It is shown in this chapter that an object-based Core profile encoder can have advantages in frame-based Simple profile encoding. The intermediate results of the CNN segmentation algorithm (Chap. 3.3) can be used in computational complexity reduction of motion estimation. The CNN intermediate results indicate areas of motion within a frame. The areas without motion can be then used to stop the motion estimation (early thresholding) or to indicate MPEG-4 skip modes. All block-based motion estimation algorithms can benefit from this knowledge.

Several algorithms, such as [128] and [129], that combine segmentation and block-based ME, have been presented. However, these algorithms do not comment on early thresholding or skip modes, as is done here. Also, as stated in Chap. 1.1.2.1, no CNN or CNN-type algorithms or implementations exist that comment on block-based motion estimation.

4.3 The Use of Shape Segmentation Intermediate Results in Motion Estimation

The *Remarkable* part of the segmentation algorithm of Chap. 3.3 indicates the areas containing motion within a frame. The structure of the segmentation algorithm is illustrated in Fig. 4.2. This motion information can be used to determine the valid search areas for motion estimation, regardless of whether the video encoding is object-based or frame-based. All the ME algorithms of Chap. 2.2.4.3 benefit from the knowledge of areas with motion. Thus, all the types of motion estimation algorithms of Chap. 2.2.4.3 can be modified to benefit from the method introduced in this chapter. With respect to the output bit-rate, the effect of coding only areas with motion is similar to using a high quantization parameter Q_p value for areas without motion and using a

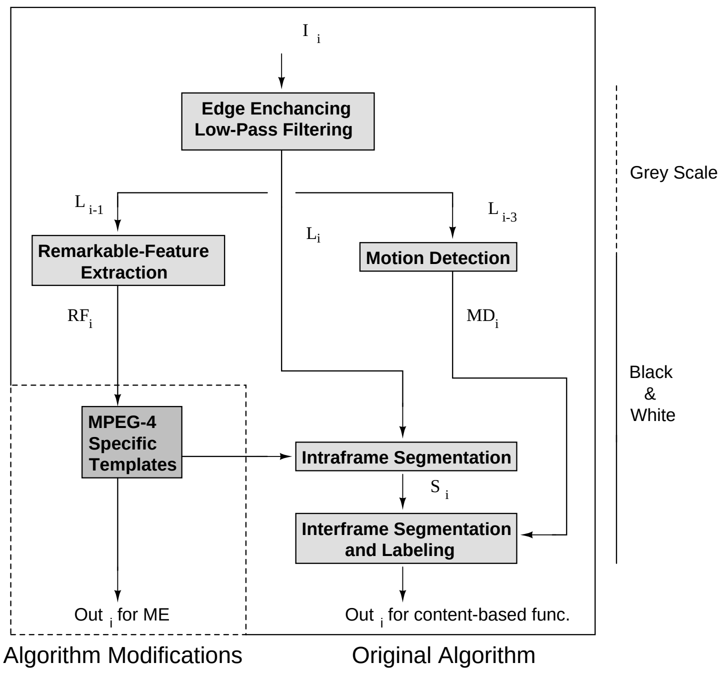

I i
Edge Enchancing Low-Pass Filtering
L i-1
L i
L i-3
Remarkable-Feature Extraction
Motion Detection
RFi
MDi
MPEG-4 Specific Templates
Intraframe Segmentation
S i
Interframe Segmentation and Labeling
Out i for ME
Out i for content-based func.
Grey Scale
Black & White
Algorithm Modifications
Original Algorithm

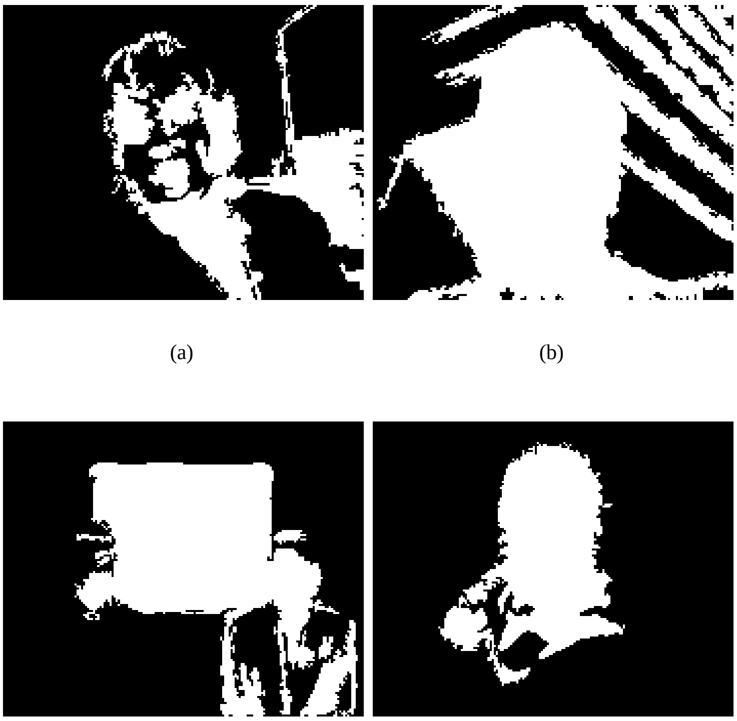
Figure 4.2 Outline of the structure of the original segmentation algorithm (Chap. 3.3) [29] and the modifications that can be used with frame-based coding (Chap. 6). Also indicated are the gray-scale and BW parts of the algorithm. The symbols are explained in Chap. 3.3.

lower Q_p value for areas with motion. With the knowledge of areas with motion the ME algorithm can be made to ignore blocks without motion.

Results from the *Remarkable* part of the algorithm can be seen in Fig. 4.3, where the areas in white indicate motion. In practice, the block of *Remarkable* data corresponding to the current block is checked whether the *Remarkable* block contains any motion (white) pixels. This operation can be achieved with various methods such as a counter or with sequential or parallel OR operations. If no motion pixels are found, the ME is signaled not to perform a search on this block. Thus, this method corresponds to early thresholding (Chap. 2.2.4.2). The no-motion information can also be implemented to indicate the macroblocks on which MPEG-4 skip modes (Chap. 2.4) should be used. For a skipped macroblock, no motion vector or transform coefficients are sent. Thus, in addition to ME, neither the DCT nor VLC have to be performed on the skipped blocks. The skip mode is conventionally determined with either early-thresholding or rate-distortion optimized algorithms (Chap. 2.2.4.6).

With high motion, a large part of the overall frame is included in the found motion. This usually implies camera panning or a changed scene. By setting an appropriate threshold, such events can be detected.

The motion was tested with two types of ME algorithms. FS was chosen as the



(a) (b) (c) (d)

Figure 4.3 Example of the CNN segmentation algorithm intermediate results. The areas with motion are indicated with white. a) Carphone b) Foreman c) News d) Silent Voice

optimal algorithm. Motion Vector Adaptive Search Technique (MVFAST) [92] was chosen as an example of a gradient-based ME algorithm.

4.4 Experimental Results

The segmentation algorithm was integrated into a Simple profile MPEG-4 encoder in Matlab software. The search area was set to [-15,16] and skip modes were enabled. The *Remarkable* results were compared to conventional Full Search motion estimation. The used values for Q_p were 2, 4, 8, 12, 16, 20, 26 and 31. The rest of the test environment is described in Chap. 4.1.2. As the optimal *th* (Chap. 3.3) for various types of video sequences is not discussed in [29], *th* was set subjectively for each sequence. Another possibility would be modifying a conventional rate-control algorithm to adjust *th* according to the rate-control parameters. As the rate-control was not implemented this possibility is not considered further.

The *Remarkable* results of the segmentation algorithm were calculated in two

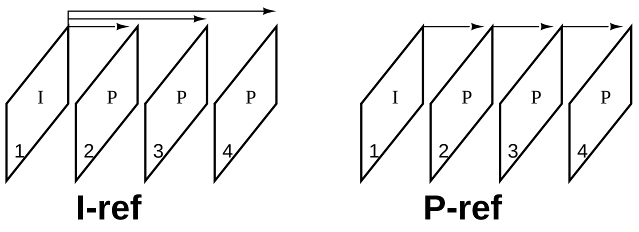


Figure 4.4 Difference between I-ref and P-ref. The arrows imply the frames from which the found motion was calculated.

ways: First, the reference frame from which the motion was calculated was the I-frame. The results from the *Remarkable* part of the algorithm are thus between the current frame and the I-frame. This is referred to as I-ref in the results. Second, the reference frame from which the results from the *Remarkable* part of the algorithm motion were calculated was a frame preceding the current frame. The number of frames between the current and reference frame was varied. This is referred to as P-ref in the results. The number of frames between the current and reference frame is also shown in the results. Fig. 4.4 shows the difference between I-ref and P-ref. The original frames were used as the input to the segmentation algorithm.

4.4.1 Effect of uncoded regions in PSNR values

Due to the unidealities of sensor arrays, all natural video sequences contain random noise [130], [131]. This random noise has a constant component and a time-variant component. For static pixels, the difference in the same pixel of sequential frames is characterized by their variance σ^2. The random noise can only be preserved by using a very low Q_p in the video coding application. In large display applications, such as HDTV, regions without this noise appear to stay unnaturally stationary. In QCIF or CIF-sized mobile applications, the lack of this noise has very little or no subjective difference.

The segmentation algorithm filters out random noise, and thus finds only the regions of true motion. The regions without motion are then either skipped or, in the case of early thresholding, no prediction error is sent. When the PSNR values are calculated from these uncoded areas of the decoded frames , the imperceptible difference arising from the noise forces the PSNR value to an upper limit. So even if no quantization was used and the regions of true motion were coded without error (assuming negligible error from the DCT), the PSNR value for the frame could not exceed the upper limit. The upper-limit value depends on the amount of motion and the variance of the noise in the sequence; an exact value cannot be calculated without knowing the noise distribution. This effect is proportional to the number of skipped blocks and early thresholding when the picture quality is high (PSNR $\overset{\sim}{\geq}$ 35 dB). In lower picture quality with higher Q_p, the difference blocks are quantized to zero, thereby having the same effect as a skipped block or early termination.

4.4.2 Simulation Results

The simulation results for the sequences Carphone, News, Foreman, and Silent Voice, are shown in Figs. 4.5, 4.6, 4.7 and 4.8, respectively. In these figures, the rate-distortion values and the average number of computed SAD values are shown in the upper and lower graphs, respectively. Both results are shown for each value of Q_p. In the graphs, the term “No Segmentation” refers to frame-based coding, e.g. no shape information is used. In the legend, thI and thP correspond to the value of *th* of I- and P-frames, respectively. Also, frP corresponds to the number of frames between the two input frames of the *Remarkable* part of the algorithm.

From the figures, it can be seen that using the intermediate results of the segmentation algorithm yields considerable savings in coded bits. From the analysis of the PSNR graphs, it would seem that the video quality reduction is unacceptably high especially for the lower values of Q_p. Although there is a considerable drop in coded bits the PSNR values can have up to an 8dB reduction for Q_p=2. This PSNR reduction is the upper PSNR boundary due to the difference in random noise (Chap. 4.4.1) in different frames and is concentrated in the uncoded areas of the frame. As the random noise has a negligible effect on the subjective quality of a coded sequence, no artifacts can be seen and the subjective quality of the low Q_p I-ref/P-ref coded sequences is equal to the sequences coded without segmentation information. As an example, the exact rate-distortion values for four middle and low bit-rate Q_p have been collected in Table 4.2. The slight improvement in PSNR that can be seen in the higher Q_p values is also caused by the random noise (Chap. 4.4.1) in the uncoded areas. The subjective quality in the Q_p=20 sequences is equal to the sequences coded without segmentation information.

From the results, it can be seen that the use of the intermediate results has the greatest advantage with video sequences where the movement is in a small area such as Silent Voice, where the movement is confined to the person. With sequences of high motion, such as the camera motion of Foreman, the advantages are reduced. With sequences with motion in extremely small but subjectively important areas, such as the faces in the sequence News, the threshold *th* has to be set with care to avoid a large error in subjectively important areas. In practice, when this algorithm is used, this means that the *Remarkable* part has to be execute once with a low *th* (*th*=2 or *th*=3) even if the segmentation only requires a higher *th*.

The average number of computed Sum of Absolute Differences (SAD) for each

Full Search

	PSNR (dB)				Coded Bits (kbps)			
	Q_p=12	Q_p=20	Q_p=26	Q_p=31	Q_p=12	Q_p=20	Q_p=26	Q_p=31
Carphone	-0.82	-0.25	-0.07	0.00	-62.9	-26.1	-17.5	-12.1
News	-0.37	0.04	0.09	0.11	-61.9	-30.4	-20.0	-16.8
Foreman	-0.21	0.02	0.05	0.10	-94.6	-47.3	-30.1	-23.1
Silent	-0.21	0.05	0.10	0.14	-47.0	-22.0	-14.8	-11.1

MVFAST

	PSNR (dB)				Coded Bits (kbps)			
	Q_p=12	Q_p=20	Q_p=26	Q_p=31	Q_p=12	Q_p=20	Q_p=26	Q_p=31
Carphone	-0.63	-0.04	0.19	0.21	-105.6	-53.6	-41.8	-31.5
News	-0.24	0.16	0.19	0.30	-80.4	-41.3	-33.3	-38.0
Foreman	-0.17	-0.00	0.11	0.19	-118.0	-51.9	-46.0	-36.6
Silent	-0.14	0.12	0.18	0.27	-61.8	-30.4	-22.0	-17.9

Table 4.2 Exact differences in rate-distortion between frame-based coding (“No Shape”) and P-ref for Q_p=12, Q_p=20, Q_p=26 and Q_p=31.

value of Q_p is shown in the lower graphs of Figs. 4.5, 4.6, 4.7 and 4.8. The reduction in the number of SADs is inversely proportional to the amount of motion in the sequence. When using MVFAST the number of SADs stays more constant with I-ref/P-ref than when the sequences are coded without the segmentation intermediate results. This can be especially seen for the sequence News in Fig. 4.6 and to a lesser extent for the sequences Carphone (Fig. 4.5) and Silent (Fig. 4.8). This can be of use when designing hardware for MVFAST-type motion estimation, as the foreknowledge of the constant number SADs leads to smaller computational complexity overhead for the hardware. However, before the design of such hardware, the phenomenon would have to be verified with a statistically wide range of video sequences.

From the results, it can be concluded that computing the motion with a single reference frame, as is the case in the I-ref results, has applications only in extremely low-power encoders where I-ref could be used in, for example, a coding scheme where the motion compensation of more than one frame refers to the same reference frame. Such a method would still offer coding advances over the intra coding used in current mobile coders intended for MMS (Multimedia Messaging Services) applications. For encoders capable of inter coding the reference frame has to be varied as in the P-ref results.

4.5 Effect of the Algorithm on Power Consumption of Motion Estimation

The effect of this algorithm on the power consumption of motion estimation depends on several factors. First, the algorithm has an advantage only in real-time codecs which include object-based and frame-based coding and pre-processing with segmentation. An example could be a transmission system with a Core profile transmitter codec and a Simple profile receiver. In such cases, the algorithm's advantages emerge with frame-based encoding.

The actual power savings depend on what type of motion estimation algorithm is used and how this algorithm is realized in hardware. Exact power consumption reduction figures would require the implementation of specific realizations. Presented below are estimates on two different ME realizations.

The first example could be Full Search implemented on a 1-D or 2-D systolic array [101]. Such an implementation has a very regular architecture with only local connections and little control circuitry. Adding the presented algorithm to such an implementation would decrease the power consumption but would require additional control overhead. For the macroblocks with motion (as indicated by the *Remarkable* data) both the 1-bit *Remarkable* data and the 8-bit pixel values have to be transferred into local memory. Also, for these macroblocks, an OR-operation has to be performed on the *Remarkable* data in addition to the SAD of the pixel values in the search area. For the macroblocks without motion only the 1-bit *Remarkable* data has to be transferred into local memory and only the OR-operation has to be performed.

By defining B_f as the total number of macroblocks in a frame and B_w as the number of macroblocks without motion, the power consumption reduction ratio with and without the *Remarkable* data can be defined as

$$power\,ratio = \frac{B_w \cdot o_w + (B_f - B_w) \cdot (o_w + o_m)}{B_f \cdot o_m} \tag{4.1}$$

where the parameters o_m and o_w are defined separately for the number of computational operations and the amount transferred data. In o_m and o_w, the subindex *m* refers to the macroblocks with motion and subindex *w* refers to the macroblocks without motion. By additionally defining

$$\rho = \frac{B_w}{B_f} \tag{4.2}$$

and

$$\varpi = \frac{o_w}{o_m}, \tag{4.3}$$

the reduction ratio can be simplified to

$$power\,ratio = 1 - \rho + \varpi. \tag{4.4}$$

As with o_m and o_w, this ratio is also defined separately for the number of computational operations and the amount of transferred data. The power consumption is then derived with

$$P_a = power\,ratio \cdot P_o \tag{4.5}$$

where P_a is the power consumption with the presented algorithm and P_o without the presented algorithm. This simplification is based on the assumption that, for both the computation and data transfer, the power consumption of the *Remarkable* data and pixel data does not correlate.

From the *Remarkable* data of QCIF-sized Carphone, Foreman, News, and Silent, for 16x16 blocks, the average $\rho = 0.48$. For the data transfer[2] $\varpi = 1/9$ and for the computational complexity[3] $\varpi = 1/((2r+1)\cdot 3 \cdot 8) \approx 0$ where N^2 is the block size and r the search area. Thus, the reduction ratio for the data transfer is 0.63 and for the SAD computation 0.52. For larger image sizes the reduction ratios would be higher as the macroblocks can be better adapted to the *Remarkable* data.

In [132], a 280 mW CIF@30fps MPEG-4 codec using FS is presented. The power consumption of the codec's motion estimation is 30 mW[4]. From [12] it can be estimated that the data transfer in FS ME is in the range of 70% of the total power consumption. Using this memory power consumption percentage, the total reduction ratio is 0.60 and the ME power consumption of [132] would drop to 18 mW by using the presented method.

Another example could be an ASIC implementation of a ME algorithm, such as PMVFAST [94], which already includes many separate power consumption reduction methods. The PMVFAST algorithm incorporates a predictor set of 6 MVs and various early thresholds. The power savings of merging such a ME algorithm with the presented algorithm would be generated from the simplification of the control circuitry. As parts, such as early thresholds, of PMVFAST and the presented algorithm overlap, the control circuitry of the ME algorithm could be simplified.

In [11], a 0.4 mW ME realization is presented. The realization incorporates a predictive gradient search algorithm. The algorithm of [11] computes a minimum of 8 SADs[5] for the blocks where the best predictor indicates the minimum SAD value. By using this value for all macroblocks, which is a conservative estimate, $\varpi = 1/(8 \cdot 3 \cdot 8) = 0.005$ for the SAD computation. For the data transfer ϖ is unchanged. With these values, the total reduction ratio remains at 0.60 and the power consumption of [11] would drop to 0.24 mW.

The power consumption of the required parts of the segmentation algorithm is analyzed in Chap. 5.4.1. The analysis also applies here, although the presented method

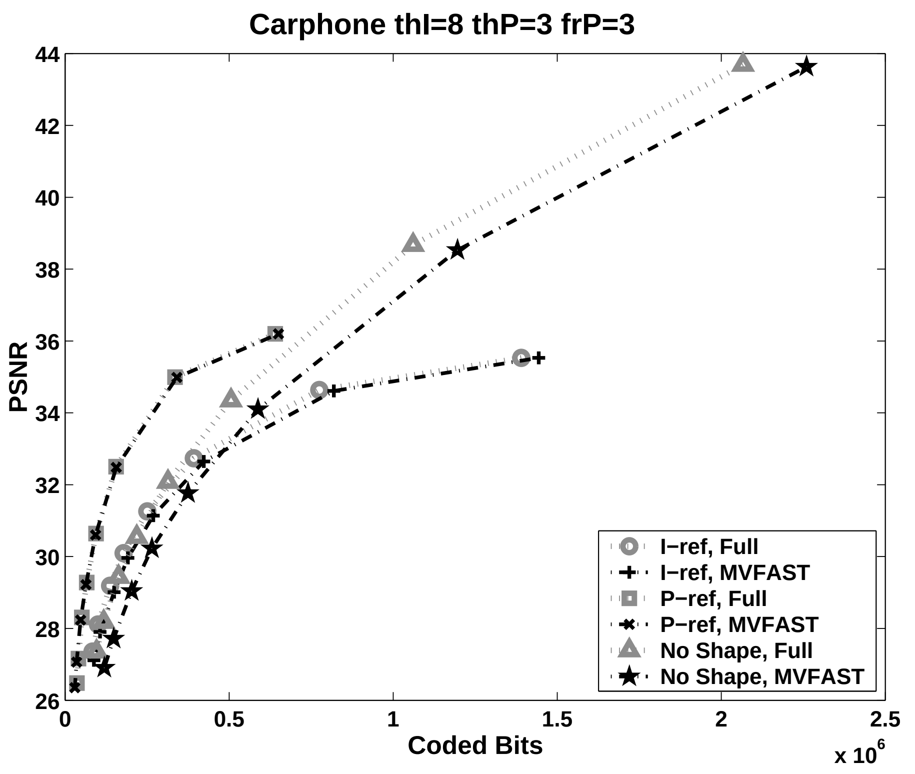
[2]Assuming all data is reused and thus fetched from memory only once.

[3]No. of operations for a block without motion $\cong N^2$(1-bit OR). No. of operations for a block with motion $\cong (2r+1)\cdot 3 \cdot N^2$(8-bit SAD).

[4]Estimated from a graph.

[5]Four predictors plus four points around the best predictor.

is intended for use in implementations where segmentation is used as a pre-processing step.



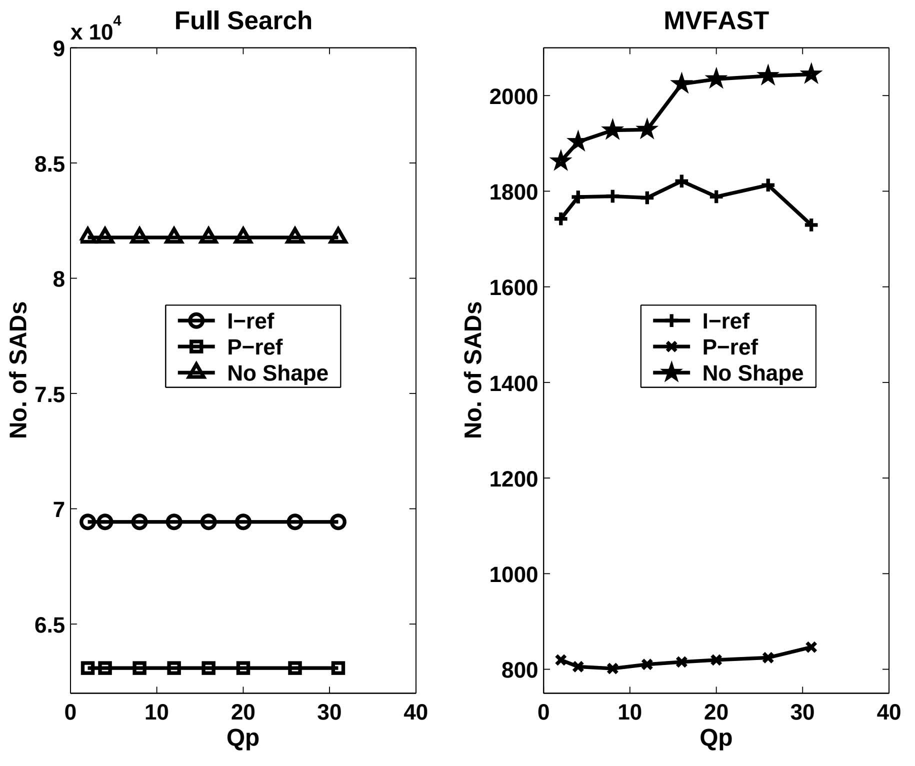


Figure 4.5 Carphone rate-distortion and number of SAD graphs with varying values of Q_p (The subjective quality of the video sequence differs from the quality presented by the PSNR values, chap. 4.4.1)

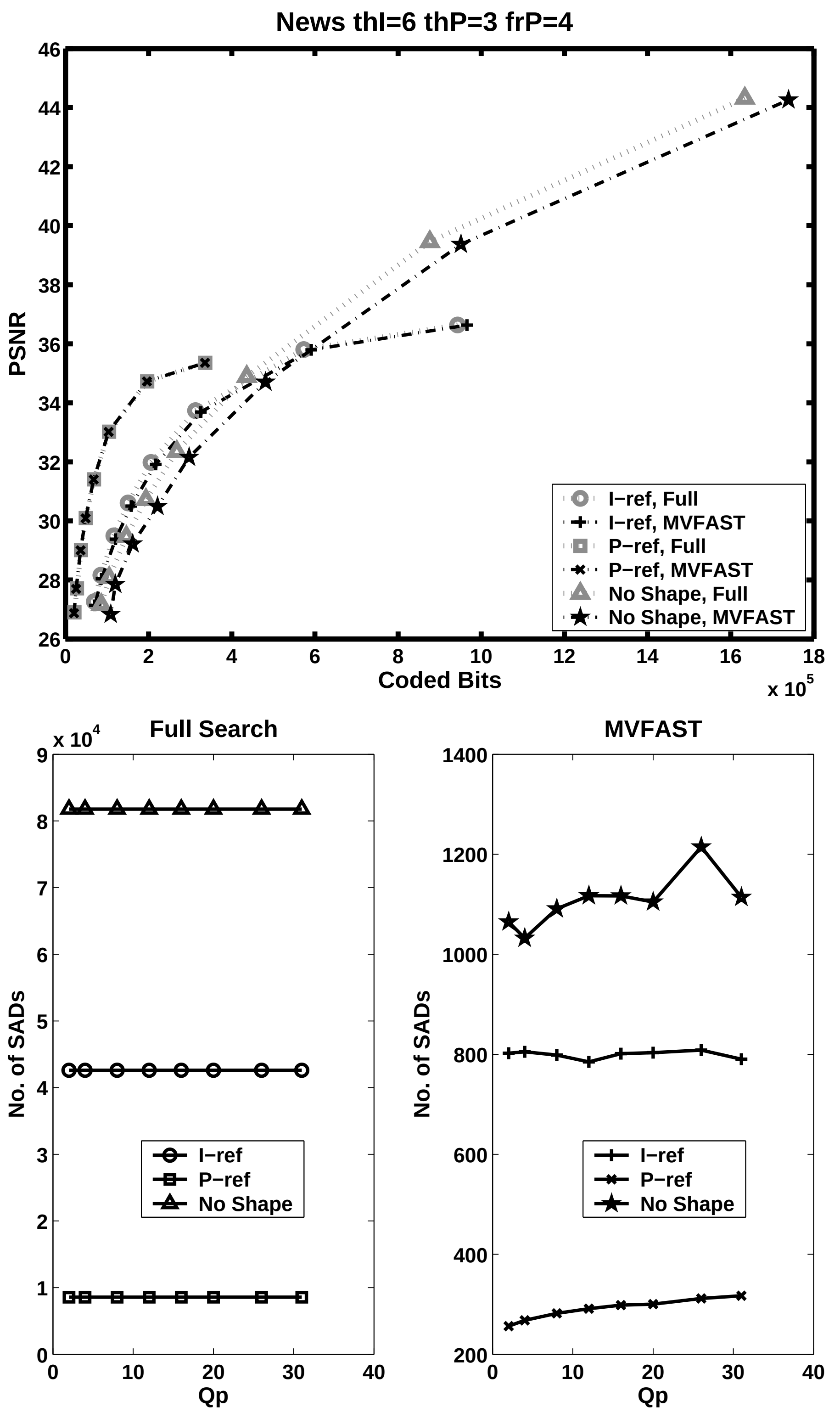
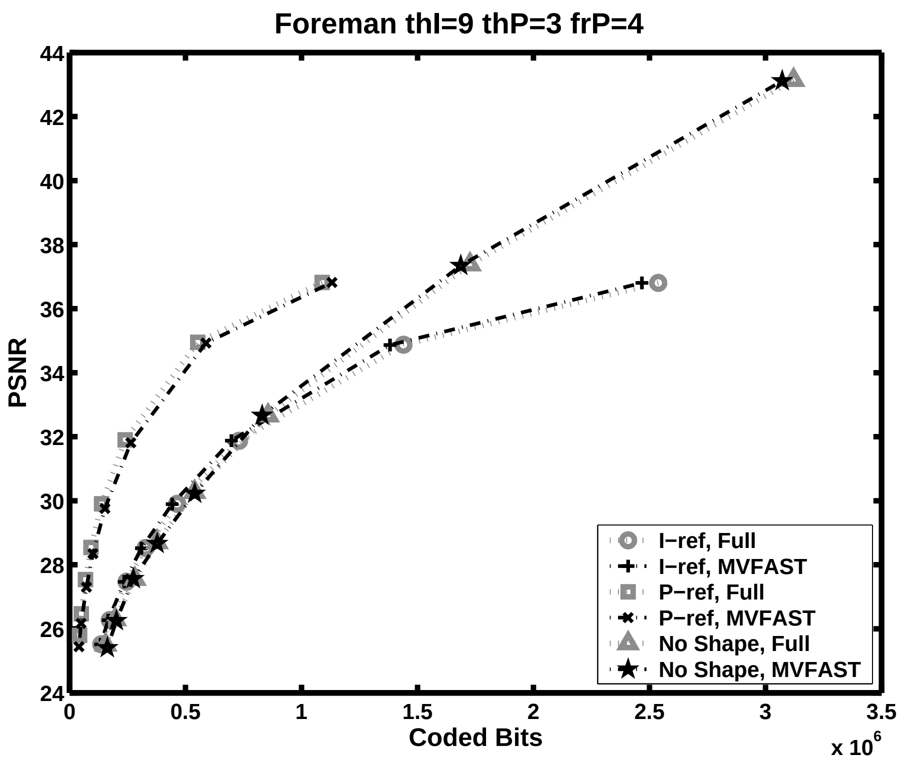


Figure 4.6 News rate-distortion and number of SAD graphs with varying values of Q_p (The subjective quality of the video sequence differs from the quality presented by the PSNR values, chap. 4.4.1)



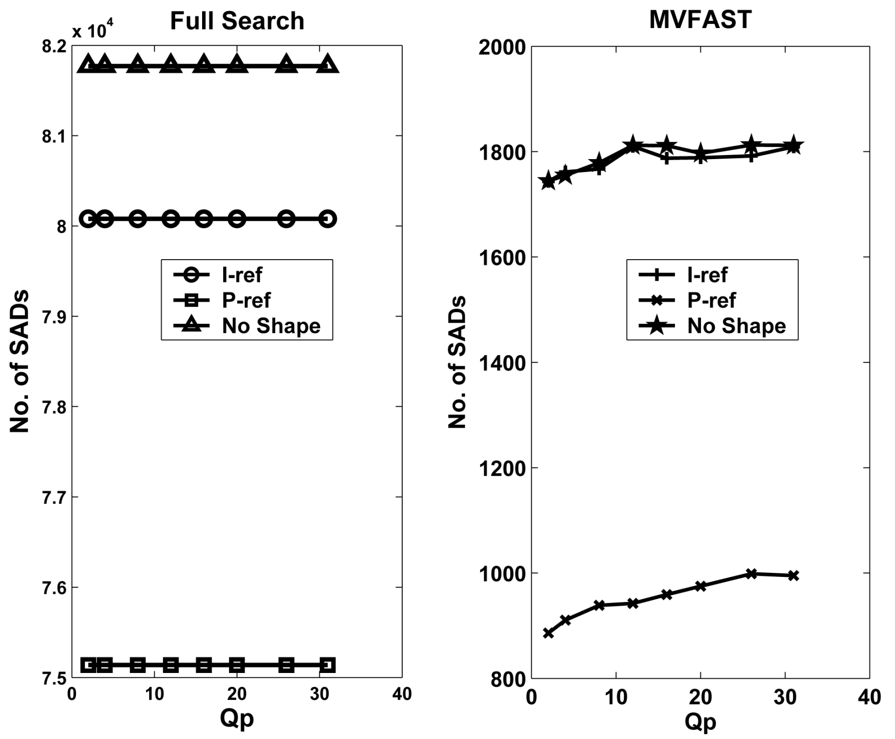


Figure 4.7 Foreman rate-distortion and number of SAD graphs with varying values of Q_p (The subjective quality of the video sequence differs from the quality presented by the PSNR values, chap. 4.4.1)

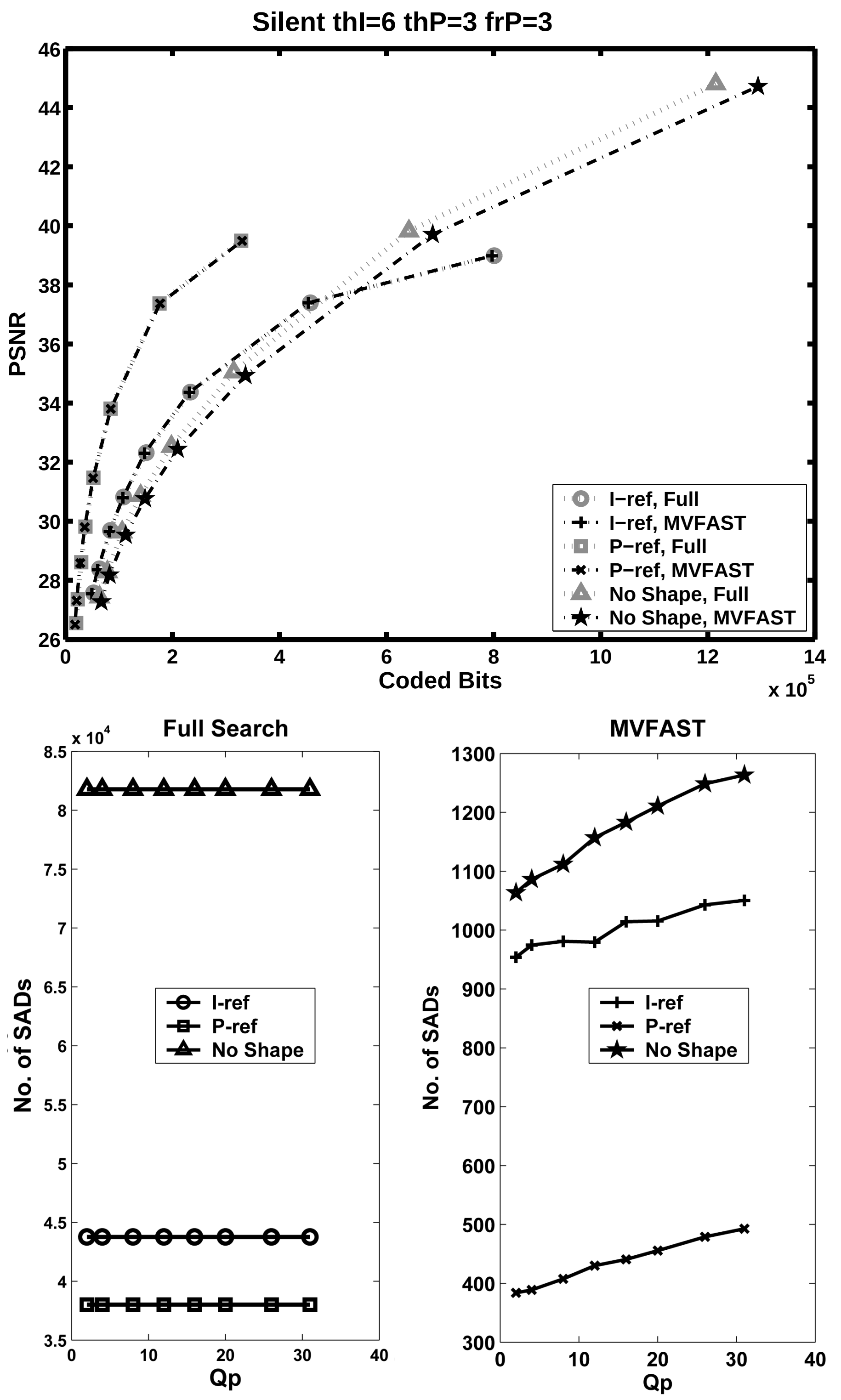


Figure 4.8 Silent Voice rate-distortion and number of SAD graphs with varying values of Q_p (The subjective quality of the video sequence differs from the quality presented by the PSNR values, chap. 4.4.1)

This page is intentionally left blank.

Chapter 5

Algorithm for H.264 Motion Estimation Partitions

5.1 Application Target

A part of the increased compression efficiency of H.264 (Chap. 2.5) is due to the seven different block sizes that can be used in H.264 ME. These block sizes, which are shown in Fig. 2.12, range from 16x16 to 4x4. The basic concept behind variable block-size ME is that smaller block sizes are more efficient in areas of spatial detail, and at the same time, larger block sizes are more efficient in homogeneous areas. When all of these sizes are used for motion estimation in H.264, the number of operations increases to NxMx(2W+1)^2, where M is the number of block types, N the number of reference frames and $\pm$W the search area. With a single block type and reference frame the number of operations is (2W+1)^2. Even though the distortion measure values for the smaller blocks can be reused for the larger blocks, the number of operations is computationally prohibitive for mobile terminals. Thus the first low-power hardware implementations of H.264 are likely to use only 16x16 block sizes. The optimum block size is determined with Lagrange optimization (Chap. 2.2.4.6) [95]

$$J\,(s_k, I_k \mid \lambda) = D(s_k, I_k) + \lambda_{Motion} \cdot R(s_k, I_k) \tag{5.1}$$

where

$$\lambda_{Motion} = \sqrt{0.85 \cdot 2^{\frac{QP_{H.264}-12}{3}}}. \tag{5.2}$$

As can be seen from Eq. 5.2 and is shown in Fig. 5.1, the optimum partition depends on QP$_{H.264}$. Fig. 5.1 shows the block-size distribution of frame 2 of the sequences Foreman and News for QP$_{H.264}$ = [1, 24, 50]. Also shown is the proportional distribu-

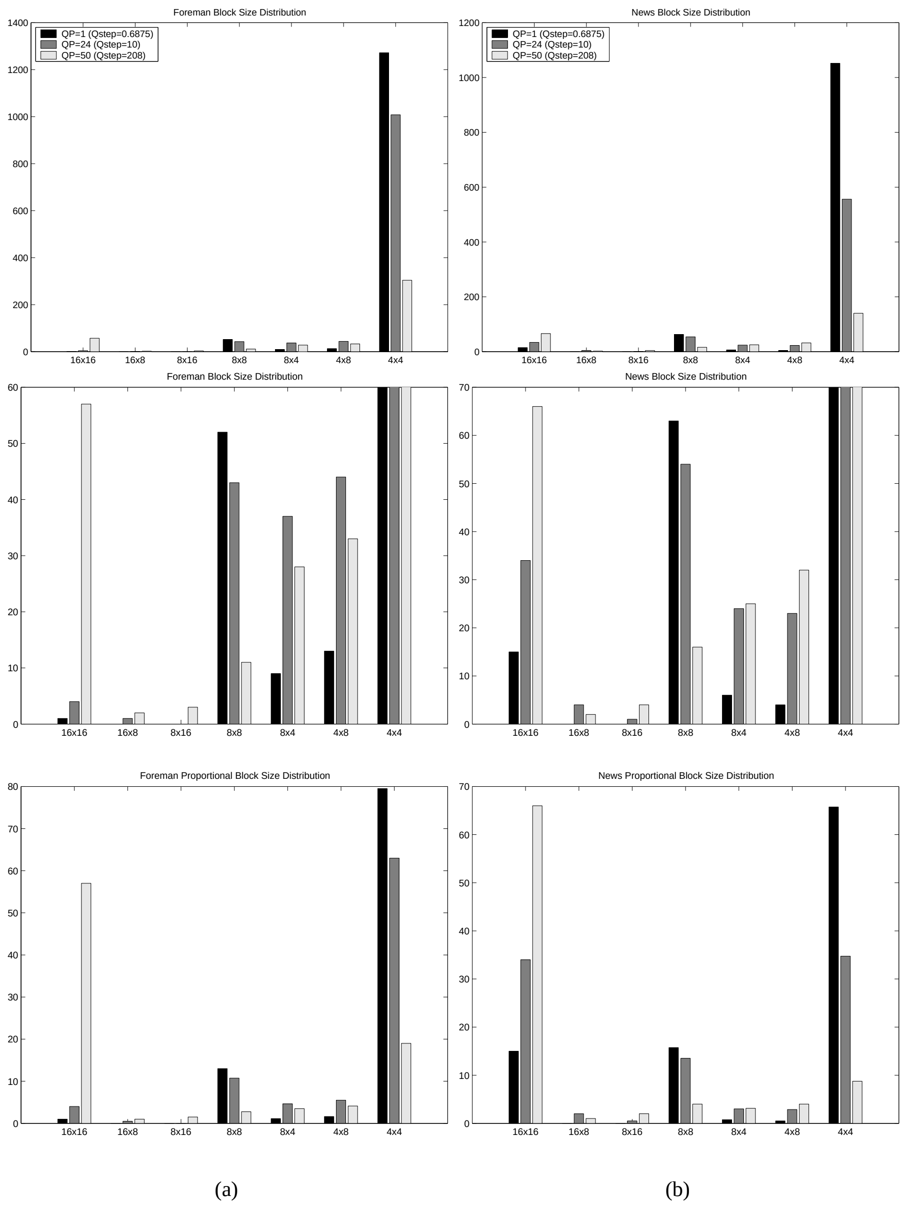
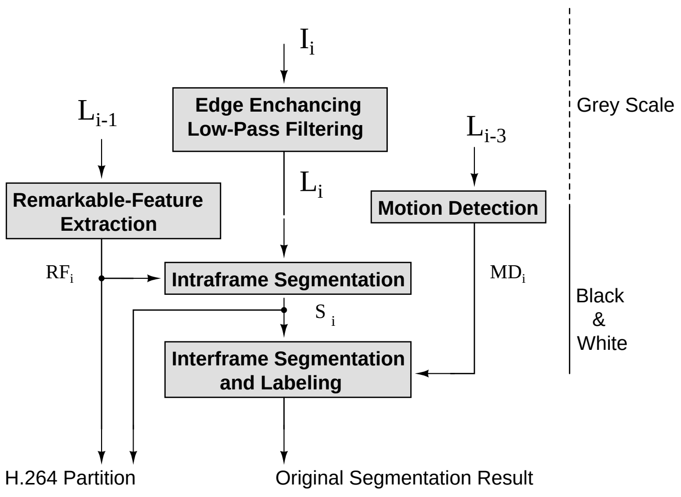


Figure 5.1 Distribution of block sizes subject to Q$_p$ (top row), close-ups (middle row), and proportional distribution (bottom row). a) Foreman. b) News

tion e.g. the size of the area covered by the various block sizes.

A method of determining the ME partition with the intermediate results of the segmentation algorithm of Chapter 3.3 is presented in this chapter with a view to decreasing the computational power requirements of the H.264 encoder. Also determined are skipped blocks and the early termination of motion estimation.



I_i
Edge Enchancing Low-Pass Filtering
L_{i-1}
L_i
L_{i-3}
Remarkable-Feature Extraction
Motion Detection
RF_i
Intraframe Segmentation
MD_i
S_i
Interframe Segmentation and Labeling
H.264 Partition
Original Segmentation Result
Grey Scale
Black & White

Figure 5.2 Outlined is the structure of the original segmentation algorithm (Chap. 3.3) [29]. Also indicated are the gray-scale and BW parts of the algorithm. The symbols are explained in Chap. 3.3.

No CNN or CNN-type algorithms or implementations exist that comment on variable block-size ME. Several conventional variable block-size estimation algorithms for H.264 have been presented. In [133], the correlation of previously computed block sizes is taken into account in predicting the partition. In [134], the partition is estimated from the MVs of 4x4 blocks. In the presented method, no search is needed in determining the partition.

5.2 Partition Algorithm

The partition algorithm has two versions, an earlier version (Chap. 5.2.1) and a later version (Chap. 5.2.2). As in Chap. 4, both methods use the intermediate results of the segmentation algorithm described in Chap. 3.3 [29]. The segmentation algorithm is shown in Fig. 5.2. Partition method 1 uses the results of the *Remarkable* part of the algorithm with three different thresholds (*th*). Partition method 2 uses the *Remarkable* and *Intra-Frame Segmentation* parts of the algorithm.

5.2.1 Partition Method 1

In Fig. 5.3, the results of *Remarkable* for frame 13 of the sequences Foreman and News are shown. The white area of Fig. 5.3 indicates the region of active motion. To attain the variable block sizes within the regions of motion, the method suggested here performs the *Remarkable* part of the algorithm three times with the value of *th* increased on every pass. In segmentation applications, a high *th* would lead to under segmentation, but here the high *th* reveals the contours within a larger region of active motion. The pseudo code for the operation is shown below

```
if(not(previously assigned))
  count(block size)
    if(all 0)
      assign(skip / early)
    if(all 1)
      assign($MV_{\text{block size}}$)
    else
      block size = block size -1
```

where block size = [16x16,16x8,. . .,4x4] and the "count" operation counts the number of white (active motion) pixels in the current macroblock (block size = 16x16) or sub-block. Naturally, for the 4x4 block size, the MV is assigned in all cases except *if(all 0)*.

The pseudo-code operation is repeated for each *Remarkable* frame. Thus, the block sizes are achieved by first assigning block sizes to the highest *th* frame, then assigning block sizes to the areas where the second highest *th* frame is, and so forth. In Fig. 5.3, the top row (*th*=9) dictates the first pass of block sizes. Secondly, the middle row (*th*=6) dictates the block sizes of the second pass. The block sizes from the first pass are kept unchanged on the second pass. The bottom row dictates the final block sizes (*th*=3). On the third pass, both the previous block sizes are kept unchanged. The number of pixels with active motion within each block determines the macroblock and sub-block sizes. This can be achieved with, for example, an OR operation. The bottom row of Fig. 5.3 shows the partition result. The 16x16 blocks without motion are assigned skip modes. As skip mode is available only for 16x16 blocks, early termination is indicated for 8x8 and 4x4 blocks. Due to the large computational power of local processor arrays (Chap. 3.3.1) performing the *Remarkable* part of the algorithm three times increases negligibly the power consumption of the processor array. With multiple reference frames, this method could be implemented on successive frames. This method does not comment on the Intra/Inter decision.

5.2.2 Partition Method 2

The partition method uses two components of the original segmentation algorithm. The results from the *Remarkable* part of the algorithm are used to indicate the active regions of motion as in partition method 1. Here, only a single *th* value is used. The *Intra-frame segmentation* results are used to indicate the edges within the frame. The

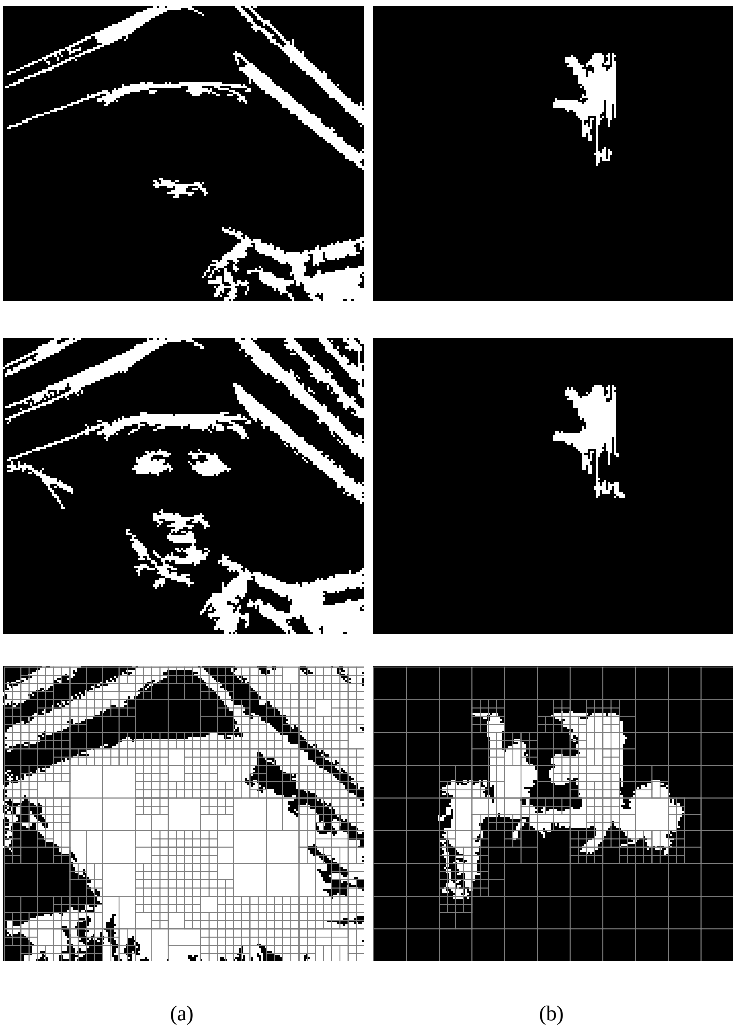


Figure 5.3 *Remarkable* results. Top row *th*=9. Middle row *th*=6. Bottom row *th*=3 and partition result 1. a) Foreman b) News.

pseudo code for the *Intra-frame segmentation* frame is

```
count(block size = 4x4)
  if(not(all 0))
    assign($MV_{4x4}$)
```

For the *Remarkable* frame the operation is the same as in partition method 1.

Thus, in this partition method, the block-size is assigned in two steps: 1) The smallest block sizes are assigned to the image gradients indicated by *intra-frame segmentation*. An example of image gradients is shown in Fig. 5.4a. The first step does not assign block sizes on the areas of the image without gradients. 2) The active regions of the image (shown in Fig. 5.4b) are assigned the largest possible block sizes that do not conflict with the assignments from the first step. An example of the partitioning result is shown in Fig. 5.4c. Skip modes and early termination thresholds are assigned as in partition method 1. These regions are shown shaded in gray in Fig. 5.4d.

5.3 Experimental Results

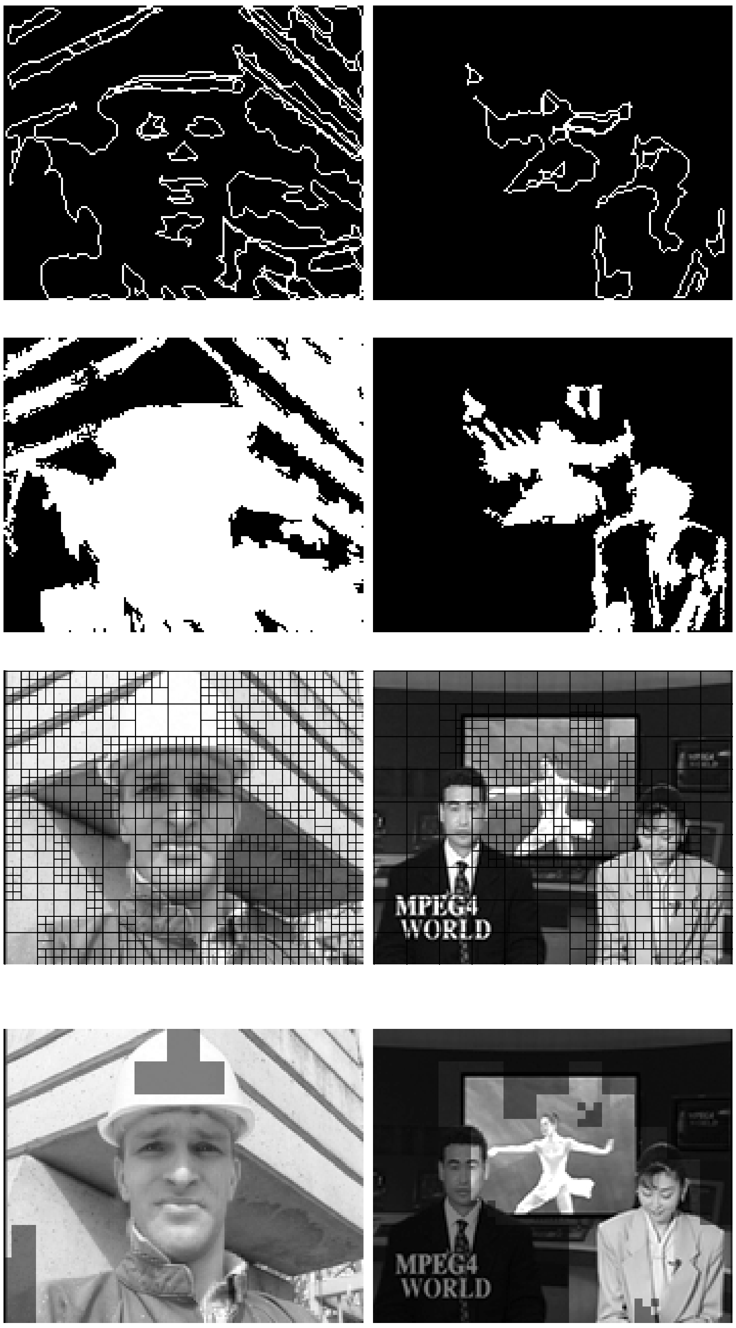
The segmentation algorithm was integrated into a Baseline H.264 encoder in Matlab software. The search algorithm was Full Search with a search area of [-15,16], skip modes were enabled, and the deblocking filter was turned off. The algorithms were simulated both with and without skipped blocks and early termination. Full Search Lagrange optimization and ME using only 16x16 blocks were used as reference. The 16x16 ME represents a typical low-power mobile terminal case. The used values for $QP_{H.264}$ were 10, 22, 28, 33, 38, and 50. The rest of the test environment is described in Chap. 4.1.2.

5.3.1 Comparison of the Methods

The partition method 1 rate-distortion results for the QCIF-sized sequences Foreman and News are shown in Fig. 5.8 and Fig. 5.9, respectively. The rate-distortion results for method 2 are shown in Fig. 5.10 and Fig. 5.11, respectively. Table 5.1 shows the average percentage of macroblocks and sub-blocks for both sequences. In Fig. 5.12, the proportional distribution of each block size is shown. From Fig. 5.12 the average area covered by each block size can be seen.

The effect of skipped blocks and early thresholding on high PSNR values, which is explained in Chap. 4.4.1, can also be seen in the R/D results of these sequences. The PSNR values for the low $QP_{H.264}$ sequences coded with skip modes are again significantly lower but, as is explained in Chap. 4.4.1, this has little subjective meaning, especially with small frame sizes such as QCIF or CIF. For high $QP_{H.264}$, the skip block sequences are objectively comparable to the Lagrange results for both video sequences. As the implementation target is low bit-rate applications, the higher $QP_{H.264}$ values are the most important.

When skip modes are not used with Foreman, for $QP_{H.264}$ values of 10, 22, and 28, it can be seen that there is a slight drop in PSNR and a slight increase in Coded Bits and, thus, a slight drop in performance. For the $QP_{H.264}$ values of 33, 38, and 50 there is a increase both in PSNR and Coded Bits. Thus, the conclusion can be made that both methods achieve comparable R/D performance with high values of $QP_{H.264}$.



(a) (b)

Figure 5.4 Top row: 1st step; Gradients. Second row: 2nd step; regions indicating motion. Third row: Partition result. Bottom row: Skipped blocks and early termination blocks shaded in gray. a) Foreman frame 11. b) News frame 96.

Foreman

	Lagrange			Method 1	Method 2
	Ave.	QP=10	QP=38		
16x16 (Skip)	4.30	0.46	10.43	1.71 (54.91)	1.48 (53.74)
16x8	0.24	0.01	0.85	0.14	0.00
8x16	0.18	0.00	0.64	0.16	0.00
8x8 (ET)	4.39	3.76	5.75	7.25 (58.78)	7.80 (57.38)
8x4	3.44	0.66	7.30	1.46	0.00
4x8	3.73	1.55	6.75	1.18	0.00
4x4 (ET)	83.72	93.56	68.28	88.11 (6.24)	90.72 (19.03)

News

	Lagrange			Method 1	Method 2
	Ave.	QP=10	QP=38		
16x16 (Skip)	26.65	11.24	41.55	19.11 (99.11)	17.45 (98.32)
16x8	0.17	0.02	0.35	0.17	0.00
8x16	0.14	0.01	0.27	0.12	0.00
8x8 (ET)	6.97	10.54	4.29	7.59 (88.74)	9.78 (88.19)
8x4	2.37	0.82	3.90	2.02	0.07
4x8	1.84	1.01	2.51	0.51	0.01
4x4 (ET)	61.87	76.36	47.13	70.47 (10.13)	72.70 (24.66)

Table 5.1 Percentage of sub-block partition averaged over 100 frames. The numbers in braces are the skip and Early Termination (ET) percentages of their respective block sizes.

With News, for all values of $QP_{H.264}$ except 50, there is a slight drop in PSNR and a slight increase in Coded Bits and, thus, a slight drop in performance. For News, the conclusion can be thus made that there is a slight overall drop in performance.

However, both methods outperform the 16x16 ME for both sequences and all $QP_{H.264}$ when the subjective effect of background random noise is taken into consideration. This is significant, considering the target application.

From Table 5.1 and Fig. 5.12 it can be seen that, for both methods, the statistical distribution of the block-size partition resembles more the low $QP_{H.264}$ Lagrange partition than the high $QP_{H.264}$ partition. Although the statistical distribution does not reveal any resemblance between the various partitions, the comparability of the R/D results would seem to indicate that the partitions also resemble one another. Exact partition resemblance analysis would require the development of a suitable resemblance metric. Also, the knowledge of the fact that various block-size partitions can have similar performance in terms of R/D could be used to derive a fast Lagrange algorithm similar to motion estimation early thresholding.

From the R/D figures the conclusion could be made that the partition methods are equal. But, as method 2 uses two frames of data as opposed to the three frames of method 1, the final conclusion that method 2 outperforms method 1 can be drawn.

5.4 Effect of the Algorithm on Power Consumption

The motion estimation power consumption analysis of Chap. 4.5 is also valid for this algorithm. This algorithm of this chapter indicates the same data as the algorithm of Chap. 4 with the addition of the indication of variable block sizes. Thus, the power consumption benefits of this algorithm can be assumed to be larger when similar comparisons of Chap. 4.5 are made. Again, exact power consumption reduction figures would require the implementation of specific realizations.

In [137], a FS variable block-size ME architecture is introduced. In the architecture, the SAD values of 4x4 blocks are first computed. These SAD values are then united to form the SAD values for the larger blocks. The MVs corresponding to these SAD values would then be passed on to a Lagrange stage which is not implemented. The compare stage takes $18.5 \cdot 10^3$ gates out of a total of $154 \cdot 10^3$ gates. By using the presented method, the compare stage could be left out of the realization resulting in a 12% percent reduction in gates. Assuming that all the stages in the realization have the same switching activity (Eq. 2.16), the reduction in the number of gates translates straight to reduction in power consumption.

With partition method 1, from Carphone, Foreman, News, and Silent, the average ratio between 4x4 blocks with and without motion is $\rho = 0.58$ (Eq. 4.2). In this algorithm, three frames of binary data are transferred making $\varpi = 1/3$ (Eq. 4.3) for the data transfer. Thus, the reduction ratio (Eq. 4.4) for the data transfer is $1 - \rho + \varpi = 0.75$. As the OR-operation has to be performed on all three binary frames, for the SAD computation, $\varpi = 3/((2r+1) \cdot 3 \cdot 8) \approx 0$ where r is the search area. Thus, for the SAD computation, the reduction ratio is $0.75 \cdot (1 - 0.12) = 0.66$. Using the power consumption percentages of Chap. 4.5 (70% data transfer, 30% SAD computation) and Eq. 4.5 the total reduction ratio is 0.73.

In addition to this power consumption reduction, the Lagrange stage would also be left out resulting in additional savings. In the Lagrange stage the VLC of each MV corresponding to each subblock has to be performed. For power consumption reduction figures in this case, encoder complexity analysis would have to be performed as is done for the decoder in [138]. Such an analysis is outside the scope of this thesis.

5.4.1 Power Consumption of the Algorithm with Dedicated CNN

Implementing all the separate parts of the segmentation algorithm of [29] just to achieve the benefits of the algorithm of this chapter is not feasible. Thus, only the relevant parts of [29] (*Low-pass filtering, Remarkable, and Intra-Frame Segmentation*) would be included in such an ME implementation. Additionally, the simplification of these relevant parts should also be investigated.

Template	Time	Template Usage	
		Remarkable	*Intra-Frame Seg.*
Hole Filler	60 ns	1	1
Erosion	11 ns	1	0
Dilation	11 ns	1	0
Restoration	0.9 μs	1	1
Skeletonization	22 μs	0	2
OR	100 ns	0	1
Line Removal	2 μs	0	1
Binary Edge Detection	100 ns	0	1

Table 5.2 Computation times of binary templates with VDD=1.2V [139], [140].

The first logical simplification choice would be the low-pass filter operation. The gray scale operations of *Remarkable* (absolute value and threshold) can be implemented with cell logic. Thus, without the low-pass operation, the only remaining gray scale operation would be the edge detection of *Intra-Frame Segmentation.* With only the gradient operation the power consumption of the gray scale implementation of [126] would drop to under 1 mW (CIF@30fps). The rest of the algorithm could be implemented with binary templates. Having only binary operations would make possible a realization that could be incorporated with the motion estimation implementation of Chap. 7.

Instead of using positive-range high-gain CNN [28], to achieve lower power consumption, the relevant binary parts of the segmentation algorithm could be realized with the techniques presented in [139]. In the positive-range high-gain CNN, the multiplier coefficients are programmable positive and/or negative real numbers whereas in the binary CNN model of [139], the multiplier coefficients are one-bit values. The binary CNN model enables a compact and fast realization. An improved version of the binary CNN model is presented in [140]. As is shown in [140], all the templates of [29] can be implemented with the binary CNN model.

The computation times of the required binary templates, acquired from the measurement data of [139][1], are shown in Table 5.2. The values of Table 5.2 are based on an CIF-sized frame. From the values of Table 5.2, the total computation time for the binary templates is 48 μs. From [139], the measured power consumption is 9 μW/cell. Thus, with Eq. 7.12 and CIF@30fps, the total power consumption for the binary templates is 1.3 mW (VDD=1.2V)[2].

For the cell logic, from Table 7.5, it can be seen that the computation time is 200

[1]The actual computation times of [139] are longer due to the required inversion between templates. With the improvements presented in [140] the inversions are no longer required. Thus, the computation times presented here are without the inversions.

[2]The chip of [139] was also simulated with VDD=0.6V, where the power consumption would have dropped approximately with a factor of 1/8. As measurement of the chip was not possible with this operating voltage, the associated power consumption values are not used here.

ns and thus the power consumption is 22 μW (VDD=2.5V, CIF@30fps. By adding the power consumption of the gray scale part, the total required power is below 2.3 mW without the low-pass operation and below 6.3 mW with the low-pass operation.

5.5 Gradient-Based Motion Estimation Realization

A gradient-based digital ME realization was also designed to work in conjunction with the partition methods presented in this chapter. The realization computes the ME result on the variable block sizes indicated by the algorithm. To achieve low power, MV prediction and search-area subsampling were used to aggressively reduce the clock frequency of the implementation.

5.5.1 Hardware-Based ME Design Aspects

5.5.1.1 Search Pattern Analysis

With search-area subsampling algorithms, the size of the local memory shown in Fig. 2.7 limits the size of the search pattern that can be used. Although with most common search patterns, which are shown in Fig. 5.5 [141], [92], some or most of the data in the local memory can be reused. For this the movement of the pattern requires that either: 1) The data that is not stored in the local memory and is required by the pattern is fetched from the frame memory or 2) all possible pattern movement locations are fetched into the local memory. In the first case, the time to fetch the data must be taken into account in the worst-case delay calculations. The second case leads to unnecessary data fetches and thus higher power consumption. After reviewing popular search patterns the range [-2,2] was selected as a good trade-off, so a 8x8 parallel memory bandwidth for the 4x4 reference block must be implemented. Such a memory bandwidth allows the computation of a search pattern where the farthest points are a maximum distance of [-2,2] from the search center.

The matching criterion computation block in Fig. 2.7 usually is composed of several single pixel or block-matching criterion computation engines that are pipelined. The number of pipelines limits the number of search points that can be computed in one clock cycle. Of these patterns, shown in Fig. 5.5, the small diamond has 5 computation points, whereas the others have 9. If the small diamond would be combined with any of these large patterns, the large pattern would dictate the number of pipelines. The small diamond would use only five of these nine pipelines, thus leading to inefficient pipeline use. Therefore, the small square was chosen as the small pattern. As can seen in Fig. 5.5, the small and large square patterns have only one common search point, but the small square and large diamond would have three common points. Thus, the large square was chosen as the other pattern. The largest pattern

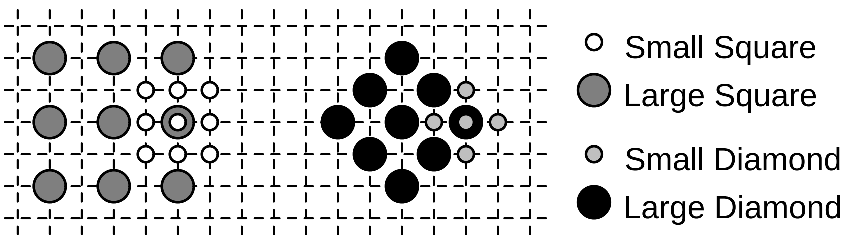


Figure 5.5 The referred search patterns.

in [141] was not considered because the number of computation points was too large for the selected number of pipelines.

5.5.1.2 Predictor Calculation

Although H.264 allows reference frame distances of up to 15 frames, the memory size limitations of mobile applications restrict the use of only the previous frame as the reference frame. Of the motion vector predictors presented in [141] and [92], the following were found to be suitable for single reference frame hardware implementation:

- Median. The median of the vectors MV1 through MV4 in Fig. 2.9
- Zero. The vector of co-located position in the reference frame
- Spatial. The four vectors MV1 through MV4 in Fig. 2.9
- 4 Adjacent temporal. The four vectors of positions ±1x, ±1y in the reference frame
- Previous best. The chosen vector from the predictor set of the co-located position in the reference frame

As this implementation is able to calculate all points in a search pattern in the same number of clock cycles as it calculates only one point (Chap. 5.5.2.2), it is possible to refine each prediction vector using a search pattern. All vectors are refined with the small square pattern. Also, the predictor-based pattern selection scheme (Eq. 2.13) from [92] is implemented in this algorithm, which decides the start search pattern based on the city-block length of MV1 through MV4 in Fig. 2.9.

5.5.1.3 Hop Counter

Instead of the search area size restriction of conventional ME, here the search area is limited by the number of hops (i.e. the number of times the pattern moves) the pattern can make. One advantage of this is that, with a conventional search range restriction, the worst-case number of hops is close to r^2, where the search range is [-(r-1),r][3], complicating the worst-case hardware design. Another advantage is that the number of hops is limited within a frame and not within the current search. This is best illustrated with the following example: In a search where the number of hops is limited to ten, the first block of the search takes two hops and the second three. The third search can now take (10-2)+(10-3)+10 =25 hops, making the effective search range ± 50. The saved hops cannot be transferred to to the following frame, due to real-time constraints. This kind of search range adaptivity has not been presented previously. Due to the center-based distribution of motion vectors [87], this method is most advantageous in occlusion and appearance cases.

5.5.2 Implementation Solutions

5.5.2.1 Processing Element

There are nine separate processing element (PE) matrices that compute the SAD values. Each matrix is composed of 4x4 processor elements. One processor element computes $|a-b|$. To simplify the implementation, the actual PE is implemented as

$$\max(a,b) + NOT(\min(a,b)) + 1 \tag{5.3}$$

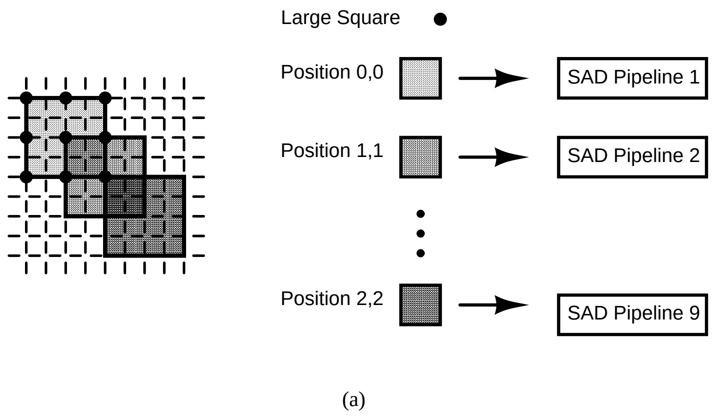
where a,b are in 2's complement format.

5.5.2.2 Pipeline Architecture

The pipeline architecture was chosen so that one pattern could be computed in one clock cycle. The variable block size does not have a significant effect on the distortion computation, as all the distortion values of the larger sizes can be composed of the 4x4 sub-block distortion values. As there are 9 points in the large pattern, 9 pipelines were needed. One search pattern for a 4x4 block is calculated in 4 clock cycles. For larger blocks, only 1 cycle is added for each 4x4 block the larger block contains, because they can subsequently be calculated in the pipelines during a clock cycle. Therefore an 8x4 block takes 5 clock cycles. Search patterns cannot be calculated each cycle because the results of the previous search pattern must be ready before it is known in what direction the pattern must be moved.

The pixel data is spread from the 8x8 memory output to 9 pipelines based on the shape of the search pattern. As the small pattern needs only a 6x6 memory bandwidth, the usage percentage of the memory is more inefficient than with the large pattern. The advantage of the architecture is that the separate blocks within the 8x8 area are partly

[3]With natural images the chance of the pattern moving through the whole search area is highly unlikely. With increased entropy cases, such as noisy images, the pattern may still traverse a long distance.



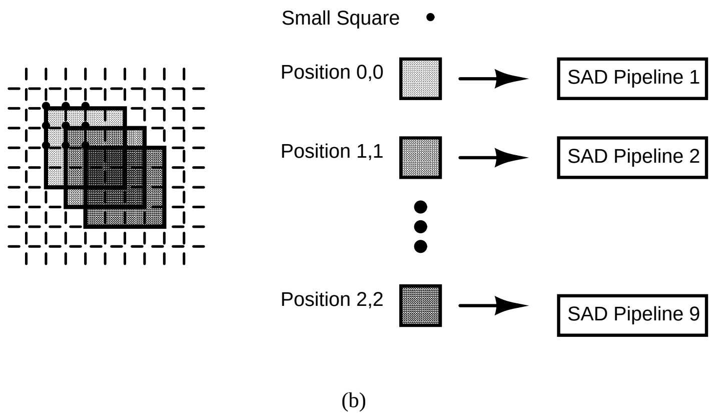


Figure 5.6 Pipeline data spreading example. a) Large square pattern. b) Small square pattern. For simplicity, only positions [-2,-2], [0,0], [2,2] and [-1,-1], [0,0], [1,1] shown in a) and b), respectively.

collocated. All of the collocated pixels are reused, and thus fetched from memory only once. This is illustrated in Fig. 5.6, for simplicity, only three positions are shown. The SAD values for the larger blocks are summed up after the pipeline stage.

5.5.2.3 Memory Architecture

The architecture of the local memory, shown in Fig. 2.7, was also considered, although the memory was not implemented on chip. The memory is split into 64 memory blocks and each memory block has a 1-pixel bandwidth. The pixels are spread with modulo 8 into the memory blocks in such a manner that pixels 1,8,16,... are in the first block and pixels 2,9,17,... are in the second block and so on. This spreading is performed in

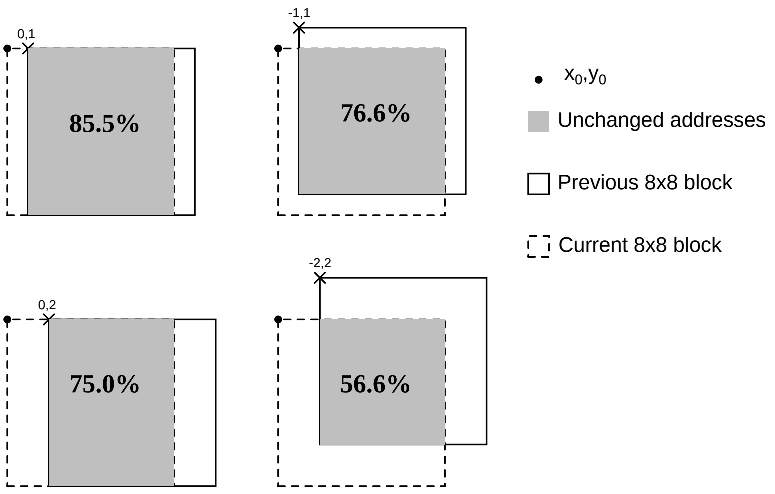


Figure 5.7 As the search pattern moves, many of the memory addresses remain unchanged. Position (0,0) is the start of the current search.

2 dimensions. Thus, first memory block holds the pixels of picture coordinates

$$\begin{array}{c}(0,0),(0,8),\ldots,(0,56)\\ \vdots \\ (56,0),(56,8)\ldots,(56,56)\end{array} \tag{5.4}$$

and the final memory block holds

$$\begin{array}{c}(7,7),(7,15),\ldots,(7,63)\\ \vdots \\ (63,7),(63,7)\ldots,(63,63)\end{array}. \tag{5.5}$$

Each of these memory blocks has a unique address bus. This enables the fetching of each pixel separately. Only four different addresses are needed for the operation, however. When the search pattern moves, because it can only move a maximum of two pixels, many of the memory addresses remain unchanged, as is shown in Fig. 5.7. Because the address changes were minimized, the parallel memory architecture does not output the data in linear form. Therefore, after being fetched from the memory the 8x8 data array must be 2-dimensionally rotated before it is ready for use.

5.5.3 Simulation Results

The algorithm was simulated in Baseline H.264 with Maxhops=10 (SA ± 20) in H.264 against FS (SA $= \pm 32$). QP$_{H.264}$ was set at 10, 20, 28, 30, and 38. The rest of the test

	PSNR	Bit-Rate (%)
Foreman 1-100	0.420	3.61
Foreman 151-250	0.233	3.25
News 1-100	0.034	0.27

Table 5.3 Collected average rate-distortion results.

environment is described in Chap. 4.1.2. The used partition algorithm is described in Chap. 5.2.2.

The rate-distortion graphs for Foreman 1-100, 151-200 and News 1-100 are shown in Fig. 5.13 a), b), and c), respectively. The average results are also collected in Table5.3. For News, the presented algorithm can be thought of as having comparable performance. For Foreman, there is a slight drop in R/D performance.

5.5.4 Realizations

The target speed of the application was chosen as CIF@30fps. The chip was designed in 0.18μm CMOS and realized, due to financial restrictions, with a Altera Cyclone EP1C20 / Stratix S80 measurement board / FPGA chip combination. The measurements were conducted with the FPGA only. These measurements

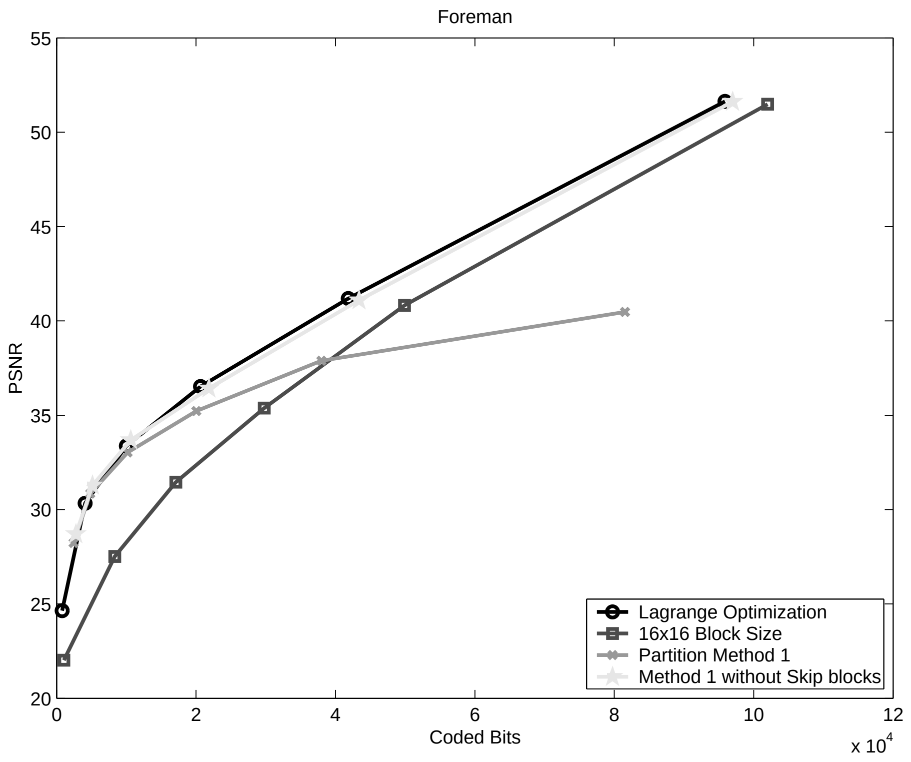
5.5.4.1 Integrated Chip

The worst case performance for a CIF image with 396 macro-blocks, containing 16 4x4 blocks, and with 10 hops, is $4*10*16*396 = 253440$ clock cycles per frame. For real-time at 30 frames per second, 7.61MHz is the required clock rate. For a 30 fps VGA-sized image with 1200 macro-blocks, the respective figure is 23.04MHz. The number of hops can also be used to decrease the required clock rate in, for example, low-battery situations.

The size of the layout is 2.8 μm^2, the layout is routing restricted. The routing restriction is mainly due to the rotation of the data. The layout is shown in Appendix A.

5.5.4.2 FPGA

The high memory bandwidth of the integrated chip could not be implemented on the 64 memory blocks of the Altera EP1C20. Thus, only the distortion measure computation could be measured. The measurements verified the simulation results of Chap. 5.5.3. The maximum measured clock frequency was 20.66 MHz.



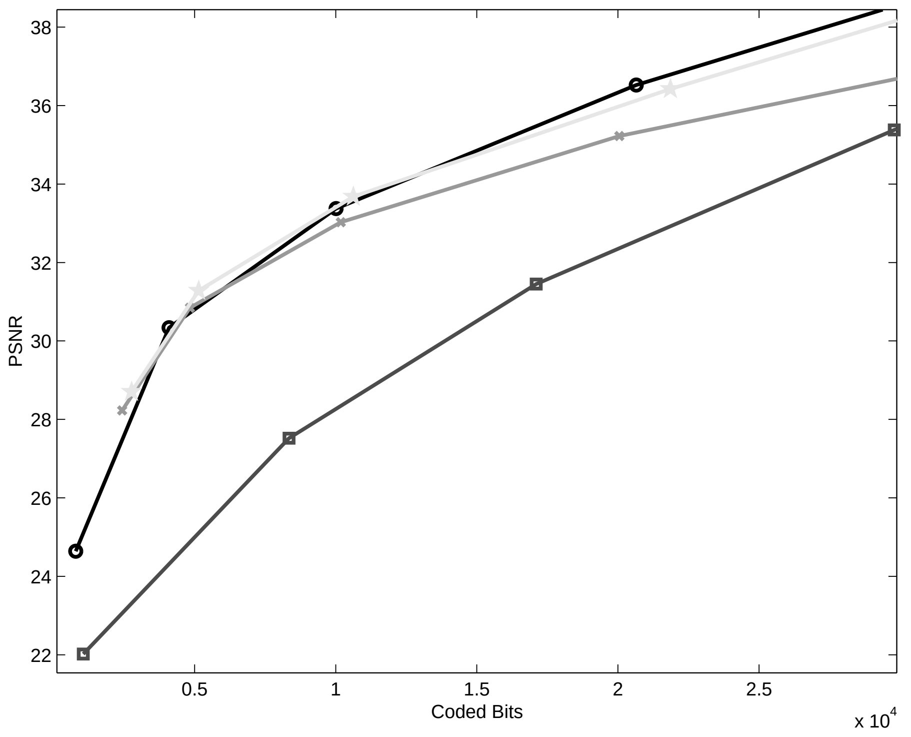


Figure 5.8 Partition method 1 Foreman rate-distortion graphs (top graph) and close-ups of low bit-rate values (bottom graph) with varying values of $QP_{H.264}$ (The subjective quality of the video sequence differs from the quality presented by the PSNR values, Chap. 4.4.1).

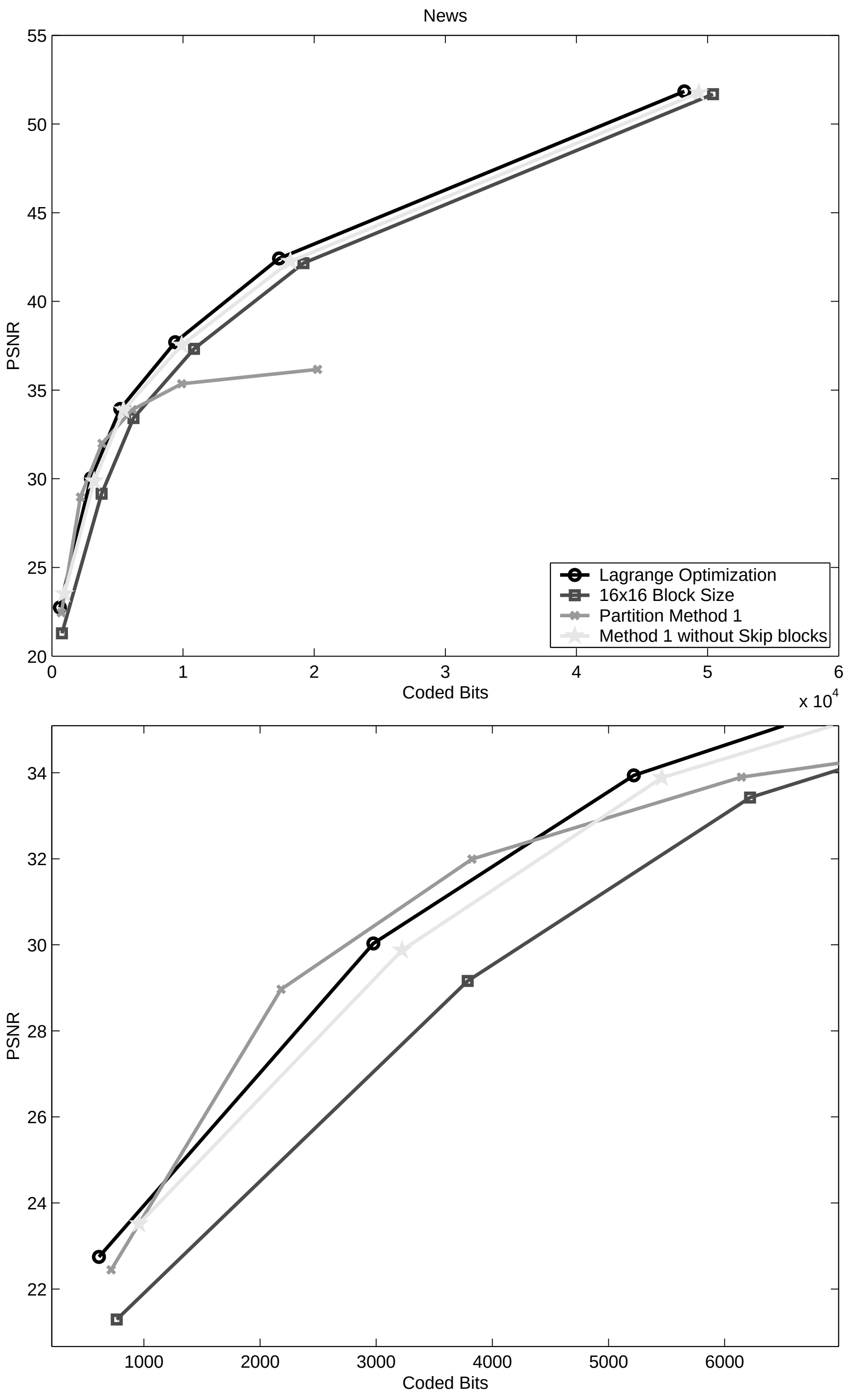


Figure 5.9 Partition method 1 News rate-distortion graphs (top graph) and close-ups of low bit-rate values (bottom graph) with varying values of $QP_{H.264}$ (The subjective quality of the video sequence differs from the quality presented by the PSNR values, Chap. 4.4.1).

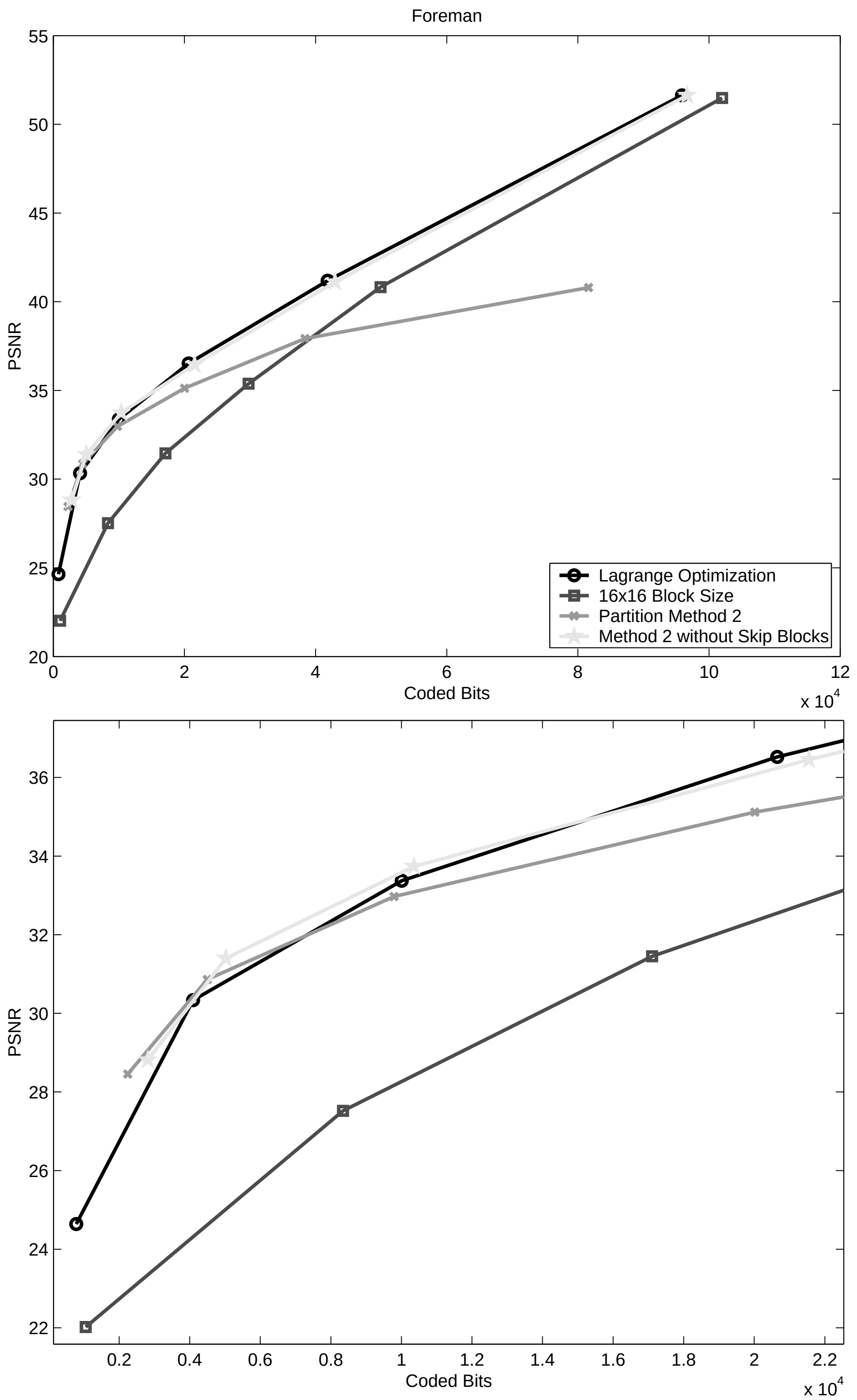


Figure 5.10 Partition method 2 Foreman rate-distortion graphs (top graph) and close-ups of low bit-rate values (bottom graph) with varying values of $QP_{H.264}$ (The subjective quality of the video sequence differs from the quality presented by the PSNR values, Chap. 4.4.1).

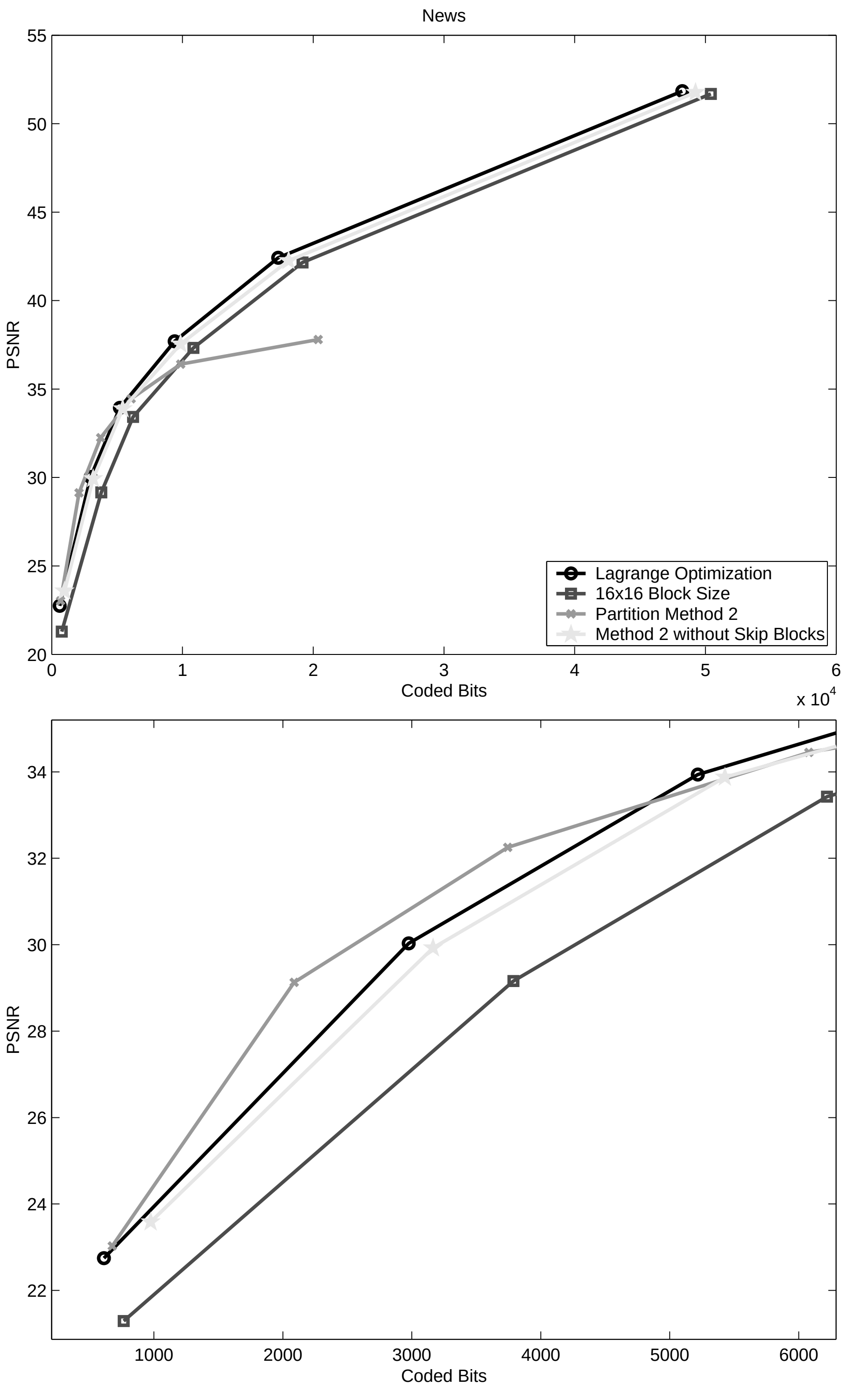
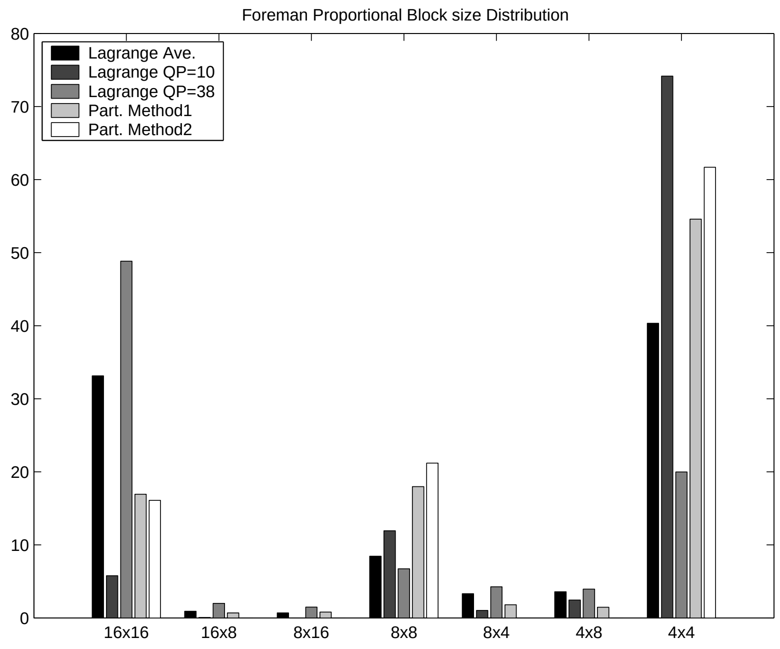
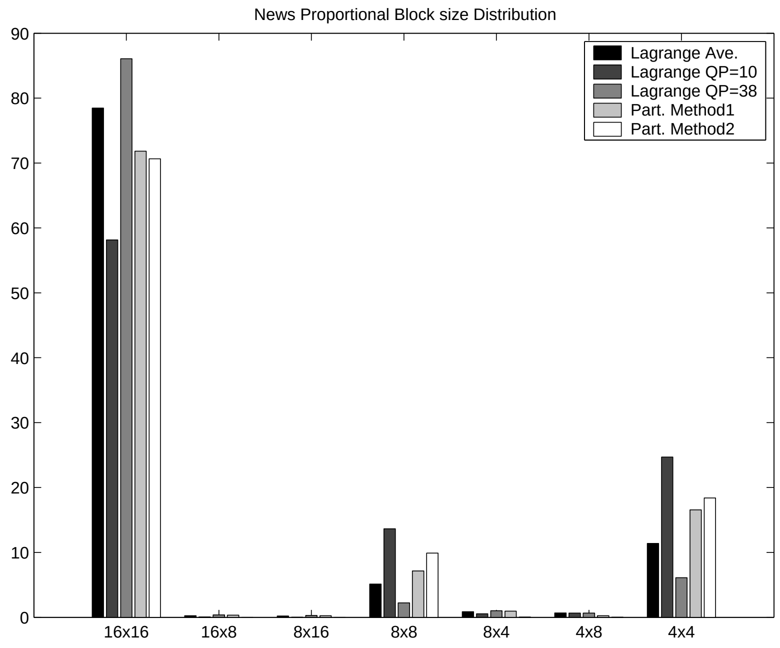


Figure 5.11 Partition method 2 News rate-distortion graphs (top graph) and close-ups of low bit-rate values (bottom graph) with varying values of $QP_{H.264}$ (The subjective quality of the video sequence differs from the quality presented by the PSNR values, Chap. 4.4.1).

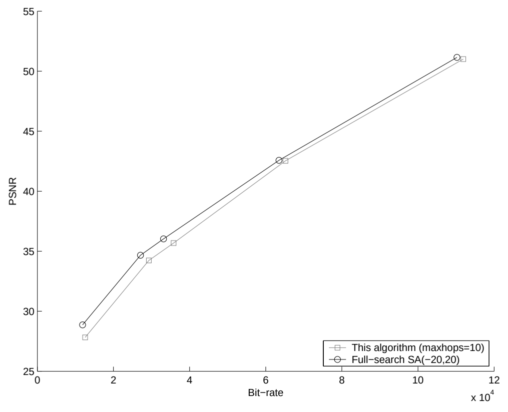


(a)

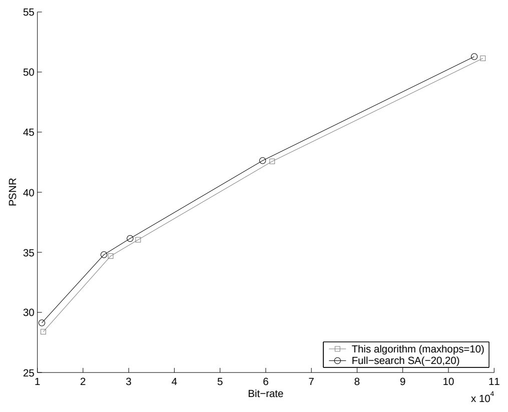


(b)

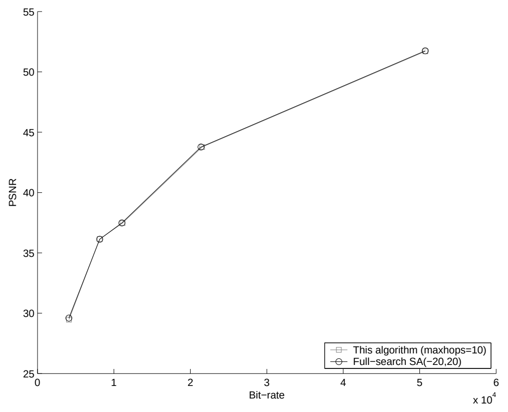
Figure 5.12 Proportional distribution of block sizes. a) Foreman. b) News



(a)



(b)



(c)

Figure 5.13 Rate-distortion graphs. a) Foreman 1-100. b) Foreman 151-250. c) News 1-100.

Chapter 6

CNN Border Smoothing Templates for MPEG-4 Shape Bit-Rate Reduction

6.1 Application Target

The shape of a Video Object (VO) in MPEG-4 (Chap. 2.4.1) [50] is indicated by the binary alpha plane. Depending on the segmentation algorithm an exact binary alpha plane may have irregular borders. In object-based MPEG-4, these borders, indicated by border Binary Alpha Blocks (BAB), are coded with the CAE algorithm (Chap. 2.2.1.4) [68]. If the border-BABs are irregular, the efficiency of the CAE algorithm is reduced. The reduction in efficiency is due to that when the borders are not smooth; the prediction of the current shape pixel is inefficient. Inefficient shape coding correspondingly raises the VO's shape bit rate.

6.2 Smoothing Templates

To smooth the borders, the three following templates [1] were developed

$$A_1 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}, B_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 0 \end{bmatrix}, z_1 = -6.5 \tag{6.1}$$

[1] In video coding terminology, white pixels belong to an object, while, in CNN terminology black pixels are treated as belonging to the object.

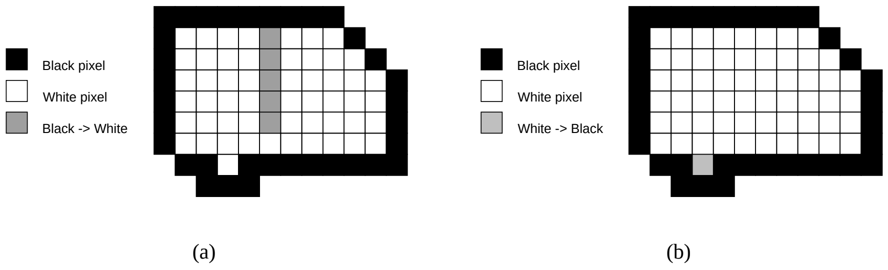


Figure 6.1 a) Effect of template 6.1 b) Combined effect of templates 6.2 and 6.3

$$A_2 = 0,\, B_2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, z_2 = -0.5 \tag{6.2}$$

$$A_3 = 0,\, B_3 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 3 & 1 \\ 0 & 1 & 0 \end{bmatrix}, z_3 = -4.5 \tag{6.3}$$

The templates are designed for the positive-range high-gain CNN model which is introduced in [28].

The first template (Eq. 6.1) changes a black pixel to white if seven of the eight surrounding pixels are white. This template has the effect of filling up black one-pixel-wide paths. The effect of this template is illustrated in Fig. 6.1a.

The second template (Eq. 6.2) removes eight-connectivity. The third (Eq. 6.3) changes a white pixel to black if three of its surrounding pixels in the horizontal or vertical direction are white. The first and third templates combine to remove single white pixels from otherwise straight borders. The combined effect of this template is illustrated in Fig. 6.1b. The actual implementation order of the templates is [2,1,3].

As the CNN implementation considered here [114] is able to process in the range of 10^3 fps, the addition of three templates results in a negligible increase in overall computational complexity.

6.3 Simulation Results

To validate the effect of the templates, the results from the *Remarkable* part of the segmentation algorithm of Chap. 3.3 [29] were used instead of the final segmentation results. As can be seen in the left-hand frames of Fig. 6.2 and Fig. 6.3, the *Remarkable* borders are much more jagged than what the final segmentation borders would be. Thus, when adding the smoothing templates, the effect on the results will be more

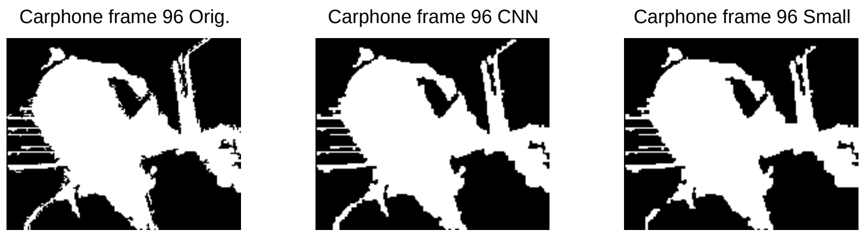


Figure 6.2 *Remarkable* Chap. 3.3 [29] results from the sequence Carphone. Left frame: *Remarkable* result. Middle frame: Effect of templates 6.1, 6.2, and 6.3. Right frame: Effect of small block removal.

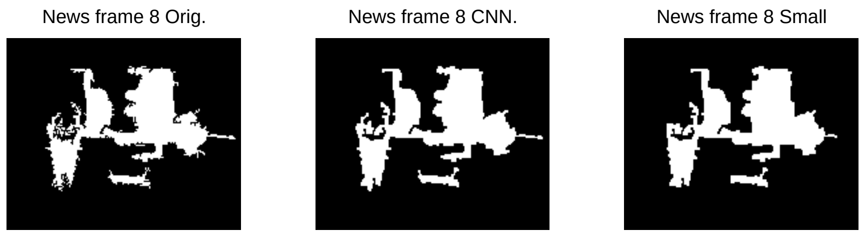
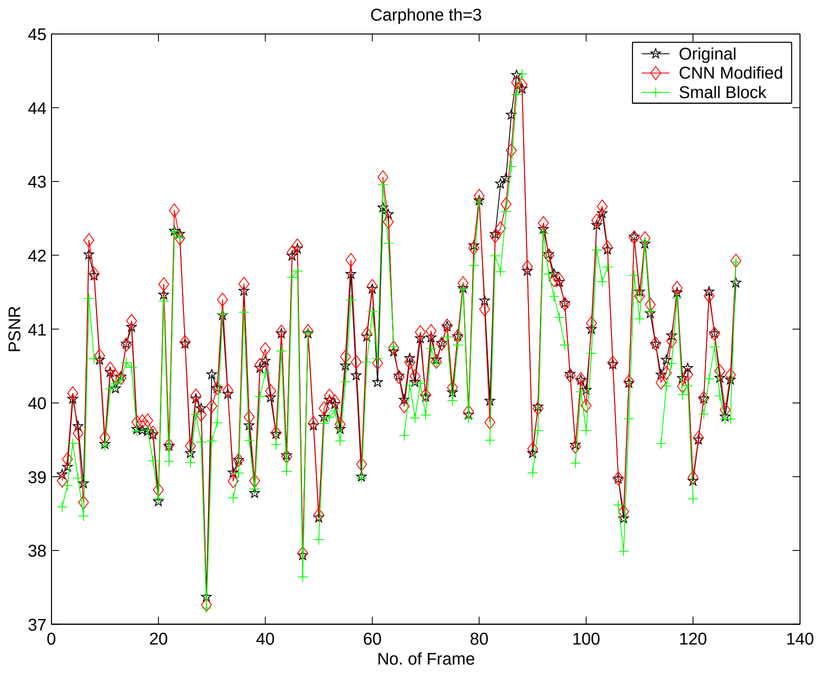


Figure 6.3 *Remarkable* Chap. 3.3 [29] results from the sequence News. Left frame: *Remarkable* result. Middle frame: Effect of templates 6.1, 6.2, and 6.3. Right frame: Effect of small block removal.

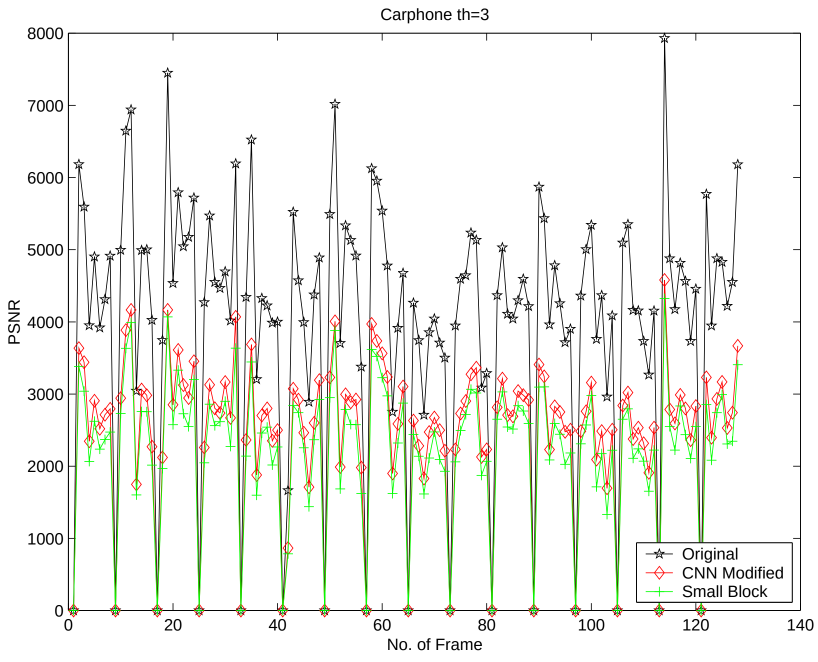
pronounced. The effect of the templates on the *Remarkable* results is shown in the middle frames of Fig. 6.2 and Fig. 6.3.

In [142], it is shown that the motion of boundary blocks correlates with the opaque blocks near the boundary block. This correlation is dependent on the used segmentation. If the correlation exists, the motion vectors for the boundary blocks can be used as starting points for the nearby opaque blocks. Irregularly shaped borders also lead to boundary MBs that contain only a few pixels that belong to the VOP. The motion vectors associated with these boundary MBs tend to be very large and the correlation of adjacent motion vectors can be small. As the motion vectors are coded differentially, large uncorrelated motion vectors are coded with a higher number of bits. A study of boundary block motion vectors revealed that boundary blocks that have only a small number of opaque pixels tend to have large motion vectors. As such blocks do not have a large effect on the picture quality blocks with 25 or less opaque pixels were transformed entirely to transparent blocks. The small block changes are shown in the rightmost frames of Fig. 6.2 and Fig. 6.3.

The segmentation algorithm was integrated into an object-based Core profile MPEG-4 encoder in Matlab software. The rest of the test environment is described in Chap. 4.1.2. The PSNR values for the sequences Carphone and News are shown in Fig. 6.4a and Fig. 6.5a, respectively. The *Remarkable* shapes with *th*=3 were used instead of the final segmentation result. The shape bit-rate after intra-CAE for the same sequences is shown in Fig. 6.4b and Fig. 6.5b, respectively. From the figures it can be seen that the implemented templates decrease the shape bit-rate of an MPEG-4 encoder without appreciable reduction in coded video quality.

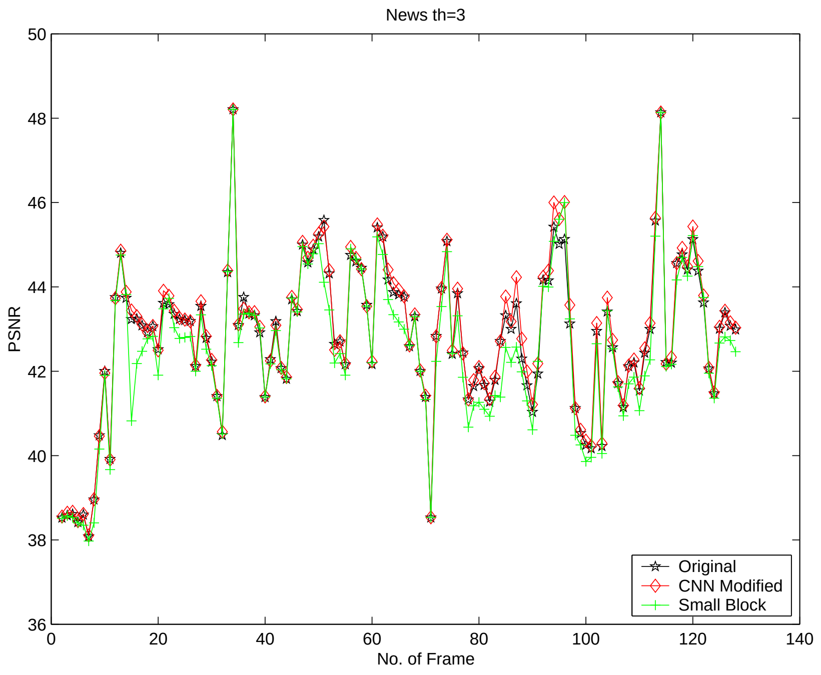


(a)

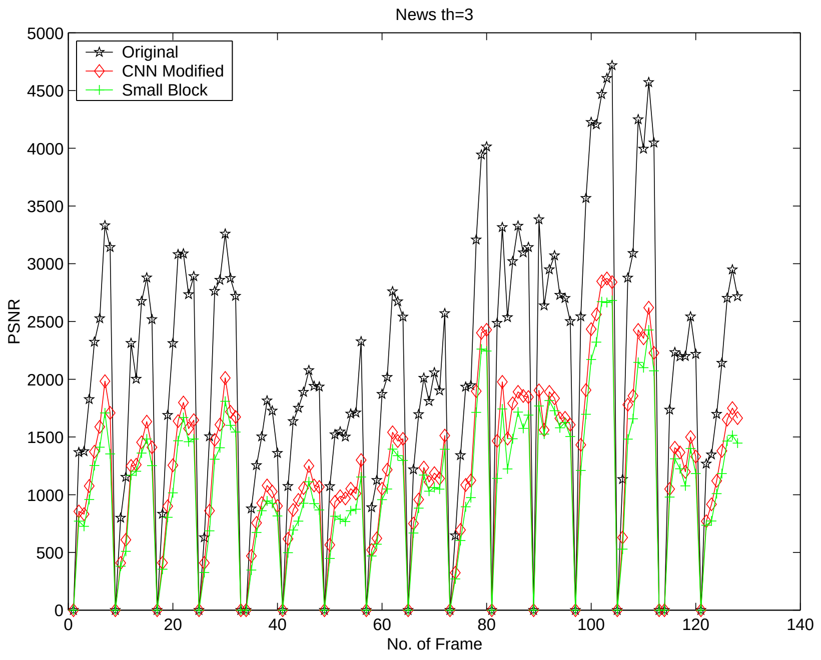


(b)

Figure 6.4 PSNR (a) and shape bit-rate (b) of Carphone frames 1-128



(a)



(b)

Figure 6.5 PSNR (a) and shape bit-rate (b) of News frames 1-128

Chapter 7

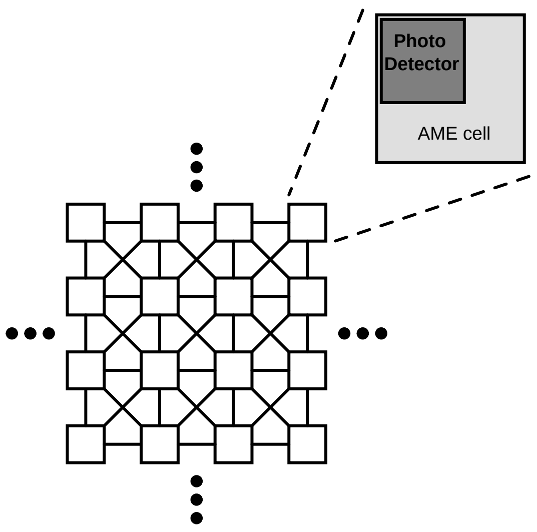
Analog Block-Based Motion Estimation

7.1 Application Target

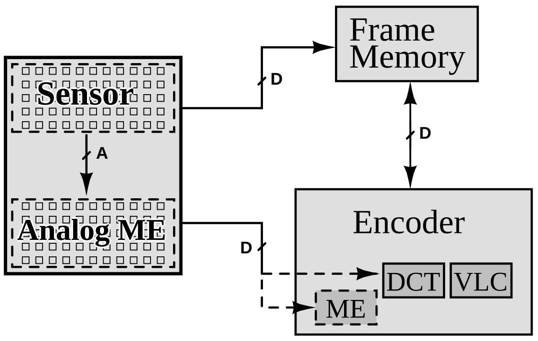
Chapter 1 states that the power consumption problem of video encoders of mobile terminals will not be solved in the near future with conventional digital hardware solutions. Motion estimation can take up to 80% of the power consumption of a video encoder [58]. For small frame sizes, such as QCIF, the power consumption of ME can be decreased sufficiently with algorithmic solutions, which are revived in Chap. 2.2.4.3 and Chap. 2.2.4.4. However, for larger frame sizes, fundamentally different hardware solutions are needed.

Due to the low-power possibilities of analog CNN-type parallel processor solutions, they can be used as building blocks for a low-power ME realization. Such an analog ME (AME) realization is presented in this chapter. Although the simulations of this chapter are performed with an MPEG-4 codec, the AME can be utilized with all codecs that incorporate 8x8 block-size ME.

From a purely implementation aspect, the most beneficial solution would be to implement the AME array in connection with the image sensor array, as is done in [143] and [144], and shown in Fig. 7.1a. However, in some cases, the computational part in the pixel processor may raise the pixel cell size to the extent that the image sensor quality is degraded. Also, with large image sensor sizes, even if a single pixel processor is feasible, the realization of the full sensor array may not be economically feasible. In such cases, the AME will have to be implemented as a separate array. The separate array can then possibly be implemented in a more compact form. For example, one possibility would be for one dimension of the array to be the same as



(a)



(b)

Figure 7.1 Principle of suggested analog motion estimation. a) AME in connection with the image sensor. b) Separate AME core **A** / **D** = analog / digital data transfer.

the image sensor dimension and the other dimension would be specified by the search range (i.e. a 352x32 or 288x32 array for CIF-sized sensor and a [-15,16] search range). This kind of an implementation would require more memory, as each cell would have to store more than one pixel of reference data.

For a separate AME array to be feasible the data transfer between the computational array and the image sensor has to be analog. Otherwise, an excess D/A conversion would be needed; this would negate much of the low-power advantage. This restricts the implementation possibilities of the AME to such that enable efficient analog data transfer. This principle is depicted in Fig. 7.1b. The analog data transfer requirement impedes a separate chip realization for the AME. The analog data transfer can efficiently be attained when, for example, the AME array is integrated on-chip with the image sensor. Otherwise, the AME and sensor can be packaged together [145], as is the case in System-on-Chip (SOC), Multichip Module (MCM), System-in-Package (SIP), and System-on-Package (SOP) realizations.

The disadvantages of AME arise from the inaccuracies inherent in all analog computation. The inaccuracies are mainly due to device mismatch. In the case of ME, the inaccuracies manifest themselves in two ways: 1) Inaccuracy in the computation of the matching criterion. 2) Inaccuracy in the transmission and storage of the reference frame data. Both cases may lead to an inferior ME result, thus deteriorating the rate-distortion.

As is reviewed in chapters 1.1.2.1 and 1.1.2.2, the use of inter-connected analog parallel processors, such as CNN, for block-based ME is novel to this thesis.

7.2 Analog Block-Based Motion Estimation

Motion estimation consists of three basic operations (Eq. 2.12):

1. Shift of reference pixel (reference block) data $ref(i_2, j_2) = ref(i+dx, j+dy)$.
2. The operation $|ref(i_2, j_2) - cur(i,j)|$ (SAD) or $(ref(i_2, j_2) - cur(i,j))^2$ (SSD).
3. Averaging the data over the current block $\sum_{i=x0,j=y0}^{x0+M-1,y0+N-1}$.

With CNN or CNN-type circuits a single shift of operation 1) can be achieved with the B-template

$$B_{sh} = \begin{bmatrix} b_{-1,-1} & b_{-1,0} & b_{-1,1} \\ b_{0,-1} & 0 & b_{0,1} \\ b_{1,-1} & b_{1,0} & b_{1,1} \end{bmatrix} \tag{7.1}$$

where $b_{i=m,j=n}$=1, $b_{i\neq m,j\neq n}$=0, and (m,n) is the direction of the current shift. For operation 2) dedicated circuits are needed in the cell logic. For a 1-neighborhood operation 3) is achieved with the template

$$B_{ave} = \frac{1}{9} \cdot \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}. \tag{7.2}$$

All previous 2-dimensional CNN or CNN-type implementations use either 4-connected or 8-connected 1-neighborhoods, which are shown in Fig. 3.1 and Fig. 3.2. To implement AME with existing arrays, the reference data would have to be shifted through as many cells as the length of the current search. For example, a search of 0,7 would be shifted through seven cells, of which each injects its own error into the shifted data, thus producing accumulating inaccuracy. This accumulating inaccuracy also applies to the averaging of the data. Another option would be to increase the size of the neighborhood. Even the averaging of an 8X8 macroblock with B-templates would require a 4-neighborhood. The number of connections per cell needed for a 4-neighborhood is 80, which makes a VLSI realization extremely inefficient. For the data shift, the neighborhood would have to be as large as the search range. With even a small search range of [-7,8], a silicon implementation would be impossible. Another option would be to implement the data shifts and averaging with A-templates. A-template implementations also suffer from accumulating analog inaccuracies and, as with IIR filters, the effect boundary of the averaging cannot be defined exactly. To calculate the irregular block sizes of H.264 with CNN architectures would require irregular connections and thereby space-variant templates.

7.2.1 Inaccuracies in Analog Motion Estimation

Whereas digital operations can be performed with sufficient precision, analog operations always include an error source due to various sources.

7.2.1.1 Device Mismatch

The manufacturing variations of CMOS processes cause process and device variations that manifest themselves from lots down to devices. These variations can be classified either as systematic or random. Systematic variations such as wafer-to-wafer variations or processing gradients can be minimized with proper circuit topologies, biasing and layout techniques. Device-to-device variations, also called “device mismatch”, are random factors that depend on the size of the device. The dominant sources [146] of device mismatch for a pair of closely spaced MOS transistors are the threshold voltage V_T and transconductance parameter β. The variances σ^2 of these parameters can be independently modeled as

$$\sigma^2(\Delta V_T) = \frac{A_{V_T}^2}{W \cdot L}, \tag{7.3}$$

$$\left(\frac{\sigma(\Delta\beta)}{\beta}\right)^2 = \frac{A_\beta^2}{W \cdot L}, \tag{7.4}$$

where $\beta = \mu C_{ox}\frac{W}{L}$, μ is the carrier mobility, C_{ox} the oxide capacitance, W and L the gate width and length of the MOS transistor, respectively, and A_{V_T} and A_β the technology dependent proportionality constants. For the sake of compactness higher-order effects are not shown in Eq. 7.3 and Eq. 7.4. From Eq. 7.3 and Eq. 7.4, it can be seen that the mismatch is in proportion to the size of the transistors making the minimizing of the error in AME a trade-off between cell size and the error magnitude.

7.2.1.2 Charge Injection

Another source of error in the AME realization is the charge injection of the MOS switches, which are caused by the release of the channel charge when the switch closes. Charge injection effects on switched-current (SI) circuits [147] are thoroughly studied in [148], but the analysis also applies here. In Fig. 7.2, with the operating voltage V$_{dd}$ as the switch-on voltage, the injected charge is

$$Q_{inj} = \alpha \cdot \left(V_{dd} - \left(1 + \frac{\gamma}{3}\right) - V_t\right) \cdot C_{g,sw} \tag{7.5}$$

where γ is the body factor, C$_{g,sw}$ the gate capacitance of M$_{sw}$, and typically $0.5 < |\alpha| < 1$. α depends on switching dynamics and the parasitic drain-source capacitances of M$_{sw}$. The voltage error induced by the injected charge is

$$\Delta V_{g,M} = \frac{Q_{inj}}{C_M + C_{g,M1} + C_{g,M2}}. \tag{7.6}$$

The drain current of M$_2$will change by $\Delta i = g_{m,M2} \cdot \Delta V_{g,M}$ where $g_{m,M2}$ is the transconductance of M$_2$.

7.2.1.3 Clock Feedthrough

Further error in the drain current of M$_2$ is caused by the clock feedthrough from the gate of M$_{sw}$ to the gate of M$_M$. The clock feedthrough is induced by the drain-gate overlap capacitance C_{dg} of M$_{sw}$ shown in Fig. 7.2. The error current is

$$\Delta i = g_{m,M2} \cdot \Delta V_{g,M} = g_{m,M2} \cdot \frac{C_{dg_{sw}}}{C_{dg,sw} + C_M + C_{g,M1} + C_{g,M2}} \cdot V_{dd}. \tag{7.7}$$

The charge injection and clock feedthrough can be canceled with the use of a half-sized dummy switch, shown in Fig. 7.3 [149]. As, in a practical realization, the transistors M$_1$ and M$_2$ are significantly larger than M$_{sw}$, the addition of the dummy transistor has a negligible impact on the cell size. The cancellation is not perfect due to the device mismatch effects and, in the case of clock feedthrough, due to the different sizes of drain-gate overlap capacitance C_{dg} of the switch and the dummy switch.

In AME, the error arising from both the charge injection and clock feedthrough phenomena is of no impact when the error is common to all the cells in the AME

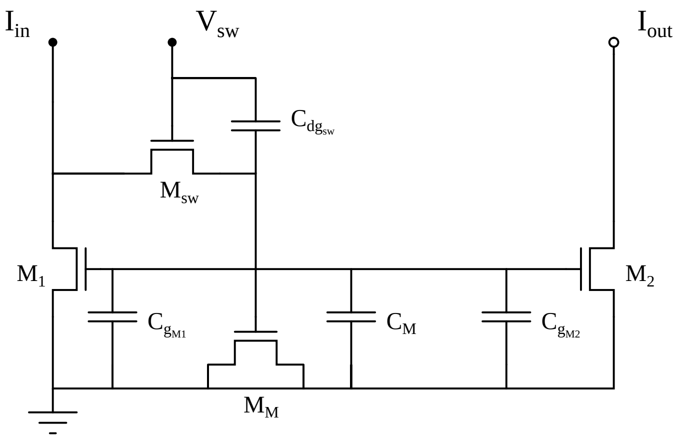


Figure 7.2 Simplified B-template cell with parasitic capacitances.

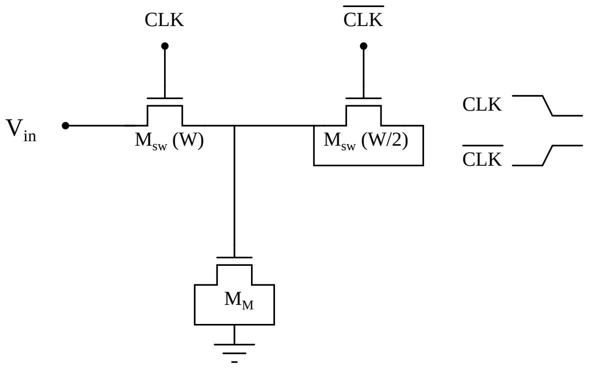


Figure 7.3 Charge injection canceling with the use of a half-sized dummy switch [149].

array. In such cases, the same amount of error is added to each of the blocks under comparison (e.g. gain is added to Eq. 2.12), which does not affect the comparison result. Thus, only the random component of charge injection and clock feedthrough caused by device mismatch is of concern in AME.

7.2.1.4 Leakage Currents

The capacitor C_M of Fig. 7.2 is slowly discharged through the leakage current of the switch transistor M_{sw}. This leakage current is due to the reverse-biased p-n junction of the switch. For deep-submicron technologies, the gate currents M1-M10 of Fig. 3.4 also can contribute to the leakage. However, the maximum storage requirement is well above the times needed in video applications [148].

7.2.1.5 Effect of the Inaccuracies on the AME Result

The effect of the inaccuracies on the ME result is very similar to the effect of the unideal matching criterion of Chap. 2.2.4.9. The effect of the total error in AME is non-linear due to: 1) Any amount of error in the ME has no effect if the block with the minimum matching criterion (the correct MV) is still chosen as the outcome of the search. 2) Even if an MV pointing to a larger matching criterion block is chosen, the difference in R/D does not depend linearly on the AME error. This is due to the fact that the minimum block and the chosen block can be very similar in terms of PSNR and bit-rate. Also, as stated in Chap. 2.2.4.9, the optimal matching criterion is input-data-dependent which means that, even with induced error, the ME result can be better in terms of R/D than the error-free ME result. Naturally, the probability that this will happen decreases with a larger error.

7.3 Analog Motion Estimation Realization

7.3.1 Novel 3-Neighborhood

To achieve an efficient shift and averaging operations (operations 1. and 3. of the list on page 91), the novel 3-neighborhood shown in Fig. 7.4 was developed. In Fig. 7.4, the 3rd neighborhood connections are shown with the thick lines, and the 1st neighborhood connections with the thin lines. The connections are shown only for the center cell. The 3-neighborhood represents an efficient trade-off between size and the number of connections in each cell. One advantage of the 3-neighborhood is that it can be added to conventional CNN or CNN-type solutions. Without adhering to this condition, more complex connections may have been designed.

The novel 3rd neighborhood reduces the number of shifts needed. The shifts are achieved with the template of Eq. 7.1 for both the first and third neighborhoods. For example, a search area of [-7,7] can be achieved with only 3 shifts. This is illustrated in Fig. 7.5 with an example of the individual shifts needed for a -7,7 shift. The shift lengths 2 and 4 have the choice of using either two 3N shifts and one 1N shift or vice versa. It is advantageous to use either one, as the transistors of the B-template implementation can then be optimized.

With the 3-neighborhood, the averaging operation can be achieved by implementing the template of Eq. 7.2 two times. First the template of Eq. 7.2 is implemented in the first neighborhood thereby averaging the data around the black cells of Fig. 7.4. Then the same operation is performed in the 3rd neighborhood. The average is then read from the center cell, which is shown in white in Fig. 7.4. The disadvantage of the averaging operation is that the effect area is a 9x9 block, as opposed to the 8x8 block

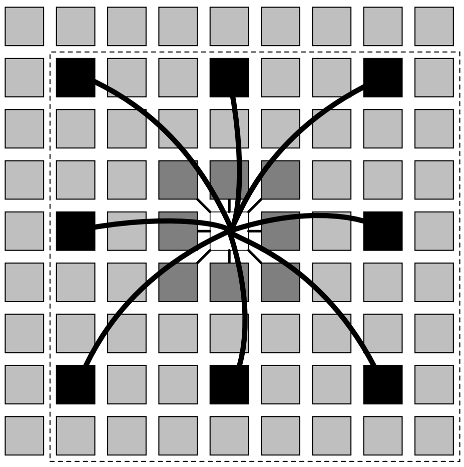


Figure 7.4 The 3-neighborhood for analog matching criterion calculation. The connections are shown only for the center cell. A normal 8X8 macroblock is also outlined.

size used in MPEG-4 and H.264 This is illustrated with the dashed line in Fig. 7.4. The basic 16X16 MVs defined by multiple video coding standards would require a 4-neighborhood, corresponding to the 3-neighborhood presented here, increasing the number of connections, so that a silicon implementation would be inefficient. Also, using only 8x8 MVs can be inefficient in some cases, for example, in homogeneous areas. However, the 16x16 MVs can be deduced from the 8x8 MVs with either Lagrange optimization or a simpler formula; for example, if all the 8x8 MVs are close to

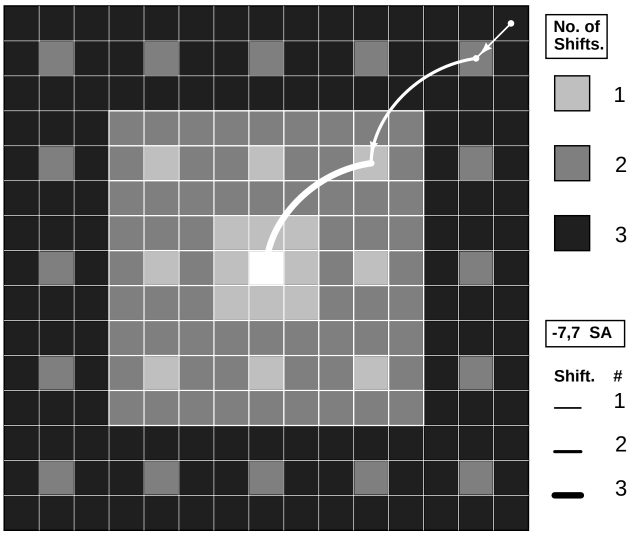


Figure 7.5 Number of shifts needed for a given search area. Example of separate shifts needed for a -7,7 shift also shown.

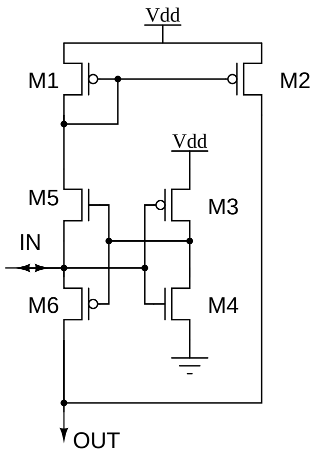


Figure 7.6 Absolute value circuit [150].

each other, an averaged 16x16 MV can be used.

7.3.2 Matching Criterion Circuits

For the second operation of the list on page 91, dedicated circuits are needed for the processor logic.

7.3.2.1 Absolute Value circuit

For a SAD calculation, an absolute-value circuit is needed. An absolute-value circuit is introduced in [150] and shown in Fig. 7.6. The circuit is composed of a comparator (M3-M6) and a current mirror (M1-M2).

7.3.2.2 Quadratic Circuits

For SSD calculation, a quadratic circuit is needed. In [151], a two-quadrant current squarer is introduced. Due to the two-quadrant operation, the circuit of [151] needs high-bias currents. A more efficient solution is to implement the absolute value circuit of Fig. 7.6, and a one-quadrant quadratic circuit separately.

The first quadratic circuit, shown in Fig. 7.7 [152], uses transistors M$_R$ (I/V converter) and M$_1$ (nonlinear V/I converter) to produce the square of the input current. When M$_R$ is biased into the linear region so that V$_r$ -V$_c$< V$_{t,R}$ making the I/V converter approximately linear. The current through M$_R$ is then [153]

$$I_{in} = K_R \left(\frac{V_{sd,R}^2}{2} + (V_c + V_{t,R}) \cdot V_{sd,R} \right) . \tag{7.8}$$

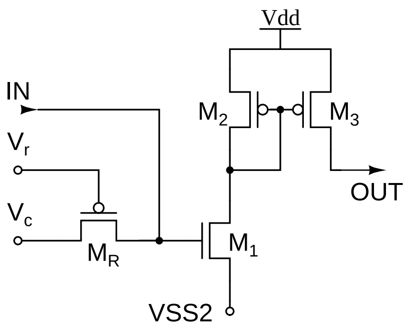


Figure 7.7 Quadratic circuit 1 [152]

M$_1$ is biased into the edge of the weak inversion region using bias voltages V$_c$ and VSS2 so that $\frac{\delta I_{M1}}{\delta I_{in}}$ is very small when I$_{in}$=0. When I$_{in}$>0 M1 shifts into strong inversion where I$_{ds,M1}$ has square law characteristics.

Another version of the circuit of Fig. 7.7 is shown in Fig. 7.8 [152]. The circuit was originally intended for cubic behavior but, by changing the bias voltages, can be made to function as a quadratic approximation. It is shown in [153] that the speed of channel resistance increase of an n-type transistor increases as a function of I$_{in}$. The speed of the increase rises with V$_{ds}$, which accounts for the exponential behavior of I$_{ds,M1}$.

A third quadratic circuit is shown in Fig. 7.9 [154]. In the circuit of Fig. 7.9, the parallel combination of transistors M$_6$ (in the linear region) and M$_7$ (in saturation) can be designed so that they convert the current I$_R$=I$_{B1}$ +I$_{in}$ approximately linearly to the voltage V$_{g,1}$. When the rest of the transistors are biased in saturation, the output current I$_{out}$ is

$$I_{out} = \frac{\beta_1}{2}(R_{6,7} \cdot I_R - V_{t,1})^2 - \frac{\beta_4}{\beta_3} - I_{B_2} \tag{7.9}$$

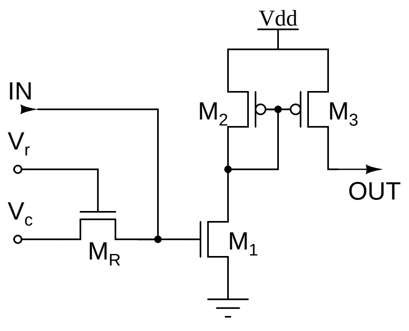


Figure 7.8 Quadratic circuit 2 [152]

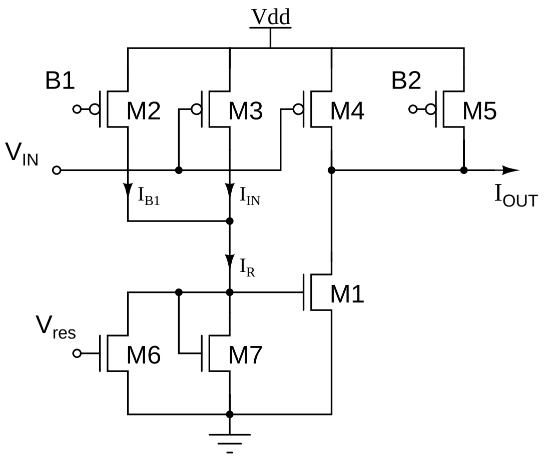


Figure 7.9 Quadratic circuit 3 [154]

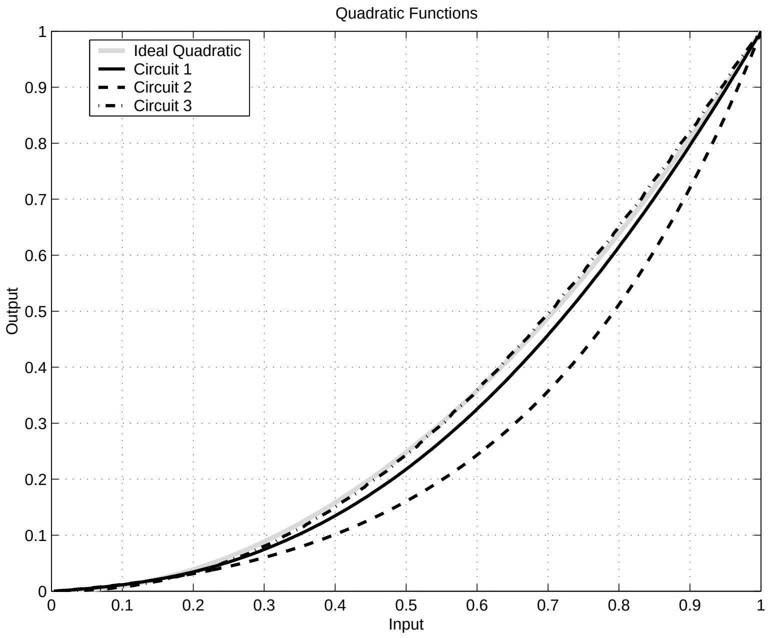


Figure 7.10 Simulated quadratic functions [152], [154].

where $R_{6,7} = \delta V_{g,1}/\delta I_R$ is the slope of the I/V converter and the current I_{B_2} is chosen so that I_{out}=0 when I_{in}=0.

The quadratic behavior of all the circuits is shown in Fig. 7.10.

7.3.3 Cell Structure

Two types of cells incorporating the 3-neighborhood were designed. The first was a preliminary version to validate the proof-of-concept. The second was an advanced version which, with the addition of reference and current pixel memories, would be fully capable of AME.

7.3.3.1 First Version

The first array realizing the introduced 3rd neighborhood had the cell structure of Fig. 7.11. The array is a CNN-type where the cell is a basic linear non-propagative cell, shown in Fig. 3.4. This is capable of realizing the B-templates of Eq. 7.1 and Eq. 7.2. The first and third neighborhood were implemented as separate non-propagative cells due to ease-of-test purposes. The cell logic includes the absolute power circuit of Fig. 7.6 and the quadratic circuit of Fig. 7.9.

In addition to the B-templates of Eq. 7.1 and Eq. 7.2, a third template

$$B_{ave2} = \begin{bmatrix} \frac{4}{64} & \frac{6}{64} & \frac{6}{64} \\ \frac{6}{64} & \frac{9}{64} & \frac{9}{64} \\ \frac{6}{64} & \frac{9}{64} & \frac{9}{64} \end{bmatrix} \tag{7.10}$$

was implemented to reduce the effect of the difference between the 9X9 block size of the analog SAD and the 8X8 block size of a macroblock. This template reduces the proportionate effect of the 3X3 blocks that include pixels that do not belong to the macroblock. These pixels are the ones outside the outlined macroblock in Fig. 7.4. An ideal solution to the oversize analog SAD block would require space-variant templates in the first neighborhood.

A 9x9 array test chip was realized; this is further described in Chap. 7.6.

7.3.3.2 Second Version

Based on the experiences in designing the first version, a more feasible cell structure was also designed. The cell structure is shown in Fig. 7.12. The reference pixel data from the previous frame is stored within the cell memory. Whether the memory is

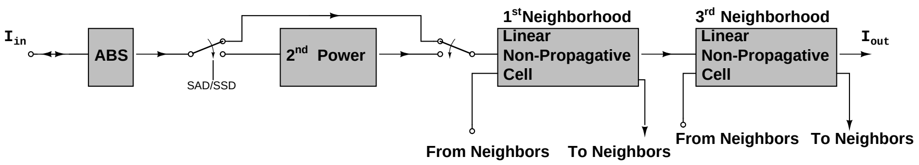


Figure 7.11 Cell structure of first version.

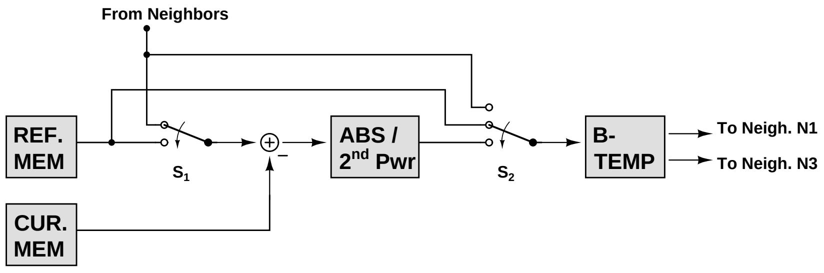


Figure 7.12 Cell structure of second version.

analog or digital is not discussed here. The memory block for the current pixel data in Fig. 7.12 could also be an interface to a photo-detector, as is shown in Fig. 7.1. Switch S_1 in Fig. 7.12 controls whether the reference data (search position 0,0) or the shifted reference data (other search positions) is used for the matching criterion computation. Switch S_2 controls whether the computed data (i.e. $|ref(i+dx, j+dy) - cur(i, j)|$ for SAD), the reference pixel of the current position, or the shifted reference pixel is input into the B-template. The cell logic includes the absolute power circuit of Fig. 7.6 and the quadratic circuit of Fig. 7.8.

The B-template implementation is shown in Fig. 7.13. To reduce the mismatch effects of the data shift, the input of the cell is switched either into an N- or P-mirror, depending on the driver. For example, if the reference pixels were shifted from position (0,4) to (0,0): The first driver would be $M_n4(0,4)$ and the current would be input

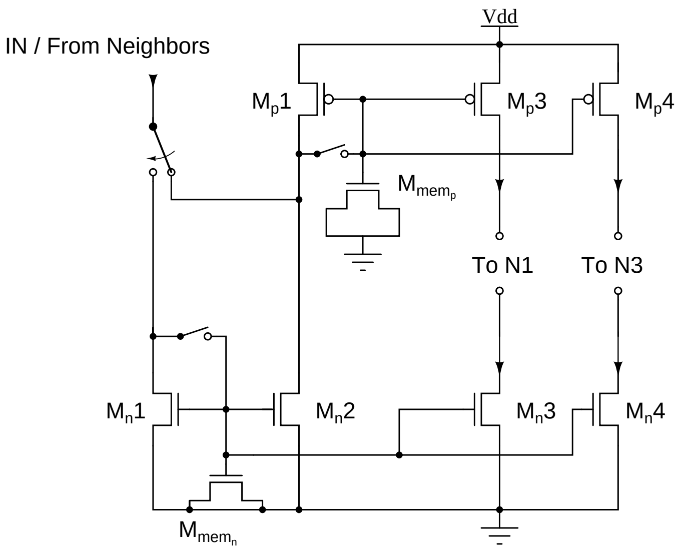


Figure 7.13 B-template structure. Switches in connection with M3 and M4 not shown. Programmability of M3 and M4 not shown.

into $M_p1(0,1)$; the second driver would be $M_p1(0,1)$ and the current would be input into the ABS block of position (0,0). The disadvantage of this method is that a second analog memory ($M_{mem,p}$ in Fig. 7.13) is required in the P-mirror. The templates of Eq. 7.1 and Eq. 7.2 are implemented with $M_{n,p}3,4$. For simplicity the programmability of the templates is not shown in Fig. 7.13. This programmability is realized with switches in the “To N1” and “To N3” outputs. Also, switch S_2 in Fig. 7.12 and the input switch in Fig. 7.13 signify the same switch, but are drawn in each figure for the sake of clarity. In place of the capacitor transistors and the separate input and output transistors, basic SI [147], or S^2I [155] memory cells could also be used. As the output of the quadratic circuit of Fig. 7.8 is a p-mirror, the averaging operation is achieved, first by M_n3 in the first neighborhood, and then M_p4 in the third neighborhood. This has the advantage that M_n4 and M_p3 do not need to be programmable, as they are only used for the shift operation.

7.4 Effect of Analog Building Blocks on Rate-Distortion

The change in rate-distortion induced by the analog inaccuracies was simulated in two parts. First, the separate effects of the three operations on page 91 were simulated independently. The purpose of this was to find out whether there is a fundamental obstacle in the AME realization. Then the simulations were performed with the cumulative effect of all inaccuracies. All simulations were done with a digital Simple profile MPEG-4 architecture using the Full Search motion estimation algorithm. As the average can only be computed on 9x9 blocks, the Full Search was made solely with 8x8 blocks (the advanced prediction mode of MPEG-4), although the conventional method would be to first calculate 16X16 block-size motion vectors and then do a refined 8X8 block-size search around the 16X16 values. The rest of the test environment is described in Chap. 4.1.2.

7.4.1 Average Operation and 9x9 Block Size

The target of the first simulation was to study the effects of inaccuracy in the averaging operation on the effect of the 9x9 block size. Monte Carlo simulations of the cell of Fig. 3.4 showed that, with reasonable cell sizes, a 5% mismatch could be easily achieved. The 5% figure was then used in the averaging operation. The cell structure was assumed to be the one shown in Fig. 7.11, which made possible the simulation with both SAD and SSD as the matching criterion. The quadratic circuit was assumed to be the one shown in Fig. 7.9. Both the SAD and SSD were simulated with averaging templates used in both neighborhoods and the template from Eq. 7.10 in the second neighborhood. The used values for Q_p were 2, 3, 6, 9, 12, 16 and 26. The reference

	Foreman		News	
	PSNR(dB)	BR(%)	PSNR(dB)	BR(%)
Ideal SAD 1	-0.123	1.14	-0.050	0.77
Ideal SAD 2	-0.907	8.00	-0.284	2.59
Worst Case SAD 1	-0.128	1.26	-0.054	0.93
Worst Case SAD 2	-0.917	8.24	-0.316	2.65
Ideal SSD 1	-0.617	2.47	-0.271	2.11
Ideal SSD 2	-0.061	-0.17	-0.040	0.05
Worst Case SSD 1	-0.653	2.96	-0.368	2.25
Worst Case SSD 2	-0.092	0.17	-0.061	0.34

Table 7.1 Drop in PSNR and percentual increase in the number of coded bits for the sequences Foreman and News. The number 1 refers to averaging templates in both neighborhoods. The number 2 refers to Eq. 7.10 template used in the third neighborhood.

matching criterion was the SAD.

The results for the sequences Foreman and News are collected in Table 7.1. The results are so closely spaced that a graph of the results would have been illegible. With the SAD, using averaging templates in both neighborhoods gives the best results. The calculation of the SSD showed better results when using the template of Eq. 7.10 in the second neighborhood. Ideally, the best results were shown in the sequence Foreman by using SSD and the template from Eq. 7.10 in the second neighborhood. Then the results showed a 0.061dB drop in PSNR and a 0.17% drop in the number of coded bits. This can be thought of as equivalent coding performance. With the SSD, the worst case results were a 0.092dB drop in PSNR and a 0.17% increase in the number of coded bits with the sequence Foreman. With the SAD, the ideal results with the sequence News showed a 0.05dB drop in PSNR and a 0.77% increase in the number of coded bits. The worst-case results showed a 0.128dB drop in PSNR and a 1.26% increase in the number of coded bits.

7.4.2 Quadratic Circuits

Chap. 2.2.4.9 postulates that motion estimation is not entirely critical of the used matching criterion. From this postulation, the conclusion can be drawn that ME is also not critical of the exact shape of the quadratic function of SSD[1]. This is important, as the analog accuracy errors in the quadratic circuits can appear as gain errors and as errors in the shape of the quadratic function. Also, noteworthy cell-size savings could be achieved by using the inferior quadratic circuits of Fig. 7.7 and Fig. 7.8 instead of the quadratic circuit of Fig. 7.9. Thus the effect of near-quadratic curves on ME must be simulated.

[1]SSD is defined with $SSD = (ref(i+dx, j+dy) - cur(i,j))^2$ and with any other curve in the matching criterion the criterion is not SSD in the strict sense. Nevertheless, the term SSD will still be used here to compass the various near-quadratic curves.

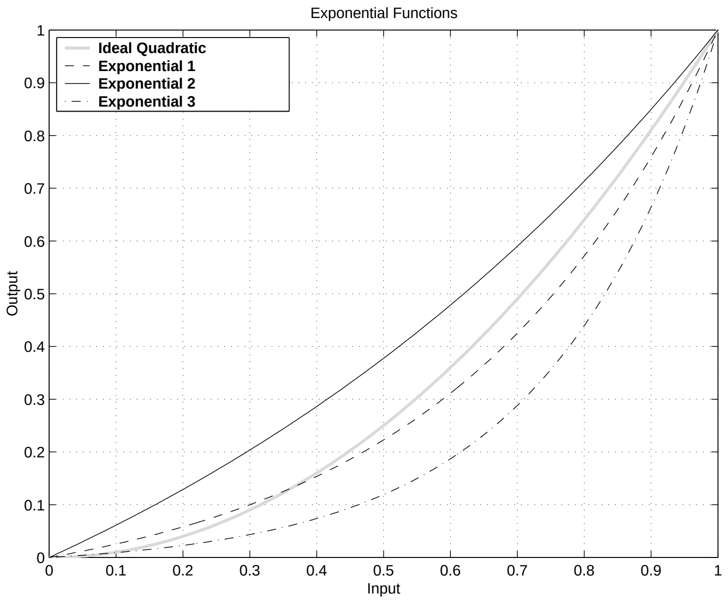


Figure 7.14 Exponential functions used in the simulations.

The simulations were performed with the curve of the circuit of Fig. 7.7 and with an ideal quadratic function with random gain of ±5% added to the slope of the quadratic function. To encompass other sources of mismatch error and other near-quadratic curves, a range of various exponential curves were also simulated. These curves are shown in Fig. 7.14. The used values for Q_p were 2, 3, 6, 9, 12, 16 and 26. Here, using MPEG-4 advanced prediction, 8x8 MVs would have brought no additional information, hence only 16x16 MVs were used. The reference matching criterion was the SSD.

The rate-distortion results for the sequences Foreman and News are collected in Table 7.2. For Foreman, the results for the first quadratic circuit show a slight drop in PSNR, and, a slight drop in coded bits. All of the exponential functions show an

	Foreman		News	
	PSNR(dB)	BR(%)	PSNR(dB)	BR(%)
Circuit of Fig. 7.7	-0.002	-0.620	-0.002	-0.700
Circ. of Fig. 7.9(σ=±5%)	-0.094	1.110	-0.022	-0.170
Exponential 1	0.015	-0.226	-0.004	-0.590
Exponential 2	0.006	0.733	-0.003	-0.710
Exponential 3	0.021	-0.110	-0.002	-0.420
Ideal SAD	-0.232	1.505	-0.042	-0.210

Table 7.2 Drop in PSNR and percentual increase in the number of coded bits compared to the ideal SSD for the sequences Foreman and News. Worst-case exponential numbers refer to Fig. 7.14.

increase in PSNR and depending on the exponential function an increase or decrease in coded bits. Also, all of the quadratic and exponential results are significantly better than the ideal SAD. The worst of the results is the simulated gain error, but this is still better than the SAD result. For News the quadratic circuit of Fig. 7.7 shows a slightly worse rate-distortion. The exponential functions all show a drop in PSNR and a drop in the number of coded bits. All the quadratic and exponential results can be thought of as having coding performance equivalent to the SSD. The results of Table 7.2 are averaged over all values of Q_p. If the results were shown only over high values of Q_p, the difference between the unideal results would be slightly larger but could still be thought of as having equivalent coding performance.

7.4.3 Data Transfer

The target of the last simulation was to study the effects of noise in the reference frame data induced by the shift. With reasonable cell sizes, Monte Carlo simulations of the circuit in Fig. 7.12 show a worst-case mismatch of 5% in each current mirror per each shift. The effect of doubling the worst-case error was also simulated. The used values for Q_p were 3, 6, 9, 16, and 26. In this simulation, using MPEG-4 advanced prediction 8x8 MVs would also have brought no additional information, hence only 16x16 MVs were used. The reference matching criterion was the SAD.

The rate-distortion results compared to the ideal SAD for the sequences Foreman and News are collected in Table 7.3, which shows the change in PSNR and the percentual increase in the number of coded bits. As the target is a low bit-rate where Q_p has high values, the values for Q_p=9 and Q_p=26 are shown along with the average through all Q_p.

For News with high Q_p, the coding performance shows a slight drop, as the bit-rate has increased over 6%, while PSNR has increased only 0.061 on the average. For the lower values of Q_p, the drop in coding performance is lower. In both sequences, doubling the worst-case error shows only a negligible drop in coding performance. From this, it can be concluded that, as the SAD is a sub-optimal matching criterion, the SAD has a larger effect on the coding performance than the error in the reference frame.

The shift error is most significant when the motion vector magnitudes are small, as in News. This can be a slight problem, as the motion vector distribution of natural video sequences is center biased [87]. Also, the error from the high Q_p quantization is much more significant than the error from the shifting of the data. Finally, it can be concluded that shift error is significantly smaller than the errors resulting from the averaging (Chap. 7.4.1) and the quadratic circuits (Chap. 7.4.2).

Error	PSNR(dB)			BR(%)		
Magnitude	Ave.	Q_p=9	Q_p=26	Ave.	Q_p=9	Q_p=26
Foreman 1-100						
5%	0.055	0.055	0.112	1.406	1.586	2.310
10%	0.055	0.055	0.112	1.407	1.590	2.310
News 1-100						
5%	0.061	0.045	0.131	1.588	1.857	6.440
10%	0.061	0.045	0.133	1.591	1.857	6.512

Table 7.3 Change in PSNR and percentual increase in the number of coded bits compared to a sequence coded with no error in the data shift.

7.4.4 Effect of Combined Unidealities

To simulate the combined effect of the inaccuracies, the cell shown in Fig. 7.12 was simulated with varying transistor sizes. The simulation was performed by doing 100 Monte Carlo simulations of the cell with the transistor sizes in Table 7.4[2] and calculating the standard deviation of these 100 values. This standard deviation was then used in creating a QCIF-sized normal distributed random array. The array had a component for the third neighborhood and a component for the first neighborhood for both the current mirrors formed by [M_1 M_3] and [M_1 M_4]. The same operation was performed on the ABS circuit of Fig. 7.6 and the quadratic circuit of Fig. 7.8. Even though the quadratic circuit of Fig. 7.8 is the worst of the three quadratic circuits, it was chosen because of its simplest implementation (i.e. least bias voltages). The AME was then simulated by accessing each component successively. For example, the mismatch computation of $d_x, d_y = [0,7]$ would be performed as follows:

First the reference value of [x_0+0,y_0+7] is multiplied by the deviation of the p_shift between [x_0+0,y_0+7] and [x_0+0,y_0+6]. The same operation is then performed with the deviation of the n_shift between [x_0+0,y_0+6] and [x_0+0,y_0+3] and a p_shift between [x_0+0,y_0+3] and [x_0+0,y_0+0]. The prefixes "n" and "p" stand for the type of current mirror used. The difference between the shifted reference and the [x_0,y_0] value is then multiplied by the deviation of the ABS circuit and the curve of the quadratic circuit. The accuracy of the quadratic curve is maintained with a wordlength of 20 bits. Finally the values in the 9x9 neighborhood of [x_0,y_0] are multiplied by the n_average deviation in the first neighborhood and the p_average deviation in the third neighborhood. In all cases, values exceeding the range were saturated on the highest level.

The Monte Carlo simulations showed that the device mismatch part of the charge injection and clock feedthrough was of no consequence when compared to the device mismatch of the current mirrors.

The results for Foreman and News are shown in Table 7.4, Fig. 7.21, and Fig.

[2] In practice, M_1, M_3 would be 8/9 of the size shown in Table 7.4 and the shift template of Eq. 7.1 would be achieved with M_3+ M_4. Thus, only M_3 would have to be programmable.

	Transistor Sizes				R/D Results			
	M_{sw}	M_{mem}	M_1, M_3	M_4	Foreman		News	
	W/L	W&L	W&L	W	PSNR	BR	PSNR	BR
1	0.32/0.4	5	2.88	0.32	-0.2293	0.0434	-0.0655	0.0102
2	0.32/0.4	5	5.76	0.64	-0.1936	0.0362	-0.0674	0.0108
3	0.32/0.4	5	8.64	0.96	-0.1763	0.0374	-0.0706	0.0103
4	0.32/0.4	5	11.52	1.28	-0.1813	0.0358	-0.0690	0.0097

Table 7.4 Transistor sizes used in simulation (W/L in μm). The L of M_4 is the same as the L of M_1 and M_3 of the same row. The cell with the transistors in question is shown in Fig. 7.13. Also shown is the change in PSNR and percentual increase in the number of coded bits

7.22. The used values for Q_p were 3, 7, 11, 16, 21, and 26. The reference matching criterion was the SAD.

For Foreman, for all transistor sizes the results show a fairly constant loss. For the first three transistor sizes, the results show a slight improvement and the difference between transistor size 3 and 4 depends on Q_p, although that cannot be seen from Table 7.4. One possible explanation for the fairly constant drop is the quadratic curve. As there is no constant improvement between the transistor sizes 3 and 4, the conclusion that the transistor size 3 is sufficiently large can be drawn.

For News, the results are extremely close to each other and the conclusion that the transistor size 1 would be sufficiently large can be drawn.

When compared to the individual results of Table 7.1, Table 7.2, and Table 7.3, with a few exceptions, the combined results are inferior to the individual results, as would be expected. The difference between the individual results and the combined results is small, which would indicate that overly pessimistic deviation values were used in the individual results.

7.5 AME Power Consumption

The power consumption of the AME array was derived by simulating the power consumption and computation time of a single cell. The various sub-circuits of the cell (n and p current mirrors, ABS circuit, quadratic circuit) were simulated separately. The load for each sub-circuit was the subsequent sub-circuit of the cell as is shown in Fig. 7.12. The smallest transistor sizes of Table 7.4 were used in the simulation. Due to the large transistor sizes and the corresponding parasitic capacitances, the effect of the wiring capacitance was negligible and was left out of the simulation.

For a [-7,7] search area $(2\cdot7+1)^2$-1 shifts and $(2\cdot7+1)^2$ computations are needed. Thus the total computation time can be derived with

$$\tau = 128 \cdot (2 \cdot t_{ps} + t_{ns}) + 80 \cdot (t_{ps} + t_{ns}) + 16 \cdot t_{ps} + 225 \cdot (t_{aq} + t_{na} + t_{pa}) \tag{7.11}$$

	I_{WC} (μA)	I_{DC} (μA)	σ_I^2	I_{RMS} (μA)	t (ns)	%t
ps	33	18	$9.8 \cdot 10^{-11}$	20	40	7.5
pa	17	9.2	$2.6 \cdot 10^{-11}$	11	100	12.0
ns	16	8.6	$2.3 \cdot 10^{-11}$	9.8	80	8.8
na	2.0	1.2	$3.3 \cdot 10^{-13}$	1.3	400	47.8
ABS	34	11	$9.8 \cdot 10^{-11}$	14	200	23.9
q	28	7.8	$5.7 \cdot 10^{-10}$	10	100	-

Table 7.5 Maximum (I_{WC}), DC (I_{DC}), variance (σ_I^2), and RMS (I_{RMS}) values of power consumption current for the sub-circuits of Fig. 7.12. Also shown are the computation time (t) and and percentage of total computation time (%t) (from Eq. 7.11).

where $t_{aq} = \max(t_{abs}, t_q)$. The subindexes *ps* and *ns* refer to the shift time of p and n mirrors, respectively. The subindexes *pa* and *na* refer to the averaging time of p and n mirrors, respectively. The subindexes *abs* and *q* refer to the computation time of the ABS and quadratic blocks, respectively. The individual computation times are shown in Table 7.5[3]. The total computation time is 188 μs which enables a maximum frame-rate of over 5300 fps.

For low-power applications the AME can put into an idle state after the computation as is done in [28]. Thus, for a frame-sized AME array, the final power consumption P_{final} is the product of the power consumption during the on-time and the ratio of τ and the frame-rate fps

$$P_{final} = \frac{\tau}{\frac{1}{fps}} \cdot P \tag{7.12}$$

where P is the power consumption during the on-time of the AME array. P can be derived with $P = V_{DD} \cdot I_{RMS}$ where I_{RMS} is the root mean square value of the power supply current. I_{RMS} is

$$I_{RMS} = \sqrt{I_{DC}^2 + \sigma_I^2} \tag{7.13}$$

where I_{DC} is the DC current (the expectation value of the current distribution) and σ_I^2 the variance of the current distribution. The dependence of the power supply current on the input current varies with each sub-circuit. This dependence for the ABS and quadratic circuits is shown in Fig. 7.15. The power consumption current of the current mirrors is linearly dependent on the input current. To derive the true value for I_{RMS} the distribution of the input current would have to be known. The power supply current

[3]From Table 7.5 it can be seen that the n-mirror is slower than the p-mirror. This is due to the larger transconductance of the n-transistor and thus the larger load caused by the Miller effect.

distribution f(I) would then be derived with

$$f(I) = \int_{I_{in,min}}^{I_{in,max}} g(I_{VDD}) \cdot h(I_{in}) \tag{7.14}$$

where g(I_{VDD}) is the power supply current as a function of the input current, h(I_{in}) input current distribution, and $I_{in,min}$ and $I_{in,max}$ are the minimum and maximum input currents, respectively. Depending on the sub-circuit h(I_{in}) is the distribution of the input luminance signal (shift circuits) or the difference between the current and reference pixel (ABS, quadratic, and average circuits). As an approximation f(I) was calculated with even distribution of h(I_{in}). Thus I_{DC} and σ_I^2 can be derived straight from g(I_{VDD}). This approximation gives somewhat conservative results.

By taking into account Eq. 7.12, the total RMS power supply current $I_{RMS,tot}$ is derived as the sum of I_{RMS} of each sub-circuit and the proportionate on-time (%t) of each sub-circuit. Thus $I_{RMS,tot}$ is

$$I_{RMS,tot} = \sum_{nc} I_{RMS}(nc) \cdot \%t(nc) \tag{7.15}$$

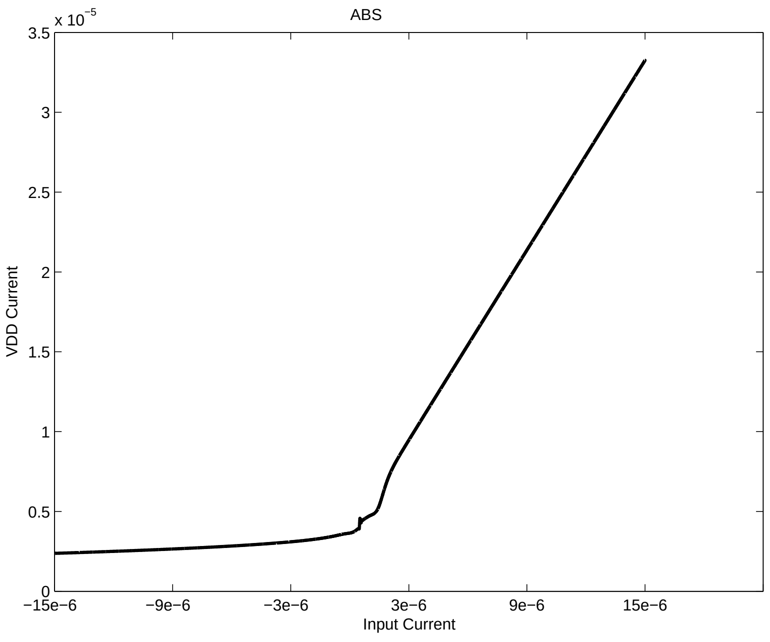
where *nc* is the number of sub-circuits in Table 7.5. From the values in Table 7.5 $I_{RMS,tot}$ is 10.3μA. With $V_{DD} = 2.5V$ and $P_{final} = V_{DD} \cdot I_{RMS,tot}$ the values for P_{final} are 2.8mW (CIF@30fps) and 0.3mW (QCIF@15fps).

Compared to previous analog ME realizations, [24] presents a 16mW (8mW computational array, 8mW analog frame memory), QCIF@15fps realization. In [25], a 11mW, QCIF@17fps realization which includes reference and current block memories, is presented.

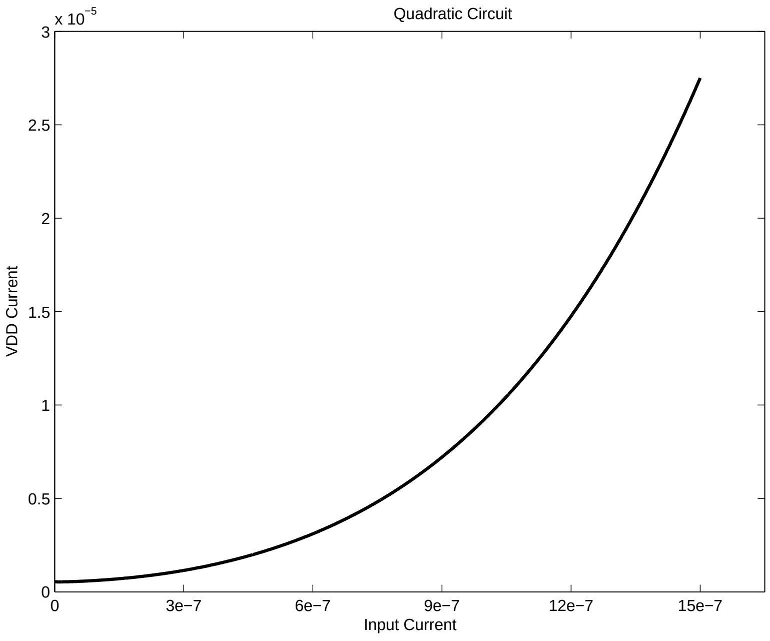
Compared to previous digital ME realizations, [11] presents a realization capable of 0.4mW (QCIF@15fps,) and 2.5 mW (CIF@30fps). The power consumption values of [11] do not include the data transfer between the frame memory and local search area memories. From the power consumption figures of [12] (4.6mW logic, 16.2mW total) it can be estimated that the total power consumption of [11] would be in the range of 1.2mW (QCIF@15fps,) and 7.5 mW (CIF@30fps).

7.6 Realized Test Chip

A test integrated circuit realizing the cell structure of Chap. 7.3.3.1 was designed and fabricated. Two 9x9 separate arrays capable of SAD and SSD computation and data averaging were included. The SAD array included only the ABS circuit of Fig. 7.6, while the SSD array included the ABS circuit and the quadratic circuit of Fig. 7.9. The chip was fabricated in 0.18 μm CMOS and the sizes of the implemented SAD and SSD cells, including the absolute value and quadratic circuits, were 25.3x34.0 μm and



(a)



(b)

Figure 7.15 The power consumption current as a function of the input current. a) ABS circuit (Fig. 7.6) b) Quadratic circuit (Fig. 7.8).

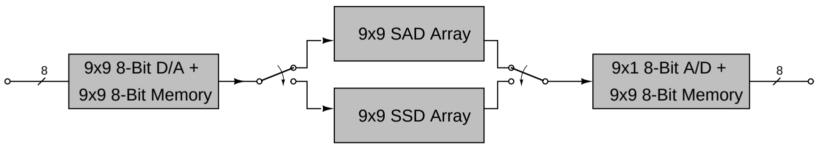


Figure 7.16 Architecture of the realized chip

45.2x33.6 μm, respectively. The chip photograph is shown in Appendix.B. The size of the core is 1.340x0.925 mm.

7.6.1 Chip Structure

As it would have not been feasible to have 81 input and output currents, the chip interface had to be digital, which meant D/A and A/D converters and memory had to be on-chip. This, however, could be achieved with no extra silicon area, as the vendor's minimum area requirement was substantially higher than the area of the computation arrays. The architecture of the chip, including these peripherals, is shown in Fig. 7.16. As the silicon size was not an issue, the memories were implemented as DFFs in order to simplify the AGU design. Not shown in Fig. 7.16 are the bias circuits for the quadratic circuit and the bias circuits to generate the input current offset, which were also included on-chip.

7.6.1.1 D/A-Converter

The D/A-converters were implemented as the binary weighted converters shown in Fig. 7.17 [156]. The dynamic range of a converter is 10 μA with a controllable offset (I_{offset} in Fig. 7.17). The dynamic range of the converter is also controllable (I_{bias} in Fig. 7.17). Also, to speed-up settling of the output current, the outputs of the binary weighted transistors (16W through 0.125W in Fig. 7.17) are switched into dummy load transistors (D_{16} through D_0 in Fig. 7.17) when their respective bits (B_7 through B_0 in Fig. 7.17) are zero. The size of a single converter is 39.7x14.2 μm. The layout of the converter is the same as in [126].

7.6.1.2 A/D-Converter

The A/D-converters were implemented as asynchronous Successive Approximation Registers (SAR) shown in Fig. 7.18 [156]. The D/A-converter in Fig. 7.18 is the same as the converter described above.

The conversion is performed, from an initial state of [11111111], by first switching the MSB B_7 to zero. The input current is then compared to the output of the D/A; if the input current is below the output of the D/A then B_7 is kept at zero. Otherwise

B_7 is switched to one. This procedure is than repeated for all the remaining bits with the previous conversion decision kept unchanged. The control logic of the conversion is asynchronous and controlled by only two bits: Start to signal the beginning of the conversion process and Zero to reset the logic. The start signal propagates through the shift register shown in Fig. 7.19. The shift register is composed of the current-starved buffer shown in Fig. 7.20. The delay of the shift register is specified with the bias voltage NBias of Fig. 7.20. The output of the comparator in Fig. 7.18 is fed into an 8-bit SRAM. Each bit of the SRAM is controlled by the output of the shift register. The result of the conversion is obtained with

$$B_i = NOR(B_{i1}, B_{i2}) \tag{7.16}$$

where B_{i1} and B_{i2} are shown if Fig. 7.19.

The asynchronous mode of operation has several advantages. A constantly switching clock signal consumes power. Also, a fast clock signal induces noise through substrate and electromagnetic coupling, which could negatively affect the conversion processes and analog computation on the chip. Furthermore, the routing of the clock signal consumes wiring and thus silicon area.

The size of a single converter is 179.8x14.2 μm. The layout of the converter is the same as in [126]. A 9x9 A/D array would increase the silicon area over the vendor's minimum area and thus only a 1x9 A/D array was implemented. The computation results are then read out column-by-column.

7.6.2 Measurements

The D/A and A/D converters of Chap. 7.6.1.2 and Chap. 7.6.1.2, respectively, were first realized in [126]. When this test chip was designed, the converters were not yet measured. The measurements [126] showed that the accuracy of the converters was not of sufficient quality to measure meaningful results from the SAD and SSD arrays.

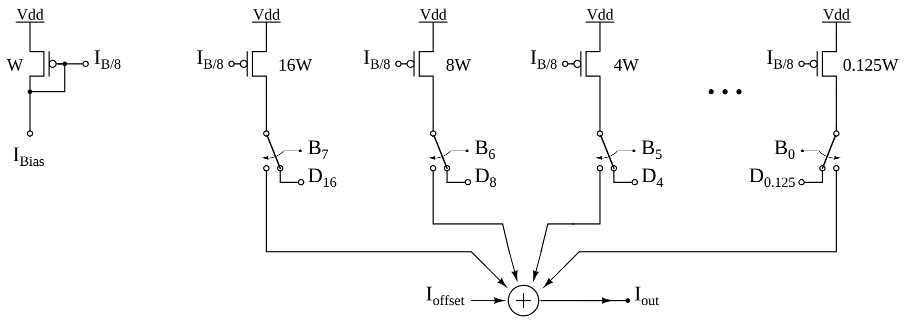


Figure 7.17 D/A-converter [156]

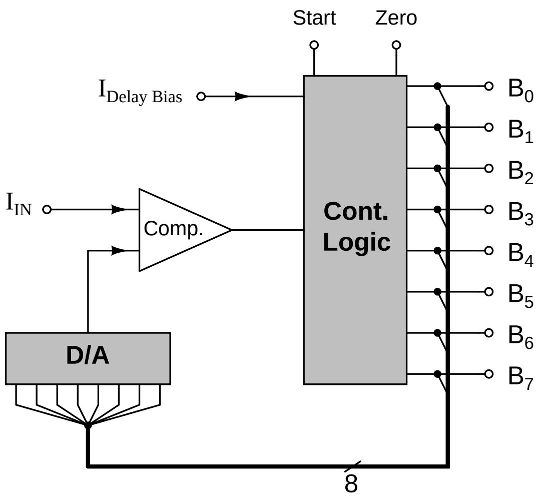


Figure 7.18 A/D-converter [156]. The D/A-converter is shown in Fig. 7.17.

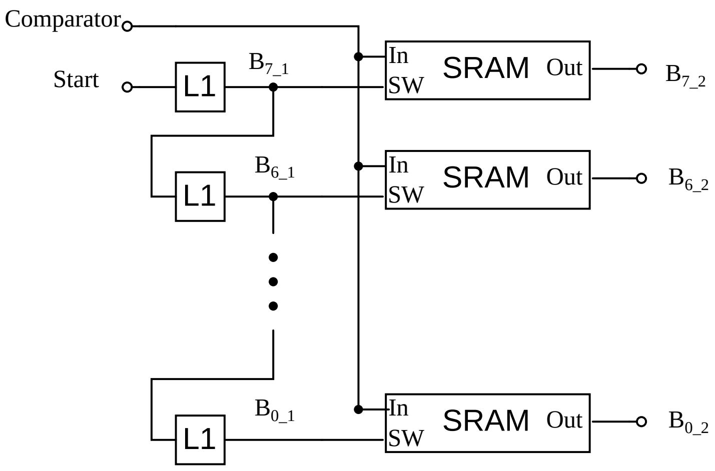


Figure 7.19 Asynchronous shift register.

As the measured converters are exactly the same down to the layout and the used process is the same the measurement results should correspond with the converters here. Unlike [126], the designed chip did not have the possibility of measuring the converters separately; this would have enabled separate calibration of the converters. Without the calibration possibility, the quality of the converters would have masked the error in the SAD and SSD arrays.

For the D/A array, the mean Integral Non-Linearity (INL) was 0.773 LSB with a standard deviation of 0.188 LSB. The D/A array's mean differential Non-Linearity (DNL) was 0.788 LSB and the standard deviation 0.186 LSB.

For the A/D array, the worst-case INL was 2.880 LSB and the DNL 5.64 LSB.

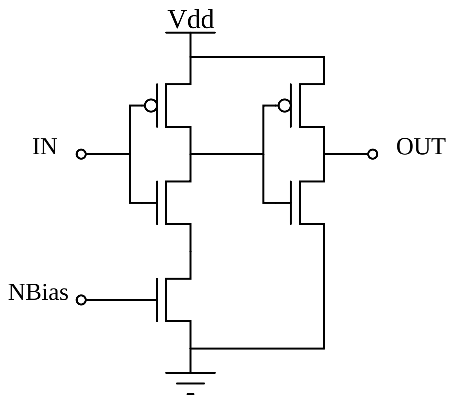
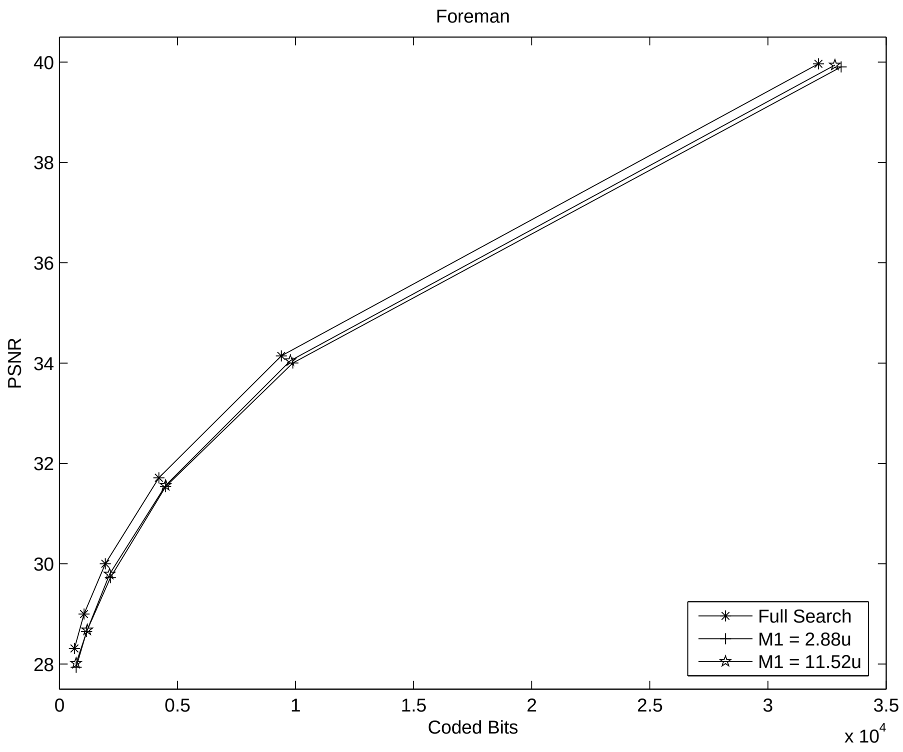


Figure 7.20 The current-starved buffer L1 of the shift register in Fig. 7.19.

The mean Effective Number of Bits (ENOB) was 5.53 bits with a standard deviation of 0.09 bits. The bandwidth of the A/D was 1.3 MHz.

The full measurement results and measurement process will be presented in the future in [157].

If a second version of the chip had been designed, the cell structure of Chap. 7.3.3.2 would have been more feasible. Unfortunately, at the time, the vendor discontinued the university price version of 0.18 μm CMOS process, which would have meant moving to 0.13 μm CMOS. Unfortunately, time constraints did not make designing a new chip with a new cell structure and a new process possible.



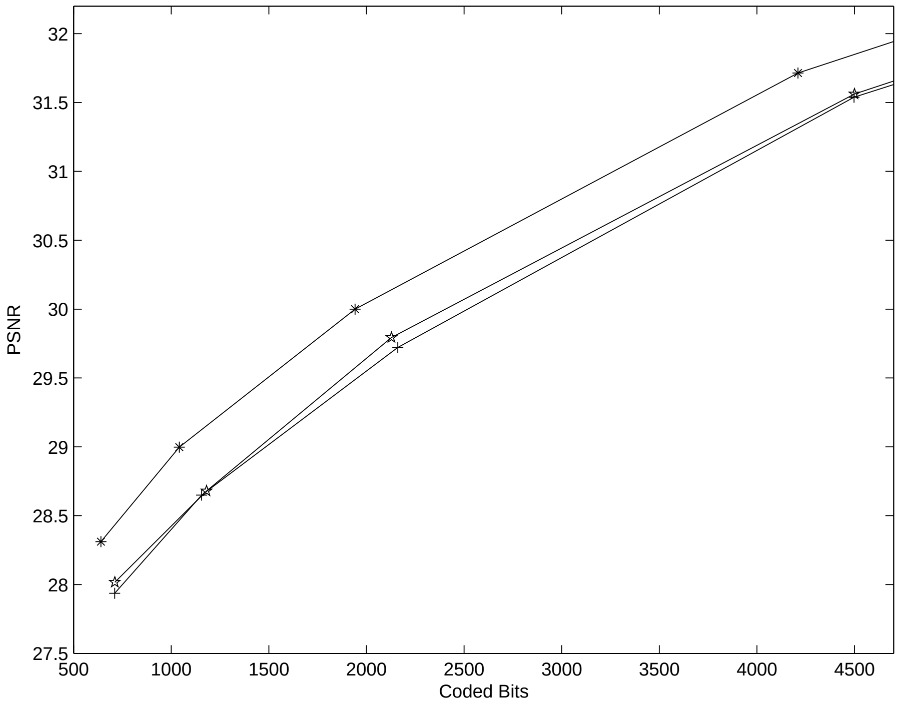
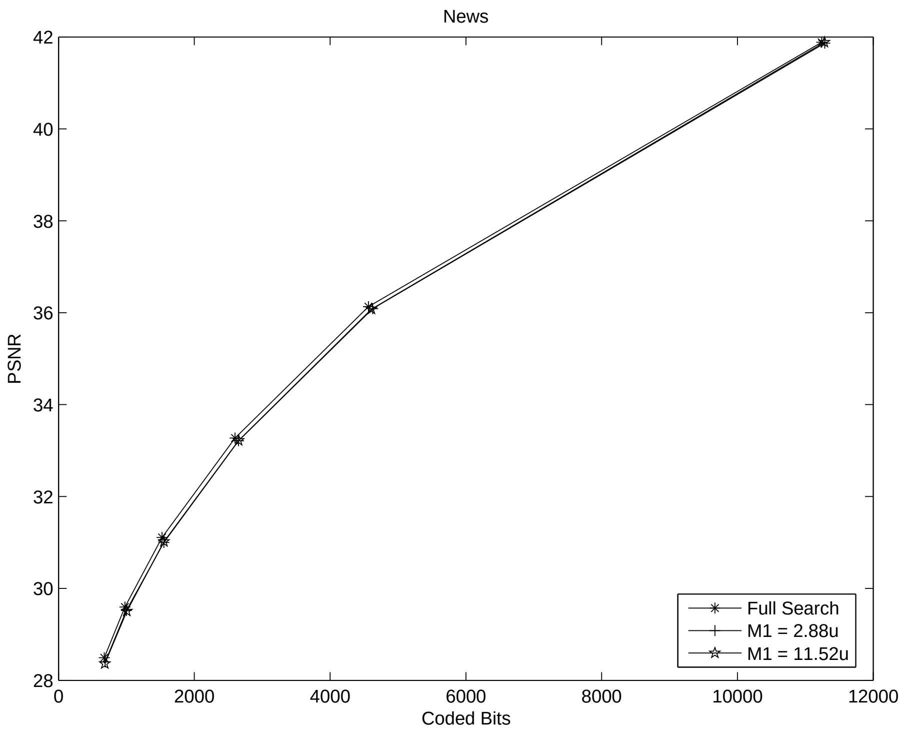


Figure 7.21 Foreman rate-distortion graphs (top graph) and close-ups of low bit-rate values (bottom graph) with varying values of Q_p.



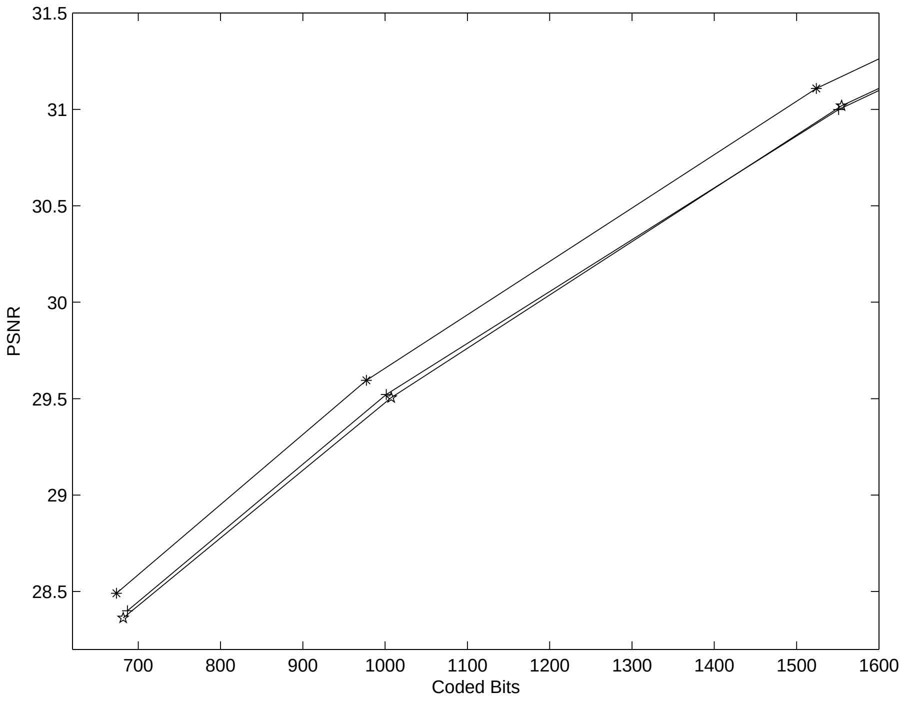


Figure 7.22 News rate-distortion graphs (top graph) and close-ups of low bit-rate values (bottom graph) with varying values of Q_p.

Chapter 8

Conclusions

This thesis presents several algorithmic and hardware solutions to reduce the power consumption of block-based motion estimation, including a full analog motion estimation array. This use of analog parallel processor arrays in computing block-based motion estimation is the new contribution of this thesis. Motion estimation will, in all likelihood, be a part of video codecs in the near future. Thus, the concept of calculating the motion estimation result within analog arrays that are integrated with the image sensor is promising and could hold one of the keys to low-power, large-frame-size video coding.

On the algorithmic level, two algorithms are presented. The first reduces computational complexity and thus power consumption of frame-based coding in an object-based Core profile MPEG-4 encoder. The second algorithm presents a method of computing a near-optimal H.264 block-size partition without resorting to computationally complex Lagrange optimization. In addition to these algorithms, CNN templates that reduce the shape bit rate of MPEG-4 are presented. On the hardware level, a novel third neighborhood connection is presented. Using this connection and specific cell logic, the analog motion estimation array is realized. Such an analog motion estimation implementation enables lower power consumption than comparable digital realizations.

To develop analog motion estimation into a marketable product, more research needs to be done on the subject. First, the integration with an image sensor needs to be investigated. The question that needs to be answered is whether it is more feasible to integrate the AME cell with the sensor cell or whether the AME array should be kept separate from the sensor. Second, the concept of very irregularly wired arrays needs to be investigated. As the concepts of this thesis were derived from a CNN segmentation implementation, the general idea was to keep the AME addable to the segmentation. In the course of the research, it was seen that the most promising avenue of design would be the above-mentioned integration with an image sensor. Thus, if the concept of merging the segmentation and AME is forgotten, the cell size could be made smaller. Small cell size would then leave more space for the wiring, which would further make possible irregular connections between the cells. Such connections could include a hexagonal connection for gradient-based ME or connections to average 4x4, 8x8, or other sized blocks.

Complex computational CNN or CNN-type arrays also may have promise. In addition to segmentation, the determining of slices could also be an avenue of research. In H.264, flexible macroblock ordering (also known as slice groups) has an impact on packet loss. The slices could be determined with algorithms similar to the ones presented in this thesis. Also, for multiple reference frame ME, the detection of uncovered areas that have their best prediction vector in a further-off reference frame should be investigated.

So, it can be seen that video coding, with its large amount of data that needs to be processed, holds many promising research subjects for analog (and digital) parallel processing.

References

[1] Wenger S.; Hannuksela M.; Stockhammer T.; “Identified H.26L Applications” ITU-T SG 16, Doc. VCEG-L34, Eibsee, Germany, 2001. Available: http://ftp3.itu.int/av-arch/jvt-site

[2] Third Generation Partnership Project; “Transparent End-to-End Packet Switched Streaming Service (PSS); RTP Usage Model” 3GPP Technical Specification 3GPP TR 26.937.

[3] Third Generation Partnership Project; “Multimedia Broadcast/Multicast Services” 3GPP Technical Specification 3GPP TR 29.846.

[4] Third Generation Partnership Project; “Multimedia Messaging Service (MMS); Media Formats and Codecs” 3GPP Technical Specification 3GPP TR 26.140.

[5] Third Generation Partnership Project; “Packet Switched Conversational Multimedia Applications; Default Codecs” 3GPP Technical Specification 3GPP TR 26.235.

[6] Stockhammer, T.; Hannuksela, M.M.; Wiegand, T.; “H.264/AVC in wireless environments” *IEEE Transactions on Circuits and Systems for Video Technology,* Volume: 13, Issue: 7, July 2003 Pages:657 - 673

[7] H. Nakayama, et al.; “An MPEG-4 video LSI with an error-resilient codec core based on a fast motion estimation algorithm”, in *Dig. Tech. Papers IEEE Int. Solid-State Circuits Conf.*, 2002, Page(s):368-369.

[8] T. Fujiyoshi, et. al.; "H.264/MPEG-4 Audio/Visual CODEC LSI with Module-Wise Dynamic Voltage/Frequency Scaling", *Digest of Technical Papers. IEEE International Solid-State Circuits Conference, ISSCC. 2005*. 2005, Page(s): 132 - 133

[9] ViBE MPEG-4 AVC encoder, Available: http://www.thomsongrassvalley.com/docs/DataSheets/transmission/-vibe_encoders/VCN-1006D.pdf

[10] Polycom VSX 3000, Available: http://www.polycom.com/common/-pw_item_show_doc/1,1276,4552,00.pdf

[11] Miyama, M.; Miyakoshi, J.; Kuroda, Y.; Imamura, K.; Hashimoto, H.; Yoshimoto, M.; "A sub-mW MPEG-4 motion estimation processor core for mobile video application" *IEEE Journal of Solid-State Circuits*, Vol. 39, Issue 9, Sept. 2004 Page(s):1562 - 1570

[12] Yoon C.-W.; Yoo H.-J.; "Low Power Motion Estimation and Motion Compensation Block IPs in MPEG-4 Video Codec Hardware for Portable Applications" *IEICE Trans. Electron.* Vol.E86-C, April 2003, Page(s):553-560

[13] Miyama M.; et. al.; "Ultra Low Power Motion Estimation Processor for MPEG2 HDTV Resolution Video " *IEICE Trans. Electron.* Vol.E86-C, April 2003, Page(s):561-569

[14] Huang Y.-W.; Wang T.-C.; Hsieh B.-Y.; Chen L.-G.; "Hardware architecture design for variable block size motion estimation in MPEG-4 AVC/JVT/ITU-T H.264" *Proceedings of the 2003 International Symposium on Circuits and Systems, ISCAS '03*. Vol.2, Page(s):II-796 - II-799

[15] Yap, S. Y.; Mccanny, J. V.; "A VLSI architecture for advanced video coding motion estimation" IEEE *International Conference on Application-Specific Systems, Architectures, and Processors, 2003. Proceedings.* Page(s):293 - 301

[16] Paasio, A.; Paakkulainen, J.; Isoaho, J.; "A Multiplier-Free Fixed-Task Digital CNN Array for Video Segmentation System. *The 2000 IEEE International Symposium on Circuits and Systems, ISCAS 2000*, Vol. 3 , Pages:710 - 713

[17] Sziranyi, T.; Czuni, L.; "Spatio-Temporal Segmentation with Edge Relaxation and Optimization Using Fully Parallel Methods" *15th International Conference on Pattern Recognition, 2000*. Vol. 4, Pages:820-823

[18] Feiden, D.; Tetzlaff, R.; "Binary Image Coding Using Cellular Neural Networks" *Proceedings of the International Joint Conference on Neural Networks, 2003*. Vol. 2, Pages:1149 - 1152

[19] Grassi, G.; Grieco, L.A.; "Object-oriented image analysis via analogic CNN algorithms. I. Motion estimation" *Proceedings of the 2002 7th IEEE International Workshop on Cellular Neural Networks and Their Applications, 2002. (CNNA 2002)*. 22-24 July 2002 Page(s):172 - 179

[20] Foldesy, P.; Zarandy, A.; Szolgay, P.; Sziranyi, T.; "Real-life application case studies using CMOS 0.8 μm CNN universal chip: analogic algorithm for motion detection and texture segmentation Cellular Neural Networks and their Applications", *Proceedings., 1996 Fourth IEEE International Workshop on CNNA-96*. 24-26 June 1996 Page(s):363 - 368

[21] Sarpeshkar, R.; Kramer, J.; Indiveri, G.; Koch, C.; "Analog VLSI architectures for motion processing: from fundamental limits to system applications" *Proceedings of the IEEE* Vol. 84, Issue 7, July 1996 Page(s):969 - 987

[22] Shi, B.E.; Roska, T.; Chua, L.O. "Design of Linear Cellular Neural Networks for Motion Sensitive Filtering" *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*. Vol. 40, May 1993 Pages:320 - 331

[23] Arena, P.; Basile, A.; Bucolo, M.; Fortuna, L.; "An object oriented segmentation on analog CNN chip" *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, Volume 50, Issue 7, July 2003 Page(s):837 - 846

[24] Tomasini, A.; Brattoli, M.; Chioffi, E.; Colli, G.; Gerna, D.; Pasotti, M.; "B/W adaptive image grabber with analog motion vector estimator at 0.3 GOPS" *IEEE International Solid-State Circuits Conference, Digest of Technical Papers*, 1996, Page(s):94 - 95, 425

[25] Panovic, M.; Demosthenous, A.; "A Low Power Block-Matching Analog Motion Estimation Processor" *IEEE International Symposium on Circuits and Systems*, 2005. ISCAS 2005, Page(s):4827 - 4830

[26] Tartagni, M.; Leone, A.; Pirani, A.; Guerrieri, R.; "A block-matching module for video compression" *IEEE Symposium on Low Power Electronics, 1994. Digest of Technical Papers*, Page(s):24 - 25

[27] Panovic, M.; Demosthenous, A.; "Architectures for Analog Motion Estimation Processors: A Comparison" *IEEE International Symposium on Circuits and Systems*, 2005. ISCAS 2005, Page(s):4566 - 4569

[28] Paasio A.; Kananen A.; Halonen K. "A QCIF Resolution Binary I/O CNN-UM Chip." *Journal of VLSI Signal Processing.* Vol. 23, No. 2/3, 1999, Page(s):281-290.

[29] Stoffels A.; Roska T.; Chua L. O. "Object-Oriented Image Analysis for Very-Low-Bitrate Video-Coding Systems Using The CNN Universal Machine." *International Journal on Circuit Theory and Applications.* Vol. 25, 1997, Page(s):235-258.

[30] Koskinen, L.; Paasio, A.; Halonen, K.; “Motion Estimation Computational Complexity Reduction With CNN Shape Segmentation” *IEEE Transactions on Circuits and Systems for Video Technology*, Volume 15, Issue 6, June 2005 Page(s):771 - 777

[31] Koskinen L.; Laiho M.; Halonen K.; “CNN Algorithm for H.264 Motion Estimation Partitions” *Proceedings of the 2004 8th IEEE International Workshop on Cellular Neural Networks and Their Applications, (CNNA 2004)* , 2004, Page(s):346-351

[32] Koskinen L.; Paasio A.; Halonen K.; “Parallel Processor Algorithm for Variable Block-Size Computation at Low Bitrates” *IEEE International Symposium on Circuits and Systems, 2005. ISCAS 2005.* 23-26 May 2005 Page(s):4122 - 4125

[33] Hämäläinen S.; Koskinen L.; Halonen K.; “A Hardware-Based Predictive Motion Estimation Algorithm” *IEEE International Symposium on Circuits and Systems, 2005. ISCAS 2005.* 23-26 May 2005 Page(s):6114-6117

[34] Koskinen, L.; Paasio, A.; Laiho, M.; Halonen, K.; “Effect of CNN shape segmentation on MPEG-4 shape bit-rate” *IEEE International Symposium on Circuits and Systems, 2002. ISCAS 2002.* Volume 4, 26-29 May 2002 Page(s):IV-552 - IV-555

[35] Koskinen, L.; Paasio, A.; Halonen, K.; “3-Neighborhood motion estimation in CNN silicon architectures *Proceedings of the 2004 International Symposium on Circuits and Systems, 2004. ISCAS ’04.* Volume 5, 23-26 May 2004 Page(s):V-708 - V-711

[36] Koskinen L.; Laiho M.; Halonen K.; “Motion Estimation Matching Criterion Computation with Analog Circuits” *Proceedings of the 2004 8th IEEE International Workshop on Cellular Neural Networks and Their Applications, (CNNA 2004)* , 2004, Page(s):216-221

[37] Koskinen L.; Paasio A.; Halonen K.; “Efficient Shift of Reference Data in Analog Motion Estimation” *Proceedings of the 2005 9th IEEE International Workshop on Cellular Neural Networks and Their Applications, (CNNA 2005),* 2005, Page(s):130-133

[38] Koskinen L.; Paasio A.; Kananen A.; Halonen K.; “MPEG-4 Encoder Architecture for a Shape Segmentation CNN Chip” *Proceedings of 2nd Workshop and Exhibition on MPEG-4 (WEMP4 2001),* 18-20.6.2001, Page(s):41-44

[39] Koskinen L.; Paasio A.; Kananen A.; Halonen K.; “A MPEG-4 Shape Segmentation Algorithm” *Proceedings of the 15th European Conference on Circuit*

Theory and Design (ECCTD 2001), Volume 1, 28-31.8.2001, Page(s):I101-I104

[40] Paasio A.; Kananen A.; Koskinen L.; Halonen K.; “Different Possibilities for Realizing the Bipolar Image Processing Tasks in CNN Field” *Proceedings of the 15th European Conference on Circuit Theory and Design (ECCTD 2001),* Volume 3, 28-31.8.2001, Page(s):III73-III76

[41] Koskinen L.; Laiho M.; Paasio A.; Halonen K.; “MPEG-4 Based Modifications for an CNN Segmentation Chip” *Proceedings of the 2002 7th IEEE International Workshop on Cellular Neural Networks and Their Applications, 2002. (CNNA 2002)*, 2002, Page(s):71-77.

[42] Koskinen L.; Paasio A.; Halonen K.; “CNN Shape Segmentation Advantages in MPEG-4 Simple Profile Encoding” *Proceeding of the Seventh International Symposium on Signal Processing and its Applications (ISSPA03)*. Volume 2, 1-4 July 2003 Page(s):117 - 120

[43] Koskinen L.; Paasio A.; Halonen K.; “A Novel 3-Neighborhood for Parallel Processor Motion Estimation Implementations” *Proceedings of the 21st Norchip (NORCHIP 2003) conference*, 2003, Page(s):141-144.

[44] Koskinen L.; Paasio A.; Halonen K.; “Improved CNN Algorithm for H.264 Motion Estimation Partitions” *Proceedings of the 2005 9th IEEE International Workshop on Cellular Neural Networks and Their Applications, (CNNA 2005)* , 2005, Page(s):142-145

[45] Kananen, A.; Paasio, A.; Halonen, K.; “Spatial Low-Pass Filtering and Gradient Calculation Blocks for Fast Image Analysis” Submitted to *IEEE Transactions on Circuits and Systems I*

[46] The JM reference software. Available: http://iphome.hhi.de/suehring/tml/

[47] ITU-T Recommendation H.261, *Video codec for Audiovisual Services at p X 64 kbit/s*. March 1993.

[48] ISO/IEC 11172: *Information Technology-Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to About 1.5 Mbit/s*. Geneva 1993.

[49] ISO/IEC 13818-2: *Generic Coding of Moving Pictures and Associated Audio Information-Part 2: Video*. 1994. Also ITU-T Recommendation H.262.

[50] ISO/IEC 14496-2: *Information Technology-Coding of Audiovisual Objects-Part 2: Visual*. Geneva 2000.

[51] ITU-T Recommendation H.263, *Video Coding for Low Bitrate Communication*. Version 1 Nov. 1995; Version 2 Jan. 1998; Version 3 Nov. 2000.

[52] Joint Video Team of ITU-T and ISO/IEC JTC 1, “Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC)” document JVT-G050r1, May 2003; technical corrigendum 1 documents JVT-K050r1 (non-integrated form) and JVT-K051r1 (integrated form), March 2004; and Fidelity Range Extensions documents JVT-L047 (non-integrated form) and JVT-L050 (integrated form), July 2004.

[53] XviD Software Package. Available: http://files.xvid.org/downloads/xvidcore-0.9.1.tar.gz.

[54] DivX Create Bundle. Available: http://www.divx.com/divx/create/download/

[55] Srinivasan S.; Hsu P.; Holcomb T.; Mukerjee K.; Regunathan S. L.; Lin B.; Liang J.; Lee M.-C.; Ribas-Corbera J.; “Windows Media Video 9: overview and applications” *Signal Processing: Image Communication*, Volume 19, Issue 9, October 2004, Pages:851-875

[56] Jain, J.; Jain, A.; “Displacement Measurement and Its Application in Interframe Image Coding“ *IEEE Transactions on Communications*, Volume: 29, Issue: 12, Dec 1981 Pages:1799 - 1808

[57] Hsiang S.-T.; Woods J. W.; “Embedded video coding using invertible motion compensated 3-D subband/ wavelet filter bank” *Signal Processing: Image Communications* Volume 16, May 2001, Pages:705-724

[58] Kuhn P. *Algorithms, Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation.* Kluwer Academic Publishers, 1999, 239 p.

[59] Gonzalez R. C.; Woods R. E.; *Digital Image Processing*. Addison-Wesley Publishing Company, 1993, p.228

[60] CCIR Recommendation 601-1. “Encoding Parameters for Digital Television for Studios”

[61] Huffman D. A.; “A Method for the Construction of Minimum Redundancy Codes”, *Proc. IRE,* vol. 40, no. 9, Page(s):1098-1101, 1952.

[62] Witten I. H.; Neal R. M.; Cleary J. G.; “Arithmetic coding for data compression” *Communications of the ACM* Volume 30, Issue 6 (June 1987) Page(s): 520 - 540

[63] Golomb, S.; “Run-length encodings” *IEEE Transactions on Information Theory*, Vol. 12, Issue: 3, Jul 1966, Page(s):399 - 401

[64] Bhaskaran V.; Konstantinides K.; *Image and Video Compression Standards, Second Edition*. Kluwer Academic Publishers, 1997, 454 p.

[65] Bjontegaard G.; Lillevold K.; “Context-Adaptive VLC Coding of Coefficients” JVT Document JVT-C028, [Online], Fairfax, VA, May 2002 Available: http://ftp3.itu.int/av-arch/jvt-site

[66] Lillevold K.; “Proposed Updates to CAVLC” JVT Document JVT-D036, [Online], Klagenfurt, July 2002 Available: http://ftp3.itu.int/av-arch/jvt-site

[67] Teuhola J; “A compression Method for Clustering Bit-Vectors”, *Information Processing Letters*, vol. 7, 1978, Page(s):308-311

[68] Brady, N.; Bossen, F.; Murphy, N.; “Context-based arithmetic encoding of 2D shape sequences” *Proceedings., International Conference on Image Processing, 1997*, Volume: 1, 26-29 Oct. 1997 Pages:29 - 32

[69] Marpe, D.; Schwarz, H.; Wiegand, T.; “Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard” *IEEE Transactions on Circuits and Systems for Video Technology*, Volume 13, Issue 7, July 2003 Page(s):620 - 636

[70] O’Neal, J.; Natarajan, T.; “Coding isotropic images” *IEEE Transactions on Information Theory*, Volume 23, Issue: 6, Nov 1977 Page(s):697 - 707

[71] Habibi, A.; Wintz, P.; “Image Coding by Linear Transformation and Block Quantization” *IEEE Transactions on Communications*, Volume 19, Issue 1, Feb 1971 Page(s):50 - 62

[72] Haskell B. G.; Puri A.; Netravali A. N.; *Digital Video: An Introduction to MPEG-2*. Kluwer Academic Publishers, 1997, 441 p.

[73] Egger, O.; Fleury, P.; Ebrahimi, T.; Kunt, M.; “High-performance compression of visual information-a tutorial review- part I: still pictures” *Proceedings of the IEEE*, Volume: 87, Issue: 6, June 1999, Pages:974 - 975

[74] Malvar, H.S.; Hallapuro, A.; Karczewicz, M.; Kerofsky, L.; “Low-complexity transform and quantization in H.264/AVC” *IEEE Transactions on Circuits and Systems for Video Technology*, Volume: 13, Issue: 7, July 2003, Pages:598 - 603

[75] Girod, B.; “Motion-compensating prediction with fractional-pel accuracy” *IEEE Transactions on Communications*, Volume: 41, Issue: 4, April 1993, Pages:604 - 612

[76] Wiegand, T.; Xiaozheng Z.; Girod, B.; “Long-term memory motion-compensated prediction” *IEEE Transactions on Circuits and Systems for Video Technology*, Volume: 9, Issue: 1, Feb. 1999, Pages:70-84

[77] Kappagantula, S.; Rao, K.; “Motion Compensated Interframe Image Prediction” *IEEE Transactions on Communications*, Volume: 33, Issue: 9, Sep 1985, Pages:1011 - 1015

[78] Gharavi, H.; Mills, M.; “Block matching motion estimation algorithms-new results” *IEEE Transactions on Circuits and Systems*, Volume: 37, Issue: 5, May 1990, Pages:649 - 651

[79] Lengwehasarit, K.; Ortega, A.; “Probabilistic partial-distance fast matching algorithms for motion estimation” *IEEE Transactions on Circuits and Systems for Video Technology*, Volume: 11, Issue: 2, Feb 2001, Pages:139 - 152

[80] Baek, Y.; Oh H. S.; Lee H. K.; “Block-matching criterion for efficient VLSI implementation of motion estimation” *Electronics Letters*, Volume: 32, Issue: 13, 20 June, 1996 Pages:1184 - 1185

[81] Chen M. J.; Chen L. G.; Chiueh T. D.; Lee Y. P.; “A new block-matching criterion for motion estimation and its implementation” *IEEE Transactions on Circuits and Systems for Video Technology,* Volume: 5, Issue: 3, June 1995, Pages:231 - 236

[82] Natarajan, B.; Bhaskaran, V.; Konstantinides, K.; “Low-complexity block-based motion estimation via one-bit transforms” *IEEE Transactions on Circuits and Systems for Video Technology*, Volume: 7, Issue:4, Aug. 1997, Pages:702 - 706

[83] Sauer, K.; Schwartz, B.; “Efficient block motion estimation using integral projections” *IEEE Transactions on Circuits and Systems for Video Technology*, Volume: 6, Issue: 5, Oct. 1996, Pages:513 - 518

[84] Lee X.; Zhan Y. Q.; “A fast hierarchical motion-compensation scheme for video coding using block feature matching” *IEEE Transactions on Circuits and Systems for Video Technology*, Volume: 6, Issue: 6, Dec. 1996, Pages:627 - 635

[85] Nam K. M.; Kim J. S.; Park R. H.; Shim Y. S.; “A fast hierarchical motion vector estimation algorithm using mean pyramid” *IEEE Transactions on Circuits*

and Systems for Video Technology, Volume: 5, Issue: 4 Aug. 1995, Pages:344 - 351

[86] Wang C. N.; Yang S. W.; Liu C. M.; Chiang T.; "A hierarchical decimation lattice based on N-queen with an application for motion estimation" *IEEE Signal Processing Letters*, Volume: 10, Issue: 8, Aug. 2003, Pages:228 - 231

[87] Gonzales C. A.; Yeo H.; Kuo C. J.; "Requirements for motion-estimation search range in MPEG-2 coded video" *IBM Journal of Research and Development* Volume 43, Number 4, 1999, Page(s):453-470

[88] J. Koga, et al. "Motion Compensated Interframe Coding for Video Conferencing" *Proceedings of the National Telecommunications Conference*, 1981, Pages:G5.3.1- 5.3.3

[89] Tham J. Y.; Ranganath, S.; Ranganath, M.; Kassim, A.A.; "A novel unrestricted center-biased diamond search algorithm for block motion estimation" *IEEE Transactions on Circuits and Systems for Video Technology,* Volume: 8, Issue: 4, Aug. 1998, Pages:369 - 377

[90] Chimienti, A.; Ferraris, C.; Pau, D.; "A complexity-bounded motion estimation algorithm" *IEEE Transactions on Image Processing,* Volume: 11, Issue: 4, April 2002, Pages:387 - 392

[91] Tourapis, A.M.; Au, O.C.; Liou, M.L.; "New results on zonal based motion estimation algorithms-advanced predictive diamond zonal search" *The 2001 IEEE International Symposium on Circuits and Systems, 2001. ISCAS 2001.* Volume: 5, 6-9 May 2001, Pages:183 - 186

[92] Hosur P. I.; Ma K. K.; "Motion Vector Field Adaptive Fast Motion Estimation" *2nd Int. Conference on Information, Communications and Signal Processing (ICICS '99), Proceedings,* November 7-10, 1999,

[93] Chiang T.; Sun H.; (ed.), *Optimized Reference Software for Coding of Audio-Visual Objects.* ISO/IEC JTC1/SC29/WG11 Coding of Moving Pictures and Associated Audio, Pattaya, Thailand N4554, December 2001.

[94] Tourapis A. M.; Au O. C.; Liou M. L.; "Predictive Motion Vector Field Adaptive search Technique (PMVFAST) - Enhancing Block Based Motion Estimation." *Proceedings of Visual Communications and Image Processing 2001 (VCIP-2001).*

[95] Wiegand, T.; Schwarz, H.; Joch, A.; Kossentini, F.; Sullivan, G.J.; "Rate-constrained coder control and comparison of video coding standards" *IEEE*

Transactions on Circuits and Systems for Video Technology, Volume: 13, Issue: 7, July 2003, Pages:688 - 703

[96] Koc, U.-V.; Liu, K.J.R.; “DCT-based motion estimation” *IEEE Transactions on Image Processing*, Volume: 7, Issue: 7, July 1998, Pages:948 - 965

[97] Choi S.-Y.; Chae S.-I.; “Hierarchical motion estimation in Hadamard transform domain” *Electronics Letters* , Volume: 35, Issue:25, 9 Dec. 1999, Pages:2187 - 2188

[98] Wien, M.; “Variable block-size transforms for H.264/AVC” *IEEE Transactions on Circuits and Systems for Video Technology*, Volume: 13, Issue: 7, July 2003, Pages:604 - 61

[99] Broida T. J.; Chellappa R.; “Estimation of object motion parameters from noisy images Source” *IEEE Transactions on Pattern Analysis and Machine Intelligence* Volume 8, Issue 1 (January 1986), Pages:90 - 99

[100] Hoetter M.; “Differential estimation of the global motion parameters zoom and pan” *Signal Processing*, vol. 16, Pages:249-265, March 1989.

[101] Cheng S.-C.; Hang H.-M.; “A comparison of block-matching algorithms mapped to systolic-array implementation” *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 7, Issue: 5, Oct. 1997 Pages:741 - 757

[102] 3GPP Technical Specification 3GPP ETSI TS 126140 V6.2.0, *Multimedia Messaging Service (MMS); Media Formats and Codecs*. http://www.3gpp.org/ftp/Specs/html-info/26140.htm

[103] MPEG Industry Forum; “*Levels for MPEG-4 Visual Profiles*” Available: http://www.m4if.org/resources/profiles/

[104] Tomic, M.; Stojkovic, N.; Kovac, M.; “Implementation and analysis of MPEG-4 dynamic resolution conversion” *Proceedings of the 3rd International Symposium on Image and Signal Processing and Analysis, ISPA 2003*. Vol.1, Page(s):317 - 322

[105] Sullivan G. J.; Topiwala P.; Luthra A;. "The H.264/AVC Advanced Video Coding Standard: Overview and Introduction to the Fidelity Range Extensions," *SPIE Conference on Applications of Digital Image Processing XXVII*, Volume 5558, part 1, Aug. 2004. Pages:454-474

[106] Haskell B.; Singer D.; “Addition of Alpha Channel to AVC/H.264 FRext” Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG Document JVT-L013 Available: http://ftp3.itu.int/av-arch/jvt-site

[107] Stolberg, H.-J.; Moch, S.; Friebe, L.; Dehnhardt, A.; Kulaczewski, M.B.; Berekovic, M.; Pirsch, P.; “An SoC with two multimedia DSPs and a RISC core for video compression applications” *IEEE International Solid-State Circuits Conference, 2004. Digest of Technical Papers. ISSCC. 2004,* 15-19 Feb. 2004 Page(s):330 - 531 Vol.1

[108] Rohrer, N.J.; Canada, M.; Cohen, E.; Ringler, M.; Mayfield, M.; Sandon, P.; Kartschoke, P.; Heaslip, J.; Allen, J.; McCormick, P.; Pfluger, T.; Zimmerman, J.; Lichtenau, C.; Werner, T.; Salem, G.; Ross, M.; Appenzeller, D.; Thygesen, D.; “PowerPC 970 in 130 nm and 90 nm technologies” *IEEE International Solid-State Circuits Conference, 2004. Digest of Technical Papers. ISSCC. 2004* 15-19 Feb. 2004 Page(s):68 - 69 Vol.1

[109] Khailany, B.; Dally, W.J.; Kapasi, U.J.; Mattson, P.; Namkoong, J.; Owens, J.D.; Towles, B.; Chang, A.; Rixner, S.; “Imagine: media processing with streams” *IEEE Micro,* Volume 21, Issue 2, March-April 2001 Page(s):35 - 46

[110] Wawrzynek, J.; Asanovic, K.; Kingsbury, B.; Johnson, D.; Beck, J.; Morgan, N.; “Spert-II: a vector microprocessor system” *Computer* Volume 29, Issue 3, March 1996 Page(s):79 - 86

[111] Dudek, P.; Hicks, P.J.; “An analogue SIMD focal-plane processor array” *The 2001 IEEE International Symposium on Circuits and Systems, 2001. ISCAS 2001.* Volume 4, 6-9 May 2001 Page(s):490 - 493

[112] Chua L.O.; Yang L.; “Cellular Neural Networks: Theory.” *IEEE Transactions on Circuits and Systems.* 1988, Vol. 35, Page(s):1257-1272

[113] Roska T.; Chua L. O.; “The CNN Universal Machine: An Analogic Computer.” *IEEE Transactions on Circuits and Systems-II.* 1993, Vol. 40, Page(s):163-146

[114] Paasio A.; *Integration of Cellular Nonlinear Network Universal Machine,* Doctor’s Thesis, Helsinki University of Technology, January 8, 1999, 118p.

[115] Kananen A.; Paasio A.; Laiho M.; Halonen K.; “CNN Applications from the Hardware Point of View: Video Sequence Segmentation.” *International Journal on Circuit Theory and Applications.* Vol. 30, , 2002, Page(s):117-137

[116] Tezlaff R.; et. al.; “Analysis of Brain Electrical Activity in Epilepsy with Cellular Neural Networks (CNN)” *Proc. of the European Conference on Circuit Theory and Design, ECCTD99,* 1999, Page(s):1007-1010

[117] Espejo S.; Carmona R.; Dominguez-Castro R.; Roriguez-Vasquez A.; “A VLSI-Oriented Continuous-Time CNN Model” *International Journal on Circuit Theory and Applications.* Vol. 24, 1996, Page(s):1007-1010.

[118] Paasio A.; Kananen A.; Halonen K. “A QCIF Resolution Binary I/O CNN-UM Chip.” *Journal of VLSI Signal Processing.* Vol. 23, No 2/3, 1999, Page(s):281-290.

[119] Harrer, H.; Nossek, J.A.; Stelzl, R.; “An analog implementation of discrete-time cellular neural networks” *IEEE Transactions on Neural Networks*, Volume: 3 , Issue: 3 , May 1992 Pages:466 - 476

[120] Raffo, L.; Sabatini, S.P.; Bo, G.M.; Bisio, G.M.; “Analog VLSI circuits as physical structures for perception in early visual tasks” *IEEE Transactions on Neural Networks*, Volume: 9 , Issue: 6 , Nov. 1998 Pages:1483 - 1494

[121] Shi, B.E.; Chua, L.O.; “Resistive grid image filtering: input/output analysis via the CNN framework” *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications,* Volume: 39, Issue: 7, July 1992, Pages:531 - 548

[122] Zarandy, A.; Keresztes, P.; Roska, T.; Szolgay, P.; “An emulated digital architecture implementing the CNN Universal Machine” *1998 Fifth IEEE International Workshop on Cellular Neural Networks and Their Applications Proceedings*, 14-17 April 1998 Pages:249 - 252

[123] Zarándy Á.; Werblin F.; Roska T.; Chua L. O.:“Spatial logic algorithms using basic morphological analogic CNN operations” *International Journal of Circuit Theory and Applications* Vol.24, Issue 3, 1996, Page(s):283-300

[124] Venetianer, P.L.; Werblin, F.; Roska, T.; Chua, L.O.; “Analogic CNN algorithms for some image compression and restoration tasks” *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, Vol.42, Issue 5, May 1995 Page(s):278 - 284

[125] CNN Software Library (CSL). Available: http://lab.analogic.sztaki.hu/Candy/csl.html

[126] Kananen, A.; Laiho, M.; Halonen, K.; Paasio, A.; “Nx16 cellular test chips for low-pass filtering large images” *Proceedings of the 2004 International Symposium on Circuits and Systems, 2004. ISCAS ’04.* Volume 5, 23-26 May 2004 Page(s):V-461 - V-464

[127] Fugunaka S.; Nakaya; Y.; Son S. H.; Nagumo T.; (ed.), *MPEG-4 Video Verification Model version 16.0.* ISO/IEC JTC1/SC29/WG11 Coding of Moving Pictures and Associated Audio, Noordwijkerhout, Netherlands N3312, March 2000.

[128] Yoon S. C.; Ratakonda, K.; Ahuja, N.; "Region-based video coding using a multiscale image segmentation" *Proceedings, International Conference on Image Processing, 1997*. Vol.2, Page(s):510 - 513

[129] Cheng S.-C.; "Visual pattern matching in motion estimation for object-based very low bit-rate coding using moment-preserving edge detection" *IEEE Transactions on Multimedia*, Volume 7, Issue 2, Apr 2005 Page(s):189 - 200

[130] Hui Tian; Fowler, B.; Gamal, A.E.; "Analysis of temporal noise in CMOS photodiode active pixel sensor" *IEEE Journal of Solid-State Circuits*, Volume 36, Issue 1, Jan. 2001 Page(s):92 - 101

[131] Al-Shaykh, O.K.; Mersereau, R.M.; "Lossy compression of noisy images" *IEEE Transactions on Image Processing*, Volume 7, Issue 12, Dec. 1998 Page(s):1641 - 1652

[132] Ohira T.; Kamemaru H.; Suzuki K.; Asano M.; Yoshimoto M.; "A Low Power Media Processor Core Performable CIF30 fr/s MPEG4/H26x Video Codec " *IEICE Trans. Electron.* Vol.E84-C No.2 Page(s):157 - 165

[133] Zhou Z.; Sun M.-T.; Hsu Y.-F.; "Fast variable block-size motion estimation algorithm based on merge and slit procedures for H.264/MPEG-4 AVC" *Proceedings of the 2004 International Symposium on Circuits and Systems, 2004. ISCAS '04*. Vol.3, Page(s):III - 725-8

[134] Kuo T.-Y; Chan C.-H.; "Fast macroblock partition prediction for H.264/AVC" *IEEE International Conference on Multimedia and Expo, ICME '04.* Vol.1, Page(s):675 - 678

[135] Rath, G.B.; et al. "Subblock Matching-Based Conditional Motion Estimation with Automatic Threshold Selection for Video Compression" *IEEE Trans. on Circ. and Syst. for Video Technology*. Vol. 13, 2003 pp. 914 -924

[136] Dubiusson S., Davoine, F.; "Motion Compensation Using Adaptive Rectangular Partitions" *Proc. on the Int. Conf. on Image Processing ICIP 99.* Vol. 1, pp. 56 -60

[137] Kim M.; Hwang H.; Chae S.-I.; "A fast VLSI architecture for full-search variable block size motion estimation in MPEG-4 AVC/H.264" *Proceedings of the Asia and South Pacific Design Automation Conference, ASP-DAC 2005*. Vol.1, Page(s):631 - 634

[138] Horowitz, M.; Joch, A.; Kossentini, F.; Hallapuro, A.; "H.264/AVC baseline profile decoder complexity analysis" *IEEE Transactions on Circuits and Systems for Video Technology*, Vol.13, Issue 7, July 2003, Page(s):704 - 716

[139] Flak J.; Laiho M.; Paasio A.; Halonen K.; “VLSI Implementation of a Binary CNN: First Measurement Results”, *Proc. of CNNA 2004* Page(s):129-134,

[140] Laiho, M.; Paasio, A.; Flak, J.; Halonen, K.; “Template Design for Binary-Programmable Cellular Nonlinear Networks” *IEEE International Symposium on Circuits and Systems, ISCAS 2005.* Page(s):3938 - 3941

[141] H.-Y.C.Tourapis et al.; “Fast Motion Estimation within the H.264 codec”, *Proc. of the 2003 Int. Conf. on Multimedia and Expo*, ICME03, pp. III 517 - III 520

[142] Hui K.; Chan Y.; Siu W.; “Priority Search Technique for MPEG-4 Motion Estimation of Arbitrarily Shaped Video Object” *Proceedings on the 2001 International Conference on Image Processing*, 2001, Vol. 3 , pp. 644 -647, 2001

[143] Dudek, P.; Hicks, P.J.; “A general-purpose processor-per-pixel analog SIMD vision chip” *IEEE Transactions on Circuits and Systems I: Regular Papers,* Volume 52, Issue 1, Jan. 2005 Page(s):13 - 20

[144] Ruedi, P.-F.; Heim, P.; Kaess, F.; Grenet, E.; Heitger, F.; Burgi, P.-Y.; Gyger, S.; Nussbaum, P.; “A 128 x 128 pixel 120-dB dynamic-range vision-sensor chip for image contrast and orientation extraction” *IEEE Journal of Solid-State Circuits,* Volume 38, Issue 12, Dec 2003 Page(s):2325 - 2333

[145] Tummala, R.R.; “SOP: what is it and why? A new microsystem-integration technology paradigm-Moore’s law for system integration of miniaturized convergent systems of the next decade” *IEEE Transactions on Advanced Packaging*, Volume 27, Issue 2, May 2004 Page(s):241 - 249

[146] Pelgrom, M.J.M.; Duinmaijer, A.C.J.; Welbers, A.P.G.; “Matching properties of MOS transistors” *IEEE Journal of Solid-State Circuits*, Volume 24, Issue 5, Oct 1989 Page(s):1433 - 1439

[147] Toumazou, C.; Hughes, J.B.; Pattullo, D.M.; “Regulated cascode switched-current memory cell” *Electronics Letters* Volume 26, Issue 5, 1 March 1990 Page(s):303 - 305

[148] Dudek P.; *A Programmable Focal-Plane Analogue Processor Array*, Dissertation for the degree of Doctor of Philosophy, University of Manchester, May 2000.

[149] Waltari, M.; *Circuit Techniques for Low-Voltage and High-Speed A/D Converters*. Dissertation for the degree of Doctor of Science (ISBN 951-22-5908-7) Available: http://lib.tkk.fi/Diss/2002/isbn9512259087/

[150] Baturone I.; et. al. “Implementation of CMOS Fuzzy Controllers as Mixed-Signal Integrated Circuits” *IEEE Transactions on Fuzzy Systems.* Volume. 5, 1997, Page(s):1-19

[151] Bult, K.; Wallinga, H.; “A class of analog CMOS circuits based on the square-law characteristic of an MOS transistor in saturation” *Int. J. Solid-State Circ.*, Volume 22, No. 3, 1987, Page(s):357 -365

[152] Laiho M.; Paasio A.; Kananen A.; Halonen K.; “Realization of Couplings in a Polynomial Type Mixed-Mode CNN” *Proceedings of the 2002 7th IEEE International Workshop on Cellular Neural Networks and Their Applications, 2002. (CNNA 2002).* pp. 436-443

[153] Laiho M.; *Mixed-Mode Cellular Array Processor Realization for Analyzing Brain Electrical Activity in Epilepsy.* Dissertation for the degree of Doctor of Science (ISBN 951-22-6598-2) Available: http://lib.tkk.fi/Diss/2003/isbn9512265982/

[154] Laiho M.; Paasio A.; Kananen A.; Halonen K.; “Realization of Couplings in a Polynomial Type Mixed-Mode Cellular Neural Network” *International Journal of Neural Systems*, Volume 13, No. 6, 2003 Page(s):443-452

[155] Hughes, J.B.; Moulding, K.W.; “S^2I: a switched-current technique for high performance” *Electronics Letters* , Volume: 29 , Issue: 16 , 5 Aug. 1993 Page(s):1400 - 1401

[156] Laiho, M.; Paasio, A.; An “Analogue-To-Digital Converter, a Current Scaler and a Method for Controlling a Function of the Current Scaler” Patent application WO03069782. Available: http://fi.espacenet.com/

[157] Kananen, A.; Paasio, A.; Halonen, K.; “Spatial Low-Pass Filtering and Gradient Calculation Blocks for Fast Image Analysis” Submitted to *IEEE Transactions on Circuits and Systems I*

This page is intentionally left blank.

Appendix A

Chip Layout

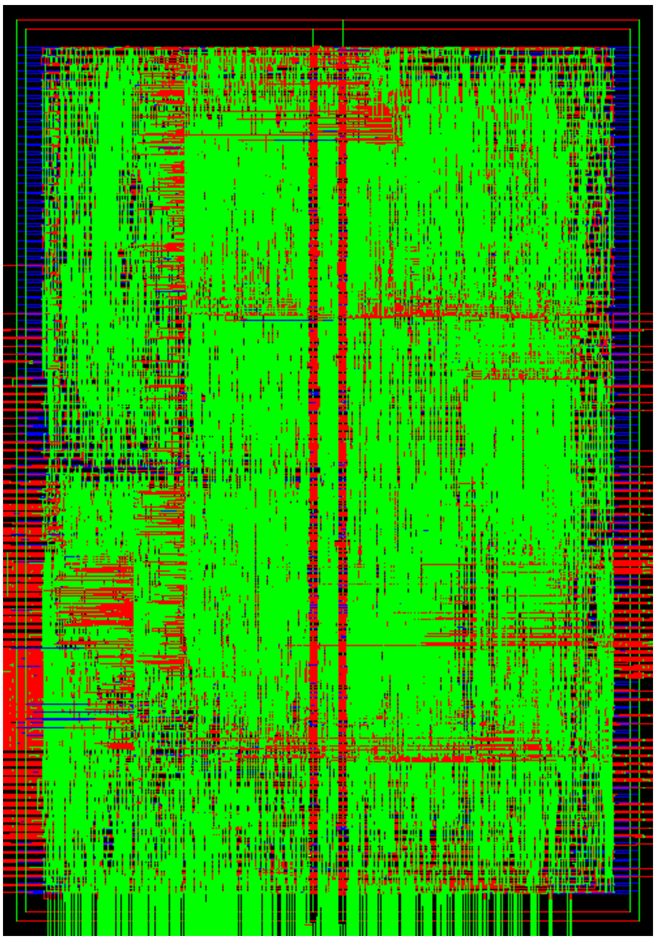


Figure A.1 Layout of the integrated chip presented in Chap. 5.5.

This page is intentionally left blank.

Appendix B

Chip Microphotograph

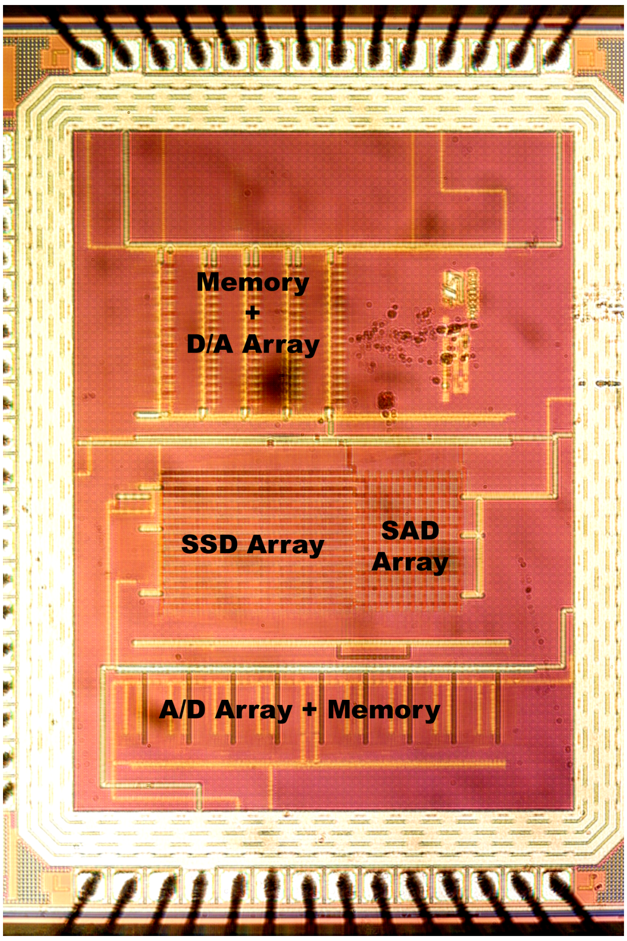


Figure B.1 Microphotograph of the integrated chip presented in Chap. 7.6