Complexity scalable MPEG encoding

Complexity Scalable MPEG Encoding

Stephan Mietens

CIP-DATA LIBRARY TECHNISCHE UNIVERSITEIT EINDHOVEN

Mietens, Stephan O.

Complexity scalable MPEG encoding / by Stephan O. Mietens. - Eindhoven : Technische Universiteit Eindhoven, 2004.
Proefschrift. - ISBN 90-386-2040-3
NUR 992
Trefw.: complexiteit van berekening / beeldcodering / codering.
Subject headings: computational complexity / video coding / encoding.
Schlagwörter: Skalierbarkeit / MPEG / Videokompression / Encoding.

Complexity Scalable MPEG Encoding

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van de
Rector Magnificus, prof.dr. R.A. van Santen, voor een
commissie aangewezen door het College voor
Promoties in het openbaar te verdedigen op
woensdag 18 februari 2004 om 16.00 uur

door

Stephan Oliver Mietens

geboren te Frankfurt am Main, Duitsland

Dit proefschrift is goedgekeurd door de promotoren:

prof.dr.ir. P.H.N. de With
en
prof.dr.ir. R.L. Lagendijk

Copromotor:
Prof. Dr.-Ing. habil. C. Hentschel

ISBN 90-386-2040-3

to my family

Contents

CHAPTER 1

Introduction

This chapter forms an introduction to the research that aims at realizing computational complexity scalability for MPEG encoding. The chapter commences with the general problem statement of scalable MPEG processing: i.e. a scalable range of picture quality with corresponding resource usage of the system. Resource usage is expressed in terms of computation power, memory and communication bandwidth. Additional sections present an exemplary flexible and programmable hardware architecture showing the concept of scalable video processing, the principles of scalable system design, and a brief description of the MPEG coding standard, in order to deepen the insights of the problem statement. Furthermore, this chapter summarizes the subsequent individual chapters and presents their publication history.

1.1 Problem statement

1.1.1 Context

The current burgeoning market for consumer digital video applications has lead to widely accepted consumer products based on MPEG coding, such as

DVD players, set-top boxes and digital video cameras. This thesis adds one extra dimension to MPEG coding, namely complexity scalable coding. Scalability offers the advantage of video algorithms that can adapt their output quality with the purpose of accordingly adapting their resource usage (like computation power or memory) to the actually resources available for the application.

One of the most popular existing applications of MPEG-2 coding is the storage of movies in compressed form on the so-called Digital Versatile Disc (DVD), which provides a real-time digital MPEG video playback system. The MPEG encoding process for creating a DVD is not time-critical and can be performed with highly complex algorithms for achieving high video quality. Until today, research on MPEG-based video compression focused primarily on maximizing the video quality for common applications such as broadcasting digital TV and DVD with a data rate of 3-8 Mbit/s, and on finding efficient implementations in hardware and software.

However, today's digital video applications have an ever-increasing range of quality requirements that exceed broadcasting digital TV and DVD. Figure 1.1 shows some examples of such extended applications operating on well-known devices. In the figure, a multi-window TV set shows a high-quality video output (decoding) from a DVD player, while simultaneously performing a video-conferencing application with medium video quality (encoding and decoding). Furthermore, the TV set is connected via a wireless channel with a mobile phone and a PDA that process video data at low quality. All these applications satisfy different constraints such as real-time execution in a software implementation, while the available system resources like computation power or memory should be well balanced.

Figure 1.1: *Multimedia devices that can execute video applications (e.g. video conferencing) while sharing available resources.*

Furthermore, future consumer terminals will combine in a flexible way high-quality video and audio with applications from the multi-media PC domain (e.g. web cameras and video editing), as found on PCs. The need for flexibility is not supported by many current products that use dedicated hardware and are thus designed for a specific application at fixed quality. Due to their inflexibility, they are for example not upgradeable for new services. To overcome this limitation, flexible software solutions running on programmable architectures are needed. In Section 1.2, an example of an emerging programmable video-processing platform [1, 2, 3] is presented in more detail, along with an outline about its programmability on the functional level and applications demonstrating the flexibility of the architecture [4]. The disadvantage of programmable architectures is that they are more expensive in hardware cost, and their power consumption is initially larger, as compared to dedicated hardware. This thesis will however open-up new ways to control the cost and the quality of SW-based algorithms.

Scalable video algorithms (SVAs) [5] that are embedded in a flexible framework provide dynamic adaptation of the resource usage to the current needs [6]. With this property, costly system redesign after changing the target architecture for a video application may become history. In addition, SVAs can optimize the resource sharing in a multi-tasking environment. The benefit of SVAs can be explained with Figure 1.1. For example, the TV set shows three different video applications, namely a DVD playback in the screen background, a TV broadcast in the bottom-right and a video conferencing in the top-left part of the screen. If the applications are based on SVAs, they may share the available resources of the TV set, and they are able to adapt their quality and resource usage according to the user focus, or they may adapt to a change in the number of running applications. Bringing the situation to mind when the user received the video call and the conference application was initiated as the third application to be executed, one can see that when using SVAs, the other applications can still be executed, albeit at lower quality. If no SVAs are used, the TV set would need to be prepared for worst-case situations with high total costs, or it could not provide this combined functionality. In Section 1.3, the design of a scalable system is presented in more detail.

1.1.2 Terms and system requirements

The development of SVAs and the design of scalable systems using SVAs was the goal of a project at Philips Research, The Netherlands. The research described in this thesis is part of this project. The objective is to design a scalable

MPEG encoding algorithm that features scalable video quality and resource usage, as described above, with respect to the desired application. New algorithms and/or techniques have been investigated to reduce the complexity of MPEG coding, while still preserving high quality. The most important requirement is that the scalable quality should match with the necessary computational resources, such that a lower quality leads to less required resources accordingly. In the following, this problem is addressed as *scalability of the computational complexity*. Note that the computational complexity relates to the amount of executed operations rather than to the simplicity of algorithms.

The development of the scalable algorithms presented in this thesis is driven by the following desires. On the one hand, a large scalability range should be provided. On the other hand, when having sufficient resources, the scalable algorithms should achieve the *same* quality as of comparable state-of-the-art algorithms that do not provide scalability. At the same time, any overhead in resource usage should be minimized. For example, this overhead is the need of additional memory to keep intermediate results for later reuse, in order to prevent multiple recomputation of the same operation.

In addition to the computational power, *memory* aspects are also considered in this thesis. Memory costs are another important factor for a system design. Aligned with the above-given definition of the computational complexity, memory aspects like the amount of memory (memory requirements) or the access to the memory (bandwidth requirements) can be seen as memory complexity. Memory complexity scalability is only partially addressed in this thesis.

In an example scalability research project [7] (see also Section 1.3), the target architectures that are considered for a scalable system design are generalized programmable architectures, for the purpose of flexibility. For this reason, the computational complexity as addressed in this thesis, is measured with e.g. the number of additions and multiplications that are required to perform the core functionality of the (developed scalable) algorithms. For a convenient comparison with existing algorithms, other implementation costs (like e.g. memory accesses) are disregarded or seen as being constant. Of course, when selecting (limiting to) a certain architecture, architecture-specific properties can be considered for the development and the implementation of (scalable) algorithms. Note that this is not the focus of this thesis. Furthermore, scalability is not limited to generalized and programmable architectures. For

example, dedicated hardware may result in reduced chip size when using scalability techniques. This aspect is partially considered in an appendix of this thesis.

Another serious constraint for the research on scalability was to maintain bitstream compatibility with the MPEG coding standard. This may not be necessarily obvious, because a specialized encoder-decoder pair may offer advanced scalability options. However, MPEG incompatibility would limit the wide applicability of the scalable encoder, which is not desirable.

1.1.3 Approaches towards MPEG complexity scalability

At this point, we assume that the reader is familiar with the basics of MPEG coding. Otherwise, the reader is referred to Section 1.4, where the MPEG video compression standard is briefly described, such that the architecture and the processing blocks are sufficiently known for understanding of this thesis.

There are several meanings of the term "scalability" in the MPEG video domain. The MPEG-2 scalability modes concentrate on the scalability of a video stream for transmission and decoding. Complexity scalability as addressed above is focused on the computational effort of the encoder. For this purpose, the ability of the MPEG core functions to complexity scalability is exploited. In MPEG encoding, computational complexity scalability can be applied at the frame level by modifying the GOP structure, because the amount of motion estimation increases for I-, P- and B-frames. This approach can be basically applied to any MPEG encoding system, without the need to change the system itself, and will therefore not be discussed further. In the sequel, the focus is on the core processing modules at the macroblock level.

For the MPEG core modules, the primary interest is on complexity scalability techniques for the algorithms that are represented by the modules. This is advantageous because improvements can be applied to various types of hardware architectures. Architecture-specific optimizations of the modules can be made after selection of a target architecture. An example to exploit features of a RISC processor for obtaining an efficient implementation of an MPEG coder is given in [8]. In this thesis, the following scalability techniques are developed for the MPEG core modules.

- **Discrete Cosine Transformation (DCT)**
  Existing DCT algorithms can be analyzed to find a specific computation

order of the DCT coefficients. The computation order maximizes the number of coefficients that can be computed at a given limited amount of computation resources.

- **Quantization**
  Since the DCT computes a subset of all possible coefficients, the quantization is scaled with the scalable DCT by preselecting coefficients for processing. The intrinsic computational complexity of quantization is not high.

- **Inverse DCT and inverse quantization**
  Coefficients that are not computed by the DCT do not have to be processed by the modules performing the inverse DCT (IDCT) and inverse quantization. If these modules are implemented in this way, they are scaled with the DCT. Note that both modules should have the same result as their unscaled pendants in order to be compliant with the MPEG standard, and therefore avoid error drift at the receiver side. For this reason, previous work on scalability of the IDCT at the receiver side [9] cannot be directly applied to the computation of the IDCT in an encoder.

- **Motion Estimation (ME)**
  It will be shown later that motion vector fields needed for MPEG coding can be generated by multi-vector-field approximations from high-quality frame-to-frame ME. Furthermore, the accuracy of the approximations can be made scalable. Finally, the refinement may be omitted or kept very simple.

- **Variable Length Coding (VLC)**
  The computational effort is scalable with the number of non-zero coefficients that are released by the quantizer.

1.1.4 Structure of the conducted research

The following chapters present the development of scalability techniques for the DCT and the ME in detail, including partial literature overviews and preliminary comparisons that have lead to a selection of algorithms that were used for further research. The DCT and the ME have been chosen, because the compression of video sequences provided by MPEG coding is significantly based on these two core modules and they require substantial computational effort. Additionally, the scalable DCT and ME are inserted into a scalable MPEG encoder and the overall system behavior is investigated.

The potentials for optimizations towards complexity scalability for DCT and ME are higher than for the other modules, which can be seen from the following. For example, the DCT is normally computing all 64 coefficients per 8×8 picture block and subsequent quantization removes the major number of DCT coefficients. Ideally, only those coefficients that remain after quantization should be computed, because the computations that are spent for the other coefficients are wasted. However, which coefficients will remain is of course not known in advance. Some optimizations can be made though. Using a selection of DCT algorithms as presented in Chapter 2, a scalable DCT computation technique is developed in Chapter 3 to optimize the number of computed coefficients under computation power limitations. The technique can include priority weighting of the coefficients to e.g. consider subsequent quantization. Later in Chapter 6, among other experiments with the scalable MPEG system, the scalable DCT technique is combined with block classification (see Section 5.4) such that the computations for the scalable DCT can be further concentrated on coefficients that are likely to be required for describing the picture block content.

The ME has a variety of possibilities in computing MV fields for the encoding process. To name a few, the ME can consider a different number of candidate MVs before selecting a best MV for a macroblock, or different strategies in ordering the selection and evaluation of candidate MVs, or the reusing of previously generated MV fields. After comparing a number of ME algorithms in Chapter 4 and selecting a basic ME algorithm for further research, two scalable ME techniques are developed in Chapter 5. The first technique is designed such that a simple frame-by-frame ME can be performed for then fast approximating and optional refining MPEG MV fields. The second technique exploits block classification (see Section 5.4) in order to concentrate computations for MV candidates that are more likely to result in a good description of the motion than other candidates.

In contrast with the DCT and the ME, the quantization processes single coefficients and consists of, roughly spoken, mainly one division and rounding. A rather obvious and simple scaling of the quantization could be to simplify or leaving out the rounding for scaling up the processing speed. Given the low intrinsic complexity of quantization, this is not further worked out. The VLC is a rather straightforward process that generates the bitstream using look-up tables, which does not leave much space for applying computational complexity scalability. Preliminary experiments for speeding up the processing of the DCT coefficients did not lead to useful results. A part of VLC is rate control,

of which the computational complexity depends the amount of quantized coefficients for obtaining a predefined bit rate. The control is usually executed at much lower frequency than the sampling rate, yielding a low complexity. Scaling the bitstream after VLC by making a trade-off between coded coefficients and coding efficiency at the expense of additional computational effort is presented in [10, 11]. Such techniques, like bitstream scaling, are not considered, because they are out of the focus of this thesis.

The remainder of this chapter has two parts. In the first part, an exemplary programmable hardware platform (see Section 1.2), the basics of scalable system design (see Section 1.3) and the MPEG-2 video coding standard (see Section 1.4) are briefly described. These sections provide more insight of the problem statement and are optional for the interested readers. The final part of this chapter summarizes the subsequent chapters of this thesis (see Section 1.5) and describes the scientific background of the individual chapter contents and their publication history (see Section 1.6).

1.2 Programmable hardware

In order to become familiar with the amount and varying complexity of video processing tasks, the author contributed to the design of an experimental video-processing platform. The contribution involved the programming of a dynamic multi-window application, in which at least one window displaying a video signal is varied in size, position and shape. This multi-window TV application requires resources that scale with the chosen window configuration. Moreover, the platform should process several video streams in parallel, thereby sharing the available resources. Because these experiments relate to the scalability aspects discussed in this thesis, this section briefly describes these preliminary studies.

1.2.1 Emerging flexible architectures

Several programmable hardware architectures for video processing have been proposed in the past years. Earlier solutions are not cost-effective, because they are more general than the TV application domain requirements [12], or do not offer sufficient parallelism to cover all processing requirements [13]. A recently developed experimental video display-processing platform (VDPP) is presented in [1, 2, 3]. This platform is depicted in Figure 1.2 and contains the following three subsystems.

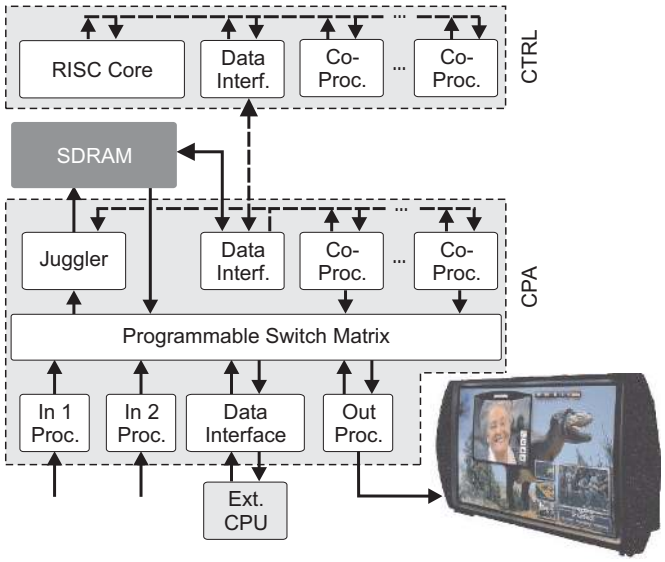

Figure 1.2: *Architecture of video processing platform (based on [1]).*

- The control subsystem (CTRL) is responsible for control-oriented tasks and contains a general-purpose RISC core and an optional number of coprocessors for frequently used tasks.

- The signal-processing subsystem is responsible for real-time video processing. It contains a coprocessor array (CPA) that provides a number of flexible application-specific coprocessors for different video-processing functions, like scaling, noise reduction, video mixing, graphics blending, etc. They are interconnected via a programmable switch matrix, suited for making any data path between the coprocessors available. This enables simple signal flow-graph designs that are applied for several video streams to be processed in parallel.

- The memory subsystem (SDRAM) handles the memory requests from the control and signal-processing subsystems.

In addition, a generic data interface is provided with the VDPP for connecting external CPUs, and therefore allow future functionality extensions.

1.2.2 Signal flow-graph programming

The programmable switch matrix of the signal-processing subsystem is an autonomous communication network that is able to redirect video-data packets between the coprocessors as required, without the need of bandwidth-costly accessing of the external memory subsystem, if not required by a video function. Figure 1.3 portrays an example of how the VDPP is programmed on a functional level for a special multi-window TV application. The solid arrows in the figure indicate video data and dashed lines indicate control data (here computing of window shapes). The depicted signal flow-graph represents a special case where a part of the picture is selected by the consumer for zooming and is shown in the background, while the original picture is displayed as a programmable Picture-in-Picture (PiP). An example for this TV application is given in Figure 1.4.

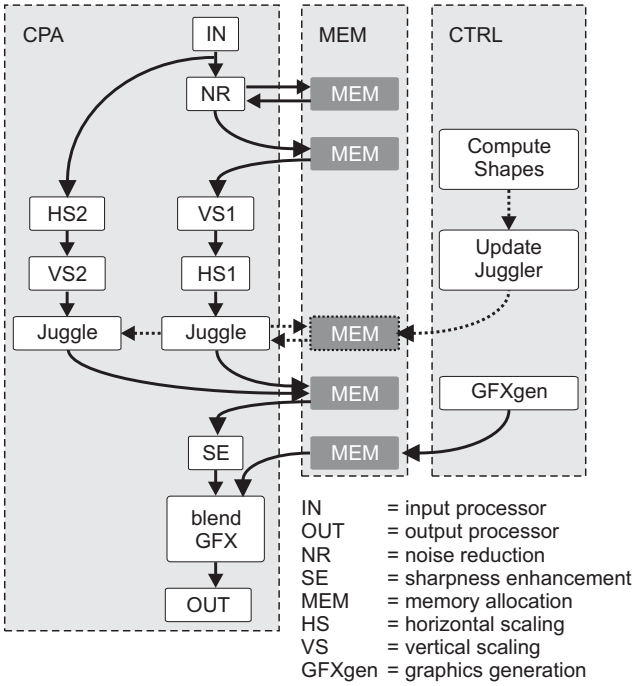

Figure 1.3: *Example of a two-window TV signal flow-graph (from [1]).*

The flow-graph in Figure 1.3 shows that an input video stream (solid arrows) is split into two streams, which are scaled horizontally (HS) and vertically

Figure 1.4: *Example of a PiP feature (based on [1]).*

(VS) and processed differently (noise reduction (NR) is applied on only one stream). Afterwards, the streams are combined in the mixing unit (called juggler, see Figure 1.2), by skipping video data that is overlapped by another video stream. Hence, an important advantage of the juggler is that only the visible parts of the incoming video streams are selected prior to storage in memory, so that the required memory-access bandwidth per frame is limited to only one full-resolution frame of a video stream for reading and writing.

The juggler is re-programmable on video field frequency (50 Hz in Europe) by the embedded RISC core inside the CTRL block, in order to obtain dynamic effects (see Section 1.2.3). Finally, the output stream is blended with graphics (GFX) after sharpness enhancement (SE). The indicated memory accesses are primarily for video stream de-interlacing, for vertical scaling and also intermediate and temporal buffering, because different data rates are generated by different scaling factors. A separate memory access, indicated with dotted lines, stores the computed parameters for composing the video lines of independent video streams. These values are addressed by the juggler. Other forms of programming are not indicated.

The created applications are layered in three levels. At the system level, the window shapes and signal inputs are computed at field rate for generating

dynamic effects. At middle level, the signal flow-graph is programmed. At the hardware level, local parameter and signal control is evaluated and programmed. This flexible way of processing video data is a novel feature that is not available in conventional high-end TV systems.

1.2.3 Dynamic multi-window TV

A more complex application that has been successfully executed on the VDPP demonstrates a high flexibility. Window position, size and shape of the input video streams are interactively controlled by the consumer. For achieving sufficient interactivity, the previous parameters should be modifiable from frame to frame, in order to create dynamical effects.

In the first stage, the experimental application was produced by using a full software-emulation of the RISC core and simulating the complete VDPP. In the second stage, an experimental real-time video chip-set was successfully programmed [14]. In the examples as shown below, the generation of the multi-window features were identical in the simulation and when running on the experimental chip. Note that the application can be further enhanced by high-level programming of additional video functions to be performed on each window, such as color enhancement or image warping.

Examples of multi-window TV features

- **Moving windows**
  Figure 1.5 shows three different video streams, one in the background of the screen and two elliptical windows. The windows move in different directions and at different speeds and overlay each other, as indicated by the (added) arrows. For visualization of the movements, the window positions at the begin of the sequence have been marked with dashed ellipses and at the end with solid ellipses. Note that the arrows and ellipses could also be generated and displayed by the VDPP, using the graphics blending coprocessor.

- **Rotating coin**
  Figure 1.6 shows two video streams, where one stream is displayed in the front window with dynamically changing position, shape and size, such that a rotating coin is simulated. It is also possible to apply horizontal and vertical scaling to the video stream to shrink each frame to the current size of the coin for improving the rotating-coin effect.

Figure 1.5: *One frame of the "Moving windows" sequence.*

The conclusion of the preliminary experiments in this section is that special video applications such as multi-windowing, imply a variable amount of resources for processing. The fixed assignment of hardware to individual tasks becomes ineffective when these tasks are variable in size and change over time. Moreover, the visibility of details depends on the available display area size and for this reason, resources could also be adapted to the size of the display area. For cost-effectiveness of the system, the implemented video functions should be able to share resources such as computing power and memory.

1.3 Scalable system design

1.3.1 Motivation for embedded applications

This section elaborates further on scalable resource usage as discussed in the previous section. The VDPP applies scalability such that the size of the video windows is programmable and depends on e.g. settings and applications. Another example of a scalable TV application using the VDPP platform is full-quality TV watching while receiving a video conference call. Because the TV application is using all available resources, the conference application cannot start until the TV set frees shared system resources (CPU time, memory usage, etc.) and thus executes both applications in parallel. A key aspect is a careful control of the quality degradation that results from limiting the resources.

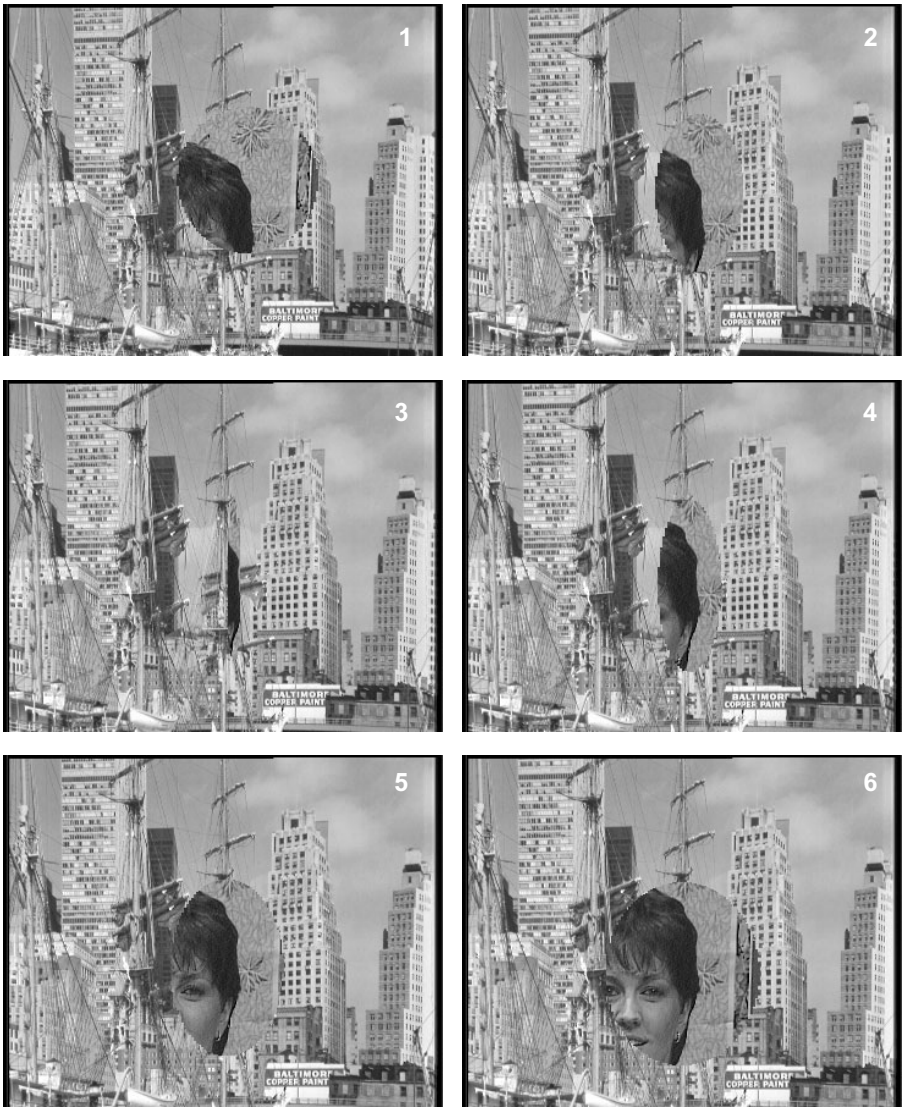

Figure 1.6: *Sequence of video frames taken from the "Rotating coin" sequence.*

In a generalized system concept using several video tasks in parallel, a scalable system design is understood from the following considerations. Firstly, the number of video tasks is varying and differs for each application. Secondly, the amount of video processing involved for a single task can also be rather different. As a result, it would become impractical to design optimized MPEG encoders for all classes of applications. Instead, our effort is concentrating on designing one algorithm that scales with the desired application, both in complexity and quality.

An application area where scalability is particularly interesting, is mobile video communication. In the case of mobile devices, a full processing of video signals such as found in TV sets is neither cost-effective nor even possible, since such devices have small displays and the observer cannot perceive fine details on such displays. This example asks for video processing that is scaled to the available resources and the display size that is used. We have found that portable equipment provides interesting cases for applying scalable algorithms. Besides scalability for portable equipment, severe system constraints have to be satisfied, such as power consumption, low system costs and reduced video format resolution.

1.3.2 Scalable systems using scalable video algorithms

Traditional systems do not support flexible run-time control of the resource usage in exchange for the resulting video quality (quality level) of an application. Figure 1.7 shows a range of programmable hardware product-families versus software-algorithm requirements. Programmable hardware equipped with a different amount of available resources exists to serve different market demands (see Figure 1.7(a)). Fixed software solutions for video applications are designed for maximum quality. Fixed solutions require resources that are available in high-end systems only, or that lead to limited functionality, depending on the number of algorithms that can run in parallel on the target architecture (Figure 1.7(b)).

A flexible system consists of software modules that contain scalable video algorithms (SVAs) [5] as shown in Figure 1.7(c). The SVAs may consist of a basic functionality at a minimum quality (dark areas) and scalable parts for improved quality (light areas). In this way, scalable algorithms are reusable for a variety of product-family members with different hardware resources.

Figure 1.8 shows an MPEG encoder using SVAs, where one module is enlarged and depicts an exemplary structure of an SVA. In this example, the

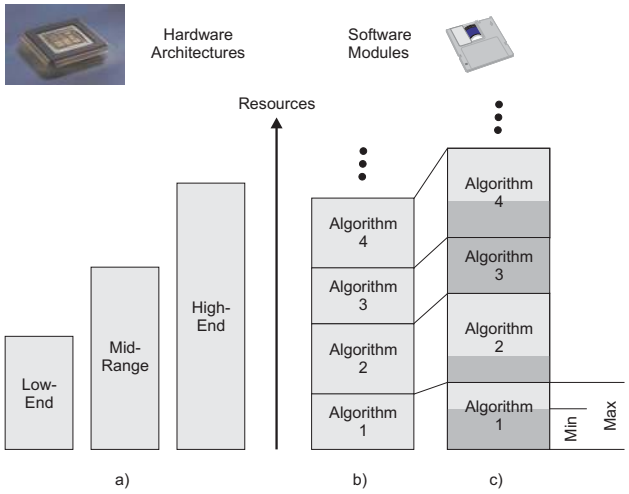

Figure 1.7: *Programmable hardware product-families and software-algorithm requirements (from [7]).*

SVA consists of four functions, of which the parameters are chosen by a quality controller. Not every function is necessarily controllable by parameters (Function 2 in this example). The resulting quality of the SVA depends on the parameter combinations of the functions. The quality controller is responsible for choosing the optimal parameter combination for highest achievable quality under given resource restrictions.

1.3.3 Overall system architecture with scalability

To ensure cost-effectiveness and flexibility of the system, SVAs are embedded in a framework containing resource and quality managers [15, 6] for dynamic resource management and quality control depending on momentary execution requirements. Figure 1.9 gives an overview of the system architecture of this framework.

The core components of the system are presented below.

- A modular set of SVAs is provided to enable different functionality as desired for a product (set-top box, multimedia PC, etc.).
- The strategy manager collects information about *application requirements* (e.g. the amount of resources needed) for all provided quality

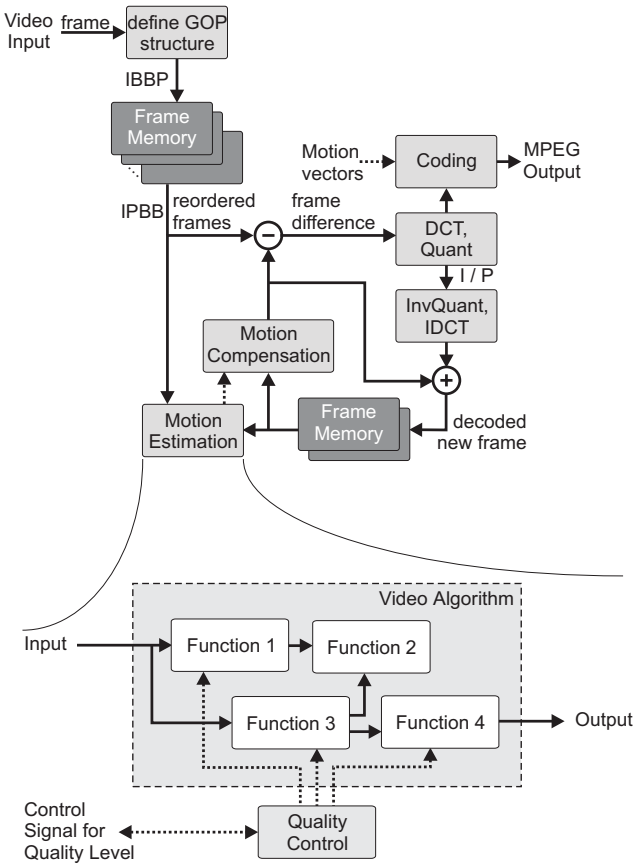

Figure 1.8: *Exemplary structure of a scalable video algorithm (based on [7]).*

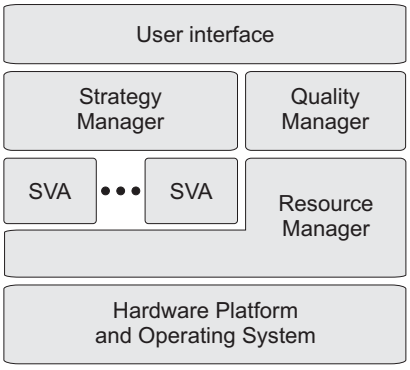

Figure 1.9: *Layered system architecture using scalable video algorithms (from [7]).*

levels and makes a strategy available to optimize the overall quality of a combination of applications.

- The quality manager optimizes the *system utilization* based on the utilization of the available applications, their importance and their estimated resource usage.
- The resource manager controls the *admission and scheduling* of the SVAs and their run-time resource needs.

The quality manager and the resource manager together constitute the Quality-of-Service resource manager (QoS-RM), which takes the responsibility for cost-effectiveness on the one hand and preservation of the overall system quality on the other hand. Let us briefly indicate the usefulness of a QoS-RM.

Figure 1.10 shows the tracking of the resource usage and the budget of two exemplary applications. The system load of application 1 is drawn from top to bottom, and vice-versa for application 2. It can be seen that application 2 exceeds its assigned budget three times. At occurrences 2A and 2B, the exceeding is granted by the QoS-RM, because the overall system utilization is below 100%. At occurrence 2C, the exceeding has to be rejected, because at the same time, application 1 has a peak in CPU load (occurrence 1B). Here, application 2 would be reset to a lower quality level to prevent system overload (the hatched area in Figure 1.10). The same figure can be used to give an example of the resource assignments to the different functions contained within a scalable MPEG encoder.

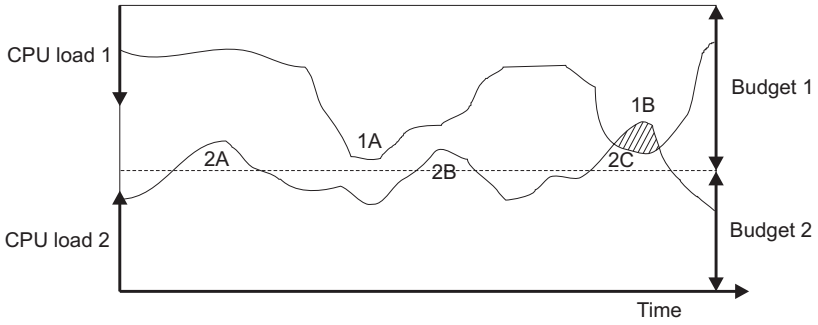

Figure 1.10: *Example of resource budgeting and changing CPU load of applications or functions (based on [7]).*

1.4 MPEG-2 video compression standard

In this section, the essentials of the MPEG video compression standard are briefly described, such that the architecture and the processing blocks are sufficiently known for understanding of this thesis. The MPEG standard is called hybrid video coding, because spatial and temporal decorrelation of video frames is performed by using Discrete Cosine Transform coding and motion-compensated prediction techniques, respectively. Compression is enabled by removing redundant information from the input signal.

A moving video sequence consisting of a series of frames, is actually a three-dimensional signal having spatial and temporal correlation. The spatial correlation can be removed by the decorrelating Discrete Cosine Transform (DCT), whereas the temporal correlation can be exploited by employing motion-compensated prediction. Both techniques are employed in the MPEG standard, which is the mostly applied video compression system today.

- **Spatial correlation** is found when looking into individual video frames (pictures) and considering areas of similar data structures (color, texture). Similar to still-picture coding standards such as JPEG, spatial information is decorrelated by converting picture blocks to the transform domain using the DCT. The result of the DCT is a block of transform coefficients, which are related to the frequencies contained in the input picture block. The meaning of the transform is that a block is decomposed into basis patterns, and the transform coefficient indicate the strength of those basis patterns. Figure 1.11 shows the basis patterns that are used for the transformation. Note that the frequency of

the patterns increases when going from left to right and top to bottom. Each picture block is a linear combination of these basis patterns. Since

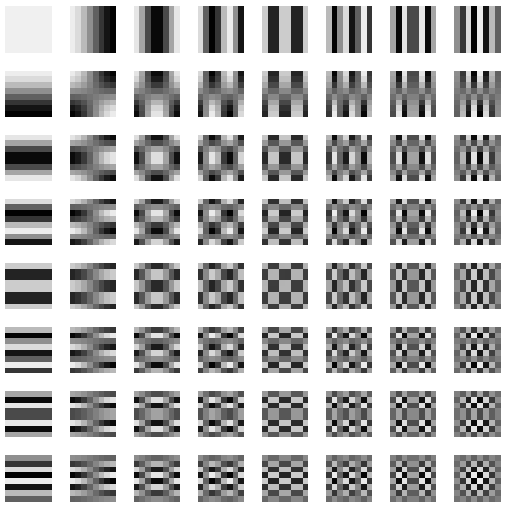

Figure 1.11: *DCT block of basis patterns used for projection (encoding) and reconstruction (decoding).*

the patterns shown in the top left corner of the DCT block (the low-frequency coefficients) are very common in video pictures, the DCT transformation concentrates the signal in these coefficients (high coefficient values). For compression purpose, in the quantization stage, coefficients are represented with fewer bits and less important coefficients are removed. Afterwards, the remaining coefficients are coded efficiently with Variable Length Coding (VLC).

- **Temporal correlation** is found between successive frames of a video sequence (see Figure 1.12) when considering that objects and background are on similar positions. The correlation is removed by predicting the contents and coding the frame differences instead of complete frames. A high compression rate is achieved by predicting picture contents using motion estimation (ME) and compensation (MC) techniques. The motion is represented by motion vectors (MV), which are computed for picture blocks, based on a simple translational model that does not consider object deformation or rotation (see middle picture of Figure 1.12). MVs are also coded efficiently with VLC.

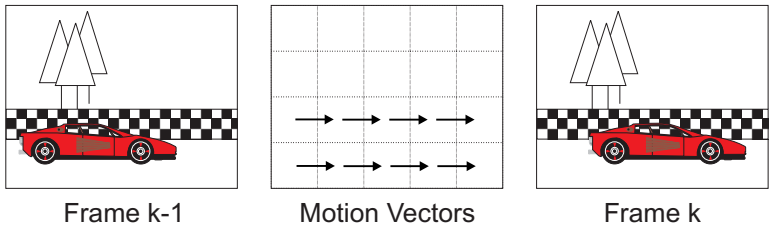

Figure 1.12: *Example of a moving object in a video sequence.*

1.4.1 Basics of MPEG video coding

The MPEG coding standard defines three different types of frames, namely intra (I) frames, predictive (P) frames and bi-directional (B) frames. I-Frames are coded as completely independent frames, thus only intraframe DCT coding is applied. For P- and B- frames, ME and MC techniques are employed in the temporal direction (interframe coding). P-frames are always predicted from a single reference frame in the past, whereas B-frames are predicted from a reference frame in the past and one in the (near) future.

Since I- and P-Frames serve as references for B- and other P-frames, they are coded with more bits than for example B-frames. B-frames are coded with the lowest amount of bits, because the prediction is most efficient and they do not serve as reference for further frames.

The frames of a video sequence are grouped into so-called Groups-Of-Pictures (GOPs), of which the structures are defined by two parameters (N, M). The parameter N represents the total number of frames in a GOP. The distance between two succeeding reference frames is represented by the parameter M. Because B-frames are predicted from upcoming reference frames, they cannot be encoded or decoded before this reference frame is processed and stored by the coder. Therefore, the video frames are processed in a reordered way, e.g. “IPBB” (transmit order) instead of “IBBP” (display order), as shown in Figure 1.13. The transmit order is also used for conveying the compressed stream to the decoder, in order to limit the required memory at the decoder side to two frame memories.

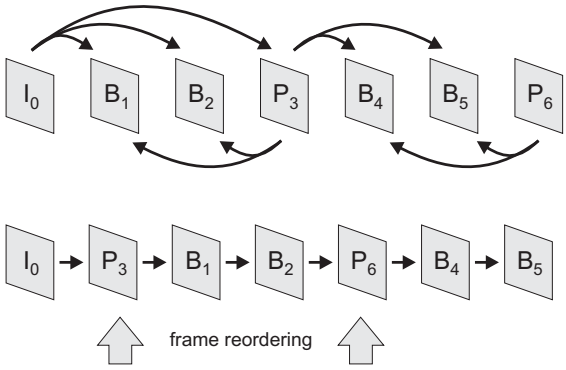

Figure 1.13: *Frame dependencies (top) and reordered frame processing (bottom) with the example of a* $(7, 3)$*-GOP.*

1.4.2 Processing of MPEG video coding

A. Coding architecture

The MPEG coding architecture is shown in Figure 1.14. The architecture includes a prediction loop that reconstructs the quantized DCT coefficients and stores this in the frame memory at the bottom of the figure for further reference. These frame memories at the bottom contain the I- and P-frames. The ME uses these frames for computing the MVs with the actual frame taken from the input frame section at the left of the figure. The MVs are supplied to the MC unit (dotted line) so that a motion-compensated prediction block can be supplied to the subtractor at the top of the figure. Usually, the prediction block closely resembles the actual input block, leading to a small frame difference signal. The complete encoder performs on a block-by-block basis. The ME and MC are based on macroblocks (16×16), whereas the DCT, Quantizer and VLC are processing 8×8-DCT blocks.

The connection from prediction to the adder at the right bottom of the figure ensures that the block-prediction is based on quantized differences, in order to ensure that encoder and decoder work with the same video data, thereby preventing error drift in the video reconstruction at the decoder side. The input video signal X_n (X to be replaced by I, P or B) contains frames in sequential (display) order. The parameter M chosen in the GOP structure requires that the overall frame memory needed for the encoding process is M frame memories. The memory located at the encoder input is for buffering the intermediate B-frames and achieving reordered processing (transmit order).

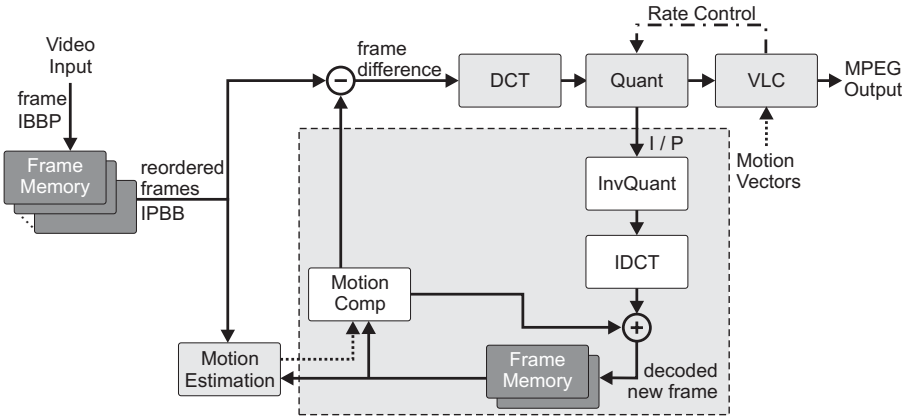

Figure 1.14: *Architecture of an MPEG encoder.*

The input frames (starting with an I-frame) are then processed in transmit order as follows.

1. Intra-frame encoding of the I-frame and after reconstruction, storage in the reference frame memory as past reference.

2. Inter-frame encoding of the P-frame and after reconstruction, storage in the reference frame memory as future reference.

3. Sequential inter-frame encoding of the B-frames, which are between the two reference frames. The B-frames are not stored and directly supplied to the output.

4. The future reference frame of Step 2 becomes the past reference frame, and return to Step 2. If the GOP ends, restart with Step 1.

Further details of the intraframe and interframe coding are discussed in the following.

B. Intraframe coding

Intraframe coding is based on three coding steps: DCT, Quantization and VLC. The MPEG encoder also contains a prediction loop that performs a local decoding, thus Inverse DCT (IDCT) and inverse quantization to create the reference frames in the sample domain.

DCT/IDCT

With the DCT, a picture block of 8×8 pixels, represented as a two-dimensional (2-D) data matrix $\underline{\underline{X}}$ with elements $x[i,j]$ for $0 \leq i,j \leq 7$, are converted to a 2-D DCT matrix $\underline{\underline{Y}}$ of coefficients with elements $y[m,n]$ for $0 \leq m,n \leq 7$ in the transform domain according to

$$\begin{aligned} y[m,n] = \; & \tfrac{1}{4}u(m)u(n)* \\ & \textstyle\sum_{i=0}^{7}\sum_{j=0}^{7} x[i,j]\cos(\tfrac{(2i+1)m*\pi}{16})\cos(\tfrac{(2j+1)n*\pi}{16}), \end{aligned} \tag{1.1}$$

where $u(k) = 1$ for $k > 0$ and $u(0) = 1/\sqrt{2}$. At sufficient accuracy, the transformation is lossless and the reconstructed picture block data-matrix $\underline{\underline{\widehat{X}}}$, with elements $\widehat{x}[i,j]$ for $0 \leq i,j \leq 7$, can be perfectly reconstructed from the DCT coefficient matrix $\underline{\underline{Y}}$ with the IDCT given by

$$\begin{aligned} \widehat{x}[i,j] = \; & \tfrac{1}{4}\textstyle\sum_{m=0}^{7}\sum_{n=0}^{7} u(m)u(n)* \\ & y[m,n]\cos(\tfrac{(2i+1)m*\pi}{16})\cos(\tfrac{(2j+1)n*\pi}{16}). \end{aligned} \tag{1.2}$$

Note that both equations indicate that the transform is based on processing real numbers and fast DCT algorithms similar to the Fast Fourier Transformation are possible. More details will be discussed in Section 2.2.

Quantization/Inverse Quantization

The quantization stage in MPEG is based on various elements: coefficient-dependent weighting, uniform block quantization and normalization. The weighting function is based on the human perception of spatial frequencies in pictures. It is known that high-frequency details can be observed, albeit at a much lower sensitivity than lower frequencies. This enables that high frequencies are represented by fewer bits than low frequencies. This introduces lossy compression. The weighting matrix $\underline{\underline{W}}$ is shown in Figure 1.15. The weighting elements $w[m,n]$ increase for higher values of m,n, leading to more suppression of the according DCT coefficients.

Due to the high sensitivity of the human eye for lower frequencies, the DC coefficient (zero frequency, the top-left coefficient of the DCT matrix), the quantization of DC coefficients is fixed.

Another element in the quantization is the uniform block quantization, which is implemented by the so-called $qscale$ parameter. This $qscale$ parameter is actually related to the step size of the uniform quantizer and influences the

$$\begin{pmatrix} 08 & 16 & 19 & 22 & 26 & 27 & 29 & 34 \\ 16 & 16 & 22 & 24 & 27 & 29 & 34 & 37 \\ 19 & 22 & 26 & 27 & 29 & 34 & 34 & 38 \\ 22 & 22 & 26 & 27 & 29 & 34 & 37 & 40 \\ 22 & 26 & 27 & 29 & 32 & 35 & 40 & 48 \\ 26 & 27 & 29 & 32 & 35 & 40 & 48 & 58 \\ 26 & 27 & 29 & 24 & 38 & 46 & 56 & 69 \\ 27 & 29 & 35 & 38 & 46 & 56 & 69 & 83 \end{pmatrix}$$

Figure 1.15: *Weighting matrix $\underline{\underline{W}}$ of intraframe DCT coefficients.*

quantized output amplitude of the remaining (AC) coefficients. The overall quantization of DCT coefficients $y^*[m,n]$ with $0 \leq m,n \leq 7$ are specified by

$$y^*[m,n] = \frac{y[m,n] + k * qfactor * w[m,n]}{2 * qfactor * w[m,n]}, \tag{1.3}$$

where $k = \text{sign}(y[m,n])$ for intraframe coding and $k = 0$ for interframe coding (see the next pages on interframe coding). In Equation (1.3), the parameter $qfactor$ is used, which is related to the $qscale$ parameter. The usual case is $qfactor = 2 * qscale$ for the uniform quantization (so-called type 0). Another possibility is using a non-uniform quantization (so-called type 1), in which $qfactor$ is non-linearly related to $qscale$. In this case, the parameter $qfactor = 1, 2, ... \ 8, 10, ... \ 24, 28, ... \ 56, 64, ... \ 120, 136, ... \ 248$ with $qscale = 1, 2, ...31$.

The inverse quantization restores the quantized coefficient values to the regular amplitude range prior to computing the IDCT. The inverse quantization is computed by

$$\widehat{y}[m,n] = \frac{(2y^*[m,n] + k^*) * w[m,n] * qfactor}{32}, \tag{1.4}$$

where $k^* = \text{sign}(y^*[m,n])$ for intraframe coding and $k = 0$ for interframe coding.

Variable Length Coding (VLC)

The next step after quantization is VLC, where the quantized coefficients are processed into serial strings of numbers, which are subsequently mapped into

bit strings. Additionally, also MVs and other system parameters are converted into bit strings of variable length.

First, the quantized DCT coefficients are reordered into a sequential string of number that starts with the most important coefficients (low frequency) and for clustering of zero values at the end of the block. Due to the energy compaction in the low-frequency DCT coefficients and the quantization, high-frequency coefficients are likely to be zero. For this reason, the coefficients are commonly scanned in zigzag scan order (see left side of Figure 1.16). MPEG-2 also offers an alternative scan order (see right side of Figure 1.16) for special coding modes that are beyond the scope of this thesis.

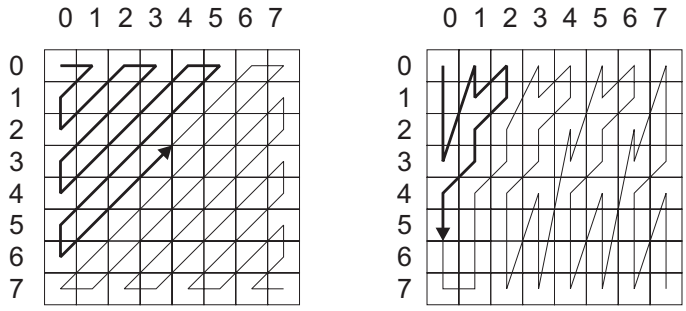

Figure 1.16: *Zigzag scan order (left) and alternative scan order (right).*

After scanning, an algorithm is applied that counts the number of occurring zeros prior to a non-zero coefficient. The combination of both numbers is mapped onto a code word using a VLC coding table. The number of zeros is also known as the run-length (or simply run) and the procedure is therefore referred to as run-length coding. The non-zero value of the quantized DCT coefficient is often called the level of the coefficient. The algorithm is described in a step-by-step approach below.

1. Load next coefficient if available. If the last coefficient has already been processed, code the end-of-block (EOB) codeword and exit.

2. Test the coefficient for zero value. If the coefficient has a zero value, increment the run counter and proceed with Step 1. If the coefficient is non-zero, proceed with Step 3.

3. Take the run-level combination and map this combination into a codeword, reset the run counter and proceed with Step 1.

The DC coefficients are treated differently and are differentially coded, which means that only the difference between DC coefficients of the actual and the previous DCT block is coded. In VLC, the coding table is organized in such a way that codewords are short if they occur more often and long if they are rarely needed (Huffman coding). In this way, the bit rate is minimized.

C. Interframe coding

In interframe coding, the motion is described by motion vectors, which indicate that a picture block of the actual frame that is going to be interframe coded, is similar to a picture block taken from a reference frame. The motion compensation reads the required picture block at the correct position in the reference frame using the MVs. This block represents the motion-compensated prediction, which is subtracted from the picture block taken from the actual frame. The result is a difference signal depending on the reference frame(s) and the actual frame, which explains the term "interframe coding". The difference signal is then DCT coded in a similar way as intraframe coding. The computation of the MVs is called motion estimation and their usage for prediction is called motion compensation. Both steps are further outlined below.

- **Motion estimation (ME)**
  The ME processes picture blocks of 16×16 pixels (macroblocks) and computes motion vectors (MVs). Since reference frames may be in the past or in the future, ME can be split into forward, backward ME, or a combination of both. The terms "forward" and "backward" indicate the usage of a past or upcoming reference frame, respectively, for the ME computation. An example of forward and backward ME is shown in the left side of Figure 1.17. The right side of the figure depicts an example of a MV for one block of the currently processed frame (light gray) and

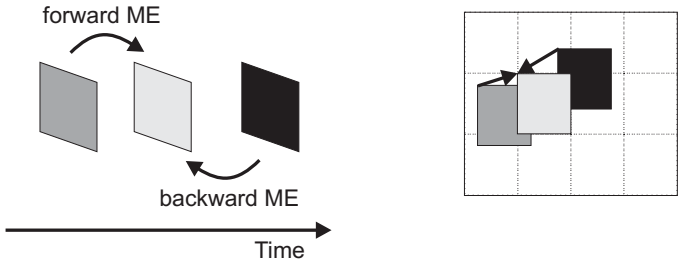

Figure 1.17: *Forward and backward motion estimation.*

the positions of the MV-referred blocks taken from the reference frames projected in the block grid of the actual frame.

The computation of a MV is usually performed by evaluating a number of MV candidates and measuring their accuracy. This will be further elaborated in Chapter 4 and 5.

- **Motion Compensation**
  The MVs resulting from the ME are used to compute a prediction of the current frame. The prediction of each macroblock may use forward prediction (P- and B-frames), backward prediction (B-frames) or both (B-frames). In addition, a fallback mode is allowed, using intraframe coding, in the case of excessive motion, inaccurate prediction or high bit-costs.

1.4.3 MPEG Scalability

There are several meanings of the term "scalability" in the MPEG video domain. Since this thesis addresses complexity scalability, the so-called MPEG scalability is briefly explained, which is mostly concentrating on distinguishing the essential part of the coded information for improving the system robustness.

A compressed video that is transmitted via a network may be corrupted at the receiver side due to channel errors and data package loss. Decoding a corrupted video stream may lead to more or less visible artifacts, or in extreme cases to temporally undecodable video. This is especially critical for streaming video, which forecloses the resubmission of lost or corrupted video data.

To increase the robustness of video streaming over networks against data loss, a two-layered coding is proposed in [16] and is also used in the recent MPEG-4 standard [17]. This coding method uses a base layer that is supposed to contain independently decodable video data providing a certain base video quality. The error-free transmission of the base layer is guaranteed by the network protocol and/or by adding special error protection. In addition, an enhancement layer is supposed to contain further video data that enhances the video quality of the base layer to full quality. The enhancement layer is transmitted with lower priority, and the receiver side, the decoder is able to enhance the video quality with each additional data package from the enhancement layer that is received.

When finalizing the draft of the MPEG-2 standard, the technique of the above-given two-layered coding is known as scalability. The MPEG-2 standard defines four different variations to achieve scalability (scalability modes), namely data partitioning (also known as frequency scalability), SNR scalability, spatial scalability and temporal scalability. These forms of scalability will be briefly described below for the sake of completeness.

- *Data partitioning* separates the block of quantized DCT coefficients such that low-frequency coefficients and motion vectors are coded in the base layer and high-frequency coefficients are coded in the enhancement layer (see Figure 1.18).

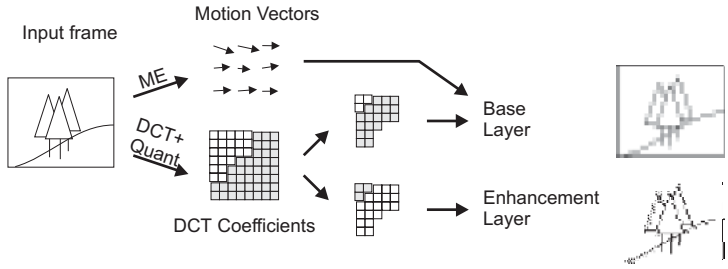

Figure 1.18: *Principle of data partitioning.*

- *SNR scalability* provides different video quality levels, where the layers have the same spatial (frame size) and temporal (frame rate) resolution. The DCT coefficients are coded with high quantization step size for the base layer, and an update for the coefficients for higher video quality achieved with finer quantization is integrated in the enhancement layer (see Figure 1.19).

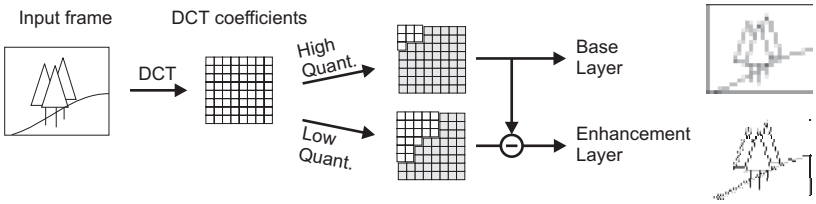

Figure 1.19: *Principle of SNR scalability.*

- *Spatial scalability* uses different spatial resolutions, where the layers have the same temporal resolution. The input frames are scaled down to lower resolution for the base layer, and the enhancement layer contains the update for achieving full resolution (see Figure 1.20).

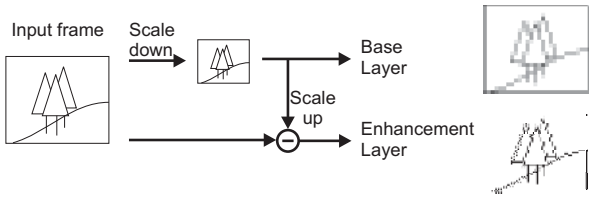

Figure 1.20: *Principle of spatial scalability.*

- *Temporal scalability* provides different frame rates. The base layer contains the frames for a certain frame rate, and the enhancement layer contains the frames that are needed to increase the frame rate. Both two layers use the same spatial resolution. The predictions of frames in the enhancement layer may use frames from both layers (see Figure 1.21).

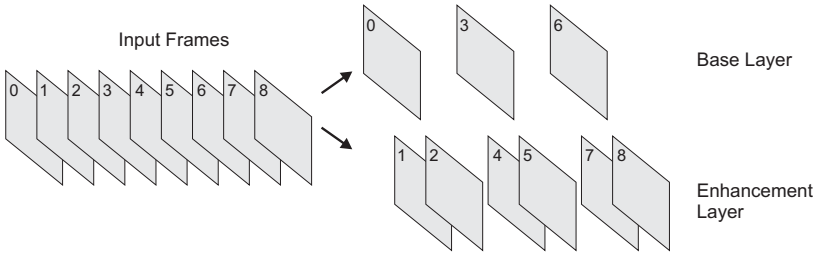

Figure 1.21: *Principle of temporal scalability.*

In MPEG-4, the scalability modes are further extended and for example forms of the above scalability options were combined into the so-called Fine-Grain Scalability (FGS) option [18], which can serve as an advanced aid in obtaining higher robustness when communicating over poor channels.

1.5 Outline of the thesis

This thesis provides complexity scalability for MPEG-2 video encoding. The MPEG coding standard leads to several functions that are considered for ap-

proaching scalability. Key functions of MPEG coding are the Discrete Cosine Transformation (DCT) and the Motion Estimation (ME), both requiring a significant amount of computing power. Algorithms for these functions are evaluated and exploited for scalability in the following chapters. Here, we structured this thesis by first concentrating on single modules (functions), without their possible interaction with other modules. Afterwards, the further results when looking to more than one module at once are exploited.

Chapter 2 discusses, the mathematical background of the DCT and compares current state-of-the-art algorithms to compute the DCT. The DCT transforms picture blocks to the transform domain to obtain a powerful compression. In conjunction with an inverse DCT, a perfect reconstruction of the picture blocks is achieved, while spending fewer bits for coding the blocks than not using the transformation. So-called fast DCT algorithms exploit the periodic property of the cosine function and find efficient implementations for full DCT computation with respect to the number of operations (additions, multiplications, etc.). Besides full DCT computation, algorithms exist that accept a reduced accuracy of the computation, for the purpose of lower the computational complexity. A complexity comparison of the algorithms providing a full DCT computation is made to decide which algorithms are taken as case studies for developing complexity scalable DCT.

Chapter 3 presents a new scalable DCT computation technique, where a DCT algorithm is analyzed for finding its shared computation nodes. The analysis results in a database that provides information about the required computational effort for computing DCT coefficients. The specifics of the target hardware/execution architecture can be taken into account with the analysis. Intermediate results that are available from already computed coefficients are considered for reusing during the computation of other coefficients. With the collected information, a DCT computation order can be found, such that the amount of computed coefficients is optimized for a constrained number of computing cycles. If certain DCT coefficients should be preferably computed (e.g. coefficients that represent horizontal or vertical lines in the sample domain), the generation of the computation order can include weighting of these coefficients.

Chapter 4 addresses the essentials of ME. The ME computes motion vector fields to indicate block displacements between frames in a video sequence. A picture block (macroblock) is then coded with reference to a block in a previous decoded frame (the prediction) and the difference to this prediction.

The displacement of macroblocks is found by evaluating their content similarity with a block-matching criterion. State-of-the-art ME algorithms aim at providing smart search strategies for efficiently finding motion vectors with pixel accuracy. The computed vectors can optionally be refined to half-pixel accuracy. Several ME algorithms are compared by their worst-case computational effort and their actual performance when processing different video sequences.

Chapter 5 presents two new techniques for complexity scalability of the ME process. The first technique processes motion vector fields (MVFs) in three stages. The first stage performs initial ME with the video frames at the entrance of the encoder and processes them in display order, rather than in the usual reordered way. The second stage efficiently derives the MVFs that are required for the final MPEG encoding process by multi-vector-field approximations, based on the MVFs computed in the first stage. Furthermore, the quality of full-search ME can be obtained with an optional refinement stage.

The second technique provides a scalable ME through block classification based on edge detection. Prior to estimating the motion between two frames, the frame contents are classified into picture areas that e.g. have horizontal or vertical lines. The classification is exploited to minimize the number of motion vector evaluations per macroblock by for example preferring vector candidates that are located across a detected line. A novelty in the algorithm is a *distribution* of reliable motion vectors *to* other macroblocks, which is opposite to other known techniques that *query* MVs *from* other blocks.

Up to this point, each function was individually investigated. In the following, the scalable functions are integrated into a complete encoder, in order to evaluate the results when combining several scalable modules (functions). The performance of the scalable encoding system is evaluated in Chapter 6. The scalable DCT has impacts on functions of the MPEG coding system that processes DCT coefficients. These functions can be scaled accordingly to the DCT by exploiting the reduced number of computed coefficients. The scalable ME affects the number of MV candidates that are evaluated, but not the motion compensation or the VLC coding of motion vectors. The resulting MPEG encoder offers a wide range of complexity scalability, video quality and compression rates.

Chapter 7 concludes that the obtained complexity scalability results in new coding algorithms, which differ from conventional MPEG encoding design.

The resulting scalable MPEG encoder offers a wide range of complexity scalability of about a factor of three, whereas a non-scalable parameterized MPEG encoder has a factor of about 1-1.5. The developed scalability techniques can be readily applied in portable MPEG coding systems and may well be used in new coding standards such as MPEG-4 and H.264.

1.6 Background of the chapters

Most parts of the chapters in this thesis have been published in conference proceedings and scientific journals. In this section, the origin of the chapters and their publication history is presented.

The initial work of this thesis started with the involvement in the Television Processor (TVP) project of Philips Research, The Netherlands. The TVP project was initiated by amongst others the promoter of this thesis. The project aimed at designing a multi-processor system suitable for executing a multitude of video functions in parallel. The description of the programmable video processing platform and its signal-flow programming in Section 1.2.3 were adopted from [1, 2]. The platform was a good framework to study scalability aspects and flexible processing as shown in the examples in Section 1.2. The scalability results were presented at the 22^{nd} International Symposium on Information Theory in the Benelux in 2001 in the paper titled "Implementation of a Dynamical Multi-Window TV System" [4], which formed the basis of parts of the introduction in Chapter 1. This paper, as well as all following papers that are published by the author of this thesis, were co-authored by the promoter and the copromoter of this thesis. In Section 1.3, several parts from [5, 7] about scalable video algorithms and quality-of-service were included. The latter formed the basis for the main focus of this thesis after joining a larger project within Philips about scalability.

The following chapters concentrate on the achievements that have been made during the research phase towards complexity scalability, starting with a comparison of algorithms for the Discrete Cosine Transformation (DCT) in Chapter 2. The chapter originates from a technical report that has been written in 2000 for Philips, which was titled “On Scalable DCT Processing Algorithms for MPEG Encoding” [19].

The work on developing a scalable DCT resulted in a patent application [20] and subsequently in the following three papers, which have been merged into

Chapter 3. The first paper was presented at the IEEE Workshop on Signal Processing Systems (SIPS) in 2001 and was titled "New Scalable DCT Computation for Resource-Constrained Systems" [21]. In the same year, the second paper was presented at the IEEE International Conference on Image Processing (ICIP 2001) and was titled "New DCT Computation Algorithm for Video Quality Scaling" [22]. An extension of the first paper has been published 2003 in a special issue of the Kluwer Academic Publishers Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology and was titled "New DCT Computation Technique based on Scalable Resources" [23].

The research on motion estimation (ME) to achieve complexity scalability is presented in a similar way. Firstly, Chapter 4 starts with a comparison of algorithms for the ME. The chapter originates from a technical report from 2000 that was titled “On Scalable Motion Estimation for MPEG Encoding” [24].

Secondly, Chapter 5 has been merged from several papers that published the results of two developed scalable ME techniques. The first technique is based on approximations of multiple motion-vector fields. The approximation technique results from a co-operation with Gerben Hekstra from Philips Research and lead to a patent application [25]. The chapter finds its roots in two papers about the first scalable ME technique that have been presented in 2002 at the IEEE International Conference on Consumer Electronics (ICCE 2002) under the title "Flexible Frame-Reordering and Multi-Temporal Motion Estimation for Scalable MPEG Encoding in Mobile Consumer Equipment" [26] and at the IEEE International Conference on Image Processing (ICIP 2002) under the title "New Flexible Motion Estimation Technique for Scalable MPEG Encoding using Display Frame Order and Multi-Temporal References" [27]. Some more aspects and details of the developed ME technique are described in two papers that have been presented in the same year at the 23^{rd} International Symposium on Information and Communication Theory in the Benelux under the title "Frame-Reordered Multi-Temporal Motion Estimation for Scalable MPEG" [28] and at the IEEE International Conference on Multimedia and Expo (ICME 2002) under the title "New Scalable Three-Stage Motion Estimation Technique for Mobile MPEG Encoding" [29]. Finally, the second scalable ME technique was published 2004 in a paper for the IEEE Transactions on Consumer Electronics under the title “Computational Complexity Scalable Motion Estimation for Mobile MPEG Encoding” [30].

Chapter 6 has its roots in three papers that indicate the performance of a complete scalable MPEG encoding system using the presented scalable al-

gorithms. The first paper has been published at the 24^{th} International Symposium on Information Theory in the Benelux in 2003 and was titled "A SW-based Complexity Scalable MPEG Encoder for Mobile Consumer Equipment" [31]. The second paper is more detailed and has been published in a special issue on multimedia of the EURASIP Journal of Applied Signal Processing in 2004 and was titled “New Complexity Scalable MPEG Encoding Techniques for Mobile Applications” [32]. A further expansion has been made for the third paper “Resource-Aware Complexity Scalability for Mobile MPEG Encoding”, which was invited for presentation in a special session of the SPIE Visual Communication and Image Processing Conference (VCIP 2004) [33].

CHAPTER 2

Discrete cosine transformation

Since the introduction of the Discrete Cosine Transformation (DCT), numerous publications have appeared about efficient computations and implementations of the DCT. The purpose of this chapter is to analyze a few DCT algorithms that are potentially suited for integration into scalable systems. The analyzed algorithms efficiently compute the DCT and have attractive properties for soft- and hardware implementations. Subsequently, two DCT algorithms are selected for building a scalable DCT subsystem that will be discussed in the next chapter.

2.1 Introduction

Since the development of the Discrete Cosine Transformation (DCT) and its usage in multiple compression standards, a large number of algorithms were proposed over the past decades for computing the DCT. This chapter describes the basic principles of some DCT algorithms, thereby serving as a partial literature overview. A simple comparison is provided between a selected number of algorithms in order to determine which algorithms are applied for further research on scalability.

For compression purpose, the luminance and chrominance values of small square blocks of an image are transformed with the DCT to the transform domain, where the representation of the blocks with the DCT coefficients in the transform domain is more efficient than with the pixel values in the sample domain. The back-transformation is performed with the inverse DCT (IDCT), where both transformations are lossless, when compared to the errors that are introduced by quantization of DCT coefficients and rounding.

The chapter is divided as follows. Section 2.2 briefly describes the mathematical formulas for the transformations. Section 2.3 presents a number of DCT algorithms that have been found in literature. These algorithms are compared by their computational effort in Section 2.4. Section 2.5 presents a number of algorithms that are designed to compute the DCT with a limited accuracy, thereby reducing the output quality. These algorithms are not further considered in particular, since it has been stated in the problem statement in Section 1.1 that the desired scalability range of a future scalable algorithm should include full quality processing. Section 2.6 concludes the chapter.

2.2 Mathematical background

2.2.1 Forward Transformation

For a given $N \times N$ picture-block, represented as a two-dimensional (2-D) data matrix $\underline{\underline{X}}$ with elements $x[i,j]$ for $i,j = 0,1,...,N-1$, the 2-D DCT matrix $\underline{\underline{Y}}$ of coefficients with elements $y[m,n]$ for $m,n = 0,1,...,N-1$, is computed by

$$\begin{aligned} y[m,n] = \; & \tfrac{2}{N} u(m) u(n) * \\ & \textstyle\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} x[i,j] \cos(\tfrac{(2i+1)m*\pi}{2N}) \cos(\tfrac{(2j+1)n*\pi}{2N}), \end{aligned} \tag{2.1}$$

where

$$u(k) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } k = 0, \\ 1 & \text{otherwise.} \end{cases}$$

Equation (2.1) can be simplified by ignoring the constant factors for convenience and by defining an $N \times N$ cosine matrix $\underline{\underline{K}}_N$ with elements

$$k[p,q] = \cos(\frac{(2p+1)q*\pi}{2N}), \tag{2.2}$$

so that Equation (2.1) can be simplified to

$$\underline{\underline{Y}} = \underline{\underline{K_N}}^{\top} * \underline{\underline{X}} * \underline{\underline{K_N}}. \tag{2.3}$$

Equation (2.3) shows that the 2-D DCT as specified by Equation (2.1) is based on two orthogonal 1-D DCTs, where $\underline{\underline{K}}_N^{\top} * \underline{\underline{X}}$ transforms the columns of the picture block $\underline{\underline{X}}$, and $\underline{\underline{X}} * \underline{\underline{K}}_N$ transforms the rows (known as row-column method). The 1-D DCT transformation of an input vector $\underline{x}$ with elements $x[i]$ (a single row or column of the picture block) for $0 \leq i \leq N-1$, into a 1-D DCT vector $\underline{y}$ with elements $y[n]$ for $0 \leq n \leq N-1$, is computed by

$$y[n] = \sqrt{\frac{2}{N}} * u(n) * \sum_{i=0}^{N-1} x[i] \cos(\frac{(2i+1)n * \pi}{2N}). \tag{2.4}$$

Ignoring the constant factors for convenience and using the definition of the cosine matrix $\underline{\underline{K}}_N$, Equation (2.4) is simplified to

$$\underline{y} = \underline{\underline{K_N}}^{\top} * \underline{x}. \tag{2.5}$$

Since the computation of two 1-D DCTs is less expensive than one 2-D DCT, state-of-the-art DCT algorithms normally refer to Equation (2.3) and concentrate on optimizing a 1-D DCT.

2.2.2 Inverse Transformation

To transform DCT coefficients back to the sample domain, the inverse DCT (IDCT) is performed. The 1-D IDCT reconstructs the data vector $\underline{\widehat{x}}$ with elements $\widehat{x}[i]$ for $0 \leq i \leq N-1$, and is computed by

$$\widehat{x}[i] = \sqrt{\frac{2}{N}} * \sum_{n=0}^{N-1} y[n] * u(n) * k[i,n], \tag{2.6}$$

and the 2-D IDCT reconstructs the data block $\underline{\underline{\widehat{X}}}$ with elements $\widehat{x}[i,j]$ for $0 \leq i,j \leq N-1$, and is computed by

$$\begin{aligned} x[i,j] = \; & \tfrac{2}{N} * \\ & \textstyle\sum_{m=0}^{N-1} \sum_{n=0}^{N-1} u(m)u(n) * y[m,n] * k[i,m] * k[j,n], \end{aligned} \tag{2.7}$$

where the cosine terms are already replaced with the definition given in Equation (2.2).

2.2.3 Periodic properties of the cosine function

An important optimization feature for the DCT computation concentrates on reducing the rotational symmetry and periodicity of the cosine function. This simplifies the matrix $\underline{\underline{K}}$ as defined in Equation (2.2).

- The periodicity means that the cosine function of angle α repeats every 2π radians, thus

$$\cos(\alpha) = \cos(n * 2\pi + \alpha);\ n\epsilon Z. \tag{2.8}$$

- Furthermore, the cosine function is anti-periodic over π radians, so that

$$\cos(\alpha) = (-1)^n * \cos(n * \pi + \alpha);\ n\epsilon Z. \tag{2.9}$$

2.3 Examples of DCT algorithms

A large number of *fast* DCT algorithms have been presented in the past few years, aiming at reducing the computational complexity of a full DCT computation. For the reduction of the computational complexity, the following algorithms arise from exploiting mathematical rules and the above-mentioned periodic properties of the cosine function. The presented algorithmic examples indicate the variety of approaches that have been developed for the DCT computation. The presented set of algorithms is not an exhaustive overview of all published DCT algorithms in literature. More extensive overviews can be found in specialized DCT handbooks.

Three DCT algorithms have been selected for their highly efficient usage of operations, namely the Lee-Huang recursive algorithm [34], the Cho-Lee 2-D algorithm [35] and the Arai-Agui-Nakajima 1-D algorithm [36]. These algorithms provide an attractive mixture from 1-D or 2-D DCT computations, implementation preferences in software or hardware, and popularity. For example, the Lee-Huang algorithm was chosen, because it was known that it is flexible for various block sizes. This property may be attractive for scalability. The Cho-Lee algorithm has attractive properties for hardware minimization, which may be exploited (partially) in scalable software implementations. The Arai-Agui-Nakajima algorithm was studied because it is widely used in JPEG [37, 38] applications and because this was a pure 1-D algorithm.

2.3.1 Lee-Huang recursive algorithm

The Lee-Huang recursive algorithm [34] reduces the computation of the cosine matrix $\underline{\underline{K_N}}$ (see Equation (2.2)) to equivalent sub-problems of a lower complexity by recursive mapping ($\underline{\underline{K_N}} = f(\underline{\underline{K_{N/2}}})$). An example is given in Figure 2.1. The figure shows the cosine matrix $\underline{\underline{K_4}}$ for cosine arguments α with $0 \leq |\alpha| < \pi/2$. A subset of the matrix elements form the reduced matrix $\underline{\underline{K_2}}$. Similarly, 2^n-DCTs are reduced to 2^{n-1}-DCTs of lower complexity.

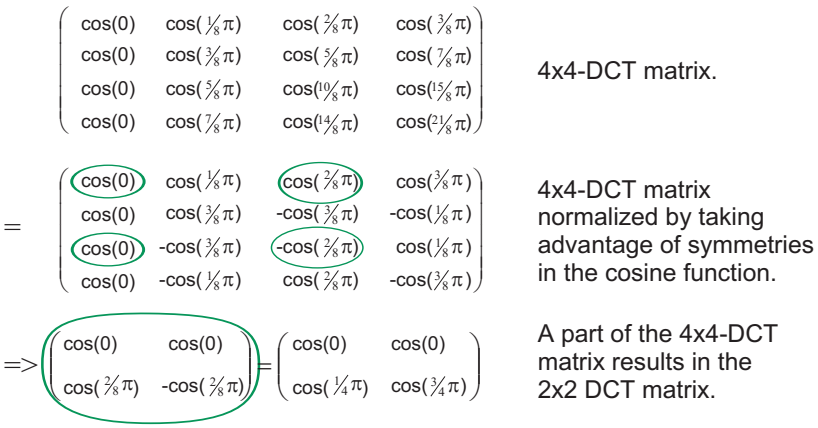

Figure 2.1: *Decomposition of matrices for a 4×4 DCT algorithm according to the Lee-Huang 1-D recursive mapping rule.*

The main advantage of this algorithm is that it provides a DCT computation for arbitrary $2^n \times 2^n$-DCTs in either the row-column method using a 1-D DCT or directly applied to a 2-D DCT. The algorithm can be implemented in software as a single recursive function. This property may be of interest for upcoming video coding standards like H.264 [39], since they do not use fixed coding picture blocks of 8×8 pixels. Furthermore, the recursivity property of this algorithm can also be exploited for advanced MPEG coding, where the transformation of a subsampled block serves as an approximation of the full-block transformation [40].

2.3.2 Cho-Lee 2-D algorithm

The 2-D algorithm by Cho-Lee [35] is based on data dependencies between both cosine matrices $\underline{\underline{K_N}}$ and $\underline{\underline{K_N}}^\top$ of Equation (2.3). By exploiting the data

dependencies, one of the matrices can be represented as a function of the other, thus $\underline{\underline{K_N}} = f(\underline{\underline{K_N}}^\top)$ or vice versa, which reduces the 2-D transformation to 1-D transformations. The selection of the 1-D DCT algorithm is free of choice. In [35] it is explained that the algorithm has the capability of saving silicon area in a hardware implementation by reducing the number of required hardware components for parallel processing. In this case, the algorithm manages the DCT computation with 2^{n-1} 1-D DCT function blocks, which is half of the number of 1-D DCT function blocks that would be used for a comparable implementation without this optimization. The basic idea of the algorithm is depicted in Figure 2.2 for a 4×4-DCT computation.

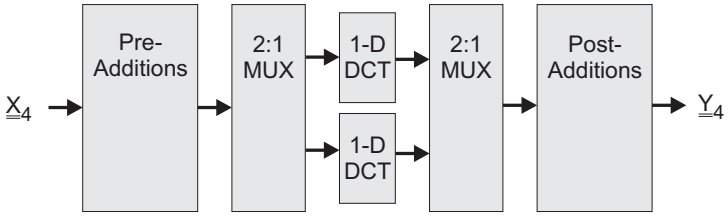

Figure 2.2: *DCT block diagram of the Cho-Lee algorithm.*

Although this thesis concentrates on software implementations of complexity scalable algorithms, this algorithm was considered, because the abovementioned property does not foreclose its utilization for scalability in e.g. saving hardware or faster software, or a combination of both.

2.3.3 Arai-Agui-Nakajima 1-D algorithm

The Arai-Agui-Nakajima 1-D algorithm [36] is deduced from a Discrete Fourier Transformation. The computations are arranged such that several multiplications (8 of the 13 multiplications in the algorithm) can be integrated in a subsequent fixed quantization stage, which is the main feature of the algorithm. The signal flow-graph of an 8×1-DCT is shown in Figure 2.3, where a bullet ($\bullet$) denotes an addition and arrows stand for sign inversion. A box ($\boxed{\ }$) represents a multiplication with the following factors.

$$\begin{aligned} \boxed{1} = \boxed{3} \quad &= \tfrac{1}{\sqrt{2}} \approx 0.707 \\ \boxed{2} \quad &= \cos(\tfrac{\pi}{8}) - \cos(\tfrac{3*\pi}{8}) \approx 0.541 \\ \boxed{4} \quad &= \cos(\tfrac{\pi}{8}) + \cos(\tfrac{3*\pi}{8}) \approx 1.307 \\ \boxed{5} \quad &= \cos(\tfrac{3*\pi}{8}) \approx 0.383 \end{aligned} \tag{2.10}$$

The scale factors are $(2 * \sqrt{2})^{-1}$ for $y[0]$ and $(4 * \cos(i/16))^{-1}$ for $y[i]$ with $1 \leq i < 8$.

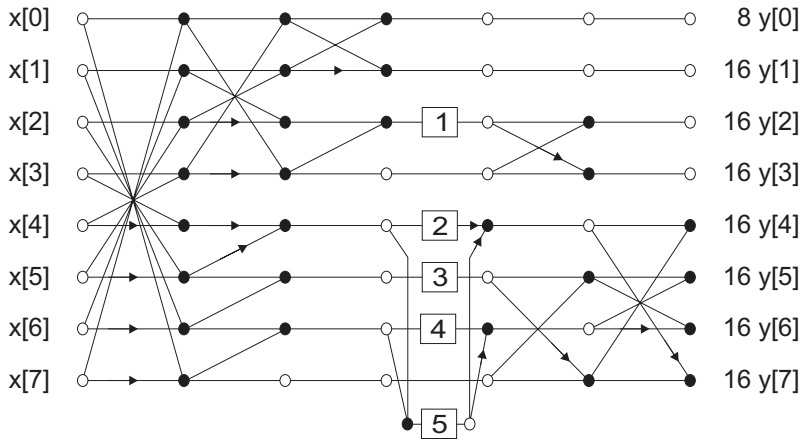

Figure 2.3: *1-D DCT signal-flow graph of the Arai-Agui-Nakajima algorithm.*

This algorithm was published in a book [38] about the JPEG still image compression standard [37], where it was presented as an example of a highly efficient DCT algorithm. The computation of Figure 2.3 can be used in a 2-D DCT algorithm using the row-column method (see Equation (2.3)), but also in the Cho-Lee algorithm from the previous section.

2.3.4 Alternative DCT algorithms

A number of algorithms belong to the group of "scaled DCT algorithms" [41]. The term "scaled" should not be mixed up with the scalability (on the computational complexity) that is proposed in this thesis. The purpose of scaled DCTs is that the quantization is integrated into the DCT transform. For this reason, scaled DCTs are implemented such that a number of multiplications are combined with those of the quantization. However, in this case, the quantization is basically fixed. The Arai-Agui-Nakajima algorithm presented in Section 2.3.3 belongs to this group of algorithms, but even without exploiting this feature, the algorithm is still efficient (see Section 2.4).

Moreover, DCT algorithms can be optimized by using combined multiply-accumulate operations [42, 43], thereby fitting to modern processor designs.

However, an optimization of the DCT computation based on such specialized operations, limits the applicability of the algorithms for arbitrary hardware architectures, where these operations are not necessarily available.

2.4 Complexity comparison of DCT algorithms

The comparison of the computational effort of DCT algorithms is usually performed based on the number of additions and multiplications (e.g. [34, 43, 44, 45, 46, 47]), rarely including shift operations (e.g. [48]). The tables in the following consider all three operations and indicate hardware and software implementation differences for the presented algorithms. Furthermore, different architectures for a software implementation of the DCT algorithms are considered by weighting the computation costs of the operations, leading to a proposal for a favorable DCT algorithm.

The computational effort required to compute the DCT algorithm examples that were presented in Section 2.3 is shown in Table 2.1, where the constant normalization factors of Equation (2.1) are ignored. The 1-D DCT algorithms are combined with the conventional row-column approach to achieve a 2-D DCT. The symbol "$\times$" denotes a multiplication and "$\gg$" denotes a shift. The symbol "$+$" represents an addition, and a subtraction is considered as a special case of an addition.

Algorithm	8×1-DCT			8×8-DCT		
	$\times$	$\gg$	$+$	$\times$	$\gg$	$+$
2-D direct implementation	-	-	-	8192	0	3584
1-D direct implementation	64	0	56	1024	0	896
2-D Lee-Huang	-	-	-	128	32	464
1-D Lee-Huang	12	5	29	192	80	464
1-D Arai-Agui-Nakajima (AAN)	13	0	29	208	0	464
2-D Cho-Lee using 1-D Lee-Huang	-	-	-	96	89	466
2-D Cho-Lee using 1-D AAN	-	-	-	104	49	466

Table 2.1: *Computational complexity of full-quality DCT algorithms.*

When evaluating the table, hardware and software implementations of the algorithms are distinguished. In a hardware implementation, shift operations are realized by wiring if no rounding is involved. For such a hardware case, the

combination of the Cho-Lee 2-D algorithm and the 1-D version of the Lee-Huang recursive algorithm is a highly efficient algorithm. In addition, this combination leads to savings of costly silicon area when exploiting the property of the Cho-Lee algorithm for parallel implementation (see Section 2.3.2).

In a software implementation, shift operations *require* computation cycles, which makes the Lee-Huang algorithm less attractive. Besides this, its recursive structure may lead to costly stack-operations. In the following, the computation complexity is expressed as a number of operations, in order to gain an efficiency indicator for the algorithms. For this purpose, multiplications are weighted against additions and shifts. While additions and shifts are each counted as one operation, the complexity of a multiplication is generally higher (1-3 operations) and may also involve rounding. Rounding is performed with one addition and one shift.

Table 2.2 shows the computational complexity of the algorithms expressed in number of operations, in which various weighting factors for multiplications are used. The minimum operations count of each table column is printed in bold. It can be seen that the Cho-Lee algorithm with integrated Arai-Agui-Nakajima 1-D algorithm is the best choice. When applying the row-column approach using 1-D algorithms, the Arai-Agui-Nakajima algorithm should be adopted.

Algorithm	Weighting of "×"				
used for an 8×8 DCT	1	2	3	4	5
2-D Lee-Huang	624	752	880	1008	1136
2-D Cho-Lee + 1-D Lee-Huang	651	747	843	939	**1035**
2-D Cho-Lee + 1-D AAN	**619**	**723**	**827**	**931**	**1035**
1-D Lee-Huang	736	928	1120	1312	1504
1-D Arai-Agui-Nakajima	672	880	1088	1296	1504

Table 2.2: *Computational complexity of an* 8×8*-DCT computation using different algorithms.*

Given the importance and cost of multiplications, the remainder of this section provides some further detail on the estimation of multiplications and summarizes an example DCT algorithm with minimum amount of multiplications. It was found in [49] that the theoretical lower bound for the number of multiplications of a 2^n-point 1-D DCT computation (see Equation (2.5)) is

$$f_{mults}(\underline{\underline{K}}_N * \underline{x}) = 2^{n+1} - n - 2, \tag{2.11}$$

with $2^n = N$. For an 8-point 1-D DCT, the equation results in the number of required multiplications of $f_{mults}(\underline{\underline{K}}_8 * \underline{x}) = 2^{3+1} - 3 - 2 = 11$.

Example

A fast 1-D DCT algorithm requiring precisely only these 11 multiplications, 29 additions and no shifts has been presented in [46] by Loeffler and Ligtenberg. The algorithm contains among others a rotation part (see Equation (2.12)), which has been exploited for reducing the number of multiplications by 1. In the end, this leads to the optimum of 11 multiplications. The exploited equivalence is given below, requiring 3 multiplications and 3 additions instead of 4 multiplications and 2 additions, because the rotators a and b are known in advance. The equivalence is given by

$$\left| \begin{array}{l} y_0 = \quad ax_0 + bx_1 \\ y_1 = -bx_0 + ax_1 \end{array} \right| \Leftrightarrow \left| \begin{array}{l} y_0 = \quad (b-a)x_1 + a(x_0 + x_1) \\ y_1 = -(a+b)x_0 + a(x_0 + x_1) \end{array} \right|, \tag{2.12}$$

where the equation rotates a vector $(x_0, x_1)^\top$ by an angle α into vector $(y_0, y_1)^\top$. The rotators a and b are $\cos(\alpha)$ and $\sin(\alpha)$, respectively.

End example

In [50], the theoretical lower bound for the number of multiplications of a 2^n-point 2-D DCT computation (see Equation (2.3)) is found as

$$f_{mults}(\underline{\underline{K_N}}^\top * \underline{\underline{X}} * \underline{\underline{K_N}}) = 2^n * (2^{n+1} - n - 2). \tag{2.13}$$

For an 8-point 2-D DCT, $f_{mults}(\underline{\underline{K_8}}^\top * \underline{\underline{X}} * \underline{\underline{K_8}}) = 2^3 * (2^{3+1} - 3 - 2) = 88$. It has been commented in [45] that when minimizing the number of multiplications, the optimal 8×8-DCT computation is found by integrating the 1-D DCT algorithm that requires only 11 multiplications [46] into the Cho-Lee 2-D DCT algorithm (see Section 2.3.2). Since the Cho-Lee algorithm is based on eight 1-D DCTs, the number of required multiplications reaches the lower bound of Equation (2.13). The aforementioned proposal of Loeffler and Ligtenberg was found during writing of this thesis, which saves one multiplication for an 8-point DCT compared to the Lee-Huang algorithm. This optimization step was not further exploited in the performed studies.

2.5 Accuracy-reduced DCT

Up to this point, this chapter has discussed DCT algorithms giving full picture quality without losses. For the sake of completeness, this section presents alternative DCT computations, where a certain loss in picture quality is introduced. The proposed algorithms aim at simplifying the DCT computation in order to improve the processing speed. The simplification leads to a reduced accuracy of the DCT and at the same time, to potentially reduced computational complexity.

The presented list of algorithms is not an exhaustive overview of DCT algorithms with reduced accuracy. Three algorithms were selected for discussion, namely the Merhav-Vasudev multiplication-free algorithm [51], the Pao-Sun adaptive DCT modeling [44] and the Lengwehasatit-Ortega variable complexity algorithm [48]. These algorithms are based on techniques such as replacing multiplications with simpler shift operations, providing DCT computations for different coefficient subsets, and performing classifications for detecting and preventing unnecessary computations that would be spent on coefficients that are zero after quantization.

Except for the above three proposals, alternatives for accuracy-reduced DCT algorithms exist, which are not a real DCT but an approximation [47, 52, 53]. The approximation is realized by replacing a part of the DCT computation by e.g. a Haar transform [54], which can be computed with a lower computational effort.

2.5.1 Merhav-Vasudev multiplication-free algorithm

The Merhav-Vasudev multiplication-free algorithm [51] was primary developed for the computation of the inverse DCT (IDCT). The same approach can be adopted for the computation of the DCT. The main idea is to separate the mathematical definition of the (I)DCT into several matrices, containing values that are basically binary fractions $F = \{0, \pm 1, \pm\frac{1}{2}, \pm\frac{1}{4}, \pm\frac{1}{8}, ...\}$ only. The advantage is that a multiplication with these values leads to simple shift operations. After the matrix separation process, two matrices are left (called M and D) that contain non-trivial multiplications. For further complexity reduction, these multiplications can be replaced with the closest element of F, which introduces an overall computation error.

Merhav and Vasudev found a better strategy for replacing the non-trivial multiplications. The authors considered that a loss-less DCT followed by a loss-

less IDCT leads to the identity matrix. Subsequently, they assumed that between the DCT and the IDCT a quantization/de-quantization stage is introduced, in which matrix D can be integrated. Afterwards, the elements of the remaining matrix M are quantized to values from F, such that the introduced overall error is preferably compensated by matrix D. The total processing chain, i.e. DCT→quantization→de-quantization→IDCT, is then close to the identity matrix.

2.5.2 Pao-Sun adaptive DCT modeling

This approach [44] is based on a statistical analysis of differently encoded video sequences with the video coding standard H.263. With this analysis, variances of the DCT coefficients are represented as a function of the Minimum Mean Absolute Error (MMAE), which is computed after motion-compensated prediction. Depending on this function and the quantization parameter used in H.263, picture blocks are processed in various ways. The following possibilities are exploited: the DCT is computed for all 64 coefficients, or for the 4×4 low-frequency coefficients, or the DC coefficient only, or the DCT is not performed at all.

2.5.3 Lengwehasatit-Ortega var.-complexity algorithm

An input-dependent DCT computation is proposed in [48], where a classification of the input block is used to decide whether a selected subset of input values (data samples) become zero-valued coefficients after quantization, or not. The DCT coefficients that are quantized to zero belong to a so-called "dead zone"[1]. Since the DCT is an orthogonal transformation, the corresponding input values also belong to the "dead zone". If the classification indicates that coefficients to be computed belong to the "dead zone", the computation for these coefficients is stopped.

At several stages during the computation of the DCT, this classification is performed for different sets of input values as follows. For an 8×1-DCT, all eight input values are first evaluated whether they all belong to the "dead zone" or not. If they are outside the "dead zone", the computation continues and the input values are split into two sets of four input values, which are then evaluated separately. The computation continues for those sets, for which

[1] This should not be confused with the dead zone in a quantizer: the area in which small input values are quantized to zero. Therefore, the alternative meaning of the word "dead zone" in this paragraph is marked with "".

it was indicated that at least one input value leads to a non-zero coefficient. These sets are then split once more into sets of two input values each, and the classification is repeated.

The input-dependent DCT computation is proposed for an exact DCT and for a DCT approximation technique that is based on mapping multiplications to simple shift operations. In both cases, the reduction of the computational complexity is well below 9% and can only be gained with high quantization.

2.6 Discussion

Although the accuracy-reduced algorithms presented in Section 2.5 contain useful starting points for low-cost solutions, the computations do not provide scalability in computational complexity. Furthermore, high-quality video applications require a DCT computation with full accuracy, which is provided by the algorithms from Section 2.3. It is beyond doubt that an algorithm providing complexity scalability up to full-quality processing on the one hand, and without overhead on the other hand, is favorable.

In literature, one proposal was found that features scalability properties up to full computational resolution. This briefly discussed here and show that it still has limitations for our purpose. The scalable DCT concept found is the data-pruning algorithm by Peng [55]. Although this approach has been proposed for the computation of an IDCT, it can directly be applied to the computation of a DCT, and it can react on the CPU-resource usage. The algorithm restricts the decoding of DCT coefficients that are found in an $N \times N$ matrix to those that are located in an $m \times n$ submatrix with $0 \leq m, n \leq N-1$. The submatrix is located in the upper left corner of the original matrix to cover at least the DC coefficient. The only overhead that is needed comes from a system resource manager that is used to select the values for m and n. However, a disadvantage of the algorithm is that the selection of the submatrix has no relation with the butterfly diagram computation, so that the really involved computation complexity is not considered explicitly. This makes the proposal not suitable for our purpose, because we are pursuing a DCT computation with constrained computational resources. This motivates the research in the next chapter.

In Chapter 3, a new scalable DCT computation technique is presented that can be applied to any DCT algorithm. Using the Cho-Lee algorithm (see Section 2.3.2) and the Arai-Agui-Nakajima algorithm (see Section 2.3.3), it

will be shown that it is advantageous to analyze the butterfly structure of DCT algorithms in order to maximize the number of computed coefficients at constrained computing resources. The selection of the computed coefficients depends on the applied DCT butterfly computation and on a coefficient weighting for e.g. considering subsequent quantization.

CHAPTER 3

Complexity scalable DCT computation

In this chapter, a complexity scalable DCT computation technique is presented that can be applied to any DCT algorithm that has been developed in the past, such that the amount of computed coefficients is optimized for a constrained number of computing cycles. This optimization leads to improved picture quality for video applications that have a limited computation power. For halved computing resources, about 2-4 dB PSNR improvement was obtained when compared to a diagonally oriented computation of coefficients, matching with the conventional MPEG scanning. Also the visual improvement in sharpness and readability can be clearly seen. It found that the applicability of this technique is only useful for well-constrained systems.

3.1 Introduction

The Discrete Cosine Transformation (DCT) is one of the two principal computations functions of MPEG coding. This chapter concentrates on finding

a technique to provide complexity scalability and being computationally efficient at the same time. Preferably, when computing resources are limited, the algorithm should not only scale down in complexity, but also the quality should smoothly scale with the computational effort. At the end of the previous chapter, the DCT algorithms from Section 2.5 ([44, 48, 51, 55]) already accept a loss of video quality for saving computations. However, these proposals do not scale in quality and they only offer fixed points of operation. Furthermore, the approaches do not optimize the butterfly DCT computation with respect to scalability. In this chapter, a new technique to compute the DCT is introduced, which exploits the butterfly structure of a DCT algorithm, in order to trade-off the saved computations and remaining video quality.

The DCT algorithms, as presented in Chapter 2, can be computationally optimized by analyzing the signal-flow graphs that are defined by the butterfly algorithms. Note that the technique developed in this chapter can be implemented for 1-D DCT and 2-D DCT computation. Our objective is to perform a scalable 8×8 2-D DCT, since this is part of the MPEG encoding standard. At limited computing power, a DCT algorithm should be modified by eliminating several computations and thus coefficients. Consequently, the output of the algorithm will have less quality, but the computing effort of the algorithms is reduced to fit within the given limitation in computing power. The key issue is to identify the computation steps that can be omitted to maximize the number of coefficients for the best possible video quality. However, also the visual relevance of the computed coefficients will be considered by introducing priority weighting.

The approach for maximizing the number of coefficients under constraints is driven by the following idea. If the computation of a coefficient (l_a) cannot be completed due to a given computation limit, it is attractive that instead other coefficients (l_b, l_c) having lower computational complexity are considered for computation. Although coefficient l_a may have a greater impact on the picture quality than coefficients l_b and l_c, the computation of coefficient l_a is useless if it cannot be completed. When assuming that the computation of the coefficients l_b and l_c can be completed within the given limit, the picture quality is improved, because more coefficient data have been computed.

The realization of the above-given idea is presented in Section 3.2. Section 3.3 describes some implementation aspects. In Section 3.3.3 and as an example, the new DCT computation is applied to the selected algorithms of Chapter 2. Section 3.4 presents an extension to the algorithm analysis using a priority

weighting of the DCT coefficients. The gain of quality that results from the new computation technique is shown in Section 3.5. Section 3.6 concludes this chapter.

3.2 DCT-algorithm analysis

3.2.1 Concept for the new technique

Prior to developing the algorithm analysis, some notations are introduced in the following using Figure 3.1. The figure shows a part of a butterfly diagram, where the bullet (•) and the small rectangle (□) are operation nodes of the butterfly, when going from the input pixel values to the output DCT coefficients.

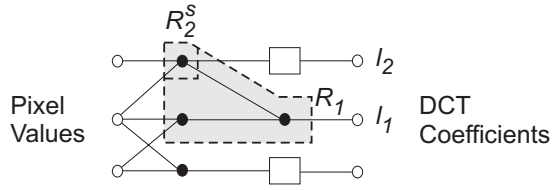

Figure 3.1: *Notations for DCT butterfly analysis.*

Let l_i be the coefficients of a sorted list L and $r(l_k)$ be a function that gives a set R_k of all the required operation nodes for a coefficient l_k (Figure 3.1 includes an example for $r(l_1) = R_1$). For convenience, let $r^s(l_k)$ be a function that gives a subset R_k^s of R_k ($R_k^s \subseteq R_k$) containing the operation nodes that have already been computed during the processing of the previous coefficients l_n with $n < k$. For example, in Figure 3.1, $r_2^s(l_2) = R_2^s$. Thus, in general, the function $r^s(l_k)$ is formed by defining the subsets R_k^s for various values of s and k such that

$$r^s(l_k) = \left\{ r(l_k) \cap \left(\bigcup_{n<k} r(l_n) \right) \right\}. \tag{3.1}$$

Since the different operation nodes involve different costs (computational effort), the following cost functions are defined. Let $op(R_k)$ be a function that returns the number of operations involved for a given set R_k of operation nodes. Furthermore, let $rop(l_k)$ be a cost function that returns the *remaining* number of operations that are required to complete the coefficient l_k, given

the fact that the coefficients l_n with $n < k$ and their intermediate results are available. The cost function $rop(l_k)$ is defined as

$$rop(l_k) = op(R_k) - op(R_k^s). \tag{3.2}$$

Equation (3.2) means that the computational complexity for the next coefficient l_k equals the overall complexity for the involved nodes, minus the number of operations for already processed nodes that are located in the computation path of the considered coefficient l_k.

3.2.2 Trade-off criterion for computations and quality

The computation path of the complete scalable algorithm is based on an overall criterion that is based on the previous equations. Prior to each computation step, a list of remaining coefficients is sorted such that in the next step, the coefficient having the lowest computational complexity is computed. More formally, the sorted list $L = \{l_1, l_2, \ldots, l_{N^2}\}$ of coefficients taken from an $N \times N$ DCT satisfies the criterion

$$\forall l_i \in L : rop(l_i) = \min\{rop(l_k) | k \geq i\}. \tag{3.3}$$

The underlying idea is that some of the intermediate results can be shared. Thus Equation (3.3) defines the minimum computational effort that is needed to obtain the next coefficient. The practical usage of this formal description including an exemplary analysis can be found in Section 3.3.

If a computation limit is given, it is preferable to compute coefficients that share computation steps. In this case, the order defined for the elements of L gives an algorithm-dependent computation order, which maximizes the number of coefficients that are computed within the computation limit. Note that using such a computation order does not change the results of a complete computation that is achieved without a computation order.

3.2.3 Priority weighting

It is known that the visual weight of DCT coefficients is not equal, because a.o. the transform concentrates energy in the low-frequency coefficients and the human visual system is frequency dependent. This is exploited in the MPEG video coding standard, where quantization and coding is performed in a zigzag scanning order of DCT coefficients. This know-how can be inserted into the design of a scalable DCT computation algorithm.

The computation order L can be perceptually optimized with *priority weighting* (see Section 3.4). For example, the weighting of the subsequent quantization stage can be incorporated, which emphasizes the use of low-frequency coefficients in the upper-left corner of the DCT coefficient matrix. The cost function $rop(l_k)$ can be combined with a priority weighting function to prefer those coefficients over high-frequency coefficients. Alternatively, coefficients can be preferred that describe special block contents, like horizontal or vertical edges (see Section 6.2.2).

The priority weighting does not exclude the choice for a coefficient having a low priority, if its computation can be completed within e.g. a single operation. Note that the computation order L is determined by the adopted DCT butterfly algorithm and the optionally applied priority weighting, and can be found in advance. For this reason, no computational overhead is required for actually computing the scalable DCT.

Within the MPEG standard, the zigzag scan is mostly used when coding DCT coefficients, to start with the most important low-frequency coefficients. If this scanning is adopted as computation order, many time-consuming computations have to be performed at the start of the complete DCT computation, in order to obtain the first coefficients. The reason is that these coefficients depend on different inputs and intermediate results cannot be reused. Hence, for reduced computation power, the zigzag order would result in less coefficients to be used afterwards.

It will be shown that at the start of the DCT decomposition, the computation of sufficient number of coefficients is important for obtaining a reasonable quality. Therefore, finding the best computation order is useful.

3.3 Implementation aspects of a fast-DCT algorithm analysis

3.3.1 Database construction

To identify the required number of operations for specific DCT coefficients, data dependencies between operation nodes within the algorithm should be explored first. Using this information, a *database* can be constructed containing every computation step, when going from the input values to the transformed coefficients at the output. The database is used to track all computations involved for a specific coefficient, and when combined with the cost

function given in Equation (3.2), it additionally provides an overview about the computations that are still required to complete the calculation of the remaining coefficients.

Afterwards, the database can be used to efficiently implement either a *dynamically-scalable* or a *dedicated-scaled* version of the adopted fast DCT algorithm.

- **Dynamically-scalable DCT computation**
  A dynamically-scalable version of the DCT algorithm allows adaptation of the computational complexity as a function of the current CPU usage in multi-tasking systems. For this reason, a dynamically-scalable video application is able to uphold real-time execution, albeit at a lower quality, despite the parallel execution of other tasks.

  This feature needs additional memory for keeping intermediate results of the computations for potential reuse. This extra memory is negligible compared to the memory needed for intermediate storage of video frames. It was found by experiments that up to 94 intermediate results are stored during computation of an 8×8 DCT, thus 1.5 times the data amount of a DCT block.

  Additional bandwidth (to external memory) is not required for these intermediate results, provided that a local cache is available for the algorithm. Dedicated and/or programmable hardware can easily implement such a cache. General-purpose processors that may not provide control over their cache require at most a cache size of 2 kB for the temporary values to prevent cache misses (and thus expensive access to external memory). Since only a small cache capacity is needed, the implementation will not be a problem and bandwidth is reduced with each coefficient that is not computed.

- **Dedicated-scaled DCT computation**
  If a fixed number of operations is selected prior to implementation, a dedicated-scaled version of the algorithm is chosen. In this case, the algorithm is optimized for a certain set of coefficients given by the database. The implementation of a dedicated-scaled algorithm automatically uses less computation power, resources and I/O (bandwidth), because less coefficients and intermediate results are computed, stored in memory or transferred from or to memory, compared to a full DCT implementation.

The remainder of this section gives an example for the structure of the above-mentioned database and construction details.

3.3.2 Construction of a database for analysis results

The construction of the database is a key for the scalable DCT computation introduced in Section 3.2. The structure and maintenance of the database is explained with an example portrayed by a typical part of a butterfly diagram in Figure 3.2. This example shows a computation with six operation nodes, where three nodes are intermediate results (ir_1, ir_2 and ir_3). The arrows in the diagram stand for sign inversion, and the box represents a multiplication with the constant c_i indicated inside the box.

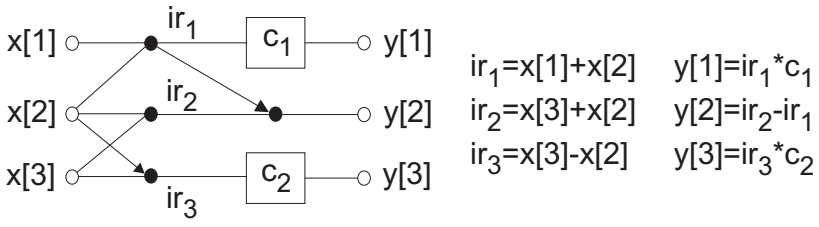

Figure 3.2: *Computation of output vector $\underline{y}$ based on input vector $\underline{x}$.*

The computational complexity for coefficients $y[1]$, $y[2]$ and $y[3]$ are determined by counting all signal-processing operations required for computing each of the coefficients, starting from the input vector $\underline{x}$. From Figure 3.2 it can be seen that

$$\begin{array}{lcll} y[1] & = & ir_1 * c_1 & = & (x[1] + x[2]) * c_1 \\ y[2] & = & ir_2 - ir_1 & = & (x[3] + x[2]) - (x[1] + x[2]) \\ y[3] & = & ir_3 * c_2 & = & (x[3] - x[2]) * c_2. \end{array} \tag{3.4}$$

In this example, the computation of e.g. coefficient $y[1]$ requires the result of node ir_1. The computation then consists of one addition (within ir_1) and one multiplication (c_1). Such computing information is stored in a database as given in Table 3.1. At the left side of the table, the operands indicate the inputs required for an intermediate result, and at the right side of the table, which intermediate results are needed to generate an output component.

This approach can be adapted in a flexible way to specific processor architectures as indicated below.

Operation	ir_1	ir_2	ir_3	$y[1]$	$y[2]$	$y[3]$
Additions	1	1	1	0	1	0
Multiplic's	0	0	0	1	0	1
Operands	$x[1]$, $x[2]$	$x[2]$, $x[3]$	$x[2]$, $x[3]$	ir_1	ir_1, ir_2	ir_3

Table 3.1: *Database of computing information.*

- **Definition of operations**
  Besides additions and multiplications, the database of computational complexity can also contain more complex operations, if a specific processor architecture is considered that supports such operations[1]. In addition, memory bandwidth aspects can be considered by defining memory-access operations if necessary. For the example in Figure 3.2, this would mean that a processor architecture is defined, such that sufficient local cache is available that can be accessed without cost. Note that the additional memory needed for the intermediate results is negligible compared to the memory needed for intermediate storage of video frames.

- **Definition of operation complexity**
  The complexity of the operations that are used in the database have to be defined. In the following, the computational complexity of one multiplication is assumed to be equivalent to three additions, thereby relating to simple processors or delay times of hardware components in VLSI designs. Note that this can be modified according to any desired processor architecture and its characteristics, which is not further elaborated (when using this assumption in experiments, its influence on the results will be considered).

In this example, the computational complexity of the coefficients shown in Figure 3.2 is computed as follows.

$$\begin{array}{lcllcll} op(y[1]) & = & op(ir_1) + 1 * 3 & = & 1 + 1 * 3 & = & 4, \\ op(y[2]) & = & op(ir_1) + op(ir_2) + 1 & = & 1 + 1 + 1 & = & 3, \\ op(y[3]) & = & op(ir_3) + 1 * 3 & = & 1 + 1 * 3 & = & 4. \end{array} \tag{3.5}$$

[1] In the Philips TriMedia processor, an example of a special multi-media operation is the $quadavg$ operation, which averages four pairs of samples in one instruction cycle. The processor contains a list of various types of special multi-media processing operations.

With Equation (3.5) it is found that the sorted coefficient $l_1 = y[2]$ will be computed in the first step, because it requires the least number of operations. The remaining coefficients $y[1]$ and $y[3]$ have the same computational complexity, so that at first glance no difference can be seen with respect to their computation order. However, from Figure 3.2 it can be noticed that coefficients $y[1]$ and $y[2]$ share node ir_1. According to the notations from Section 3.2.1, this leads to the following definitions

$$\begin{array}{lcl} R(y[1]) & = & \{ir_1\}, \\ R(y[2]) & = & \{ir_1, ir_2\}, \\ R(y[3]) & = & \{ir_3\}. \end{array} \tag{3.6}$$

In the second step, the cost function $rop(i)$ indicates less remaining computational complexity for $y[1]$ than for $y[3]$, because

$$\begin{array}{lcl} rop(y[1]) & = & op(y[1]) - op(R(y[1]) \cap R(y[2])) \\ & = & op(y[1]) - op(\{ir_1\}) \\ & = & 4 - 1 = 3, \\ rop(y[2]) & = & op(y[3]) - op(R(y[3]) \cap ((R(y[1]) \cup R(y[2])) \\ & = & op(y[3]) - op(\{ir_3\} \cap \{ir_1, ir_2\}) \\ & = & op(y[3]) - op(\emptyset) \\ & = & 4 - 0 = 4. \end{array} \tag{3.7}$$

As a conclusion, it is preferable to compute the given coefficients in the order $L_A = \{y[2], y[1], y[3]\}$, which satisfies the condition from Equation (3.3). If the computation power would be reduced to six operations for this example, the first two coefficients $y[2]$ and $y[1]$ can be computed. With a computation order of $L_B = \{y[2], y[3], y[1]\}$, only $y[2]$ could be computed, because in this case the first two coefficients $y[2]$ and $y[3]$ together need seven operations. Therefore, it is obvious that order L_A leads to a higher picture quality than order L_B.

3.3.3 Algorithmic example

The computation technique from Sections 3.2 and 3.3 has been used to find an scalability-optimized computation order for a *dynamically-scalable* version of the 2-D DCT algorithm by Cho and Lee (ChoLee, see Section 2.3.2), when inserting the 1-D DCT algorithm by Arai, Agui and Nakajima (AAN, see Section 2.3.3). Both algorithms were adopted, because their combination provides a highly efficient DCT computation (see Section 2.4). The derivation

of the computation order is presented in Appendix A, in order to avoid lengthy discussions. This derivation includes a weighting of coefficients for enhancing the obtained computation order, which is presented in the next section.

3.4 Enhancements using priority weighting

The computation order can be improved, if the coefficient weighting of the quantization stage after the computation of the DCT is considered. The quantizer weighting emphasizes the use of low-frequency coefficients in the upper-left corner of the coefficient matrix. Therefore, a priority weighting is incorporated into the computation technique to favor those coefficients. In Figure 3.3(a), the scalability-optimized computation order is shown that is found with the algorithm analysis that was presented in Section 3.2, based on the assumptions that are made in Section 3.3.3 and its corresponding Appendix A. Figure 3.3(b) is a modified computation order with an additional priority for the upper-left corner of the coefficient matrix. This modified computation or-

a)

	0	1	2	3	4	5	6	7
0	1	49	9	53	3	58	11	50
1	17	33	22	41	27	47	31	39
2	10	54	5	61	15	56	7	64
3	21	45	29	37	19	35	25	44
4	4	59	12	51	2	52	13	60
5	26	43	24	36	20	38	30	46
6	14	63	8	57	16	62	6	55
7	18	40	32	48	28	42	23	34

b)

	0	1	2	3	4	5	6	7
0	1	33	9	41	5	44	14	36
1	17	49	21	57	29	63	31	55
2	10	37	3	42	11	39	7	48
3	25	61	26	53	18	51	24	60
4	6	45	15	34	2	35	16	46
5	28	59	23	52	19	54	27	62
6	12	47	8	40	13	43	4	38
7	20	56	32	64	30	58	22	50

Figure 3.3: *Scalability-optimized computation orders of coefficients in a DCT matrix using priority function (b) or not (a).*

der was found by multiplying the cost function given in Equation (3.2) with a priority weighting as given by

$$p(i,j) = 2(i+j) + |i-j| + 1. \tag{3.8}$$

The arguments i and j denote the position of the coefficient in the DCT block, and are specified for $0 \leq i, j \leq N-1$. Note that this function results in lower numbers for coefficients that are weighted with higher priority. The priority function $p(i,j)$ was found experimentally and considered suitable for a first implementation.

Table 3.2 shows the effect of the proposed priority weighting. The resulting computation order is shown in Figure 3.3(b). Once more, the computational

complexity of one multiplication is assumed to be equivalent with three additions. Let us suppose that the first two coefficients $x[0,0]$ and $x[4,4]$ and their intermediate results are already available. It is clear that the next coefficient to be computed is $x[0,4]$ without using priority weighting, and $x[2,2]$ when using priority weighting with function $p(i,j)$.

	$x[0,4]$	$x[2,2]$
Additions left to compute coefficient	7	9
Multiplications left to compute the coefficient	4	4
Operations count - without priority weighting	**19**	22
Priority function $p(i,j)$	13	9
Operations count - scaled with priority weighting	247	**189**

Table 3.2: *Example for the decision of the next coefficient to be computed.*

3.5 Experimental results

3.5.1 Computational results of the experiments

The 2-D DCT algorithmic example of the previous section has been used to illustrate the effect of the analysis for finding a computation order. Note that the results in this section concentrate on the scalability of the DCT alone, and not about its influence in a full MPEG encoder. The latter aspects will be discussed in Chapter 6.

Figure 3.4 shows the result of the analysis under different conditions. The two matrices on the left side no not apply priority weighting, while the two matrices on the right side are found with using priority weighting, by assigning a higher priority to the coefficients in the top-left corner (see Equation (3.8)). The variable $ratio_{ma}$ in the figure indicates the computational complexity of a multiplication relative to an addition. The parameter $ratio_{ma} = 1$ was chosen for high-end CPUs and $ratio_{ma} > 1$ for low-cost systems. The matrices have been shaded with different gray levels to mark the first and the second half of coefficients in the sorted list. It can be noticed that the proposed computation order is optimized for the chosen processor architecture. Note that for e.g. $ratio_{ma} = 1$, the column-based structure of Figure 3.4(a) and (d) is similar, but the order of the coefficients is slightly different.

The experiments show that the computation order in Figure 3.4(c) uniformly selects the coefficients from the matrix. Therefore, this order can be used

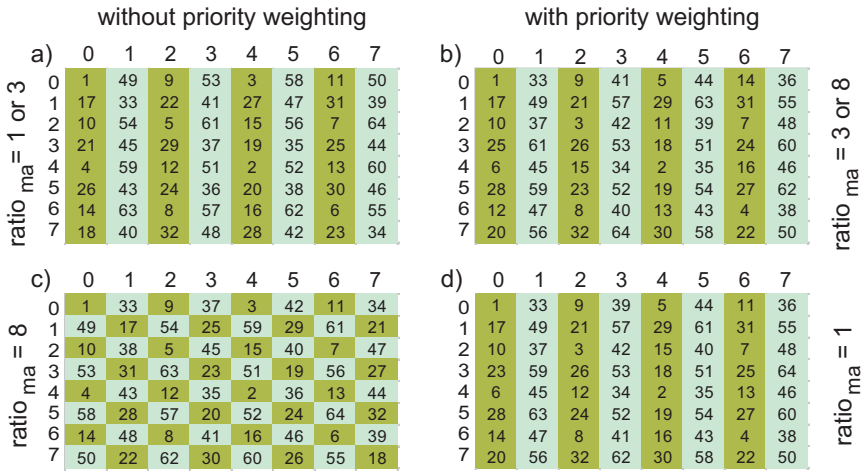

Figure 3.4: *Different computation orders of coefficients.*

for blocks that have no clear structure. The computation order shown in Figure 3.4(a,b,d) clearly favors vertical detail. It is emphasized here that the order (b) and (d) were found using a priority weighting by assigning a higher priority to the coefficients in the top-left corner. This can be verified for the $ratio_{ma} = 3$ case, because the number of coefficients in the top-left corner of the DCT matrix using computation order (b) is equal or higher than using order (a) that is build without priority weighting. Note that if a zigzag order would be preferred right from the start, another DCT algorithm would be selected for optimization of the computation and may yield better results (low-cost operations have been used as a primary requirement in the previous section).

The operations count for a given number of coefficients using the MPEG-zigzag order has been compared with the scalability-optimized order given in Figure 3.4(b). For comparison, lets us assume that one addition counts for one operation ($op('+') = 1$) and a multiplication counts for three operations ($op('\times') = 3$). The resulting number of coefficients of the comparison are given in Figure 3.5. It can be noticed that the proposed scalability-optimized order leads to significantly more computed coefficients, resulting in an improved picture quality. It has been verified that the main characteristic of the curves does not change with other addition-to-multiplication complexity ratios (e.g. $ratio_{ma} = 1$), although the computation order varies.

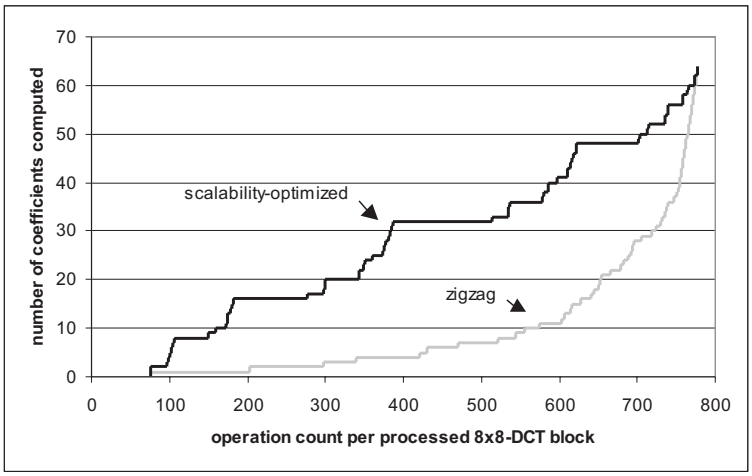

Figure 3.5: *Zigzag-ordered computation vs. scalability-optimized-ordered computation.*

3.5.2 Pictorial results of the experiments

The Peak-Signal-to-Noise Ratio (PSNR) improves between 1-5 dB, depending on the amount of available operations. This is for example shown for the first frame of the “Voit” sequence in Figure 3.6. Figure 3.7 shows the improvements in PSNR difference (PSNR gain) using a scalability-optimized order instead of the zigzag order. This is shown using a frame of the sequence “Voit" and the sequence “Renata". The lowest "Renata" curve in the figure is caused by including a standard MPEG quantization stage. Note that for computing the PSNR values, a perfect IDCT was used, but its computational effort is not included in the figures.

It can be seen in Figure 3.6 that the PSNR values increase slowly in the beginning of the computation and augment faster at the end. However, the perceptual quality increases inversely, because at the start, every additional coefficient contributes to a much higher quality. As no quantization was used, the PSNR grows significantly at the end of the computation, although the perceived video quality hardly improves. A standard quantization stage that employs the default MPEG quantizer-weighting matrix for intracoded frames [56] would limit the PSNR of the images to 32 dB for the standard-definition images (720×576 pixels) used in the experiments. However, the shape of the curves shown in Figure 3.7 does not change, although the PSNR

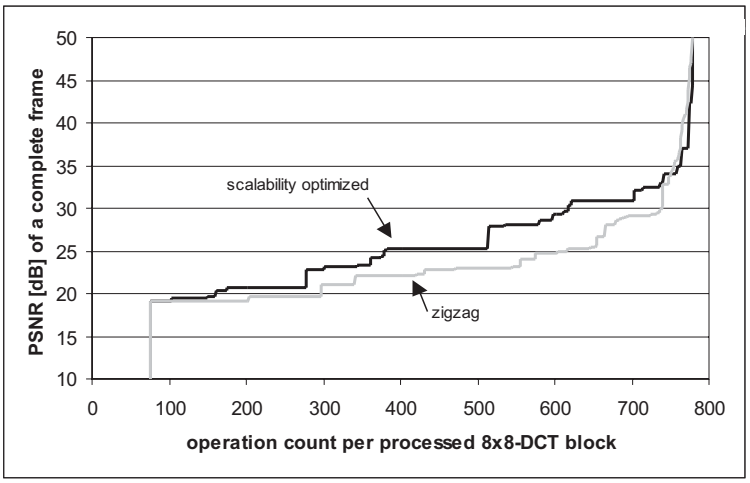

Figure 3.6: *PSNR obtained as a function of the available number of operations (1^{st} frame of the "Voit" sequence).*

differences are reduced when quantization is added. As an example, Figure 3.7 contains two curves from the "Renata" sequence, where the (bottom) light gray curve is caused by a standard MPEG quantization.

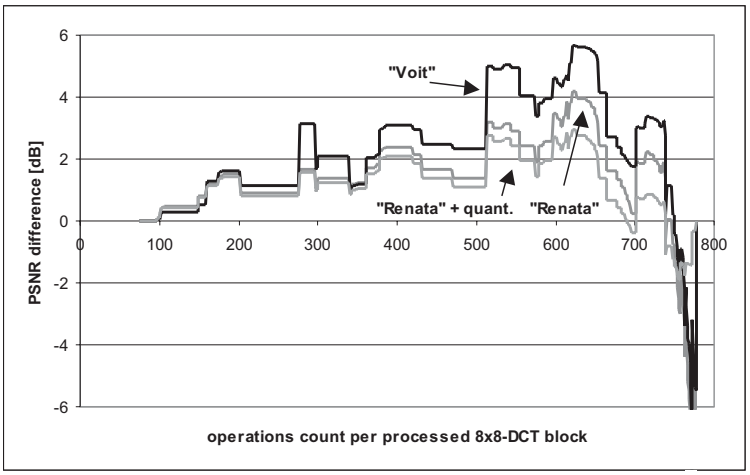

Figure 3.7: *Improvement expressed in PSNR difference.*

Figure 3.8 shows two image pairs (based on zigzag and scalability-optimized order) sampled during different stages of the computation (representing low-cost and medium-cost applications). The index $n_1(n_2)$ in the bottom-right corner of each image indicates that n_1 operations are spent for the computation of n_2 coefficients. Perceptive evaluations of the experiments have revealed that the quality improvement of the scalable DCT computation technique is the largest between 200 and 600 operations per block. In this area, the amount of coefficients is still relatively small, so that the benefit of having much more coefficients computed as compared to the zigzag order, is fully exploited. Although the zigzag order yields perceptually important coefficients from the beginning, the computed number is simply too low to show relevant details. For example, for the well-known "Stefan" sequence (tennis scene)

Figure 3.8: *Visual results from the scalable DCT using the "Renata" sequence.*

and the "Foreman" sequence (construction site), the zigzag-ordered computation blurs all details like persons, trees and text, whereas the scalable DCT presented in this chapter preserves significant details. This can be seen in Figure 3.9, which shows the visual results of an experiment using the previously

mentioned two sequences. A quantization factor of $qscale = 8$ was used in this experiment.

At more than 600 operations per block, the perceptual quality becomes comparable. In this case, the designer can trade-off blurring of data (occurring in the zigzag computation) against high-frequency blockiness (the scalable proposal). The technique presented in this chapter is particularly attractive for use with small displays, because a good sharpness impression is achieved at an early stage.

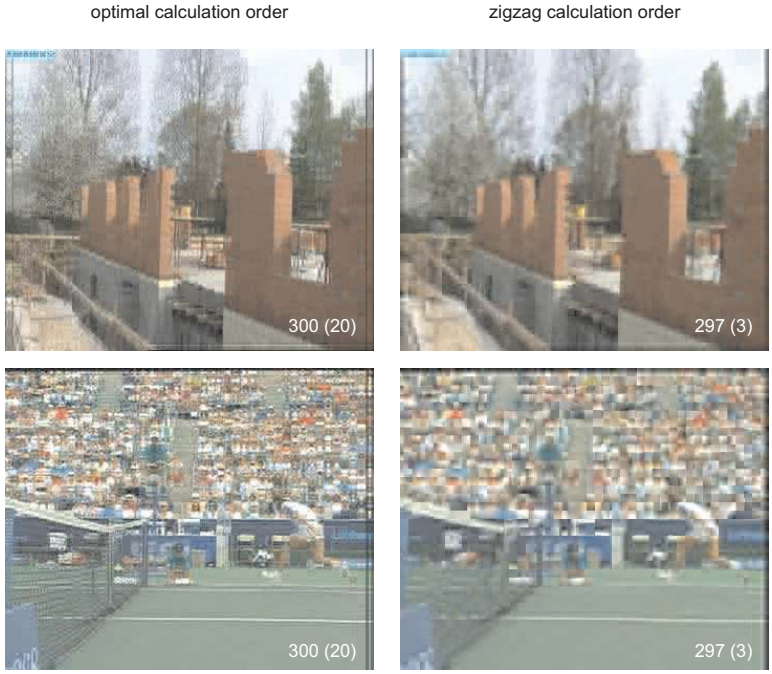

Figure 3.9: *Visual results from the scalable DCT using the "Foreman" sequence (top) and the "Stefan" sequence (bottom).*

3.6 Discussions and conclusions

In this chapter, a new scalable DCT computation technique has been developed for optimizing the number of coefficients that can be computed when

computation resources are constrained. The main idea behind the development of the new technique is that computations should be spent on coefficients that can be fastly computed and provide additional picture quality in terms of PSNR. Other coefficients may have a greater impact on the quality, but would need computation resources that may not be available. Priority weighting of the coefficients is possible to consider e.g. subsequent quantization. The results that are presented in this chapter can be generalized for other algorithms, like the Inverse Discrete Cosine Transformation (IDCT).

Section 3.3 has provided the construction of a database of computing information that includes the involved operations and data dependencies for each operation node and DCT coefficient. The type of involved operations can be adapted to the target architecture. Based on the database, a dynamically-scalable or dedicated-scaled version of a basic DCT algorithm can be implemented. The dynamically-scalable version allows adaptation of the computational complexity, since its execution allows interruption or stopping of computations at any time. The achieved number of computed coefficients depends on the available execution time. The dedicated-scalable version is optimized for a preselected number of coefficients, thereby minimizing for example memory requirements.

The dynamically-scalable version of the DCT computation technique should be used in cases where the amount of the computing power is not known in advance or can vary over time. If the limitation in computing power is predetermined, then the dedicated-scaled version can lead to better results, provided that the database is properly used to trade-off the predefined available computations with the selection of DCT coefficients that would be achievable.

The experimental results show that the scalability-optimized computation order leads to significantly more computed coefficients compared to a computation order based on the zigzag scan that is used in MPEG. At the same time, the picture quality is improved by 1-5 dB PSNR, depending on the available amount of computations. Visual results show an improved sharpness impression of the scalability-optimized ordered computation, which is particularly attractive when using small displays. This is shown in Figure 3.10, where the first frame of the “Renata” sequence has been coded and displayed at different sizes and using both computation orders. The DCT computation for this experiment was limited to 300 operations, thereby representing low-cost applications. The frames in the middle are coded at full broadcast TV resolution. When going to the top in the figure, the display size is reduced for these

frames. It can be seen that the enhanced sharpness impression is maintained when reaching the top of the figure, where the display size is QCIF, thereby representing e.g. a GSM-phone display size. When going to the bottom, the frame was directly coded at lower resolution. More artefacts can be seen here, where the scalability-optimized computation order shows more coded details than using the zigzag order.

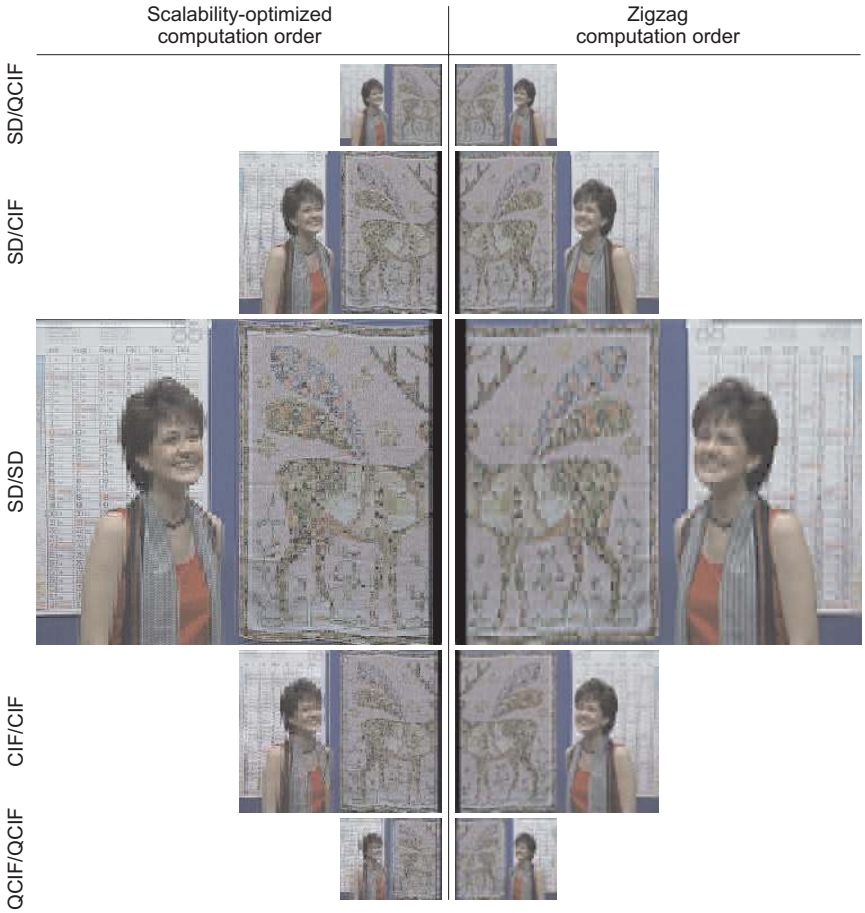

Figure 3.10: *Compare of coding and displaying a "Renata" frame in different sizes, which is indicated at the left side of the figure (encoding size/displaying size).*

When looking back to the design of the proposed scalable DCT technique in this chapter, the major criterion for computational complexity scalability was sharing of operations in the computation of two or more coefficients, so that the decision criterion strongly emphasizes the number of DCT coefficients computed after transformation. In hindsight, an alternative criterion for selection of coefficients could have been chosen. For example, a desirable criterion with respect to picture quality could be to select the coefficients that contribute mostly to the PSNR. This is clearly a generalization of the research results of this chapter that is worthwhile to consider in future research. With respect to power consumption, a criterion could be chosen that minimizes the number of memory accesses, because they contribute most to the power usage of a system.

The technique presented in this chapter can also be extended in the dimension of content-adaptive scalable DCT computation. This means that incoming pixel blocks are classified with respect to their contents and a specialized DCT that is optimized for the type of content contained, is chosen. This concept will be further elaborated in Section 6.2.2. This enhancement increases the probability that the scalable DCT selects coefficients that are important for describing the block content.

The proposed DCT computation in this chapter is an overall management technique for controlling the amount of operations, in which existing fast DCT-butterfly diagrams are inserted. With the aid of our proposal, these diagram computations are optimized with respect to the amount of operations, if the computation power is limited. Usage of alternative fast DCT algorithms will give similar, but different results. Thus the selected combination of DCT butterfly algorithms is efficient and attractive, but there may be another combination that yields a (slightly) better quality at the same condition.

It should be noted that the proposed technique is fully compliant with the MPEG standard. The computation order for computing the DCT coefficients should not be confused with the scan order for coding the coefficients. During the scalable DCT transformation, the meaning of the intermediate results, i.e. computing a coefficient matrix, is not different from performing conventional DCTs. However, the selection of coefficients for computation leads to different coefficient patterns for quantization and VLC coding, so that there is an influence on the coding efficiency. At this stage, we still do not have a complete scalable system, thus the influence on the coding efficiency is discussed later in this thesis.

CHAPTER 4

Motion estimation

In the past decades, numerous publications have appeared about Motion Estimation (ME). The purpose of this chapter is to analyze a few popular ME algorithms and evaluate their efficiency. ME algorithms require a criterion for selecting the best motion vector from a set of vector candidates. The analysis in this chapter includes three commonly used criteria. At the end of this chapter, an algorithm and criterion having a high efficiency are selected to design a scalable ME subsystem that will be discussed in the next chapter.

4.1 Introduction

Motion Estimation (ME) is the most computationally intensive function of MPEG encoding. This chapter describes briefly the basic principles of several ME algorithms, thereby serving as a partial literature overview. A simple comparison of their quality and computational complexity is provided for the algorithms. The two metrics, quality and complexity, are merged into an efficiency indicator in order to make a choice which algorithm should serve as a basis for further research. The chosen algorithm efficiently performs the ME, while giving near-optimal quality.

ME in an MPEG encoder is used for finding a block in a reference frame that closely resembles the actual block that has to be coded. A close resemblance can be expressed by subtracting two blocks (the reference and the actual block) on a pixel-by-pixel basis. This process is called block matching. The computation of block differences can be seen as an error measure. Two blocks having the lowest error measure are then considered as the best match. Besides evaluating all block combinations in a predefined search area in the reference frame, in the past few years, fast algorithms have been proposed to decrease the number of block evaluations significantly.

This chapter presents some error measures that are used as criteria for finding block matches in Section 4.2. Section 4.3 presents a number of fast ME algorithms, in which a small number of vector candidates is used to come to a best match. For comparison, full search is also discussed. The algorithms lead to motion vectors that refer to the best block matches based on the full-pixel grid. The accuracy of the vectors can be enhanced by refining the vectors to the half-pixel grid as presented in Section 4.4. A comparison of the ME algorithms is given in Section 4.5. It should be noted that the comparison of algorithms is not exhaustive, but based on a few popular proposals from literature.

4.2 Block-matching criteria

The ME process in MPEG systems divides each frame into rectangular macroblocks (16×16 pixels each) and computes motion vectors (MVs) per block. As shown in Figure 4.1, a MV $mv(dx, dy)$ signifies the displacement of a block $b(x_0, y_0)$ in the x-y pixel plane with respect to a reference frame. The coordinates x_0 and y_0 refer to the top-left most pixel of the block. The purpose of this block-matching process is to find a block $b^*(x_0 + dx, y_0 + dy)$ in the reference frame that fits best for the current block b.

The error measure that is computed for each MV candidate involves various ways of pixel difference computations. The Mean-Square-Error and Sum-of-Absolute-Differences are commonly used in video coding and therefore discussed. As an alternative for low complexity, the Minimum Maximum Error [57] has been added. These three measures are briefly presented below.

- The Mean-Square-Error (MSE) is computed similar to the PSNR and is based on the sum of squared pixel differences. For this reason, the

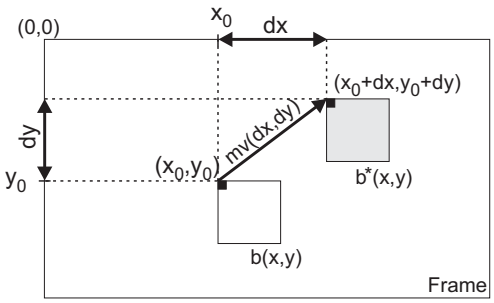

Figure 4.1: *Block matching of the ME process.*

MSE can result in a higher PSNR of the predicted frames as compared to other measures when using the same MV candidate set.

$$MSE(b,b^*) = \frac{1}{16^2}\sum_{x=0}^{15}\sum_{y=0}^{15} \left(b(x+x_0,y+y_0) - b^*(x+x_0+dx,y+y_0+dy)\right)^2 . \tag{4.1}$$

- The Sum-of-Absolute-Differences (SAD) is the mostly used measure, because it is simple to compute and comparable in performance to MSE.

$$SAD(b,b^*) = \frac{1}{16^2}\sum_{x=0}^{15}\sum_{y=0}^{15} \left|b(x+x_0,y+y_0) - b^*(x+x_0+dx,y+y_0+dy)\right| . \tag{4.2}$$

- The Minimized Maximum Error (MiniMax) [57] is the most simple criterion of the three presented, because it only involves comparisons.

$$MiniMax(b,b^*) = \max_{0\le x,y\le 15} \left|b(x+x_0,y+y_0) - b^*(x+x_0+dx,y+y_0+dy)\right| . \tag{4.3}$$

The computational complexity of one error measure can be reduced by limiting the processing to a subset of all block pixels. In [58] it was found that the quality of the error measure hardly suffers from sub-sampling. Block sub-sampling has been applied for example in [59] for complexity scalable ME. Another technique that is presented in [60] predicts the result of a complete error measure computation from intermediate results that are based on block sub-sampling. Appendix C.2 shows that the computational complexity of the

error measure can be lowered by reducing the bit representation of pixels. This is best exploited in hardware solutions, which is not in the scope of this thesis and therefore not further considered.

4.3 Fast ME algorithms

In this section, algorithms are presented that have been frequently referred to in literature. These algorithms are the one-dimensional full search, block-based gradient descent search, diamond search, three-step search, new three-step search and simple recursive ME. For reference, the discussion starts with the full-search ME, which considers all possible candidates in the search area. A popular ME algorithm that is not found in the list is the 3-D recursive search (3DRS) block matcher [61, 62]. The 3DRS is mainly developed for frame rate up-conversion, where true-motion estimation is required for interpolating frames from each pair of successive frames of the video sequence. True-motion estimation is not required for video coding, where coding efficiency is more important. However, a simplified version of 3DRS is used for evaluations.

4.3.1 Full Search (2DFS)

The simplest way to find the optimal MV for a given search area is to compute the error measures for all possible block comparisons. The search area is bounded by a maximum allowed block displacement of $\pm dx_{max}$ in horizontal direction and $\pm dy_{max}$ in vertical direction. The corresponding MV candidates that are evaluated range from $mv = (0, 0)$ to $mv = (\pm dx_{max}, \pm dy_{max})$ (see Figure 4.2). Commonly, $(n \times n)$ search areas are implemented with $n = 2 * dx_{max} - 1$ and $dx_{max} = dy_{max}$, exploiting less MVs for positive-integer displacement. The MV that leads to the minimum error is adopted as best match. The drawback of this search strategy is the high computational effort.

4.3.2 One Dimensional Full Search (1DFS)

This search strategy [63] is similar to the 2DFS, except that it exploits only one dimension at a time and performs two steps (see Figure 4.3(a)). In the first step, the error measures for all motion vectors $mv = (dx, dy)$ with $dy = 0$ are computed. This will result in a vector mv_{1a}, pointing to the position $1a$ of the best-matching block in the horizontal bar. Afterwards, the error measures of

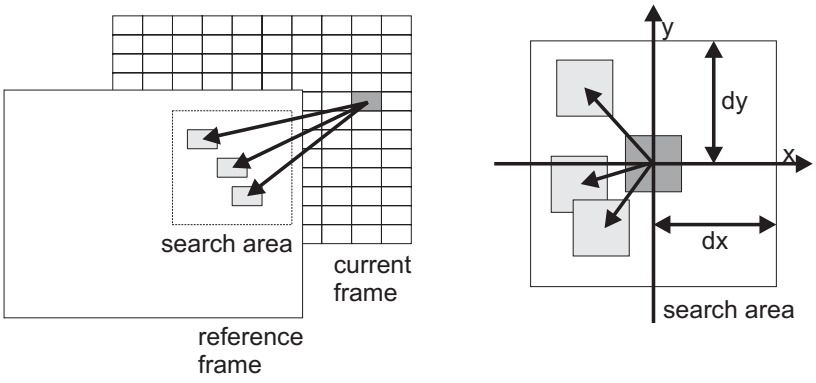

Figure 4.2: *Principle of full-search motion estimation.*

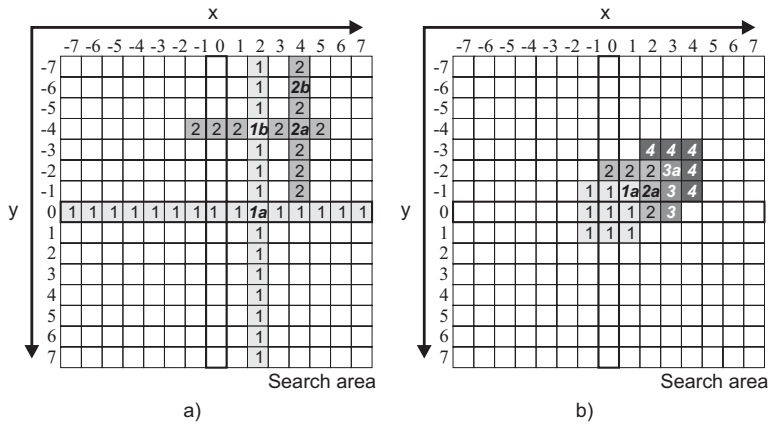

Figure 4.3: *Example of 1DFS motion estimation (left) and block-based gradient descent search (right).*

all motion vectors $mv = mv_{1a} + (dx, dy)$ with $dx = 0$ are computed, which results in a vector mv_{1b}. This is repeated in a second step with halved horizontal and vertical dimensions leading to the final MV mv_{2b}. In the Figure 4.3(a), $mv_{2b} = (4, -6)$ is the result of the 1DFS.

4.3.3 Block-Based Gradient Descent Search (BBGDS)

This algorithm [64] starts with computing the error measures for all motion vectors $mv_{msa} = (dx, dy)$ with $dx, dy \in \{-1, 0, +1\}$, thereby defining a 3×3 mini-search-area. The algorithm performs several steps, where in each step i, the algorithm computes the best vector mv_i of such a 3×3 mini-search-area, which is centered on the position that is referred by mv_{i-1}. Thus the vectors $mv = mv_{i-1} + mv_{msa}$ are evaluated for finding mv_i. Figure 4.3(b) shows an example of the search process, where four possible steps are visualized. The vectors mv_i refer to the positions i that are marked with a subsequent "a" . The algorithm terminates when the position of the best-matching block is found in the center of the mini-search-area, which means that $mv_i = mv_{i-1}$ (referring to position $3a$ in the example given in Figure 4.3(b)).

4.3.4 Diamond Search

The diamond search [65] is similar to the BBGDS algorithm (see Section 4.3.3), as both algorithms evaluate nine MVs in each step and center their correspond-

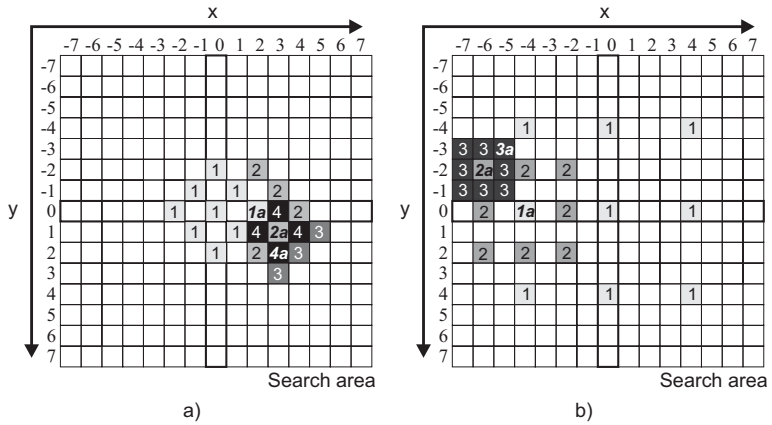

Figure 4.4: *Example of diamond (left) and three-step (right) search.*

ing mini-search-area on the position that is referred to by the best MV of the previous step. The difference between the algorithms is the shape of the mini-search-areas, which is for the diamond search equal to a diamond shape (see start positions 1 in Figure 4.4(a)). A further but smaller difference in finding the best MV is that in case of identical MVs for two succeeding steps, the MVs referring to positions of the inner diamond shape (see end positions 4 in Figure 4.4(a)) are evaluated in addition.

4.3.5 Three Step Search (TSS)

The TSS algorithm [66] is closely resembling the BBGDS algorithm (see Section 4.3.3), because in each step, nine MVs are evaluated, and the covered search area is centered around the position of the best candidate of the previous step. Two main differences are that TSS performs three fixed search steps and the distance between the evaluated MVs decreases with a factor of two in each step. In each step $i \in [1..3]$, the best vector mv_i is found by evaluating the vectors $mv = mv_{i-1} + mv(dx, dy)$ with $dx, dy \in \{-2^{n-i}, 0, 2^{n-i}\}$ and $mv_0 = (0, 0)$. An example is portrayed by Figure 4.4(b).

4.3.6 New Three Step Search (NTSS)

NTSS is globally equal to TSS, except for the first step, where additional eight MVs with $mv = (dx, dy)$ with $dx, dy = \pm 1$ surrounding the zero vector are evaluated. In case that the best MV of the first step is close to the zero vector, one additional step is performed for refinement, where eight vectors around the best MV of the first step are evaluated. Otherwise, the algorithm continues like the TSS algorithm (see Section 4.3.5). An example of this algorithm is given in Figure 4.5, where the first minimum error measure was found near the zero vector.

4.3.7 Simple Recursive Motion Estimation (simple RME)

This algorithm is described in somewhat more detail, because it forms the basis of the new algorithms explained in the next chapter. Based on the work of [67], the simple RME was developed for coding high-quality television signals [68]. The algorithm is based on the fact that motion in a video sequence is in most cases either caused by camera movement or moving objects in the video scene. Both types of motion generate a fluent (coherent) field of motion vectors over the image. The result is that the motion vectors of neighboring blocks are (almost) identical. For this reason, the probability is high

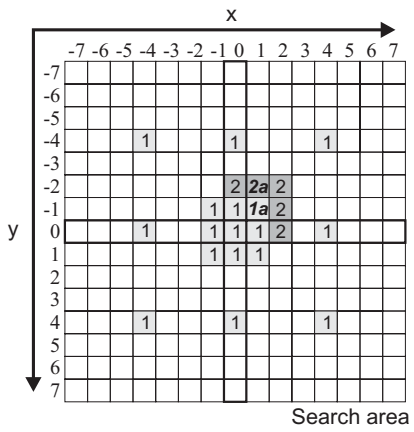

Figure 4.5: *Example of new three step search.*

that a vector is similar to one of the neighboring vectors, and therefore, vector candidates are predicted form previously processed blocks.

The prediction is visible in the first phase of the simple RME algorithm, where the MVs of the previous and upper three neighboring blocks are evaluated. Thus, the candidate vectors are equal to the adopted MVs of the blocks $b(x_b - 1, y_b)$, $b(x_b - 1, y_b - 1)$, $b(x_b, y_b - 1)$ and $b(x_b + 1, y_b - 1)$. The blocks are processed in a block-by-block and line-by-line order (see Figure 4.6). The variables x_b and y_b indicate the block coordinates of the currently processed block. In addition to these blocks, the zero MV is evaluated, since many blocks have a small motion component only. The best vector of the first phase that comes with the minimum error measure is labeled as mv_{1a}.

In the second phase, the MV prediction is enhanced by evaluating all MVs that refer to the positions of the 3×3 mini-search-area centered on the position that is referred to by the best candidate mv_{1a} of the first phase. In addition, four vectors referring to larger distances than vector mv_{1a} are evaluated for considering the case of sudden motion change. A sudden motion change occurs for example at the edge of objects that move differently from the background. The vectors "look around" in alternating directions, in order to detect a motion change as soon as possible. These additional four vectors are

$$mv = mv_{1a} + mv(dx, dy) \text{ with}$$

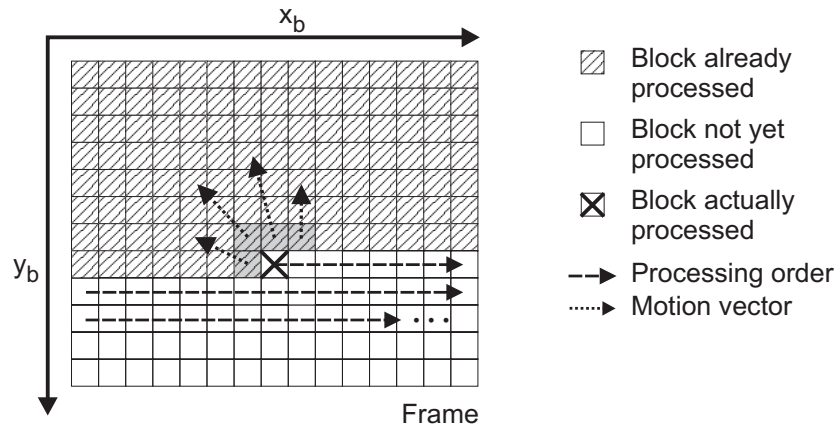

Figure 4.6: *Processing of macroblocks in simple recursive motion estimation.*

$$(dx, dy) = \begin{pmatrix} (-1)^n * k_1 \\ 0 \end{pmatrix}^T \vee \begin{pmatrix} 0 \\ (-1)^n * k_1 \end{pmatrix}^T \vee \begin{pmatrix} (-1)^{n+1} * k_2 \\ 0 \end{pmatrix}^T \vee \begin{pmatrix} 0 \\ (-1)^{n+1} * k_2 \end{pmatrix}^T .$$

where n is the horizontal block index. Suitable values for the constants k_1 and k_2 are $k_1 = 2$, and for SDTV scenes, $k_2 = 4$. The best vector of the evaluated vectors in phase two is labeled mv_{2a}.

The first two phases of this algorithm are visualized in Figure 4.7, where the indicated MVs are taken from the surrounding blocks that are already processed, and the MV that points to position $1r$ marks the vector that is taken from the same block in the previous frame (temporal candidate).

The last phase performs a vector refinement based on half-pixel accuracy, where $mv = mv_{2a} + mv(\pm 0.5, \pm 0.5)$. Vector refinement is an additional step that is suitable for all ME algorithms (see Section 4.4).

Finally, the simple RME algorithm includes a quantized SAD as block-matching criterion as given in Equation (4.4).

$$SAD_{RME} = \frac{SAD}{2^q} \tag{4.4}$$

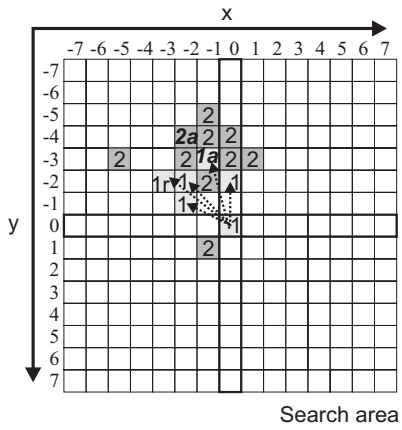

Figure 4.7: *The first two phases of simple recursive motion estimation.*

Typical values for the exponent q are $q \in \{7, 8, 9\}$. The SAD_{RME} ensures that only motion vectors leading to a substantially lower SAD error measuring then the current one are considered as an alternative, thereby improving the consistency of the motion-vector field.

4.4 Motion-vector refinement

The motion vectors that are computed by the ME algorithms have full-pixel accuracy and their referred search positions are located on the full-pixel grid. This accuracy may not be sufficient for describing the actual motion in a video sequence. Higher accuracy can be reached when the pixel grid is refined by a factor of two in each direction. In this case, search positions that are located on a half-pixel grid can be considered in addition. ME with half-pixel accuracy requires that pixel values on the half-pixel grid have to be additionally computed by using linear interpolation (see Figure 4.8).

Due to the additionally required computational complexity, the MV refinement is performed after finding the best motion vectors $mv_{fp}(x, y)$ with full-pixel accuracy. Subsequently, the best candidate vector is refined by evaluating the vectors $mv_{hp} = mv_{fp} + mv(\pm 0.5, \pm 0.5)$.

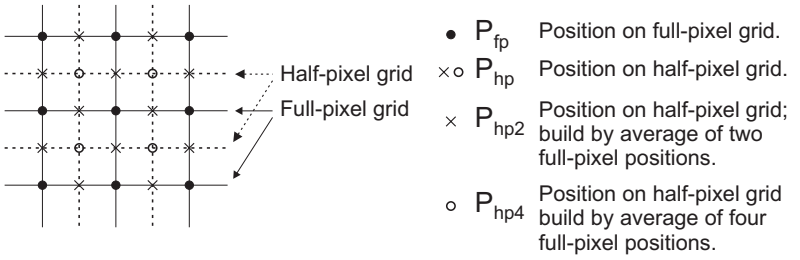

Figure 4.8: *Motion vector refinement full-pixel to half-pixel accuracy.*

4.5 Comparison of ME algorithms

Section 4.5.1 compares the previously presented ME algorithms with respect to their frame-prediction quality. A detailed breakdown of the computational complexity of the individual steps of ME is presented in Appendix B, in order to avoid lengthy discussions. Section 4.5.2 summarizes the results. Section 4.5.3 presents a final comparison using an algorithm-efficiency indicator, based on both complexity and picture quality.

4.5.1 Picture quality of motion-compensated frames

In this section, an experiment measures the picture quality, expressed in PSNR, of the previously discussed ME algorithms using the video sequences "Voit", "Girl" and "Teeny". A search area of 32×32 pixels is used, thereby allowing a maximal block displacement of 16 pixels in each direction for finding a suitable MV. In the experiment, the PSNR between the motion-compensated prediction and the corresponding original input frame is used as performance measure, based on frame-by-frame processing. To exclude effects from DCT and quantization, a perfect difference signal is used, such that the frame reconstruction is identical to the original frame. Although this condition is not satisfied in MPEG coding, it is required that a suitable algorithm should give a high quality in order to be applicable in regular MPEG compression conditions.

The algorithms are evaluated both on full- and half-pixel accuracy. Figure 4.9 shows the achieved PSNR of the frame predictions when processing the "Voit" sequence. The MSE is used as block-matching criterion. The value of this figure is that it shows clearly the quality improvement of the frame predictions when refining MVs to half-pixel accuracy, which is almost constant

about 1 dB PSNR (except for the BBGDS algorithm). For a fair comparison, the simple RME algorithm was tested without a quantization of the block-matching criterion. Figure 4.9 also shows that the difference in performance between the algorithms is much larger than the improvement from using half-pixel accuracy. For this reason, algorithms with high quality in full-pixel accuracy are required.

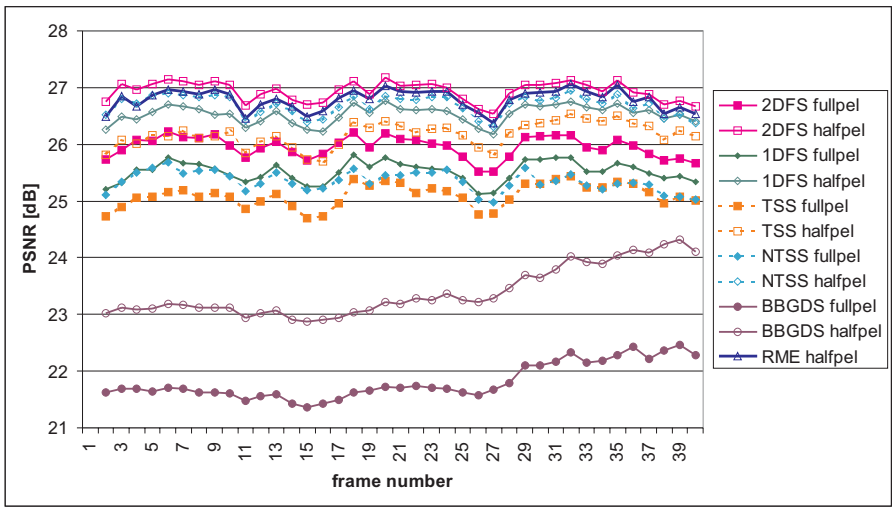

Figure 4.9: *Evaluation of ME algorithms using the "Voit" sequence as input and MSE as block-matching criterion.*

Figure 4.10 portraits a more detailed ME comparison, where the average PSNR of the frame predictions is compared for different block-matching criteria and video sequences. The measured PSNRs are plotted with respect to the PSNR obtained without ME (PSNR gain).

MSE and SAD criterion show a comparable performance and clearly outperform the MiniMax criterion. Since the SAD computation is less complex than the MSE, the SAD is mostly used in state-of-the-art ME. The MiniMax criterion is too simple, because only one pixel difference is used. Figure 4.10 also shows that the algorithm performance is strongly dependent on the video content of the sequences. The little motion in sequence "Girl" leads to good prediction quality even without ME, thus only small enhancements of the prediction quality can be gained with ME. The "Voit" sequence has more motion.

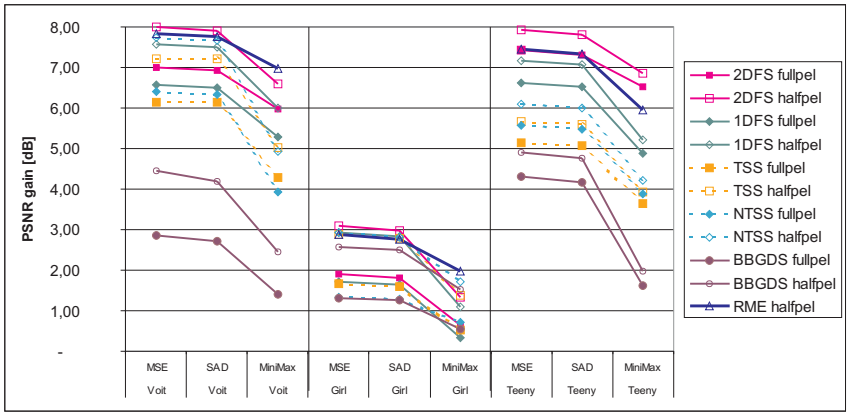

Figure 4.10: *PSNR gain of various ME algorithms using block-matching criteria MSE, SAD and MiniMax for sequences "Voit", "Girl" and "Teeny".*

Due to edges in the image content, almost all algorithms can find suitable motion vectors for this sequence. The high motion in the "Teeny" sequence has the consequence that the achieved PSNR-gain mainly depends on the number of evaluated search positions.

4.5.2 Computational complexity comparison

A detailed breakdown of the computational complexity of the individual steps of ME is presented in Appendix B. In this subsection, we summarize the results of this analysis below.

- The algorithmic comparison shows that the complexity of 2DFS grows quadratically with the number of block comparisons and 1DFS grows linearly. All fast algorithms have a constant number of motion-vector evaluations, except for the BBGDS.
- The complexity of all three block-matching criteria grows linearly with the number of pixels in the block.
- The complexity of interpolating pixels on half-pixel accuracy grows linearly with the number of pixels.
- Using television signals on CCIR-601 resolution, the worst-case complexity of performing ME for one frame varies between 40M operations

for a fast algorithm and 2.14G operations for 2DFS.

- Experiments reveal that for regular video sequences, the average amount of operations is a factor 3-4 lower than in the worst-case condition. This is obtained when the computation of error measures is aborted as early as possible and motion vectors around the zero vector are favored over large motion vectors.

4.5.3 Complexity vs. picture quality

In this section, the picture quality and the computational complexity are combined into a criterion in an attempt to express a so-called algorithm-efficiency parameter. This parameter is measured with the average number of operations required for each dB of PSNR gain (see Section 4.5.1).

Figure 4.11 shows a detailed overview of the algorithm efficiency for all considered ME algorithms when processing the "Voit", "Girl" and "Teeny" sequence. Small values indicate a high efficiency for achieving an average relative quality improvement of 1dB. The figure portrays that the TSS, NTSS, BBGDS and RME algorithm are much more efficient in achieving quality improvements than the other algorithms for a low number of operations. However, in order to make a final choice, it is required to look at Figure 4.10 for the absolute values of the obtained picture quality.

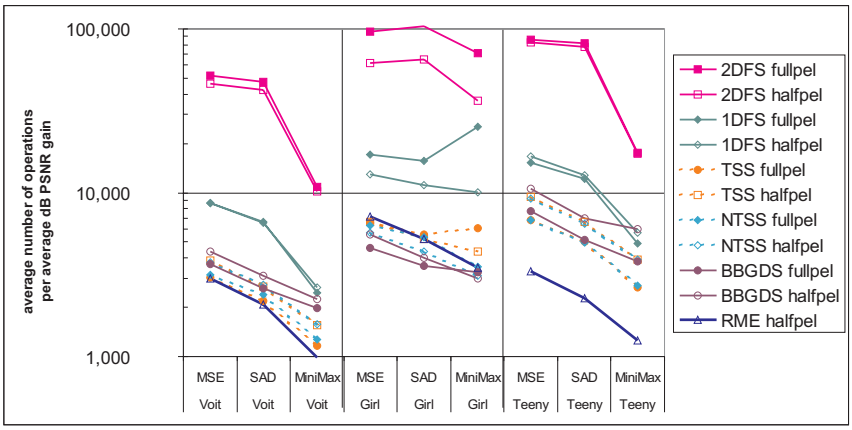

Figure 4.11: *ME algorithm efficiency indication based on number of operations required per average dB PSNR gain.*

4.6 Conclusion

From Figure 4.10, it follows that 2DFS gives the best picture quality (every possible MV in the given search area is evaluated). The picture quality of the simple RME is slightly less than the 2DFS, but at the same time, it can be seen from Figure 4.11 that the algorithm efficiency in achieving the picture quality is much higher than 2DFS. Although the BBGDS is of high efficiency (Figure 4.11), it yields significantly less picture quality (Figure 4.10), and therefore it should not be adopted.

It was found that the selected block-matching criterion has a minor effect on the obtained picture quality. The SAD results in slightly lower PSNRs than the MSE, whereas the MiniMax shows a considerable decrease in quality compared to the SAD and MSE.

MV refinement to half-pixel accuracy results in a clear picture quality enhancement. However, the refinement is useless, if it is performed in combination with an ME algorithm that results in inaccurate predictions on the full-pixel grid. For this reason, it should be considered as a last refinement step towards high-quality prediction.

The final conclusion is that the simple RME algorithm forms a good basis for starting investigations on scalable ME algorithms, due to its high efficiency and picture quality. In Chapter 5, a new scalable ME technique will be presented that changes the way of computing MV fields compared to common MPEG MV field processing. The new technique can basically be applied to any ME algorithm. Therefore, the development of the new technique in Chapter 5 and the performed experiments will be based on simple RME using the SAD matching criterion.

CHAPTER 5

Scalable motion estimation

In accordance with the scalable DCT technique that was presented earlier in this thesis, a computational complexity scalable ME technique is presented in this chapter that can be applied to any conventional ME algorithm. This technique is based on frame-oriented processing and performs the ME on a frame-by-frame basis to approximate MPEG MV fields and optionally refine them afterwards. Complexity scalability is obtained by varying the number of MV fields that are processed. Furthermore, a new scalable ME algorithm is presented aiming at optimizing the number of MV candidates by using block-content classification. Changing the classification strength varies the number of processed MV candidates, thereby leading to complexity scalability.

5.1 Introduction

5.1.1 A view on the ME process

The Motion Estimation (ME) is one of the time-consuming functions and a key element of MPEG coding [8]. In this preliminary section, we motivate

the research on scalable ME by taking different viewpoints on the same processing task. These viewpoints are primarily intended to classify existing algorithms and to explain why we have designed our new techniques for complexity scalability. These viewpoints emerge by recognizing that the signal processing is performed in several levels. These levels will now be introduced below.

The levels used for analyzing the ME processing consider the processing of input video frames, the processing of MV candidates, and the metric for measuring block similarities, respectively. A new technique based on frame-oriented processing is presented that performs ME on a frame-by-frame basis to approximate MPEG MV fields and refine them afterwards. Another new technique that is presented aims at optimizing the number of MV candidates by using block-content classification.

We consider ME as a signal-processing task applying the following three levels, where each level can contribute to enhance ME with scalability.

- **Accuracy level**: the accuracies of (candidate) motion vectors (MVs) are evaluated in order to decide whether the vectors are suitable to describe picture block displacements.
- **Vector-selection level**: an algorithm provides the selection of MVs for evaluation, in order to estimate the motion between two frames.
- **Structural level**: the structure of the Group-Of-Pictures (GOP) defines which MV fields (MVFs) are needed for the MPEG encoding process. The computation of according MVFs is initiated at this level.

5.1.2 State-of-the-art ME

In the past, a large number of algorithms have been proposed for enhancing the vector selection, thereby reducing the computational effort of a full-search ME. In Chapter 4, ME algorithms have been presented that make a trade-off between complexity and the quality of the computed vector fields. However, the accuracy of the MVs that is achieved by using popular algorithms like New Three Step Search [69] and Center-Biased Diamond Search [65], is limited for fast motion in the video sequence.

It was found in Chapter 4 that high quality and high efficiency of ME are obtained by using Recursive ME (RME, already discussed in [61, 62, 68], and

now included in MPEG-4), that derives candidate MVs from previously computed MVs in both the current MVF ("spatial" candidates) and the previous MVF ("temporal" candidates). Up to now, the RME algorithms have been used on GOP structures of fixed size and B-frames were not considered for long-term tracking of the motion.

A more sophisticated approach for ME on the structure level employing RME is presented in [70], featuring a two-step estimation process and enhanced vector-field prediction. The first step of this approach is a coarse RME to pre-estimate the forward vector-fields. The second step uses the vector fields computed in the first step as prediction and performs an additional RME. Vector fields that are used as prediction are scaled to the appropriate temporal distance that is actually needed.

Computational complexity scalability has been introduced for the ME process by providing an algorithm that operates on the vector-selection level and on the accuracy level [60]. The algorithm approximates the result of a full SAD computation based on partial SAD computations for pixel subsets of a block. The number of pixel subsets that is used for the approximation is the first degree of scalability. The second degree results from varying the number of MV candidates that are taken for evaluation from a predefined vector set.

The drawback of conventional ME algorithms is that they process each picture block in the same content-independent way, thereby spending many computations on computing MVs for e.g. relatively flat blocks. Unfortunately, despite the effort, the ME process then yields MVs of poor quality. Block classification is a known technique for concentrating the ME on blocks that may lead to accurate motion vectors [71].

In a further step, the number of blocks that are evaluated is controlled via block classification, thereby leading to complexity scalability. A recent publication on scalable ME using block classification was presented in [72]. The classification that is used in the algorithm presented in [72] is performed for distinguishing textured and less textured blocks and making a binary choice between the two. It does not use the content of the blocks for providing the ME algorithm with the information whether it is more likely to find a good MV in up-down or left-right search directions, as compared to the block-classification algorithm that is presented in this thesis (see Section 5.4). The algorithm in [72] provides scalability on the vector-selection level by varying the number of MV candidates that are taken from predefined candidate sets, depending on the classification and by adapting the classification thresholds.

5.1.3 Contribution of this chapter to scalable ME

The problem of the aforementioned ME algorithms is that a larger GOP size increases the prediction depth, implying a larger frame distance between reference frames. This dependence hampers accurate ME. Furthermore, although state-of-the-art RME has a high efficiency, still unnecessary MV evaluations are performed. For example, MVs are re-evaluated if candidate vectors that are taken from previously processed macroblocks are equal. This phenomenon occurs if the currently processed block is a part of a larger area (e.g. the image background) that has a nearly equal MVF everywhere.

To overcome these inefficiencies, two new complexity scalable ME techniques are presented in this chapter. The first technique operates on the structural level and processes MVFs in three stages. The first stage performs initial ME with the input video frames in display order ("IBBP") and is independent of GOP structures. The second stage efficiently derives MPEG MVFs by approximations based on multiple vector fields that are computed in the first stage. Furthermore, the quality of full-search ME can be obtained with an optional third refinement stage. The aforementioned stages form the new Scalable MVF Approximation and Refinement Technique (SMART). Since the second stage of SMART is the important stage for ensuring the availability of the MVFs that are required for MPEG encoding, the first and the third stage are used for varying the computation of MVFs, thereby leading to complexity scalability.

In addition, SMART ME algorithm works not only for the typical (pre-determined and fixed) MPEG-GOP structures, but also for more general cases. This feature enables on-the-fly selection of GOP structures depending on the video content (e.g. detected scene changes, significant changes of motion, etc.).

The second ME technique operates on the vector-selection level and provides Content-Adaptive REcursive Scalable ME (CARES) through block classification based on edge detection. Prior to estimating the motion between two frames, the macroblocks inside a frame are classified into areas having horizontal, vertical edges or no edges. The classification is exploited to minimize the number of MV evaluations for each macroblock by e.g. concentrating vector evaluations along the detected edge (across the line that is upright to the edge). A novelty in the technique is a *distribution* of good motion vectors *to* other macroblocks, even already processed ones, that differs from other known recursive ME techniques reusing MVs *from previously* processed

blocks. Varying the block-classification strength and thereby the number of evaluated MV candidates, results in complexity scalability.

The sequel of this chapter is organized as follows. Section 5.2 introduces some notations for ease of discussion. Section 5.3 presents the SMART technique. Section 5.4 presents a block classification based on edge detection, which is used in a following ME algorithm. The CARES technique employing the previously mentioned block classification, is presented in Section 5.5. Section 5.6 concludes this chapter.

5.2 Notations

For ease of discussion, Sub-Groups-Of-Pictures (SGOP) are formed that have the form $(I/P)BB...B(I/P)$ within an MPEG GOP. For simplicity, pictures are addressed as frames, although interlaced pictures have two fields. The following explanation refers to Figure 5.1.

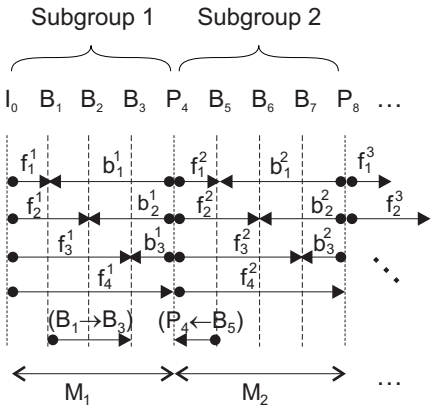

Figure 5.1: *Example of vector fields used for motion estimation in MPEG encoding after defining a GOP structure. In this example, a GOP with a constant $M = 4$ was chosen.*

The prediction depth of a subgroup k is denoted by M_k for $k \epsilon N_0$, in accordance with the prediction depth M of a GOP, and can vary from SGOP to SGOP. The MPEG forward vector-field, which is used in the prediction of the i^{th} frame, is denoted by $\underline{f}_i^k$. The MPEG backward vector-field is denoted by $\underline{b}_i^k$. Arbitrary MVFs are denoted by $(X_m \rightarrow X_n)$ for the forward case

and $(X_m \leftarrow X_n)$ for the backward case, indicating motion between frame X_m and X_n with $n > m$. If the frame type is known, the arbitrary frame type X can be replaced by I, P, or B. The given subscript parameters (k, i, m, n) may be left out where they are not needed.

5.3 SMART, a structural-level technique

5.3.1 Algorithm

Temporal candidate MVs play a key role within this scalable technique for ME. For this reason, we have adopted Recursive ME (RME), which is based on block matching. RME algorithms employ temporal candidates and provide a consistent ME in video sequences, while using a small number of candidate vector evaluations.

Obviously, the prediction quality of an ME algorithm improves with a smaller temporal distance D, where the parameter D denotes the difference between the frame numbers of the considered frames. Therefore, we commence with estimating the motion using the minimum temporal distance $|D| = 1$, which results in an accurate ME that can be performed at low computational effort, mainly due to smaller search areas that need to be considered (see Figure 5.2).

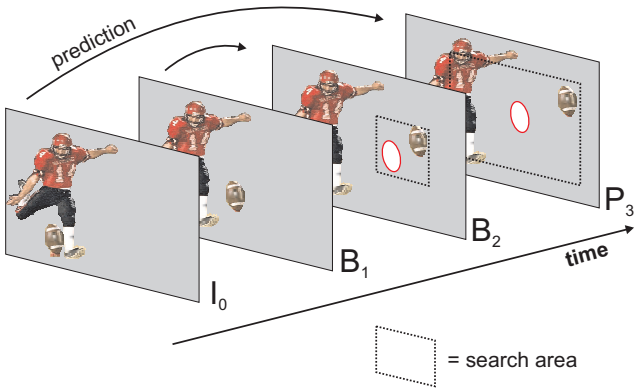

Figure 5.2: *Search area versus prediction depth. Higher prediction depth leads to larger search areas and therefore increased computation effort.*

Since a common MPEG-GOP structure has $M > 1$ and thus some of the required vector fields must have $M > D > 1$, this is considered as a first

stage to derive a *prediction of vector fields*. In a second stage, these predicted vector fields are used to compute the required vector fields according to the MPEG standard (using larger D). In the third stage, the vector fields can be refined by using an additional –although simple– ME process. This stage is optional and only required if the highest quality has to be obtained (e.g. a conventional MPEG ME algorithm). Summarizing, the new concept results in a three-stage process, which is described more formally below.

- *Stage 1.* Prior to defining a GOP structure, we perform a simple RME for every consecutive frame X_n and compute the forward vector field $(X_{n-1} \rightarrow X_n)$ and then the backward field $(X_{n-1} \leftarrow X_n)$. For example, in Figure 5.1 this means computing vector fields like $\underline{f}_1^1$ and $\underline{b}_3^1$, but then for every pair of sequential frames (see the left side of Figure 5.3).

- *Stage 2.* After defining a GOP structure, all the vector fields $\underline{F} \in \{\underline{f}, \underline{b}\}$ required for MPEG encoding (right side of Figure 5.3) are approximated by appropriately accessing multiple available vector fields $\underline{F}_A$ and $\underline{F}_B$ and combining them using the linear relation

$$\underline{F} = \alpha * \underline{F}_A + \beta * \underline{F}_B. \tag{5.1}$$

  The scaling factors α and β depend on the processed fields and are chosen according to the required temporal distance for computing $\underline{F}$. For example, $\underline{f}_2 = (X_0 \rightarrow X_1) + (X_1 \rightarrow X_2)$ (see middle of Figure 5.3), thus having $\alpha = \beta = 1$. Note that α and β will have different values if the frame distances changes, or when complexity scaling is applied (see below).

- *Stage 3.* Optionally, a second iteration of RME is performed for refining the computed approximated MPEG vector fields from Stage 2. For example, the approximated vector fields from Stage 2 serve as temporal candidates in the final refining RME process at this last stage.

Note that the GOP structure is chosen after performing Stage 1, which enables that GOP structures can be chosen dynamically, based on e.g. an analysis of the computed MVs (see next section). In this way, the GOP structure is modified according to the measured motion in Stage 1. An example of this approach is visualized in Figure 5.3, in which a MVF $(X_0 \leftarrow X_1)$ is computed but not used afterwards for approximation, because the adopted GOP structure does not require approximations from it. This situation depends on the approximation process, and in the given example, the approximation of the

MPEG vector fields is considered sufficient. With predefined GOP structures, the computation of vector field $(X_0 \leftarrow X_1)$ is not necessary.

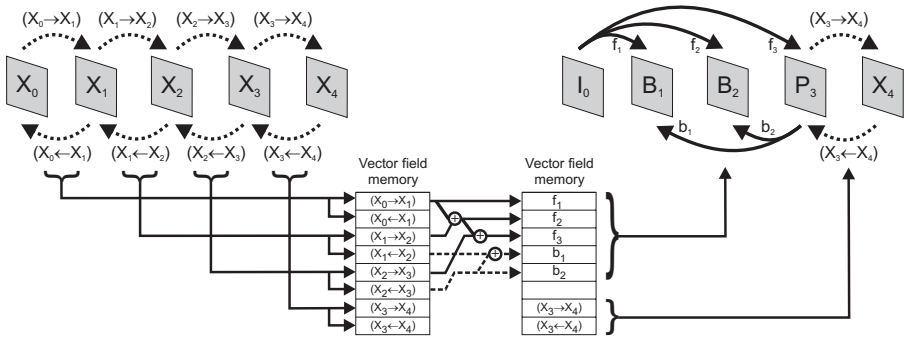

Figure 5.3: *Overview of the SMART ME process.*

Rovati *et al.* [70] have proposed an approach that at first glance looks similar to the algorithm presented here. However, there are a number of important differences. Firstly, they initially estimate the MPEG vector fields and then process these fields for a second time, while keeping restricted to the MPEG GOP structure. This means that they have to deal with an increasing temporal distance to derive the vector fields already in the first step. This limits the accuracy of the computed first-step predictions. The processing of pictures in MPEG order is a second difference. Thirdly, the proposed ME does not provide scalability. The possibility of scaling vector fields, which are also used for the above-mentioned multiple predictions, is mentioned in [70], but not further exploited. SMART ME makes explicit use of this feature, which is a fourth difference. In the sequel, the important system aspects of SMART are explained.

5.3.2 Modifications in the MPEG encoder architecture

The SMART ME technique processes video frames both in the input signal order and the MPEG processing order. It is therefore relevant to consider the architectural changes and the complexity when using the SMART ME technique. Figure 5.4 shows the architecture of the SMART ME technique embedded in an MPEG encoder. In the figure, the markers 1, 2 and 3 as indicated in the modules refer to the stages in the SMART ME technique.

Note that the amount of memory needed for the new architecture is the same as used for the architecture as shown in Figure 1.14, except for the relatively small additional MV memory. The frame memory costs are equal, because the entrance frame memory for Stage 1 can be integrated with the memory for the reordering process. The additional memory required to store a vector field is negligible compared to the memory requirement of a video frame (a vector needs only 2 bytes vs. a luminance macroblock using 256 bytes). The three stages are decoupled from the actual coding process and are connected with each other via the central MV memory. This concept based on a central MV memory enables the three SMART stages for ME to be implemented as individual communicating processes. This property is attractive for heterogeneous systems.

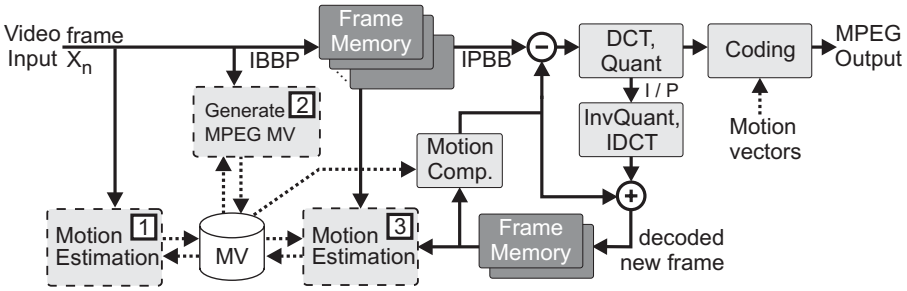

Figure 5.4: *Architecture of an MPEG encoder with the SMART ME technique.*

Several architectural aspects of the individual stages of SMART are discussed below.

- The initial ME process in Stage 1 is performed on succeeding frames, thus ($|D| = 1$). Furthermore, Stage 1 uses original frames, without quantization errors from the actual coding process. For these reasons, the RME yields a high-quality prediction with accurate MVFs.

- Stage 2 can enhance the choice of a GOP structure by e.g. analyzing the computed MVFs. For example, if motion occurs in a sequence, the first frame that initiates a group of frames having (almost) zero motion in-between, can be defined as a reference frame. On a sequence level, flexible scene-change detection can be added.

- With Equation (5.1), Stage 2 introduces a new concept called *multi-field* ME. Multi-field ME is an abbreviation of multi-MVF ME[1], indicating that a plurality of vector fields is used for creating the desired vector field. The term "multi-field ME" refers to two aspects. Firstly, the computation of Equation (5.1) means that one vector field is constructed from two other vector fields. Secondly, the total prediction of a vector field can be based on various vector fields (more temporal references).

  The second aspect can be used for high-quality applications to approximate different real-life motions like video-object velocity, acceleration, zoom, rotation, etc. To give an example of multi-field ME for a video object with constant motion speed, we predict a MVF $\widehat{\underline{f}}$ by specifying the motion model "most recent velocity" as

$$\widehat{\underline{f}_i^k} = \begin{cases} 2 * \underline{f}_{i-1}^k - \underline{f}_{i-2}^k & \text{if } M_k \geq i \geq 3 \\ 2 * \underline{f}_1^k & \text{if } i = 2 \\ -\underline{b}_{M_{k-1}}^{k-1} & \text{if } i = 1,\ M_{k-1} > 1 \\ \underline{f}_1^{k-1} & \text{if } i = 1,\ M_{k-1} = 1, \end{cases} \tag{5.2}$$

  where the term "most recent" refers to the previously processed frames in display order.[2]

5.3.3 Scalability aspects

The main advantage of the proposed SMART architecture is that it enables a broad scalability range of resource usage and achievable picture quality in the MPEG encoding process. This is illustrated by the following statements.

- Stage 1 and 3 can omit the computation of vector fields (e.g. the backward vector fields) or compute only significant parts of a vector field to reduce the computational effort and memory.

- If the refinement in Stage 3 is omitted completely, the new technique can be executed at a further reduced computational effort, because the

[1] This should not be confused with an alternative concept, called multi-temporal, which is used in H.264 coding, where the prediction of a frame is based on multiple (up to five) reference frames.

[2] Note that besides our framework, any state-of-the-art ME algorithm can be improved by using multiple vector-field predictions. This implies that more than one prediction is generated for the computation of one vector field.

processing of vector fields in Stage 1 and 2 is much simpler than with regular ME, where the desired MPEG GOP leads to large temporal frame distances for the regular ME process.

- Note that bi-directional ME (usage of B-frames) can be realized at the same cost of single-directional ME (usage of P-frames only) when properly scaling the computational complexity, which makes it affordable for mobile devices that up to now rarely make use of B-frames.

In this thesis, the complexity scalability of the SMART ME is realized by varying the number of MV fields that are computed in Stage 1 and 3 for one GOP. In the following, an example is given for the approximation of an MPEG vector field in Stage 2 of SMART when scalability is applied. For the example, the situation shown in Figure 5.5 is considered, where two vector fields from Stage 1 are simply skipped, i.e. not computed. These two fields are $(X_1 \rightarrow X_2)$ and $(X_2 \leftarrow X_3)$, denoted by the dotted gray arrows in the figure. The computations of the fields $(X_2 \rightarrow X_3)$ and $(X_1 \leftarrow X_2)$,thus denoted by the solid gray arrows, are replaced by the computations of the fields $(X_1 \rightarrow X_3)$ and $(X_1 \leftarrow X_3)$, indicated by the dashed arrows. Besides the reduced computation, this required less memory bandwidth compared to the full processing of Stage 1. In the special situation shown in the figure, frame X_2 is not accessed at all in the first two stages.

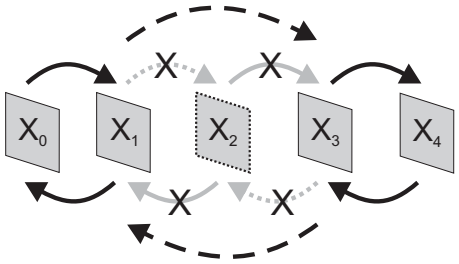

Figure 5.5: *Example for SMART complexity and memory scalability.*

Figure 5.6 depicts two approximation alternatives for the MPEG MVFs. The left side of the figure shows the situation after defining an SGOP for the first four frames. Vector field $(I_0 \rightarrow B_2)$ cannot be directly approximated from the available vector fields. The approximation requires scaling of vector fields. The alternative shown at the top right of the figure approximates field $(I_0 \rightarrow B_2)$ by $(I_0 \rightarrow B_1) + (B_1 \rightarrow B_2)/2$, and the alternative shown at the bottom right approximates the field by $(I_0 \rightarrow B_1) * 2$.

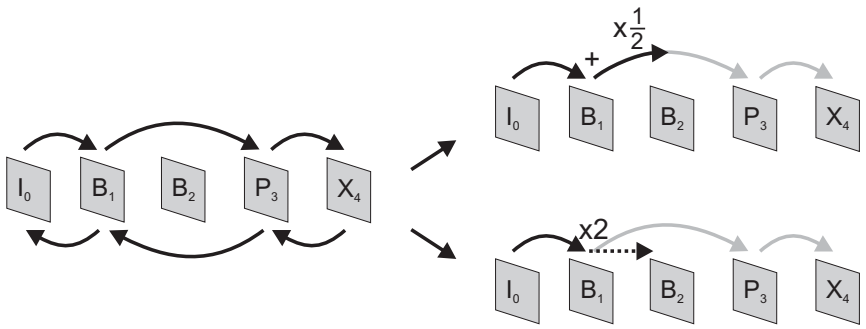

Figure 5.6: *Exemplary alternatives for MVF approximation.*

Note that other alternatives for approximating vector fields are possible. An optimal strategy for selecting the MVFs that should be computed when scaling the computational complexity, or for choosing how MPEG MVFs are approximated, cannot be derived easily. The reason for this results from the fact that each decision for computing a particular vector field and the way for approximating the MPEG MFVs, influences other vector fields and thus the prediction quality of the motion-compensated frames, also beyond (post-defined) (S)GOP structures. Optimization techniques based on e.g. the Lagrangian multiplier or other multi-dimensional search algorithms that can find an sub-optimal path through a decision tree as outlined above, could be applicable for solving this problem, but this has not been elaborated.

Note that optimizing the strategy for selecting the MVFs is more challenging than for selecting the coefficients in the DCT, because for the DCT, at least the computation costs of the coefficients was known in advance and did not change with the processed video content. In ME, the computation costs of MVFs is not constant or known, even not for fixed ME algorithms that do not vary the number of evaluated MV candidates (e.g. when smartly implemented, the computation of the block-matching criteria is stopped if the accumulated error measure exceeds a certain value). It is expected that finding the optimal computation order of MVFs would be impractical during coding time. A more practical approach would be to make assumptions about the computation costs and the obtainable quality of the MVFs, based on different contents of the video, and then perform some preliminary experiments with selected sequences. This approach can lead to some basic decision rules, which should be applied during the encoding process. In the following subsection, we define a simple priority of vector fields by considering the different importance

of MVF properties, such as their temporal distance or their direction of prediction.

5.3.4 Experimental verification

The scalability performance of the new technique is shown with an initial experiment using the "Stefan" (tennis) sequence. The sequence is encoded based on a GOP size of $N = 12$ and $M = 4$ (thus "IBBBP" structure). The RME taken from [68] (limited to pixel-search) is used in Stage 1 and 3. Note that the RME in Stage 1 used for this initial experiment could have been simplified, because of the minimum temporal distance at this stage. A scalable version of RME that performs equally to [68] but at much lower computational complexity is presented in Section 5.5.

In this experiment with $M = 4$, the number of vector fields (forward and backward motion) that are considered in an SGOP in Stage 1 is $2 * M = 8$ and in Stage 3, $M + (M - 1) = 7$. To realize scalability, the amount of vector field computations in Stage 1 and 3 is gradually decreased. To select vector fields for computation, we define a simple priority of the vector fields (similar to the earlier introduced computation order of DCT coefficients) with the following rules.

- Rule A: from the construction of the SMART ME technique it follows that Stage 1 is more important than Stage 3.
- Rule B: forward motion is considered as more important than backward motion, because P-frame predictions do not use backward motion.
- Rule C: MPEG vector fields in Stage 3 (or their equivalents in Stage 1) with high temporal distance are considered less important than the ones with lower temporal distance (just for the sake of this experiment).

These rules leads to MVF priorities (ABC) as given in Figure 5.7. In the figure, the dashed arrows indicate vector fields in Stage 1, solid arrows are the MPEG vector fields in Stage 3, and the number below the vectors the priority weight. Small numbers (reading (ABC) as integer values) indicate high priority. The MVF priorities are used for selecting vector fields for computation, and does not change the computation order of vector fields as defined in Section 5.3.1. Vector fields such as $(I_0 \leftarrow B_1)$ are not computed in this experiment, because they are not required for computing the MPEG vector fields. Note that in order to realize scalability and still keep track of the motion in

Stage 1, the computation of vector fields is proceeding such that a new vector field starts where a previous field has ended (there are no "holes").

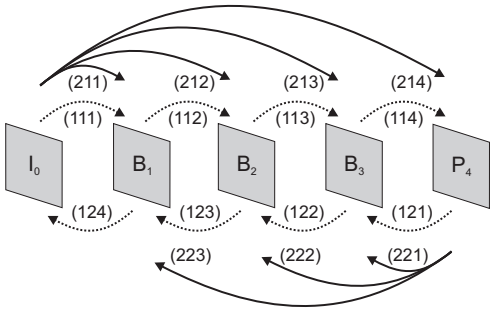

Figure 5.7: *Visualization of the priority order for motion-vector fields (numbers below the arrows). Small numbers indicate a high priority.*

The result of this scalability experiment is shown in Figure 5.8, where P-frames are not shown for the sake of clarity ($N = 12$, $M = 4$). The area with the white background is the quality range that results from scaling the computation complexity by varying the amount of computed MVFs as described above.

For comparison, the computation effort of the simple RME used by a standard MPEG encoder (which computes four forward vector fields and three backward vector fields per SGOP) is defined as 100%, which is used as reference. Each vector field then requires 14% of the reference computational effort. The percentage as shown at the right side of Figure 5.8 results from omitting the computation of vector fields in Stage 1, or from performing additional refinement in Stage 3. If all vector fields of Stage 1 are computed and the refinement Stage 3 is performed completely, the computational effort is 200% (not optimized). Figure 5.8 shows that a large quality range is covered, matching with the large differences in computational effort.

The average PSNRs of the motion-compensated P- and B-frames (taken after motion compensation and prior to computing the frame-difference signal) of this experiment are shown in Figure 5.9. Again, the percentages show the different computational efforts that result from omitting the computation of vector fields in Stage 1 or performing additional refinement in Stage 3. For a full-quality comparison (200%), we consider full-search block-matching

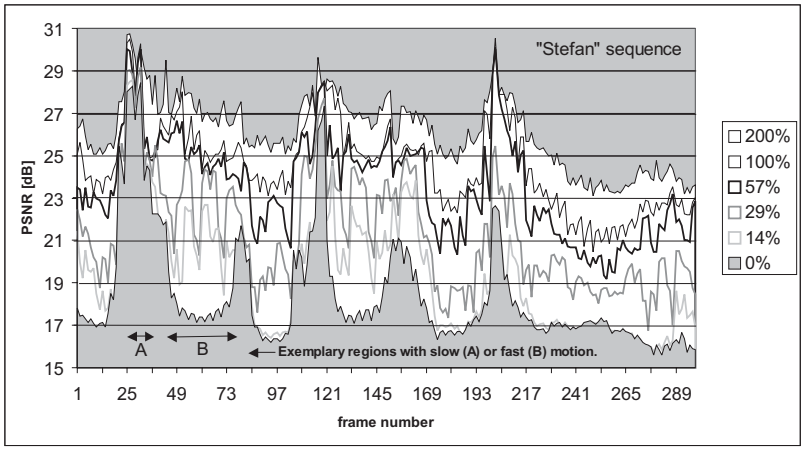

Figure 5.8: *PSNR of motion-compensated B-frames of the "Stefan" sequence for different computational efforts. The percentages show the different computational efforts, resulting from omitting the computation of vector fields in Stage 1 or performing additional refinement in Stage 3. In this example, computing one vector field requires* 14% *of the reference computational effort.*

with a search window of 32×32 pixels. The new ME technique slightly outperforms this full search by 0.36 dB PSNR measured from the motion-compensated P- and B-frames of this experiment (on the average, 25.16 dB instead of 24.80 dB).

Further comparisons are made with the scalable SMART ME running at full and "normal" quality. Table 5.1 shows the average PSNR of the motion-compensated P- and B-frames for three different video sequences and ME algorithms with the same conditions as described above (same N, M, etc.). The first data column (Evals./MB) shows the average number of vector evaluations performed per macroblock in the "Stefan" sequence. Note that MV evaluations pointing outside the picture are not performed, which results in numbers that are lower than the nominal values (e.g. 926.2 instead of 1024 for 32×32 FS). The simple RME algorithm results in the lowest quality here, because only three vector-field computations out of $4 * (4 + 3) = 28$ can use temporal vector candidates as prediction. However, our new SMART ME algorithm based on simple RME performs comparable to full-search ME at 200% complexity. At 100% complexity, it is comparable to the other fast ME algorithms.

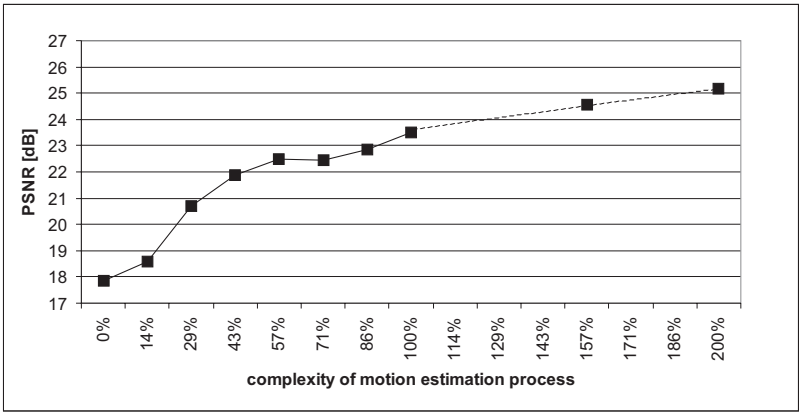

Figure 5.9: *Average PSNR of motion-compensated P- and B-frames of the "Stefan" sequence at different computational efforts for the SMART ME technique.*

Algorithm	Evals./MB	(A)	(B)	(C)
2DFS (32×32)	926.2	24.80	29.62	26.78
NTSS [69]	25.2	22.55	27.41	24.22
Diamond [65]	21.9	22.46	27.34	26.10
Simple RME [68]	16.0	21.46	27.08	23.89
SMART ME 200% (incl. [68])	37.1	25.16	29.24	26.92
SMART ME 100% (incl. [68])	20.1	23.52	27.45	24.74

Table 5.1: *Average luminance PSNRs of the motion-compensated P- and B-frames for sequences "Stefan" (A), "Renata" (B) and "Teeny" (C) with different ME algorithms. The second column (Evals./MB) shows the average number of SAD-based vector evaluations per MB (based on the "Stefan" sequence).*

5.4 Block classification supporting the DCT and ME

The conventional MPEG encoding system processes each picture block in the same content-independent way. In this section, a simple block classification algorithm is presented for optimizing the coding process by content-dependent video processing. This algorithm forms the basis of the CARES ME algorithm that will be presented in Section 5.5. This algorithm also improves the scalable DCT (see Section 3) on the system level by applying differently scalable DCTs to different block classes, which will be presented in Section 6.2.2.

Classification of picture blocks with respect to their video data content is a frequently-used technique for content-dependent processing.

- Block classification is used for quantization to distinguish between flat, textured and mixed blocks [73] and then apply different quantization factors for these blocks for optimizing the picture quality at given bit-rate limitations. For example, quantization errors in textured blocks have a small impact on the perceived image quality. Blocks containing both flat and textured parts (mixed blocks) are usually blocks that contain an edge, where the disturbing ringing effect[3] deteriorates with high quantization factors.

- ME can benefit from classifying blocks with respect to their contents. The drawback of conventional ME algorithms is that they spend many computations on computing MVs for e.g. relatively flat blocks. Unfortunately, despite the effort, the ME process then yields MVs of poor quality, because flat blocks do not have strong discriminating data properties. When employing block classification, computations can be concentrated on blocks that lead to reliable MVs [71, 72]. In this context, reliable MVs means that the MVs have small error metric and are computed for a block with a structured content (e.g. texture, lines).

Of course, in order to be useful, the costs to perform block classification should be less than the saved computations. Given the above considerations, in the following, we will adopt content-dependent adaptivity for coding and motion processing. The next subsection explains the content adaptivity in more detail.

[3] Ringing is sometimes visible at edges in the video signal. This is caused by inappropriate filtering or by coarse quantization of the video data in a video compression system.

5.4.1 Algorithm

The proposed simple block classification is based on detecting horizontal and vertical transitions (edges) in the block content for two reasons.

- The ME can be provided with the information whether it is more likely to find a good MV in up-down or left-right search directions. Since ME will find equally good MVs for every position *along* the line that is upright to such an edge (where a displacement in this direction does not introduce large displacement errors), searching of motion vectors *across* this line will rapidly lead to finding the minimum displacement error and thus it results into a reliable MV. Horizontal and vertical edges can be detected by significant changes of pixel values in horizontal and vertical direction, respectively.
- From the scalable DCT, computation orders are available that prefer coefficients representing horizontal or vertical edges (see Figure 3.4). In combination with a classification, the computation order that fits best for the block content can be chosen. As a result, the amount of coefficients computed is optimized for a given computation constraint and block content.

Known edge-enhancing/detecting algorithms like the Sobel or Roberts-Cross operator or the Canny algorithm are found to be too computationally expensive for our purpose and therefore have not been considered. The finally applied edge-detecting algorithm is in principle based on continuously summing up pixel differences along rows or columns and subsequently counting how often the sum exceeds a certain threshold.

Let p_i with $i = 0, 1, .., 15$, denote the pixel values in a row or column of a macroblock (size 16×16). We then define a range where pixel divergence (expressed as d_i) is considered as noise if $|d_i|$ is below a threshold t. The pixel divergence is defined by Table 5.2.

The area preceding the edge yields a level in the interval $[-t; +t]$. The middle of this interval is at $d = 0$, which is modified by adding $\pm t$ for the case that $|d|$ exceeds the interval around zero (start of the edge). The term $(p_i - p_{i-1})$ updating the pixel divergence d is the relative change in pixel values when processing a row or column. This mechanism will follow the edges and prevent noise from being counted as edges. Figure 5.10 shows an example of this algorithm. The value of the pixel divergence during the first four samples

Condition	Pixel divergence d_i
$i = 0$	0
$i = 1..15 \wedge \lvert d_{i-1} \rvert \leq t$	$d_{i-1} + (p_i - p_{i-1})$
$i = 1..15 \wedge \lvert d_{i-1} \rvert > t$	$d_{i-1} + (p_i - p_{i-1}) - sgn(d_{i-1}) * t$

Table 5.2: *Definition of pixel divergence d, where the divergence is considered as noise if it is below a certain threshold.*

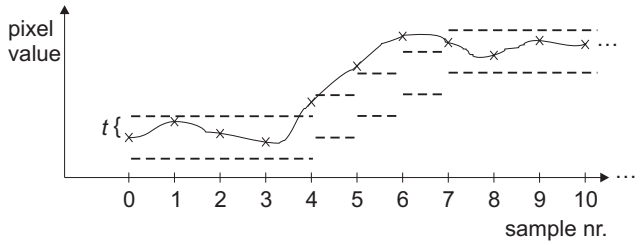

Figure 5.10: *Example of the edge-detecting algorithm. The area between the dashed lines portrays the pixel divergence range that has to be exceeded before considering a pixel divergence a sufficiently relevant.*

remains within the allowed interval. At sample 4, the start of an edge leads to a distinct pixel divergence. The area between the dashed lines indicate the allowed pixel divergence. A counter c indicates how often the actual interval is exceeded, which is defined by

$$c = \sum_{i=1}^{15} \begin{cases} 0 & \text{if } \lvert d_i \rvert \leq t, \\ 1 & \text{if } \lvert d_i \rvert > t. \end{cases} \tag{5.3}$$

The decision of finding an edge is defined by the value of c in Equation (5.3). The above edge-detection algorithm is made scalable by carefully selecting the threshold t, varying the number of rows and columns that are considered for the classification, and choosing a typical decision value for c. Experimental evidence has shown that in spite of the complexity scalability of this classification algorithm, the evaluation of a single row and column in the middle of a picture block provides sufficient quality for the full 2-D classification.

5.4.2 Experimental verification

Figure 5.11 shows the result of an experiment to classify image blocks of size 16×16 pixels (macroblock size). For this experiment, a threshold $t =$

25 was used. We considered a block to be classified as "horizontal edge" if $c \geq 2$ holds for the central row computation and class "vertical edge" if $c \geq 2$ holds for the central column computation. Note that with a horizontal/vertical edge, a line is visible that stretches in the vertical/horizontal direction (see Figure 5.11). Finally, we define two extra classes, called "flat" for all blocks that do not belong to "horizontal edge" and "vertical edge", and "diagonal/structured" for blocks that belong simultaneously to both classes "horizontal edge" and "vertical edge". In the figure, the left (right) picture shows blocks where vertical (horizontal) edges are detected. Blocks that are visible in both pictures belong to the class "diagonal/structured", while blocks that are black in both pictures are considered as "flat".

The visual results of Figure 5.11 are just an example of a more elaborate set of test sequences for conducting experiments. The results showed clearly that the algorithm is sufficiently capable of classifying the blocks for further content-adaptive processing.

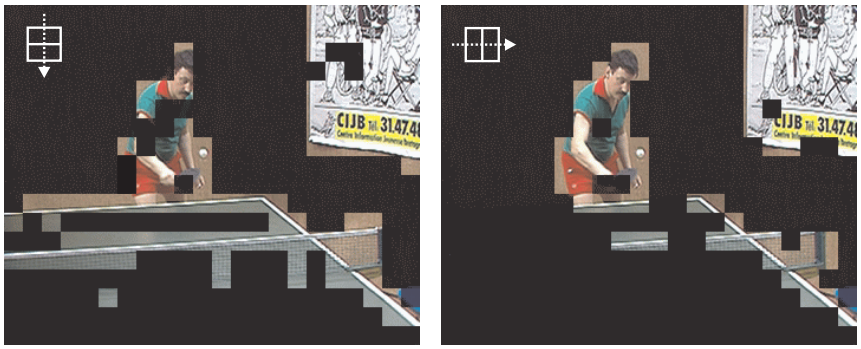

Figure 5.11: *Block classification of vertical (left) and horizontal edges (right) using a picture of the "Table tennis" sequence.*

5.5 CARES, a vector-selection level technique

5.5.1 A concept for content-dependent ME

In this section, a new vector-selection level concept for a scalable RME is presented, that can be integrated into SMART (see Section 5.3), the three-stage ME technique on the structural level (see Section 5.1.1), as a more advanced

replacement for the simple RME that is used in Stage 1 and 3 of SMART. The block-classification algorithm that is presented in the previous section is designed to support content-adaptive processing, and it will be used in this section to create a new ME algorithm with high scalability. The block classification is used to concentrate the ME on structured blocks with content that normally leads to reliable MVs (with small error metric), whereas the remaining "flat" blocks become an MV assigned without any further MV evaluation. When changing the strength of the block classification, the number of processed blocks varies, thereby leading to complexity scalability. This idea was implemented in a similar form in [72], but a later comparison revealed two differences. First, we analyze the line structure of the blocks (see Section 5.4) to come to detailed decisions. Second, in our algorithm, good vector candidates are proposed to "future" blocks such as the neighboring blocks at the right and just below the actual block (see below). Therefore, our algorithm is more suited for software-oriented implementations. Further system aspects are discussed at the end of this chapter.

Besides content-adaptivity, a second key feature of the new concept is a more advanced MV prediction, as compared to conventional RME. In state-of-the-art RME, the set of candidate MVs evaluated to find the displacement of a certain macroblock, contains a few vector candidates that are adopted from already processed macroblocks (the vectors are queried from a MVF memory that stores the finally selected MVs for the already processed blocks). This prediction mechanism leads to re-evaluation of the same MV for a block, if for example the block is a part of a larger equally moving area (e.g. the picture background). Instead, in the new concept, certain MVs of high reliability are distributed from actually processed blocks to surrounding blocks, which are then triggered to evaluate the distributed vector candidates. This may lead to a series of vector evaluations, but in practice the total amount of evaluated vectors is scalable and lower than with conventional RME. The MVs that yield less reliability for ME are not distributed.

Summarizing, the basic approaches of the new concept are

- distinguish blocks which are suitable for reliable ME, and
- distribution of reliable MVs to surrounding blocks and immediately trigger the evaluation of these MVs for the surrounding blocks.

5.5.2 Algorithm

The new proposed ME algorithm is called CARES (Content-Adaptive REcursive Scalable) ME. The algorithmic steps are as follows.

1. Using the block classification presented in Section 5.4, two lists L_+ and L_- are created containing macroblocks that are classified as having horizontal or vertical edges (L_+) or being flat (L_-).

2. All macroblocks $mb_+ \in L_+$, are initialized with an approximated motion vector (temporal candidate) if available, or with the zero vector otherwise.

3. Based on the best motion vector mv_b found so far for the current macroblock $mb_+ \in L_+$, the following MV candidates $mv_c = (dx, dy)$ are evaluated:

   - $mv_c = (0, -1)$ and $mv_c = (0, +1)$
     for macroblocks having a horizontal edge,
   - $mv_c = (-1, 0)$ and $mv_c = (+1, 0)$
     for macroblocks having a vertical edge.

   If a new best motion vector mv_b^* is found for the current macroblock, this motion vector is proposed to other macroblocks by inserting the eight surrounding macroblocks of the current macroblock into a temporary list L_{tl}. The macroblocks that are inserted to this list are restricted to be from list L_+, thus $L_{tl} \subseteq L_+$. The macroblock of list L_{tl} are then processed in the following Step 4.

4. If L_{tl} is not empty, all macroblocks $mb_{tl} \in L_{tl}$ evaluate the vector mv_b^* of Step 3 as candidate vector. Each macroblock $mb_{tl} \in L_{tl}$ further distributes vector mv_b^* by inserting its adjacent macroblocks into L_{tl}, provided that they belong to list L_+ and if mv_b^* is a better motion vector than the current best motion vector of mb_{tl}. Afterwards, the actually processed block is removed from the list L_{tl}. This Step 4 is repeated until L_{tl} is empty.

5. If L_+ is not empty, the next macroblock mb_+ in list L_+ is processed with Step 3.

6. Finally, all macroblocks $mb_- \in L_-$copy the MV of one of its neighbors mb_n, if $mb_n \in L_+$, or the zero vector as default. In both cases, the vector is not further evaluated.

5.5.3 Experimental verification

An experiment has been set up to compare the CARES ME of this section with the simple RME [68] that was used within the SMART technique in Section 5.3. A GOP size of $N = 12$ and $M = 1$, thus "IPPPP" structure, is used for this experiment in order to examine if CARES can serve as a replacement for the simple RME. This GOP structure was chosen, because in the first stage of SMART, frame-by-frame ME is performed. The measured PSNRs of the motion-compensated frames of the "Stefan" sequence when using the simple RME or the CARES ME, are compared. The block-classification algorithm (see Section 5.4) was used with different thresholds t. Figure 5.12 shows the results of this experiment.

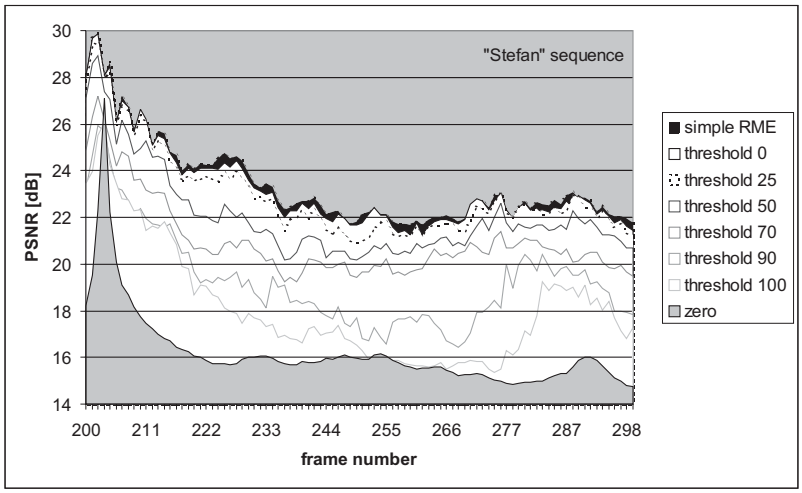

Figure 5.12: *Comparison of simple RME and CARES ME for various classification thresholds t. The fat black areas close to the upper curve indicate the small quality gap between simple RME and CARES ME. The white area is the complexity scalability range of CARES ME.*

In the figure, the frame-number interval refers to a part of the "Stefan" sequence. In this interval, the PSNR differences are well visible. The fat black areas close to the upper curve indicate the small quality gap between simple RME and CARES ME. The white area is the complexity scalability range of CARES ME. The figure shows a large scalability range between 16 dB

and 24 dB PSNR, which results from just varying the threshold t. By decreasing the classification threshold t to a lower number, the quality comes arbitrary close to simple RME. A typical setting of $t = 25$ results in a PSNR that is rather close to the simple RME.

The average PSNR and the average number of MV evaluations per macroblock for different sequences are shown in Figure 5.13. In the figure, the top curves show the average PSNR for three different video sequences, and the bars show the average number of vector evaluations per macroblock. The ultimate left part of the figure refers to simple RME, and the ultimate right part of the figure refers to performing no ME. The rest of the figure refers to CARES ME. It can be seen that for the typical threshold setting of $t = 25$, the obtained PSNR is within 1 dB from the simple RME algorithm. However, the simple RME uses 18.35 vector evaluations on the average, whereas CARES requires only between 1.9 and 4.9 vector evaluations, depending on the sequence. In the worst-case situation using a threshold of $t = 0$ (all macroblocks are processed), virtually the same quality as with simple RME is

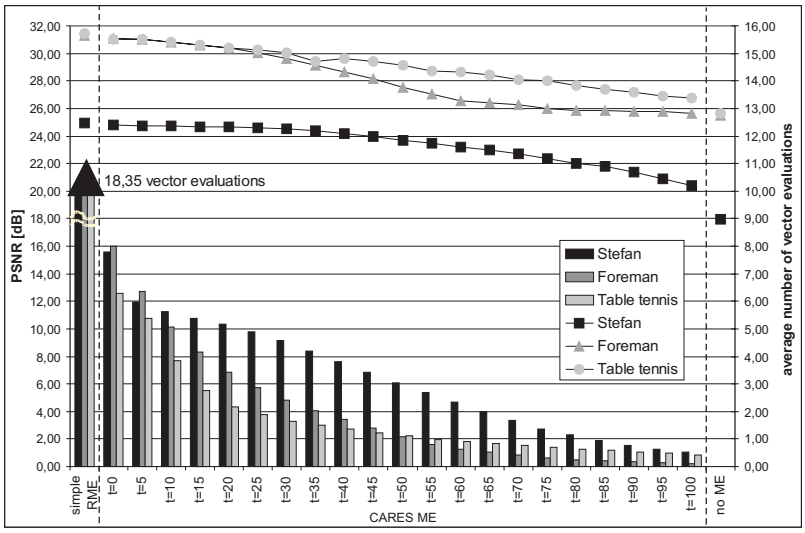

Figure 5.13: *Comparison of simple RME and CARES ME. The top curves show the average PSNR for three different video sequences, and the bars show the required average number of vector evaluations per macroblock.*

obtained. For example, even for the worst-case "Stefan" sequence, the computational complexity is reduced by at least 58%, resulting in an average of 7.78 MV evaluations per macroblock. The "Tennis table" sequence, which has less motion, needs less than one vector evaluation per macroblock, leading to medium quality. The complexity is smoothly scalable until zero-vector fields are reached.

5.6 Discussion and conclusions

In this chapter, two new techniques have been developed for obtaining computational complexity scalable ME. The scalability is obtained by scaling the number of processed MVFs (SMART ME) and the number of vector evaluations (CARES ME). The first term relates to the GOP structure in MPEG, whereas a second term relates to the intelligent selection of MVs for computation.

Experiments with the SMART ME technique have shown that the picture quality of SMART is comparable to high-quality 32×32 full-search ME, while obtaining a complexity comparable to fast state-of-the-art ME algorithms. The advantage of SMART is that it offers a large range of scalability in saving computations. In the experiments, the computational complexity is scaled by a factor of 14, e.g. resulting in an average PSNR ranging from 17.84 dB to 25.16 dB for the motion-compensated P- and B-frames of the "Stefan" sequence, where the visual quality aligns with the measured PSNRs. Another advantage is that the first stage of SMART performs the estimation process on original frames, which results in reliable MVFs. Although the MPEG MVF approximations resulting from Stage 1 and 2 of the algorithm are not optimized for high PSNR numbers, the above-mentioned experiments show that both the quality and the required computing resources of fast ME algorithms are already outperformed. This means further that the optional refinement stage should be used for high-quality requirements and if the additional resources for computation and memory access are available.

To fully exploit the frame-by-frame ME of Stage 1 of the SMART ME algorithm, a second technique called CARES was developed. Experiments have shown that CARES leads to a reduced set of MV candidates that are evaluated in comparison with the simple RME that was used for SMART. The CARES algorithm is an improved RME that can be inserted into the SMART

algorithm, leading to an improved SMART algorithm. The disadvantage of CARES ME is that a significant cache memory is required, so that the algorithm is more suited for a software-oriented implementation. This can be seen from the fact that in the CARES ME, we first concentrate on blocks with a structured detail giving reliable vector candidates, and we simply assign vectors to the remaining blocks. Second, we provide reliable vector candidates to all surrounding blocks, also at the right of and below the current block position. For this reason, the acceptation process for a reliable vector can distribute itself within the frame in all directions. In order to implement this, the required worst-case cache memory is one video frame. However, given the usual size of objects in most cases, a much smaller cache can be used without sacrificing too much quality.

The algorithms in this chapter are based on a software-oriented approach, because the scalability range was implemented in the direction of portable devices having constrained computational resources. The architecture of such devices is usually based on the combination of a DSP processor with a programmable RISC core, so that multi-media applications are implemented in software in any case.

With the development of SMART and the CARES ME, a flexible modular framework for creating scalable ME is presented in this chapter. Both algorithms address scalability in ME at different levels: number of processed MVFs in the GOP structure (structural level) and number of vector evaluations per MVF (vector-selection level). The algorithms can be used individually to introduce scalability, but the combination (discussed in the next chapter) provides an even more powerful proposal for highly scalable ME implementations. This flexibility is important, because ME is the most expensive step in MPEG encoding.

CHAPTER 6

System experiments and enhancements

Up to this point, the scalability of single MPEG modules have been evaluated. In this chapter, the system behavior of a scalable MPEG-encoder system is evaluated, which incorporates the complexity-scalability techniques developed earlier in this thesis. First, first-step effects on the encoder system when scaling the DCT or ME, are investigated. For optimization of the system, the emphasis is on reusing important input/output parameters or control data in related signal-processing modules (e.g. the results from the block classification is reused in the DCT and ME module). Second, all modules are scaled such that nearly all possible parameters and means for scalable control are used, in order to evaluate the overall system performance (the design space) for different test sequences, bit rates and GOP structures.

6.1 Experimental environment

In this chapter, experiments are conducted with an MPEG-encoder framework, in which the scalable DCT and ME modules are integrated. In addition,

the remaining modules for the IDCT, (de-)quantization and VLC were adapted towards scalability, such that the scaling of these modules is controlled by the scalable DCT (this is the topic of the next section). The architecture of the applied encoder framework originates from MPEG software-based encoding experiments [73], and was reused for inserting our scalability techniques. The framework will also be used later for comparing the results of the obtained scalable computational complexity and the corresponding picture quality. In order to visualize the obtained scalability of the computations, the scalable modules are executed at various parameter settings, leading to effectively varying key parameters such as the number of DCT coefficients and the number of evaluated MV candidates. When evaluating the overall system complexity, these two key parameters are considered jointly.

The experiments in this chapter were conducted on a Pentium-III Linux system running at 733 MHz. A parameter used for comparisons is the execution time, which will be motivated below. In order to be able to measure the execution time of single functions being part of the complete encoder execution, it was necessary to compile the C++ program of the encoder without compiler optimizations. Additionally, it should be noted that the experimental C++ code was not optimized for fast execution or usage of architecture-specific instructions (e.g. MMX). For these reasons, the encoder and its measured execution times cannot be compared with existing already optimized software-based MPEG encoders. However, in our case, it was ensured that the measured change in execution time results from the scalability of the modules, while programming style, code structures or common coding parameters were kept identical.

It can be argued whether the elapsed execution time of the encoder is a good basis for comparisons. One one hand, this time parameter highly depends on the underlying execution architecture, the programming and the applied operating system. However, on the other hand, it should be considered that the high amount of operations involved in processing and encoding of video sequences decreases the relative importance of the platform architecture. From the system view-point, the parameters that scale the computational complexity of the encoder modules and thus of the whole encoder system, basically vary the number of DCT coefficients computed and processed and the number of performed MV evaluations. Both the processing of the DCT coefficients and the evaluation of MVs need processing cycles on any architecture, where the processing cycles relate to execution time. Of course, the number of processing cycles required to process coefficients of MVs depend on the architecture.

For example, MMX instructions significantly speed up the operations for DCT computation and MV evaluation. This means that the part of the execution time required for such operations changes relatively to the overall execution time, but the scalable techniques still scales the number of operations and thus the execution time of the system. We therefore conclude that it is valid to use the elapsed execution time as a basis for comparisons.

6.2 Inserting scalable DCT and related optimizations

6.2.1 Effect of proposed scalable DCT

The scalable DCT technique that was presented in Chapter 3 computes only a subset S of all possible 64 DCT coefficients C. This feature can be used for the optimization of other modules as follows. The subset S is known prior to the processing inside quantization, dequantization, VLC and the IDCT module. Evidently, coefficients that are not computed are set to zero, and therefore they do not have to be processed further in any of these modules. Since the subset S is known in advance, no additional detection of these zero coefficients is required. The applied scaling of the processing modules of the MPEG encoder is described below.

- The quantization and dequantization requires a fixed amount of operations per processed intraframe or interframe coefficient. For this reason, each coefficient $c \in C \setminus S$ (each produced zero coefficient) is not processed and therefore saves $1/64$ of the total complexity of the quantization and dequantization module.
- The VLC processes the DCT coefficients in zigzag or alternate order and generates runlength-amplitude pairs for coefficients that are unequal to zero, where "runlength" indicates the number of zero-coefficients that are skipped before reaching a non-zero coefficient and "amplitude" is the actual value of this succeeding non-zero coefficient. The usage of the scalable DCT instead of a full DCT increases the probability that zero-coefficients occur, for which no computations are spent.
- The IDCT can be simplified when knowing in advance which coefficients are zero. It is obvious that e.g. each multiplication and addition with zero input-values can be skipped.

The execution times of the modules when coding the "Stefan" sequence and scaling the modules that process coefficients, are visualized in Figures 6.1

and 6.2. Figure 6.1 shows the results from an experiment, where the sequence was coded with I-frames only. In Figure 6.2, similar results are observed from another experiment, for which P- and B-frames are included using a $(12, 4)$-GOP structure.

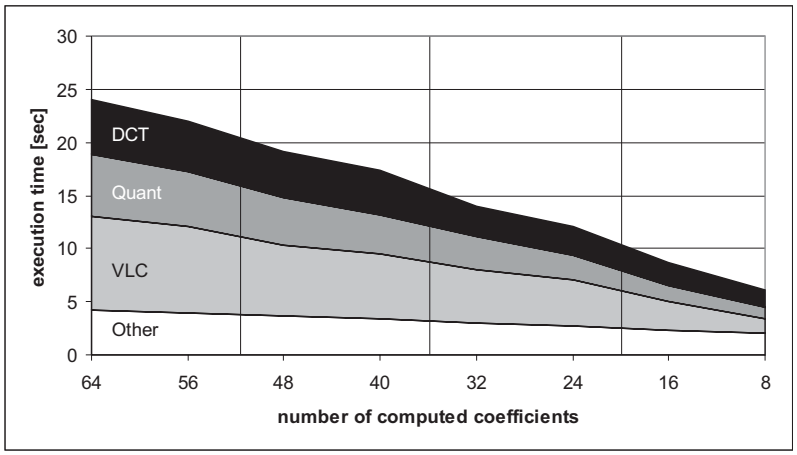

Figure 6.1: *Visualization of the relative computational complexity of DCT coding modules, resulting from integrating the scalable DCT into an I-frame MPEG encoder.*

In both figures, the category "other" is used for functions that are not exclusively used by the scalable modules. When the encoder performs full processing (the results at the left side computing 64 coefficients), the fractions of the elapsed time of the complete encoder (100%) consumed by the individual modules are 25% (DCT), 24% (quant), 34% (VLC) and 17% (other) in Figure 6.1, and 8% (DCT/IDCT), 5% (quant/dequant), 11% (VLC) and 12% (other) in Figure 6.2. The fraction of the elapsed time for the motion estimation and compensation resulting from the experiment in Figure 6.2 is 64% at full processing. The corresponding curve is not shown for convenience, because motion estimation and compensation are not affected by processing a different number of DCT coefficients.

To remove the effect of quantization, the experiments were performed with $qscale = 1$. In this way, the figures show results that are less dependent on the coded video content. The measured PSNRs of the scalable encoder

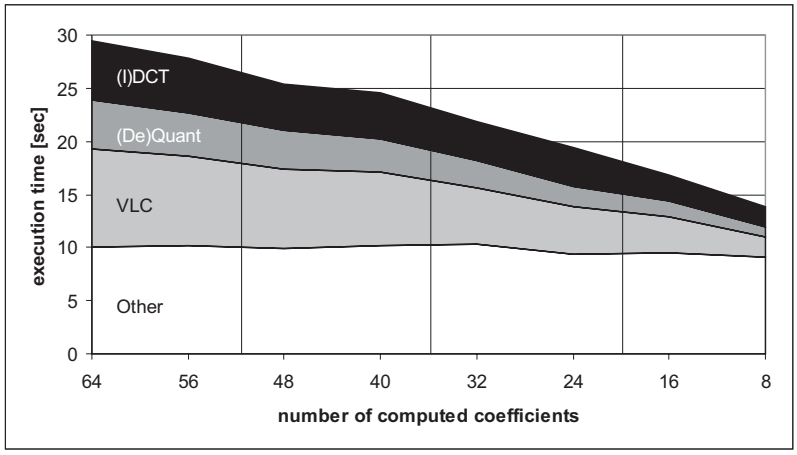

Figure 6.2: *Visualization of the relative computational complexity of DCT coding modules, resulting from integrating the scalable DCT into an MPEG encoder using* $(12, 4)$*-GOPs.*

running at full quality is 46.5 dB for Figure 6.1 and 48.16 dB for Figure 6.2. When the number of computed coefficients is gradually reduced from 64 to 8, the PSNRs drops gradually to 21.4 dB (Figure 6.1) and 21.81 dB (Figure 6.2), respectively. In both figures, the visual quality gradually reduces from "no noticeable differences" down to "severe blockiness".

The experiments show that the scalable DCT and its computational efficiency provide a scalability behavior that is comparable to the other DCT coding modules. It can even be seen that due to the algorithmic optimization, the relative importance of the DCT was decreased and the VLC becomes a significant part of the computational complexity. Note that the (de-)quantization, IDCT and VLC were adapted to the outcome of the scalable DCT. In a refinement, the VLC algorithm could be studied for further scalability. The key parameter that is reused in the quantization and coding is the number of coefficients. Because the remaining steps were modified to scale with this key parameter, the overall range of scalability of the complete encoder is enhanced. This can be seen from the steepness of the top curves upperbounding the black area of the DCT in Figures 6.1 and 6.2. In the following subsection we will show that the applicability of the scalable DCT technique is further improved by reusing data from the block classification that was introduced earlier in this thesis in Section 5.4.

6.2.2 Selective DCT computation based on block classification

When reducing the number of DCT coefficients that are computed, it is obviously preferable to concentrate on those coefficients that correspond with the block content in the sample domain. Generally, the most important coefficients describing the the block contents are found in the upper left corner of the DCT coefficient matrix after transformation. These coefficients represent a mixture of low frequencies that correspond with horizontal and vertical edges in the sample domain. However, this general case is not optimal for picture blocks that are containing for example horizontal or vertical edges only. For this reason, it is beneficial to perform a simple test on the blocks to detect edges. With limited computing power, the computations should be concentrated on coefficients corresponding with the edges as found by the block classification. In order to find different computation orders that concentrate on the relevant coefficients, the priority weighting presented in Section 3.4 is used.

In order to introduce content-adaptive scalable DCT computations at system level, we propose to combine the scalable DCT module with block classification as introduced in Section 5.4. This classification provides simple edge detection that already has been used for scalable ME (see Section 5.5). Depending on the classification, the output quality of the scalable DCT is enhanced by using different computation orders, which are specifically designed for blocks in different classes (see Figure 6.3).

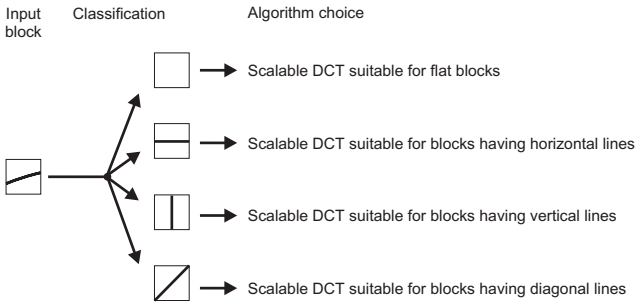

Figure 6.3: *Example of input block leading to vertical edge classification and corresponding scalable DCT computation.*

As a first step, in adopting the previously mentioned concept, the following simple experiment indicates the benefit in quality improvement. In the experiment, the average of the DCT-coefficient magnitudes are computed when

coding the "Table tennis" sequence with I-frames only. The DCT blocks are not quantized ($qscale = 1$). Figure 6.4 shows the statistics of blocks that are classified as having a vertical (left graph) or horizontal (right graph) edge only. It can be seen that the classification leads to a frequency concentration in the DCT coefficient matrix in the first column and row, respectively.

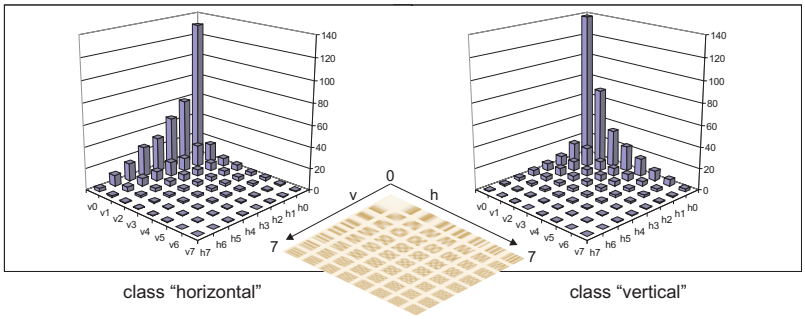

Figure 6.4: *Statistics of the "Table tennis" sequence of the average DCT-coefficient magnitudes with no quantization ($qscale = 1$).*

As a second step, we have evaluated which fast DCT algorithm fits best with a certain preferred orientation in the block data. We have found that the DCT algorithm of Arai, Agui and Nakajima (AAN, see Section 2.3.3) can be used best for blocks with horizontal or vertical edges, while flat blocks have a better quality impression when using the algorithm of Cho and Lee (CheLee, see Section 2.3.2). The difference in perceptual quality can be easily noticed when a limitation in the computation power is applied. Figure 6.5 portrays the results of an experiment, using the two selected DCT algorithms under the constraint of 256 operations, showing clear performance differences between the table edges (AAN is better) and the background (ChoLee is better). The computation orders of both algorithms are designed such that good visual horizontal lines should be obtained. The computation limit was set to 256 operations, leading to 9 computed coefficients for AAN and 11 for ChoLee. The computed coefficients are encircled in the corresponding matrix of DCT basis images. It can be seen that AAN covers all coefficients with mainly vertical energy, while ChoLee covers a mixture from coefficients with both high and low, vertical and horizontal energy. The resulting overall PSNR values are 26.58 dB and 24.32 dB, respectively.

Figure 6.6 shows the visual result of selective DCT computation based on full block classification. Almost all of the background blocks were classified as

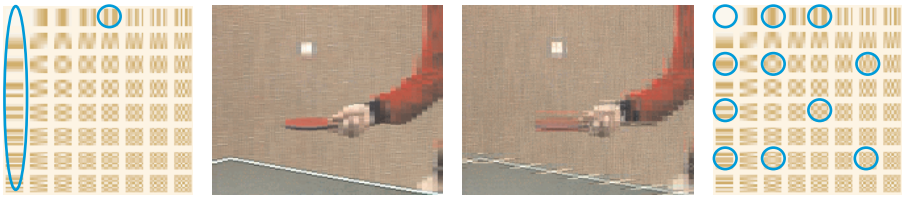

Figure 6.5: *Example of visual results with scalable AAN-DCT (left, with better results for vertical edges) and ChoLee-DCT (right, with better results for the background) with a computing constraint of 256 operations.*

flat blocks and therefore the ChoLee algorithm was chosen for transforming these blocks, while blocks that e.g. contain the table edges are processed by the AAN algorithm. For convenience, both algorithms were set to compute 11 coefficients. Blocks having both horizontal and vertical detected edges at the same time were treated as blocks with vertical edges only, because an optimized computation order for blocks with such complex structures was not readily available and could not be designed in the desired time frame. Treating such blocks as blocks with vertical edges were visually more attractive than treating them as blocks with horizontal edges. The resulting PSNR is 26.91 dB.

Figure 6.6: *Visual result from a full selective DCT transform where flat blocks were processed with the ChoLee-DCT and blocks with detected edges using the AAN-DCT.*

6.2.3 Cyclical DCT for interframe coding

With interframe coding, the DCT will be performed on interframe-coded pictures and intraframe-coded pictures, where interframe coding refers to frame

differences. For applying a selective scalable DCT in an encoder system, we therefore distinguish between original frames and difference frames. The DCT computation on frame differences occurs more often than for original frames ($N-1$ times for (N, M) GOPs). For this reason, a more closer look to interframe DCT coding is required, were a special phenomenon was discovered from using the scalable DCT.

It was found that the DCT-coded frame differences show temporal fluctuations in frequency content. The temporal fluctuation is caused by the motion in the video content combined with the special selection function of the coefficients computed in our scalable DCT. Due to the motion, the energy in coefficients shifts over the selection pattern, so that the quality gradually increases over time. Figure 6.7 shows this effect from an experiment when coding the "Stefan" sequence with "IPP" frames ($(12, 1)$-GOPs are used), while limiting the computation to 32 coefficients. The camera movement in the shown sequence is panning to the right. It can be seen that for example the artifacts around the background text decrease over time.

Figure 6.7: *Gradual reduction of coding artifacts in time for interframe coding using a scalable DCT.*

The aforementioned phenomenon was mainly found in sequences containing modest motion. The described effect leads to the idea of temporal data partitioning using a cyclical sequence of several scalable DCTs with different coefficient selection functions. The complete cycle would compute each coefficient at least once. Temporal data partitioning means that the computational complexity of the DCT computation is spread over time, thereby reducing the average complexity of the DCT computation (per block) at the expense of a delay in build-up of visual quality. Using this technique, picture blocks having a static content (blocks having zero motion like non-moving background) and which do not show temporal fluctuations in their frequency content, will obtain the same result as a non-partitioned DCT computation after full computation of the partitionized DCT.

Based on the concept of temporal data partitioning, N subsets s_i (with $i = 0, 1, ..., N-1$) of coefficients are defined such that

$$\bigcup_{i=0}^{N-1} s_i = S, \tag{6.1}$$

where the set S contains all 64 DCT coefficients. The subsets s_i are used to build up functions f_i that compute a scalable DCT for the coefficients in s_i. The functions f_i are applied to blocks with static contents in a cyclical sequence (one per intercoded frame). After N intercoded frames, each coefficient of these blocks is computed at least once.

In order to measure the effect of the cyclical DCT during interframe coding, we have conducted an experiment using the "Table tennis" sequence as follows. The computation of the DCT (for intraframe coding and interframe coding) is limited to 32 coefficients. The coefficient subsets that are used are shown in Figure 6.8.

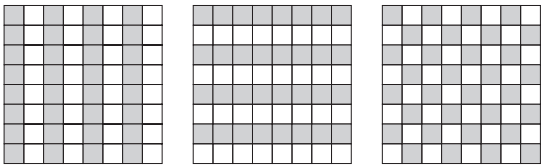

Figure 6.8: *Example of coefficient subsets (marked gray) used for cyclical DCT computation in an interframe coding environment with a limitation of 32 coefficients per subset.*

Figure 6.9 shows the improvement in PSNR that is obtained with this concept. Three curves are shown in this figure, plotting the obtained PSNR of the coded frames. The medium gray curve results from coding all frames as I-frames, which is taken as reference in this experiment. The other two curves result from applying a GOP structure with $N = 12$ and $M = 4$. First, all blocks are processed with a fixed DCT (light gray curve) computing only the coefficients as shown in the left subset of Figure 6.8. It can be seen that when the content of the sequence changes due to movement, the PSNR raises. Second, cyclical DCT transformation during interframe coding, based on temporal data partitioning as described above, is applied to the coding process, which results in the dark gray curve. The dark gray curve shows an improvement to the light gray curve in case of no motion. The comb-like structure of the curve results

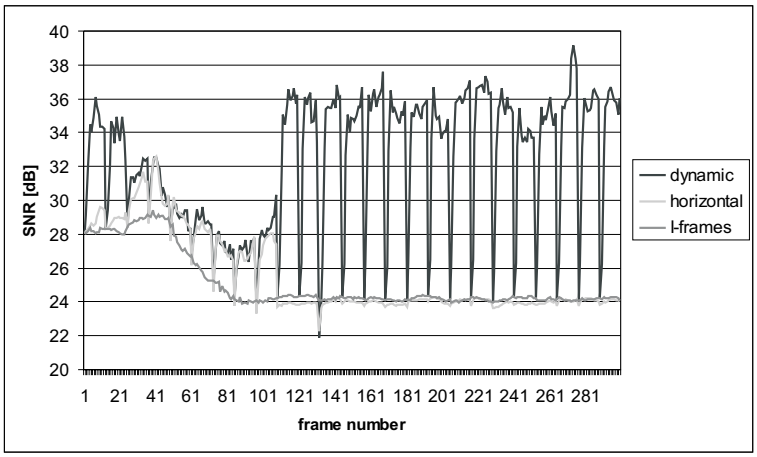

Figure 6.9: *PSNR measures for the coded "Table tennis" sequence, where the performance of a cyclical scalable DCT computation is compared with other forms using only 32 coefficients.*

from the periodic I-frame occurrence that restarts the quality build-up. The low periodicity of the quality drop gives a visually annoying effect that can be solved by computing more coefficients for the I-frames. Although this seems interesting, this was not further pursued because of limited time.

6.3 Combining SMART and CARES into one scalable ME system

In Sections 5.3 and 5.5, we presented the novel ME techniques SMART (located at the structural level) with frame processing in display order and the CARES ME algorithm (located at the vector-selection level), based on block classification. Both techniques have been combined and integrated in the MPEG encoder framework for evaluating their combined scalability performance. We experimented with different video sequences and we scaled the computational complexity with two scalability parameters.

The first scalability parameter (labeled "priority level") controls the SMART ME technique and defines the number of motion-vector fields that are computed per SGOP (we employ a GOP size of $N = 12$ and $M = 4$, thus an "IBBBP" structure). The priority order of vector fields was identical to the

evaluation of the SMART technique in Section 5.3.4. Summarizing, vector fields from Stage 1 have higher priority than vector fields from Stage 3, forward motion has higher priority than backward motion and fields with shorter temporal distances have higher priority than fields with larger temporal distances. The priority order of MPEG vector fields to be refined in Stage 3 is again $\underline{f_1}$, $\underline{f_2}$, $\underline{f_3}$, $\underline{f_4}$, $\underline{b_3}$, $\underline{b_2}$, $\underline{b_1}$. The priority order of the vector fields in Stage 1 is defined such that the concatenated chain of vector fields covers the total prediction depth of a SGOP k, starting from frame $(I/P)_{k*M}$ to $(I/P)_{(k+1)*M}$ for the M forward fields (with $k \epsilon N_0$) and then vice-versa for the M backward fields. The second parameter (labeled "threshold") for achieving scalability varies the classification threshold t of the block classification (with $t \epsilon N_0$) used in the CARES ME algorithm.

Let us now consider the scalable performance of the combination of SMART and CARES ME. We have evaluated the number of vector evaluations and the resulting PSNR of the motion-compensated frames (without DCT, quantization and coding) as a function of the two scalability parameters. Figure 6.10 results from an experiment using the "Stefan" sequence, where both scalability parameters are varied over a broad range. The figure shows the average number of vector candidates that were evaluated for each macroblock of the computed MPEG motion-vector fields. The priority level refers to the number of computed vector fields as given by the priority order used for SMART, and the indicated threshold in the horizontal direction is the same as used in CARES. The figure shows that the MV evaluations scale smoothly with the threshold and the number of computed vector fields. The impact of both parameters on the scalability is roughly equal. Experiments with other sequences revealed that the threshold parameter becomes more important for sequences having less flat blocks, whereas the priority-level parameter dominates for sequences with more motion. Note that the choice of a small non-zero threshold already leads to a significant reduction of the average number of vector evaluations.

The obtained average PSNR of the predicted P- and B-frames (taken after motion compensation and prior to computing the frame-difference signal) is shown in Figure 6.11. In the figure, the bold black line on the surface follows the path of the highest PSNR. For a full-quality comparison, we have considered full-search block matching with a search window of 32×32 pixels. The new combined ME technique slightly outperforms full search by up to 0.39 dB PSNR measured from the motion-compensated P- and B-frames of this experiment (25.31 dB instead of 24.92 dB). From the two Figures 6.10

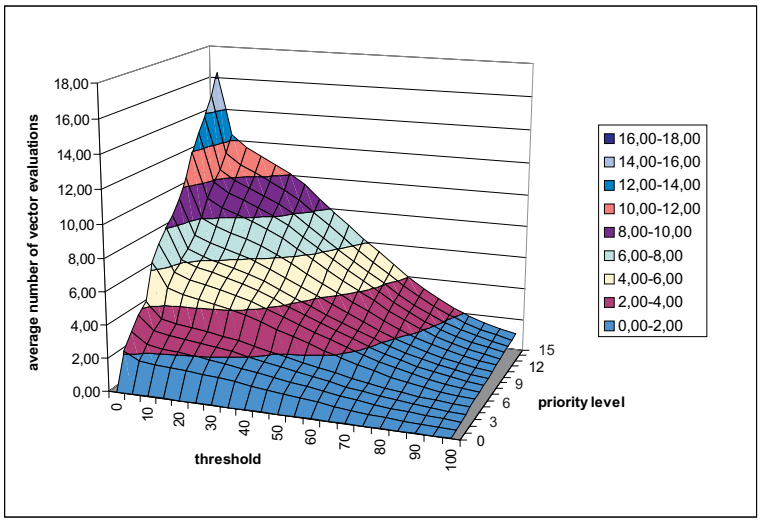

Figure 6.10: *Average number of MV evaluations performed per macroblock for the "Stefan" sequence as a function of two scalability parameters.*

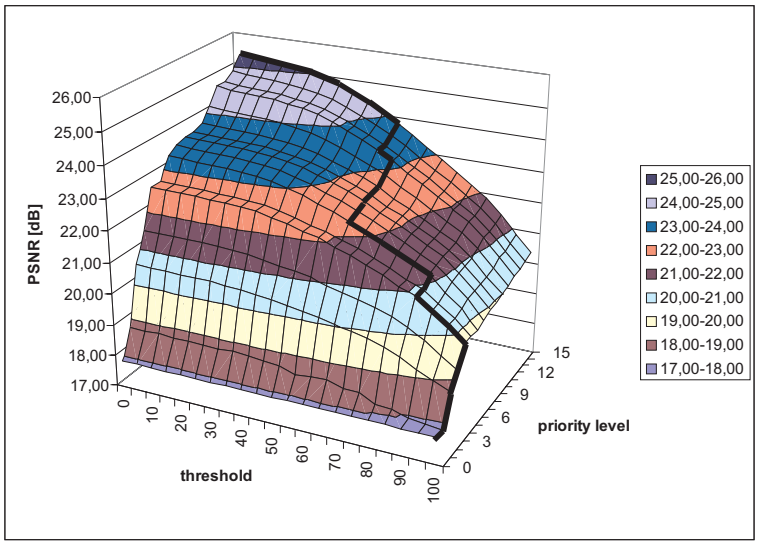

Figure 6.11: *Average PSNR of the motion-compensated P- and B-frames for the "Stefan" sequence as a function of the scalability parameters.*

and 6.11, we have concluded that for further system evaluation, we can easily exclude operating points with a zero classification threshold in CARES (giving the drop of the maximum number of MV evaluations), because the PSNR grows only marginally at very small threshold values.

Table 6.1 gives a more detailed insight in the results of the "Stefan" sequence, which contains complex motion. The first data column (Evals. per MB) shows the average number of vector evaluations that were performed per macroblock by the listed ME algorithms, and the second column ("PSNR") shows their obtained quality. For comparison with our scalable ME system, the row below each ME algorithm entry is named "scalable ME" and contains the average number of vector evaluations required for our system (at optimal configuration for this experiment) to reach the same quality as the compared (fast) ME algorithms listed in the table. Note that MV evaluations pointing outside the picture are not performed, which results in numbers that are lower than the nominal values (e.g. 923.52 instead of 1024 for 32×32 full-search (FS)). From the experiment, it can be seen that our scalable ME system requires only a fraction of MV evaluations compared to the other ME algorithms (even the two fast ones), while obtaining the same quality as the other ME algorithms.

Algorithm	Evals/MB	PSNR
2DFS (32×32)	923.52	24.92
Scalable ME	9.42	24.92
NTSS[69]	25.13	22.63
Scalable ME	2.69	22.64
Diamond[65]	21.77	22.53
Scalable ME	2.31	22.55

Table 6.1: *Average PSNR of the motion-compensated P- and B-frames using the "Stefan" sequence for different ME algorithms (MB is macroblock).*

A PSNR comparison as a function of the number of vector evaluations for a set of video sequences is shown in Figure 6.12, where we used the sequence "Stefan" with high motion, and "Renata" and "Table Tennis" having less motion. In the figure, the difference in gray levels indicate the different sequences. The dots show all possible configurations of the scalable ME system and the curve of connected points follows the optimum. For this PSNR comparison, the sequences were also processed using a diamond search (diamond-shaped

symbols in the figure) and NTSS (square-shaped symbols in the figure). The horizontal dotted lines indicate the PSNR obtained with a 32×32 FS.

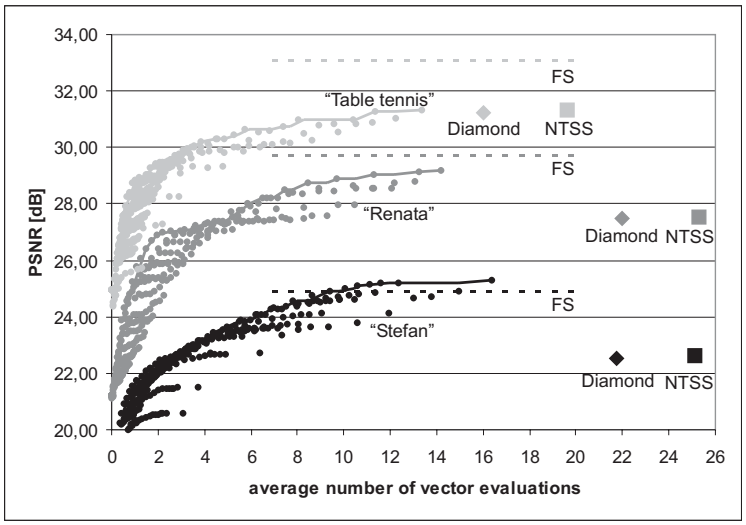

Figure 6.12: *PSNR comparison as a function of the number of vector evaluations for the scalable ME system (dots = all configurations, solid line = optimum) with diamond search (diamond) and NTSS (square), using the "Stefan" (black), "Renata" (medium gray) and "Table tennis" sequence (light gray). The dotted horizontal lines indicate the PSNR obtained for a* 32×32 *FS.*

The curves in Figure 6.12 show a waterfall effect, which means that a fast quality build-up is realized at the beginning of the scalability range, and that finally, the quality generally approaches full-search quality. The bending point of the waterfall is between 3 and 5 vector evaluations per block. It can also be seen that for more complex scenes, the scalable ME system obtains the quality of FS at about 10-16 vector evaluations per block. For the "Table tennis" sequence, it seems that the provided scalability range does not reach the full-search level. However, because the absolute quality level is already much higher than for more critical scenes, this was not explored further.

As discussed at the beginning of this chapter, the execution time will be used as basis for system comparisons. In this section, the scalable ME techniques SMART and CARES are combined into one scalable ME system and up-to-now experiments for this ME system are performed based on the number of

MV evaluations. To show the relation between the number of MV evaluations and execution time, the scalable ME system has been integrated into our MPEG encoder framework. In Figure 6.13, the execution times of the MPEG modules when encoding the "Stefan" sequence is shown, while scaling the computational complexity of the scalable ME system as a function of MV evaluations per macroblock. It can be seen that the curve for the ME block

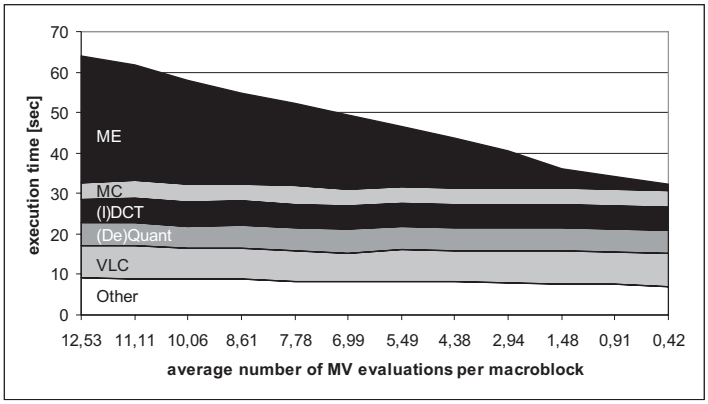

Figure 6.13: *Example of ME scalability inside a complete encoder when using a* $(12, 4)$*-GOP ("IBBBP" structure) for coding.*

scales roughly linearly with the number of MV evaluations, whereas the other processing blocks remain constant. To remove the effect of quantization, the experiments were performed with $qscale = 1$. In this way, the figure shows results that are relatively independent from the coded video content. More specifically, the comparison of execution times of the MPEG encoding modules should be compared of the ultimate left side of the Figure 6.13, where it shows that the scalable ME consumes roughly half of the execution time of the complete encoder (compare this with approximately 62% for the Diamond search, see Section 6.2). At this point, the integration of all scalable modules in the MPEG encoder framework has been finalized and enables experiments with a full scalable MPEG encoder (see next section).

6.4 Combined effect of scalable DCT and scalable ME

In this section, the complete scalable ME (SMART and CARES) and DCT techniques are combined in the MPEG encoder and the scalability rules for

(de-)quantization, IDCT and VLC are applied as described in Section 6.2.1. A large design space is obtained when all MPEG modules are scalable at the same time. In the following experiments, the resulting performance of the scalable encoder is usually presented in the form of obtained quality (PSNR) or bit rate, for a predetermined value of the execution time (computational complexity). The execution time can be controlled as a function of our two key scalability parameters as discussed in the previous two sections, i.e. MV evaluations and computed DCT coefficients. By varying these parameters, the scalability and complexity can be manipulated over a large range (the design space).

6.4.1 Open-loop MPEG encoder

Figure 6.14 portrays the obtained average PSNR of the coded "Stefan" sequence (CIF resolution) and Figure 6.15 shows the obtained bit rate corresponding with Figure 6.14. The experiments are performed with a $(12, 4)$-GOP and $qscale = 1$. Both figures indicate the large design space that is available with the scalable encoder without quantization and open-loop control. The horizontally-oriented curves refer to a fixed number of DCT coefficients (e.g. 8, 16, 24, 32, ..., 64 coefficients), whereas vertically-oriented curves refer to a fixed number of MV evaluations. A normal coder would compute all 64 coefficients and would therefore operate on the top horizontal curve of the graph. The figures should be jointly evaluated. Under the above-mentioned measurement conditions, the potential benefit of the scalable ME is only visible in the reduction of the bit rate (see Figure 6.15), since an improved ME leads to less DCT coefficients for coding the difference signal after motion compensation in the MPEG loop.

In Figure 6.15, it can be seen that the bit rate decreases when computing more MV candidates (going to the right). The reduction is only visible when the bit rate is high enough. For comparison, the markers "A", "B" and "C" refer to three points within the design space. With these markers, the obtained bit rate of the scalable encoder is compared with the encoder using alternative ME algorithms. Marker "A" refers to the configuration of the encoder using the scalable ME system, where the same bit rate and video quality (not the computational complexity) is achieved compared to diamond search. As mentioned earlier, the diamond search performs 21.77 MV candidates on the average per macroblock. Our scalable coder, operating under the same quality and bit rate combination as the diamond search in marker "A", results in 10.06 average MV candidates, thus 53.8% less than diamond search. Mark-

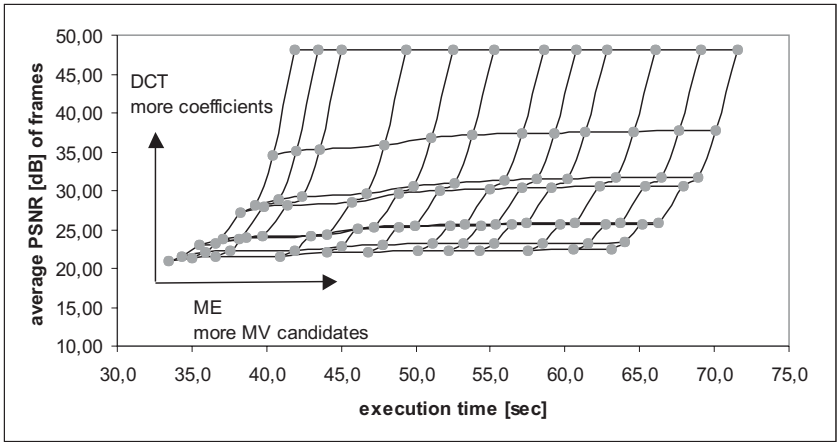

Figure 6.14: *Design space of obtained execution times and PSNRs of a scalable full MPEG encoder, which scales as function of computed DCT coefficients and MV evaluations.*

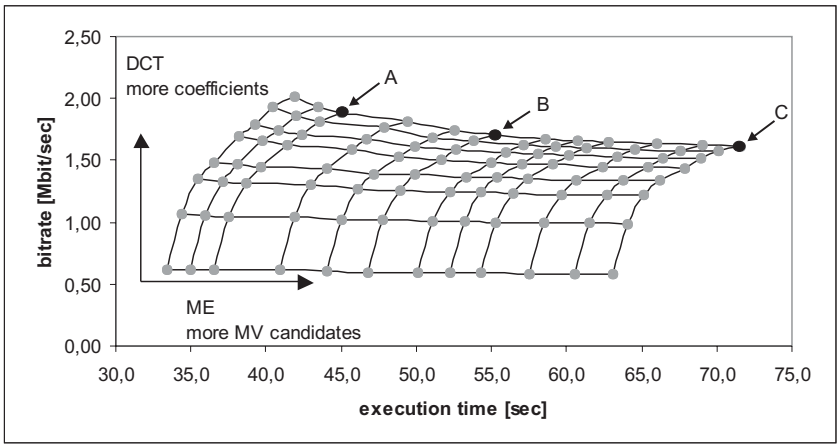

Figure 6.15: *Design space of obtained execution times and bit rates of a scalable full MPEG encoder, which scales as function of computed DCT coefficients and MV evaluations.*

ers "B" and "C" result from using full-search ME with a 32×32 and 64×64 search area, respectively, requiring substantially more MV evaluations (1024 and 4096, respectively).

6.4.2 Closed-loop MPEG encoder

Figure 6.14 and 6.15 both present a large design space, but in practice this is limited due to the quantization and bit-rate control. Figure 6.16 shows the influence of the processed video sequences on the obtained design space. For the experiment, we used three different sequences having CIF resolution, which are coded at a target bit-rate of $1500\,kbps$. The sequences are "Stefan" (solid curve, fast motion), "Foreman" (dashed curve, medium motion) and "Table tennis" (dotted curve, slow motion). With the target bit-rate of $1500\,kbps$, for all sequences and at any time during the encoding process, the rate controller was active (closed-loop coding) and able to select a valid $qscale$ value for accurately realizing the chosen bit rate.

The curves shown in Figure 6.16 are constructed in the same way as the results from Figure 6.14, but in Figure 6.16, the presentation of the curves is simplified for clarity by drawing the boundary curves only. The resulting execution times were normalized for comparing sequences of different lengths.

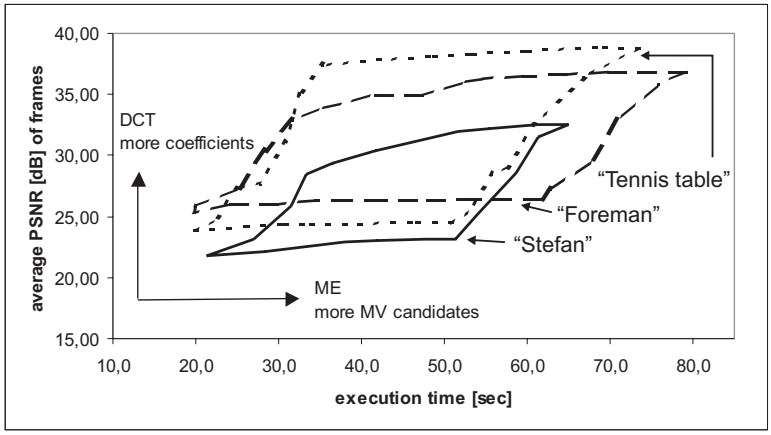

Figure 6.16: *Obtained PSNRs for different sequences with CIF-resolution, which were coded using* $(12, 4)$*-GOPs and bit-rate control to target* $1500\,kbps$.

When comparing the curves from Figure 6.16 with Figure 6.14, the quantization forces the design-space area to a lower PSNR interval. Furthermore, less computation time is required leading to a shift to the left, since a lower number of DCT coefficients is computed. When going upwards in the graph, the number of computed DCT coefficients increases, and when going to the right, the number of evaluated MV candidates increases. It can be seen that the obtained design space is larger for sequences having less motion. Since the bit rate is fixed for this experiment and therefore we do not have to find a good balance between PSNR and bit rate, only the top part of the curves are of interest. It can be seen that under these conditions, it is preferable to first up-scale the DCT prior to improving the ME, because it adds more to the PSNR[1].

6.4.3 MPEG encoder with variable GOP and bit rate

In this section, the same encoder of the previous section is used, but now we investigate the influence of using various GOP structures and bit rates. For the conducted experiments, we have used the "Stefan", "Foreman" and "Table tennis" sequence for showing the obtained improvements when using a scalable MPEG encoder with respect to a non-scalable encoder. We varied the bit rates and GOP structures, because both are primary coding parameters in the MPEG coding standard, having influence on both the execution time and quality of an encoder[2]. Figure 6.17 is an extended and more detailed view on the most interesting area of Figure 6.16. The depicted graphs are located on the top curve of the design space as shown in Figure 6.16, resulting from the scalable encoder under variable GOP/bit-rate/sequence conditions. For comparison, the graphs include the performance of the same encoder framework when using non-scalable algorithms (full fast DCT computation and the popular and efficient "Diamond" ME [65]). The non-scalable performance curves are shown at the right side of each graph. The left three figures in the array of pictures (when rotating the page 90 degrees clockwise) are based on a

[1] It is estimated that the DCT and the ME are scaled simultaneously for optimal performance when configuring the scalable MPEG encoder such that the scalability algorithm for the DCT computation is different for intracoded frames and intercoded frames (as already mentioned in Section 6.2.3). Although this seems interesting, this was not further pursued because of time limitations.

[2] Note that we consider an encoder that does not modify the sampling standard of the input video-signal. This means that the full spatial and temporal resolution of the input signal is processed. We do *not* consider a system that reduces the spatial and/or temporal resolution for reducing the processing speed and re-interpolate the original resolutions afterwards at the decoder, because the encoder in such a system is not directly affected.

fixed target bit-rate of $1500\,kbps$ and various GOP structures. The right three figures of the array have a fixed $(12, 4)$-GOP structure and vary the bit rate.

A first major result is that in all cases, the scalable encoder outperforms the non-scalable encoder in both quality and execution time. Apparently, the special attention given to the individual processing steps and the limited amount of extra memory has been worthwhile to obtain a substantial performance improvement. A second result deals with the obtained scalability range. The control of parameters (bit rate and GOP structure) of a non-scalable MPEG encoder does offer some form of scalability. This scalability reduces the execution time with about 20%-30%. The scalable encoder shows a much larger scalability range of about 50%-60%, which provides more fine-grained complexity scalability and over a larger range. With respect to the absolute quality level of the encoded sequences, the scalable encoder obtains higher PSNR values than the non-scalable encoder with increased motion in the sequence. It can be concluded that the presented scalable MPEG encoding system enables high-quality processing for high-end devices at lower computing power than coders using non-scalable algorithms. However, it should be noted that we have taken advantage of a larger freedom in a software-based implementation that employs a slightly increased memory usage.

A final experiment has been performed to make a challenging quality comparison between the scalable and non-scalable MPEG encoder. In this experiment, the non-scalable encoder uses a 32×32 full-search ME. As for earlier experiments, the three sequences "Stefan", "Foreman" and "Table tennis" have been coded using $(12, 4)$-GOPs and a target bit-rate of $1500\,kbps$. In Table 6.2, the (highest obtained) PSNR numbers that result from our scalable encoder and the non-scalable encoder are compared. In addition, the table shows the average $qscale$ value that the bit-rate controller adopted for achieving the required bit rate (of course a lower $qscale$ relates to higher PSNR numbers). It can be seen from the table that our scalable encoder performs better for the "Stefan" sequence that has high motion (1.28 dB higher PSNR), and equal for the "Foreman" sequence that has medium motion (0.05 dB higher PSNR). For the "Table tennis" sequence, a 0.43 dB lower PSNR is observed when using our scalable encoder, where the required average $qscale$ values are low and almost equal. Note that PSNR differences are less noticeable in perceptual quality at high PSNR number ranges (as obtained for the "Table tennis" sequence) than at lower PSNR number ranges (as obtained for the "Stefan" sequence).

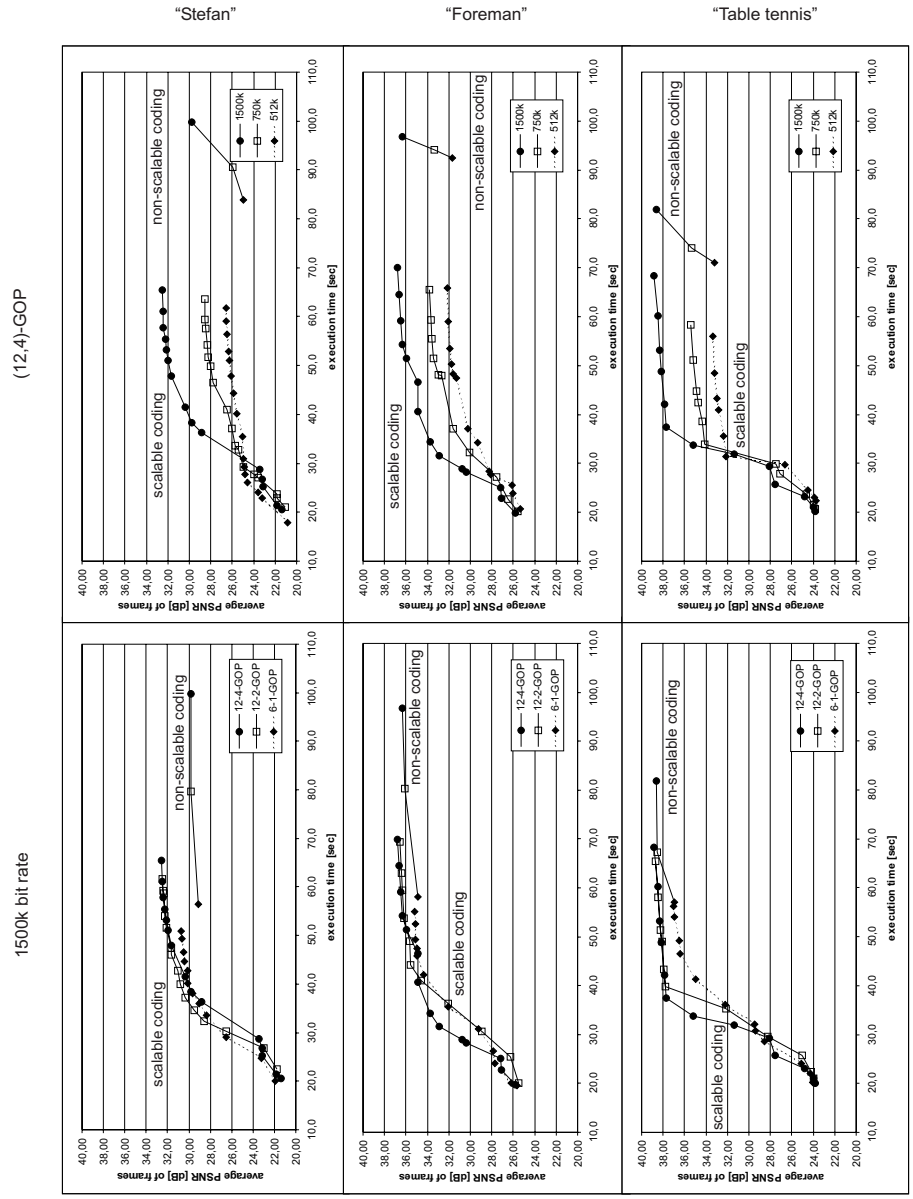

Figure 6.17: *PSNR vs. execution times of the scalable encoder for various bit rates, GOP structures and sequences. The left graphs are coded with fixed target bit-rate of* $1500\ kbps$*, and the right graphs are coded with a fixed* $(12, 4)$*-GOP structure. The graphs include operations points of the encoder using non-scalable algorithms.*

	non-scalable coder		scalable coder	
Sequence	PSNR	average $qscale$	PSNR	average $qscale$
"Stefan"	31.33	13.4	32.61	11.0
"Foreman"	36.82	5.7	36.87	5.8
"Table tennis"	39.26	4.5	38.83	5.0

Table 6.2: *Comparison of PSNRs and average $qscale$ numbers achieved by MPEG encoders using quality-optimized non-scalable algorithms or our scalable algorithms.*

6.5 Discussion on further optimization of the scalable encoder

The complexity vs. PSNR results that are obtained with the scalable MPEG encoder presented in this thesis, are based on a number of experiments for evaluating the scalability of encoder modules and assumptions on good settings for the evaluated modules. As can be seen from the previous section, the obtained results show faster and scalable processing than the reference encoder design from which we started, and higher quality of the video output has been achieved. Unfortunately, despite the good results, it cannot be judged how close the proposed configurations are to the theoretically optimal system. The reason for this is found in the high amount of configurations for the scalable encoder modules and their dependencies on the actual video-input signals. Let us first discuss the dependencies below, in order to show the complexity of the complete optimization problem.

The computational complexity of the scalable (I)DCT and (de-)quantization are independent from the video-input signal, so that the complexities of these modules are known in advance. Whereas the scalable IDCT and (de-)quantization are designed such that their output is equivalent with full processing, their computational complexity depends on the scalable DCT (the number of computed coefficients) and the actually processed video signal. The DCT output-quality is somewhat predictable, because DCT coefficients occur with different probabilities, which can be measured. The contribution of a coefficient to the resulting output quality is more accurately predictable when a classification of a the block contents is made in advance.

The VLC is designed not to take decisions on the output quality (considering bit-rate control as a separate step) and codes the quantized DCT coefficients

according to the MPEG-2 standard. The computational complexity of the VLC is somewhat predictable and varies mainly with the measurable number of non-zero DCT coefficients that are coded.

The scalable ME in our system consists of two techniques, namely SMART ME with integrated CARES ME. The computational complexity and output quality of the CARES ME highly depends on the processed video sequence. In order to predict the computational complexity of the scalable ME, a model of the complexity should be defined based on the number of non-flat blocks per frame, the amount of motion and the computational complexity resulting from coding different types of video sequences. Similarly, it should be possible to model the quality of the coded sequences resulting from scalable ME. However, modeling the quality will be difficult, because SMART ME works independently of GOP structures and the accuracy of a MV field has an impact on the accuracy of other (later computed) vector fields that do not belong to the same GOP. This may finally result in an optimization of the complete video sequence as a whole.

The aforementioned different considerations with respect to computational complexity and quality, yield a pluriform multi-dimensional design space for which the definition of an optimal configuration of a scalable MPEG encoder is currently unknown. Such optimization problems may be formulated with e.g. the use of Lagrangian multipliers incorporating each dimension, or other techniques. In the past, video compression coders were optimized using such approaches and aiming at the best quality for a given bit rate, or vice versa (e.g. see [74]). In our case, such an optimization should be extended with the parameter "scalability". This parameter has not been fully modeled yet. In order to continue this discussion, let us assume that we would take our two scalability parameters (MV evaluations and computed DCT coefficients). For these extended dimensions, the encoder should be theoretically optimized. As already stated, this would still lead to an optimum for one encoder configuration, whereas we have many possibilities. Furthermore, the parameter MV evaluations has to be spread over two scalable ME techniques, which is an optimalization problem of itself. Finally, we already indicated that the motion vectors do have a recursive influence outside the GOP boundaries and the content-dependent behavior of ME. This discussion clearly points out that finding the theoretical optimum configuration is a scientific problem that is worth a study on itself.

6.6 Conclusions

In this chapter, the scalable DCT and ME techniques are integrated into a complete scalable MPEG encoding system for performance evaluations and showing the design space that is obtained with the scalability techniques. It has been shown that the computational complexity of the IDCT, quantization, dequantization and VLC processing blocks scale with the number of DCT coefficients that are computed by the scalable DCT. Therefore, this number has been selected as a primary parameter for scalability in the complete MPEG encoder framework. Another second result of our experiments is that the scalable DCT has an integrated coefficient selection function, which may enable a quality increase during interframe coding. This phenomenon can lead to an MPEG encoder with a number of special DCTs with different selection functions, and this option should be considered for future work. This should also include different scaling of the DCT for intra- and interframe coding of video pictures.

For the ME block in the scalable MPEG encoder, we have found that the combination of the SMART and CARES ME technique leads to a highly scalable and efficient motion estimator in terms of computational complexity and resulting PSNR quality. With respect to quality, experiments with the new scalable combined ME show that a full processing of the scalable ME compares well with a 32×32 full-search ME (or even outperform it). With respect to computational complexity, a comparison with existing fast ME algorithms published for MPEG reveals that our scalable ME requires roughly 30% to 90% less MV evaluations for different sequences. The complexity scalability of the complete scalable ME proposal is between roughly 1 and 15 vector evaluations per macroblock on the average, leading to a global PSNR variation of 7.0 to 7.4 dB PSNR of the motion-compensated frames. For scalable ME, future work should examine the scalability/efficiency potentials of using various fixed and dynamic GOP structures, and on concentrating or limiting the ME to frame parts of which the content (could) have the current viewer focus (region-of-interest coding).

The scalability of the complete scalable MPEG encoder, expressed in execution time, can be gradually reduced to roughly 30% of its original execution time, while as the same time providing a wide range of video quality levels (roughly from 20 dB to 48 dB PSNR in average) and compression ratios (from 0.58 to 2.02 Mbit/s, CIF resolution). The introduced new techniques for scalability show a large range of obtained computational complexity. Per-

formance evaluations have been carried out to show the design space of the scalable MPEG encoder system when scaling all system modules at the same time. Experiments have been conducted with various bit rates, GOP structures and sequences. It has been shown that our scalable MPEG encoder clearly outperforms a non-scalable encoder in all measured cases (computational complexity and PSNR quality). The obtained scalability range of our scalable encoder in terms of computational complexity is about a factor of three, whereas a non-scalable parameterized MPEG encoder has a factor of about 1-1.5. This improvement offers fine-grain complexity scalability. At the same time, our scalable MPEG encoder provides a wide range of video quality levels of roughly between 21.5 dB and 39.0 dB. The obtained computational-complexity scalability is seen sufficient for a large range of stationary and portable MPEG coding applications.

It has been debated that the actual scalable MPEG encoder represents a single point in a large pluriform multi-dimensional design space. Presently, it cannot be judged how close the proposed scalable encoder configurations are to the theoretically optimal system. In order to do so, it would be required to define a scientifically exact definition of complexity scalability, which has not been studied. Instead, we have found at least two important scalability parameters for the computational complexity of an MPEG encoder. Due to the recursion in the scalable ME and the aforementioned remarks, the formulation of the optimization problem is rather complicated and requires a follow-up study.

CHAPTER 7

Conclusions

The preceding chapters have resulted in a number of complexity scalable processing modules for MPEG encoding. The scalable modules have been integrated into an MPEG encoding framework, for evaluation of the obtained scalability on the system level. This chapter discusses the applicability of the results of this thesis for various possible video systems and alternative encoding standards, such as H.264.

7.1 Conclusions of the thesis chapters

In this thesis, the scalability for MPEG encoding was studied and various results in terms of algorithms and system design have been presented. The described research was part of a larger cooperation framework, in which encoder and decoder systems were studied separately. The research focus of this cooperation was unique of its kind. This also holds for the contents of this thesis. As far to the knowledge of the author, this thesis presents the first comprehensive approach for obtaining the computational complexity scalability of a *complete* MPEG-2 encoding system, where both the individual processing blocks and their interaction was examined. This section aims at summarizing

the research results. In the remainder of the chapter, we will discuss possible future work and applicability in alternative applications or coding standards.

The introductory Chapter 1 has discussed the need for flexible solutions of video encoding algorithms for usage in multiple devices and in certain applications, such as e.g. high-end TV. We have found that the large variety of mobile devices show various resource limitations that motivate the design of complexity scalable systems. Complexity scalability offer a more efficient reusage of algorithms and encoding systems for a set of devices and architectures. The MPEG-2 coding standard was briefly presented. The scalability concept of the MPEG-2 standard does not provide efficient solutions for the outlined application range, because the control of coding parameters is the only way to implement scalability and its range is too limited.

Two key functions of MPEG encoding have been examined in detail, namely DCT and motion estimation (ME), because they require the largest computational effort (in original form). In Chapter 2, several fast DCT computation algorithms were studies in detail, in order to explore possibilities for scalability. This study did not lead to new fast algorithms, but it provided direct insight in different computing approaches and it lead to a new computation technique that is suited for scalability and can be applied to any fast DCT algorithm.

In Chapter 3, it was found that the DCT computation can be enhanced with a scalability feature by analyzing the data-flow of the fast DCT algorithm. The purpose of the algorithm analysis is to find shared computation nodes that can be reused during the computation of different DCT coefficients, thereby preventing costly recomputation of these nodes. The analysis leads to a computation order that maximizes the number of computed coefficients under computing power limitations, where the properties of the underlying computing architecture, like special operations for computation or the costs for accessing memory, are taken into account. The computation order can be enhanced by priority weighting of the coefficients. Experiments have shown that the scalability-optimized computation order computes significantly more coefficients under computing power limitations, than when using the well-known zigzag order. With considerable computing limitations, the difference in coefficients is so significant that a considerable quality increase (PSNR and perceptual) is obtained. When computing power is less constrained, the scalability-optimized computation order still outperforms the zig-zag computation order in obtained PSNR. However, the perceptual difference becomes

small and sometimes the choice is in favor of the conventional technique. The implementation of a scalable DCT offers a large range in scalability and it controls the scalability of subsequent encoding blocks (IDCT, (de-)quantization and VLC) that process a the variable number of DCT coefficients.

Chapter 4 discusses a number of fast ME algorithms, which are typically used in MPEG encoder systems. The focus of this exploration was on selecting an algorithm that could be used for scalable ME concepts. It was found by various comparisons (complexity and quality, and the combination of these two) that algorithms based on recursive ME are attractive for the construction of a scalable ME for MPEG encoding, since they already provide near full-search quality at low cost. The algorithmic study revealed that the algorithm complexity itself has a negligible importance, rather than the resulting number of motion vector (MV) evaluations.

Chapter 5 presents two techniques for introducing scalability in MPEG ME. The first scalability technique, called SMART, processes the MV fields in three stages. Stage 1 precomputes MV fields for succeeding frames at the entrance of the encoder. In Stage 2, MV fields are scaled and added or subtracted (thus having multi-field references) to approximate the MV fields that are required for MPEG coding. These computations are less complex than performing advanced vector search. The computation of e.g. the backward ME can be omitted to save computational effort and memory-bandwidth usage. Stage 3 optionally refines the approximated MPEG MV fields for approaching the quality of a conventional MPEG encoder employing a high-quality ME. The second scalability technique for ME, called CARES, is a recursive ME (RME) algorithm that is based on block classification, depending on the picture content. The applied block classification provides (in addition to other classifiers) the information whether it is more likely to find a good MV in up-down or left-right search directions. This feature leads to a reduced set of MV candidates that are evaluated, in comparison with state-of-the-art ME algorithms for MPEG applications. The CARES ME algorithm proposes good MVs to surrounding blocks for further evaluation. This extension prevents the re-evaluation of identical MVs for a block, which regularly occurs in state-of-the-art RME. The disadvantage is that a significant cache memory will be required. It seems that the CARES algorithm is most attractive for a software-based implementation.

The presented scalable ME algorithms are flexible with respect to the chosen GOP structure, which preserves freedom in the application domain. When

more closely observed, it can be noticed that SMART and CARES can be combined into one scalable ME system, because they operate on different levels of processing (structural and vector-selection level). This combination was explored in a following chapter. We have found that a reduction of the overall computational effort of the ME can be reduced by content-adaptivity of the processing. In the CARES ME algorithm, a simple block classification based on edge detection was already sufficient to provide this adaptivity.

In Chapter 6, the above-mentioned scalability techniques were integrated in a complete MPEG encoding system. However, a full scalable MPEG encoder did not exist, so that all remaining blocks of the MPEG encoder had to be redesigned for scalability and cooperation with the included new scalable DCT and ME. The effect of the scalable techniques on the computational complexity of the encoding process and their mutual dependencies were evaluated. It was shown that when exploiting the scalable DCT inside an encoder system, the encoder performance is further enhanced by taking the variable number of coefficients of the scalable DCT into account. The enhancement leads to scalable modules for the IDCT, (de-)quantization and VLC, such that they process only the coefficients computed by the scalable DCT module. With respect to ME, the combination of the two scalable ME techniques SMART and CARES leads to an efficient and scalable ME system, providing a wide range of frame-prediction quality and outperforming state-of-the-art fast ME algorithms in quality and/or computational complexity.

The complete complexity scalable MPEG encoder smoothly reduces the quality as a function of limited resources. We have found two key parameters of scalability, namely the number of processed DCT coefficients and the number of evaluated MV candidates, resulting in a large design space for complexity scalability and resulting video quality. For evaluation of the complete system, we combined those two scalability parameters into one joint measurable parameter, namely the execution time of the encoder. It was found that the overall execution time of the scalable encoder can be gradually reduced to roughly 50% of its original execution time, resulting in a quality-level range from roughly 21.5 dB to 39.0 dB for different video sequences coded at various bit rates and using several GOP structures. Since the optimization of the scalability parameters was found to be a study of its own, only limited optimizations have been made and appropriate parameter settings were used. Fortunately, despite a lack of optimization, the obtained results already outperform a non-scalable encoder in execution time and picture quality (PSNR).

The obtained computational complexity scalability offers a large range of stationary and portable scalable MPEG coding applications. Scalability was mainly developed to scale down the computational complexity, starting from a defined reference processing effort (full DCT computation and state-of-the-art ME). For this reason, resource-aware applications can exploit the flexibility of the proposed scalable encoder for achieving e.g. low-power processing. For applications running in e.g. stationary high-end devices, the obtained flexibility of the scalable MPEG encoder embedded in a scalable system environment enables flexibility in scheduling real-time encoding tasks executed in parallel. With respect to research, the develop scalable MPEG encoder presented in this thesis is the first fully complexity scalable framework, which can form a basis for further research on complexity scalability and extensions to other video coding standards.

7.2 Memory-resource usage

In this thesis, the main focus was to obtain scalability for the computational complexity of the MPEG encoding process. A secondary effect of the presented techniques is the scalability of memory accesses. The scalable MPEG functions that process DCT coefficients ((I)DCT, (de-)quantization and VLC) reduce the number of memory accesses with each coefficient that does not belong to the set of computed coefficients that is defined by the scalable DCT. The scalable ME functions reduce the number of memory accesses with each discarded MV evaluation.

With the scalability of both the computational complexity and the memory access, the power consumption of a scalable encoding system is scaled accordingly, because the amount of computations and memory accesses are of primary importance for power consumption. This aspect was not further elaborated.

7.3 Future work on MPEG scalability

7.3.1 System optimization

The developed scalable MPEG encoder can be optimized on the basis of various criteria. One way of doing this is the introduction of cost functions. When investigating the DCT, we defined a cost function for certain DSP operations, in order to define and optimize the computational complexity. Similarly, cost

functions for memory accesses and power consumption can be defined and integrated into our analysis approach for obtaining a scalable DCT. Furthermore, these cost functions can be adapted to different hardware architectures.

Such a simple way of changing the optimization parameter, as used with the scalable DCT, has not been worked out for the scalable ME techniques. The reason for this is as follows. It has been debated at the end of the previous chapter that the actual scalable MPEG encoder represents a single point in a large pluriform multi-dimensional design space. Presently, it cannot be judged how close the proposed scalable encoder configurations are to the theoretically optimal system. In order to do so, it would be required to define a scientifically exact definition of complexity scalability, which is a challenging topic for further research. Similarly, it can be studied how to optimize memory accesses and power consumption of a scalable ME algorithm by e.g. ordering the evaluation of the vector candidates, such that for each vector evaluation, a maximum of video data in a potentially available cache memory is reused.

7.3.2 Bit-based processing

The work of this thesis concentrates on algorithmic optimization for a scalable MPEG encoder, which may be implemented in software only. During the research work, we have considered possible hardware implementations for achieving scalability. In Appendix C it is indicated that bit-based processing of video data seems promising for achieving complexity scalability. The advantage of bit-based processing is that intermediate results of good quality are fastly obtained. Furthermore, the processing can be better adapted to the requirements of an application. For example, it is known that displays of mobile devices have low contrast, especially in outdoor light-conditions. Therefore, mobile video conferencing should be scalable for low-power processing by reducing the video color-depth. Unfortunately, new hardware should be designed supporting bit-based processing, because this is not available in commonly used programmable hardware devices.

7.3.3 Modifying the input signal

In this thesis, we consider an encoder that does not modify the spatial and/or temporal resolution of the input video signal. The consequence of this would be that an interpolation to the original resolution after MPEG decoding would be required. Such a concept is certainly feasible for exploring complexity

scalability, but the computational complexity of the additional pre- and post-processing steps cannot be neglected. Although our techniques can be inserted into an MPEG-4 encoder, the development of a full complexity scalable layered MPEG-4 encoder was not finished.

7.3.4 Differentiating scalable DCT for intra- and intercoding

In the previous section, the experiments with the scalable DCT integrated in an MPEG encoder framework revealed temporal fluctuations in the frequency content during intercoding of the frame differences, which lead to a quality build-up over time. The quality augmentation is periodical and reset by the occurence of an I-frame, leading to a comb-like structure within the PSNR curves. This phenomenon leads to the idea of using various scalable DCTs employing different coefficients-selection functions. The complete sequence of scalable DCTs should compute each DCT coefficient at least once, and apply these scalable DCTs to the picture blocks in a cyclical sequence (one per intercoded frame). The result is that the computational complexity of the DCT computation is spread over time, thereby reducing the average complexity of the DCT computation (per block), at the expense of a delay in the build-up of visual quality. The fluctuating visually annoying effect can be solved by computing more coefficients for the I-frames, which makes it a promising approach for future research on complexity scalability.

7.4 Applicability to other video applications

7.4.1 MPEG-incompatible DCT coefficient coding

The scalable DCT presented in this thesis introduces a DCT-coefficient selection-function, which results from a special computation order of coefficients. Even when using a scalable DCT, the encoder is fully compatible with a non-scalable encoder. However, the VLC coding efficiency is reduced, because the scalable DCT may introduce zero coefficients at unusual locations in the coefficient matrix. Consequently, the scalable DCT changes the statistics of the generated runlength-amplitude codewords. A higher coding efficiency is achievable when using a scalable DCT, by measuring the modified statistics and adapting the scanning pattern and VLC coding tables to it. However, this solution is not compatible with the MPEG coding standard, but could be of interest for dedicated applications.

7.4.2 Segmentation (MPEG-4)

MPEG-4 is the successor of MPEG-2 and aims at the coding of individual objects inside a video scene. The main area of application was seen in video communication over the Internet. In computer-based systems, the use and display of both natural and synthetic objects is attractive and the MPEG-4 standard offers a flexible framework for object-based coding. Besides this, the set of coding techniques was enhanced in order to achieve higher compression factors than with MPEG-2.

Recently, a high number of publications have shown up proposing algorithms to retrieve objects from video, which is called segmentation. It has become clear that segmentation processing is a complex task, and the computational requirements for accurate segmentation clearly exceeds the computational complexity of motion estimation. For this reason, scalable segmentation is a key topic for scalable MPEG-4 encoding. One approach to apply scalability to the segmentation process is to e.g. start with a segmentation based on 16×16 pixel blocks and successively refine the segmentation up to pixel accuracy. In such an approach, the scalability parameter is obviously the processed block size, and the computational complexity will be a function of this scalability parameter. In the segmentation process, reusable data from the encoding process from previous frames are for example motion vectors (coherent vector field per object) and detected edges (objects are bordered by edges). The system optimization of a scalable MPEG-4 encoder will be even more sophisticated than for a scalable MPEG-2 encoder, because the segmentation in MPEG-4 has impact on the object-based coding modules, the video quality and compression/coding efficiency.

7.4.3 Multi-temporal ME (H.264)

In the H.264 video coding standard, the motion estimation and compensation is allowed to have multiple (up to five) reference frames from which picture blocks are taken for predicting the actual frame. Furthermore, the block sizes used for motion estimation are variable. Since the number of reference frames and the different block sizes are relatively high, the computational requirements for accurate motion estimation in H.264 coding are substantially higher than for MPEG-2 having at maximum two reference frames. Scalability is applicable to the ME in H.264 coding in the same way as it is proposed for MPEG-2 in this thesis. Optimizing a scalable H.264 encoder will be more sophisticated than for a scalable MPEG-2 encoder due to the additional freedom

in (among others) the ME process in H.264, as indicated above. An opportunity for significantly reduced computational complexity is seen when applying the SMART ME technique presented in this thesis to the ME in H.264. Because SMART performs frame-by-frame ME, it can be used for multiple frame prediction. For example, the vector fields can be concatenated for each individually processed block in order to find the appropriate reference frame for this block, without the need of a full vector search for each reference-frame alternative. After the appropriate reference frame for a block was found, it can be chosen to split the block to obtain a better prediction quality, as defined in the H.264 standard.

References

[1] E.G.T. Jaspers, P.H.N. de With and J.G.W.M. Janssen, "A Flexible Heterogeneous Video Processor System for Television Applications," *IEEE Transactions on Consumer Electronics*, vol. 45, no. 1, pp. 1–11, Feb. 1999.

[2] P.H.N. de With, E.G.T. Jaspers, J.L. van Meerbergen, A.H. Timmer and M. Strik, "A Video Display Processing Platform for Future TV Concepts," *IEEE Transactions on Consumer Electronics*, vol. 45, no. 4, Nov. 1999.

[3] E.G.T. Jaspers and P.H.N. de With, "Chip-Set for Video Display of Multimedia Information," *IEEE Transactions on Consumer Electronics*, vol. 45, no. 3, Aug. 1999.

[4] S. Mietens, P.H.N. de With and C. Hentschel, "Implementation of a Dynamical Multi-Window TV System," *Proceedings of* 22^{nd} *International Symposium on Information Theory in the Benelux*, pp. 139–146, May 2001.

[5] C. Hentschel, M. Gabrani, K. van Zon, R.J. Bril and L. Steffens, "Scalable Video Algorithms and Quality-of-Service Resource Management for Consumer Terminals," *IEEE International Conference on Consumer Electronics (ICCE), Digest of Technical Papers*, pp. 338–339, June 2001.

[6] R.J. Bril, C. Hentschel, E.F. M. Steffens, M. Gabrani, G. van Loo, J.H.A. Gelissen, "Multimedia QoS in Consumer Terminals," *Proceedings of IEEE International Workshop on Signal Processing Systems (SIPS)*, pp. 332–343, Sept. 2001.

[7] C. Hentschel, R.J. Bril and Yingwei Chen, "Invited: Video Quality-of-Service for Multimedia Consumer Terminals - An Open and Flexible System Approach," *Proceedings of IEEE International Conference on Communication Theory and Systems and West Sino Expositions*, vol. 1, pp. 659–663, June 2002.

[8] R.S.V. Prasad and K. Ramkishor, "Efficient Implementation of MPEG-4 Video Encoder on RISC Core," *IEEE International Conference on Consumer Electronics (ICCE), Digest of Technical Papers*, pp. 278–279, 2002.

[9] Yingwei Chen, Zhun Zhong, Tse-Hua Lan, S. Peng, K. van Zon, "Regulated Complexity Scalable MPEG-2 Video Decoding for Media Processors," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 8, pp. 678–687, Aug. 2002.

[10] D. Farin, M. Käsemann, P.H.N. de With and W. Effelsberg, "Rate-Distortion Optimal Adaptive Quantization and Coefficient Thresholding for MPEG Coding," *Proceedings of* 23^{rd} *Symposium on Information Theory in the Benelux*, pp. 131–138, May 2002.

[11] K. Ramchandran and M. Vetterli, "Rate-Distortion Optimal Fast Thresholding with Complete JPEG/MPEG Decoder Compatibility," *IEEE Transactions on Image Processing*, vol. 3, no. 5, pp. 700–704, Sept. 1994.

[12] R.J. Gove, "The Multimedia Video Processor (MVP): An Architecture for Advanced DSP Applications," in *Proceedings of 5th International Conference on Signal Processing, Application and Technology*, Oct. 1994, vol. 1, pp. 854–859.

[13] V.M.Jr. Bove and J.A. Watlington, "Chepos: A Reconfigurable Dataflow System for Video Processing," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 5, pp. 140–149, Apr. 1995.

[14] M. Strik and E.G.T. Jaspers, "Integrating Dynamic Multi-Window TV Feature in Software Stack," Philips Research Labs. Eindhoven, The Netherlands, personal communication, 2000.

[15] R.J. Bril, M. Gabrani, C. Hentschel, G. van Loo, E.F.M. Steffens, "QoS for Consumer Terminals and its Support for Product Families," *International Conference on Media Futures (ICMF)*, pp. 299–302, May 2001.

[16] M. Ghanbari, "Two-layer coding of video signals for VBR networks," *IEEE Selected Area Communications*, vol. 7, no. 5, pp. 771–781, June 1989.

[17] ISO/IEC, *"Coding of Moving Pictures and Audio: MPEG-4 Overview"*, N4668 edition, Mar. 2002.

[18] M. van der Schaar-Mitrea, *System and network constrained scalable video compression*, Ph.D. thesis, University of Technology Eindhoven, The Netherlands, Dec. 2001.

[19] S. Mietens and P.H.N. de With, "On Scalable DCT Processing Algorithms for MPEG Encoding," Tech. Rep. 2000-VDP-003, Dept. Circuitry and Simulation, University of Mannheim, Germany, D-68131 Mannheim, Feb. 2000.

[20] S. Mietens, P.H.N. de With and C. Hentschel, "Coding: New DCT computation algorithm for video quality scaling," patent application WO02056250/US2002173952, 2002.

[21] S. Mietens, P.H.N. de With and C. Hentschel, "New Scalable DCT Computation for Resource-Constrained Systems," *Proceedings of IEEE International Workshop on Signal Processing Systems (SIPS)*, pp. 285–296, 2001.

[22] S. Mietens, P.H.N. de With and C. Hentschel, "New DCT Computation Algorithm for Video Quality Scaling," *Proceedings of IEEE International Conference on Image Processing (ICIP)*, vol. 3, pp. 462–465, Oct. 2001.

[23] S. Mietens, P.H.N. de With and C. Hentschel, "New DCT Computation Technique based on Scalable Resources," *Kluwer Academic Publishers Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology, Special Issue on IEEE Workshop on Signal Processing Systems (SIPS 2001)*, vol. 34, no. 3, pp. 189–201, July 2003.

[24] S. Mietens and P.H.N. de With, "On Scalable Motion Estimation for MPEG Encoding," Tech. Rep. 2000-VDP-004, Dept. Circuitry and Simulation, University of Mannheim, Germany, D-68131 Mannheim, Feb. 2000.

[25] G. Hekstra, S. Mietens and P.H.N. de With, "Motion estimation with 3D-RS candidates from multiple temporal vector fields," patent application PHNL020034.

[26] S. Mietens, G. Hekstra, P.H.N. de With and C. Hentschel, "Flexible Frame-Reordering and Multi-Temporal Motion Estimation for Scalable MPEG Encoding in Mobile Consumer Equipment," *IEEE International Conference on Consumumer Electronics (ICCE), Digest of Technical Papers*, pp. 342–343, June 2002.

[27] S. Mietens, G. Hekstra, P.H.N. de With and C. Hentschel, "New Flexible Motion Estimation Technique for Scalable MPEG Encoding using Display Frame Order and Multi-Temporal References," *Proceedings of IEEE International Conference on Image Processing (ICIP)*, pp. 701–704, Sept. 2002.

[28] S. Mietens, P.H.N. de With and C. Hentschel, "Frame-Reordered Multi-Temporal Motion Estimation for Scalable MPEG," *Proceedings of* 23^{rd} *International Symposium on Information Theory in the Benelux*, 2002.

[29] S. Mietens, P.H.N. de With and C. Hentschel, "New Scalable Three-Stage Motion Estimation Technique for Mobile MPEG Encoding," *Proceedings of IEEE International Conference on Multimedia and Expo (ICME)*, pp. 685–688, Aug. 2002.

[30] S. Mietens, P.H.N. de With and C. Hentschel, "Computational Complexity Scalable Motion Estimation for Mobile MPEG Encoding," *IEEE Transactions on Consumer Electronics*, Feb. 2004.

[31] S. Mietens, P.H.N. de With and C. Hentschel, "A SW-based Complexity Scalable MPEG Encoder for Mobile Consumer Equipement," *Proceedings of* 24^{th} *International Symposium on Information Theory in the Benelux*, 2003.

[32] S. Mietens, P.H.N. de With and C. Hentschel, "New Complexity Scalable MPEG Encoding Techniques for Mobile Applications," *Eurasip Journal on Applied Signal Processing, Special Issue on Multimedia over Wireless Networks*, vol. 2, 2004.

[33] S. Mietens, P.H.N. de With and C. Hentschel, "Invited: Resource-aware Complexity Scalability for Mobile MPEG Encoding," *Proceedings of SPIE International Conference on Visual Communications and Image Processing (VCIP)*, 2004.

[34] PeiZong Lee and Fang-Yu Huang, "Restructured Recursive DCT and DST Algorithms," *IEEE Transactions Signal Processing*, vol. 42, no. 7, pp. 1600–1609, July 1994.

[35] Nam Ik Cho and San Uk Lee, "Fast Algorithm and Implementation of 2-D Discrete Cosine Transform," *IEEE Transactions Circuits and Systems*, vol. 38, no. 3, pp. 297–305, Mar. 1991.

[36] Y. Arai, T. Agui and M. Nakajima, "A Fast DCT-SQ Scheme for Images," *Transactions of the IEICE*, vol. 71, no. 11, pp. 1095, Nov. 1988.

[37] ISO/IEC, *"Digital Compression and Coding of Continuous-Tone Still Images"*, 10918-1 edition, 1997.

[38] William B. Penneberger and Joan L. Mitchell, *JPEG: Still Image Data Compression Standard*, Van Nostrand Reinhold, 1993.

[39] ISO/IEC, *"ITU-T Rec. H.264, Advanced Video Coding, Final Committee Draft, Document JVT-F100"*, 11496-10 edition, Dec. 2002.

[40] R.L. de Queiroz, "Reduced DCT Approximations for Low Bit Rate Coding," *Proceedings of IEEE International Conference of Image Processing (ICIP)*, vol. 1, pp. 233–236, Sept. 2002.

[41] V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards*, chapter 3.5.2, Kluwer Academic Publishers, 1995.

[42] V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards*, chapter 3.5.3, Kluwer Academic Publishers, 1995.

[43] E. Linzer and E. Feig, "New Scaled DCT Algorithms for Fused Multiply/Add Architectures," *Proceedings of International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 3, pp. 2201–2204, Apr. 1991.

[44] I-Ming Pao and ing-Ting Sun, "Modeling DCT Coefficients for Fast Video Encoding," *IEEE Transactions Circuits and Systems for Video Technology*, vol. 9, no. 4, pp. 608–616, June 1999.

[45] Hong Ren Wu and Zhihong Man, "Comments on "Fast Algorithms and Implementation of 2-D Discrete Cosine Transform"," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 2, pp. 128–129, Apr. 1998.

[46] C. Loeffer, A. Ligtenberg and G.S. Moschytz, "Practical fast 1-D DCT Algorithms with 11 Multiplications," *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 2, pp. 988–991, May 1989.

[47] Sung-Hwan Jung, S.K. Mitra and D. Mukherjee, "Subband DCT: Definition, Analysis and Applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, pp. 273–286, June 1996.

[48] K. Lengwehasatit and A. Ortega, "DCT Computation based on Variable Complexity Fast Approximations," *Proceedings of IEEE International Conference of Image Processing (ICIP)*, vol. 3, pp. 95–99, Oct. 1998.

[49] P. Duhamel and H. H'Mida, "New 2^n DCT Algorithms suitable for VLSI Implementation," *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 12, pp. 1805–1808, Apr. 1987.

[50] E. Feig and S. Winograd, "On the Multiplicative Complexity of Discrete Cosine Transforms," *IEEE Transactions on Information Theory*, vol. 38, pp. 1387–1391, July 1992.

[51] N. Merhav and B. Vasudev, "A Multiplication-Free Approximate Algorithm for the Inverse Discrete Cosine Transform," *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, vol. 2, pp. 759–763, Oct. 1999.

[52] A. Hossen and U. Heute, "Fast Approximation DCT: Basic-Idea, Error Analysis and Applications," *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 3, pp. 2005–2008, Apr. 1997.

[53] R.L. de Queiroz, "DCT Approximation for Low Bit Rate Coding using a Conditional Transform," *Proceedings of IEEE International Conference on Image Processing (ICIP)*, vol. 1, pp. 237–240, 2002.

[54] K.R. Rao and P. Yip, *Discrete Cosine Transform: Algorithms, Advantages, Applications*, Academic Press, 1990.

[55] S. Peng, "Complexity Scalable Video Decoding via IDCT Data Pruning," *IEEE International Conference on Consumer Electronics (ICCE), Digest of Technical Papers*, pp. 74–75, June 2001.

[56] ISO/IEC, *"Information Technology - Generic Coding of Moving Pictures and Associated Audio Information: Video"*, 13818-2 edition, 2000.

[57] Mei-Juan Chen, Liang-Gee Chen, Tzi-Dar Chiueh and Yung-Pin Lee, "A new Block-Matching Criterion for Motion Estimation and its Implementation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 5, no. 3, pp. 231–236, June 1995.

[58] G. de Haan and H. Huijgen, "Motion Estimation for TV Picture Enhancement," in *Signal Processing of HDTV*, H. Yasuda and L. Chiariglione, Eds., vol. III, pp. 241–248. Elseviers Science Publishers B.V., 1992.

[59] R. Braspenning and G. de Haan, "Efficient Motion Estimation with Content-Adaptive Resolution," *Proceedings of International Symposium on Consumer Electronics (ISCE)*, pp. E29–E34, Sept. 2002.

[60] K. Lengwehasatit, A. Ortega, A. Basso and A. Reibman, "A Novel Computationally Scalable Algorithm for Motion Estimation," *Proceedings of SPIE International Conference on Visual Communications and Image Processing (VCIP)*, pp. 68–79, Jan. 1998.

[61] G. de Haan, P.W.A.C. Biezen, H. Huijgen and O.A. Ojo, "True-Motion Estimation with 3-D Recursive Search Block Matching," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 3, no. 5, pp. 368–379, Oct. 1993.

[62] G. de Haan and P.W.A.C. Biezen, "Sub-pixel motion estimation with 3-D recursive search block-matching," *Signal Processing: Image Communication*, vol. 6, pp. 229–239, 1994.

[63] Mei-Juan Chen, Liang-Gee Chen and Tzi-Dar Chiueh, "One-Dimensional Full Search Motion Estimation Algorithm for Video Coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 4, no. 5, pp. 504–509, Oct. 1994.

[64] Lumg-Kuo Liu and E. Feig, "A block-based Gradient Descent Search Algorithm for Block Motion Estimation in Video Coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 4, pp. 419–423, Aug. 1996.

[65] Jo Yew Tham, S. Ranganath, M. Ranganath and A.A. Kassim, "A Novel Unrestricted Center-Biased Diamond Search Algorithm for Block Motion Estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 4, pp. 369–377, Aug. 1998.

[66] T. Koga, K. Iinuma, A. Hirano, Y. Iijima and T. Ishiguro, "Motion-Compensated Interframe Coding for Video Conferencing," *Proceedings of the NTC*, vol. 81, pp. C9.6.1–9.6.5, Nov. 1981.

[67] G. de Haan and H. Huijgen, "New Algorithm for Motion Estimation," *Signal Processing of HDTV*, vol. 73, no. 4, pp. 523–548, Apr. 1989.

[68] P.H.N. de With, "A Simple Recursive Motion Estimation Technique for Compression of HDTV Signals," *Proceedings of IEE International Conference on Image Processing and its Applications (IPA)*, pp. 417–420, 1992.

[69] Reoxiang Li, Bing Zeng and M.L. Liou, "A new Three-Step Search Algorithm for Block Motion Estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 4, no. 4, pp. 438–442, Aug. 1994.

[70] F.S. Rovati, D. Pau, E. Piccinelli, L. Pezzoni and J.-M. Bard, "An Innovative, High Quality and Search Window Independent Motion Estimation Algorithm and Architecture for MPEG-2 Encoding," *IEEE Transactions on Consumer Electronics*, vol. 46, no. 3, pp. 697–705, Aug. 2000.

[71] T. Kummerow and P. Mohr, "Method of Determining Motion Vectors for the Transmission of Digital Picture Information," EP 0 496 051, Nov. 1991, European Patent Application.

[72] R. Braspenning, G. de Haan and C. Hentschel, "Complexity Scalable Motion Estimation," *Proceedings of SPIE International Conference on Visual Communications and Image Processing (VCIP)*, vol. 4671(1/2), pp. 442–453, 2002.

[73] D. Farin, N. Mache and P.H.N. de With, "A Software-Based High-Quality MPEG-2 Encoder Employing Scene Change Detection and Adaptive Quantization," *IEEE Transactions on Consumer Electronics*, vol. 48, no. 4, pp. 887–897, Nov. 2002.

[74] P.H. Westerink, *Subband Coding of Images*, Ph.D. thesis, TU Delft, The Netherlands, Oct. 1989.

[75] M.-T. Sun, T.-C. Chen and A.M. Gottlieb, "VLSI Implementation of a 16x16 Discrete Cosine Transform," *IEEE Transactions on Circuits and Systems*, vol. 36, no. 4, pp. 610–617, Apr. 1989.

[76] O. Hauck, S.A. Huss, "Asynchronous Wave Pipelines for High Throughput Datapaths," *Proceedings of IEEE International Conference on Electronics, Circuits and Systems*, pp. 1283–1286, Sept. 1998.

[77] S. Mietens, "VLSI-Entwurf eines asynchronen SRT-Ringdividierers," M.S. thesis, Darmstadt University of Technology, Germany, Aug. 1998.

[78] O. Hauck, S. Mietens, S.A. Huss, "Giga-Hertz SRT-Division mit asynchronen Wave-Pipelines," Proceedings of 9. Workshop "Entwurf Integrierter Schaltungen", Darmstadt/Germany, Sept. 1999.

[79] B. Natarajan, V. Bhaskaran and K. Konstantinides, "Low-Complexity Block-Based Motion Estimation via One-Bit Transforms," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, no. 4, pp. 702–706, Aug. 1997.

APPENDIX A

Derivation of computation order

In this appendix, the computation technique from Sections 3.2 and 3.3 has been used to find an scalability-optimized computation order for a *dynamically-scalable* version of the 2-D DCT algorithm by Cho and Lee (ChoLee, see Section 2.3.2), when inserting the 1-D DCT algorithm by Arai, Agui and Nakajima (AAN, see Section 2.3.3). Both algorithms were adopted for the following derivation, because their combination provides a highly efficient DCT computation (see Section 2.4).

In Figure 3.3(a), the scalability-optimized computation order is shown that is found with the algorithm analysis that was presented in Section 3.2. The corresponding Figure 3.3(b) is a modified computation order with an additional priority for the upper-left corner of the coefficient matrix. This priority weighting was explained in the previous section. The remainder of this section presents a detailed derivation of the coefficient computation order, including this weighting.

The nominal computational complexity of the combination of ChoLee and AAN for the 8×8-DCT is 104 multiplications, 49 shifts and 466 additions. The shift operations are located at the end of the butterfly diagram, and therefore can be integrated into the constant normalization factors that are used for

completing the DCT computation (see the factors left to the double summation in Equation (2.1)).

The first step of the analysis is the counting of the number of operations that are performed to compute a single coefficient. The following matrices show the number of multiplications ($\underline{\underline{M}}_m$) and additions ($\underline{\underline{M}}_a$) that are needed when using the above-mentioned combined algorithm.

$$\underline{\underline{M}}_m = \begin{pmatrix} 4 & 16 & 8 & 16 & 4 & 16 & 8 & 16 \\ 16 & 11 & 24 & 11 & 20 & 11 & 24 & 11 \\ 8 & 24 & 6 & 24 & 12 & 24 & 6 & 24 \\ 16 & 11 & 24 & 11 & 20 & 11 & 24 & 11 \\ 4 & 20 & 12 & 20 & 4 & 20 & 12 & 20 \\ 16 & 11 & 24 & 11 & 20 & 11 & 24 & 11 \\ 8 & 24 & 6 & 24 & 12 & 24 & 6 & 24 \\ 16 & 11 & 24 & 11 & 20 & 11 & 24 & 11 \end{pmatrix}.$$

$$\underline{\underline{M}}_a = \begin{pmatrix} 63 & 79 & 67 & 79 & 63 & 79 & 67 & 79 \\ 79 & 79 & 95 & 79 & 87 & 79 & 95 & 79 \\ 67 & 95 & 67 & 95 & 75 & 95 & 67 & 95 \\ 79 & 79 & 95 & 79 & 87 & 79 & 95 & 79 \\ 63 & 87 & 75 & 87 & 63 & 87 & 75 & 87 \\ 79 & 79 & 95 & 79 & 87 & 79 & 95 & 79 \\ 67 & 95 & 67 & 95 & 75 & 95 & 67 & 95 \\ 79 & 79 & 95 & 79 & 87 & 79 & 95 & 79 \end{pmatrix}.$$

The number of operations is determined by considering an addition as one operation and a multiplication as three operations.

Let $\underline{\underline{M}}_o$ be the matrix containing the number of the resulting operations. The derivation of matrix $\underline{\underline{M}}_o$, given the aforementioned relation between additions and multiplications is trivial.

$$\underline{\underline{M}}_o = \underline{\underline{M}}_a + 3 * \underline{\underline{M}}_m = \begin{pmatrix} 75 & 127 & 91 & 127 & 75 & 127 & 91 & 127 \\ 127 & 112 & 167 & 112 & 147 & 112 & 167 & 112 \\ 91 & 167 & 85 & 167 & 111 & 167 & 85 & 167 \\ 127 & 112 & 167 & 112 & 147 & 112 & 167 & 112 \\ 75 & 147 & 111 & 147 & 75 & 147 & 111 & 147 \\ 127 & 112 & 167 & 112 & 147 & 112 & 167 & 112 \\ 91 & 167 & 85 & 167 & 111 & 167 & 85 & 167 \\ 127 & 112 & 167 & 112 & 147 & 112 & 167 & 112 \end{pmatrix}$$

In this example, the following priority weighting $p(i,j)$ is used, which is represented with the matrix

$$p(i,j) = 2(i+j) + |i-j| + 1 = \begin{pmatrix} 1 & 4 & 7 & 10 & 13 & 16 & 19 & 22 \\ 4 & 5 & 8 & 11 & 14 & 17 & 20 & 23 \\ 7 & 8 & 9 & 12 & 15 & 18 & 21 & 24 \\ 10 & 11 & 12 & 13 & 16 & 19 & 22 & 25 \\ 13 & 14 & 15 & 16 & 17 & 20 & 23 & 26 \\ 16 & 17 & 18 & 19 & 20 & 21 & 24 & 27 \\ 19 & 20 & 21 & 22 & 23 & 24 & 25 & 28 \\ 22 & 23 & 24 & 25 & 26 & 27 & 28 & 29 \end{pmatrix}.$$

The decision about the first coefficient to be computed is made by finding the lowest entry (marked with a circle) of matrix $\underline{\underline{M}}_d$, which results from multiplying $\underline{\underline{M}}_o$ with the weighting function $p(i,j)$. The decision matrix $\underline{\underline{M}}_d$ contains all possible results of the cost function $rop(l_0)$ as defined in Equation (3.3).

$$\underline{\underline{M}}_d = \underline{\underline{M}}_o * p(i,j)$$

$$= \begin{pmatrix} \textcircled{75} & 508 & 637 & 1270 & 975 & 2032 & 1729 & 2794 \\ 508 & 560 & 1336 & 1232 & 2058 & 1904 & 3340 & 2576 \\ 637 & 1336 & 765 & 2004 & 1665 & 3006 & 1785 & 4008 \\ 1270 & 1232 & 2004 & 1456 & 2352 & 2128 & 3674 & 2800 \\ 975 & 2058 & 1665 & 2352 & 1275 & 2940 & 2553 & 3822 \\ 2032 & 1904 & 3006 & 2128 & 2940 & 2352 & 4008 & 3024 \\ 1729 & 3340 & 1785 & 3674 & 2553 & 4008 & 2125 & 4676 \\ 2794 & 2576 & 4008 & 2800 & 3822 & 3024 & 4676 & 3248 \end{pmatrix}.$$

The first sorted coefficient l_0 in the list L of coefficients that defines the computation order is therefore the DC coefficient. In the following, the superscript (i) indicates the actual iterations count of the analysis.

For the second iteration with $i = 1$, the matrices $\underline{\underline{M}}_m^{(1)}$ and $\underline{\underline{M}}_a^{(1)}$ are updated by considering that the intermediate results from the computation of l_0 are available and can be reused during the computation of the next coefficient. The updated matrices $\underline{\underline{M}}_m^{(1)}$ and $\underline{\underline{M}}_a^{(1)}$ are given by

$$\underline{\underline{M}}_m^{(1)} = \begin{pmatrix} 0 & 16 & 8 & 16 & 4 & 16 & 8 & 16 \\ 16 & 11 & 24 & 11 & 20 & 11 & 24 & 11 \\ 8 & 24 & 4 & 24 & 12 & 24 & 4 & 24 \\ 16 & 11 & 24 & 11 & 20 & 11 & 24 & 11 \\ 4 & 20 & 12 & 20 & 0 & 20 & 12 & 20 \\ 16 & 11 & 24 & 11 & 20 & 11 & 24 & 11 \\ 8 & 24 & 4 & 24 & 12 & 24 & 4 & 24 \\ 16 & 11 & 24 & 11 & 20 & 11 & 24 & 11 \end{pmatrix},$$

$$\underline{\underline{M}}_a^{(1)} = \begin{pmatrix} 0 & 79 & 19 & 79 & 7 & 79 & 19 & 79 \\ 47 & 79 & 63 & 79 & 55 & 79 & 63 & 79 \\ 19 & 95 & 9 & 95 & 27 & 95 & 9 & 95 \\ 47 & 79 & 63 & 79 & 55 & 79 & 63 & 79 \\ 7 & 87 & 27 & 87 & 1 & 87 & 27 & 87 \\ 47 & 79 & 63 & 79 & 55 & 79 & 63 & 79 \\ 19 & 95 & 9 & 95 & 27 & 95 & 9 & 95 \\ 47 & 79 & 63 & 79 & 55 & 79 & 63 & 79 \end{pmatrix}.$$

Repeating the matrix computations of iteration 1 leads to the decision matrix $\underline{\underline{M}}_d^{(1)}$ as given below, where already computed coefficients are marked as "ok".

$$\underline{\underline{M}}_d^{(1)} = \underline{\underline{M}}_o^{(1)} * p(i,j)$$

$$= \begin{pmatrix} ok & 508 & 301 & 1270 & 247 & 2032 & 817 & 2794 \\ 380 & 560 & 1080 & 1232 & 1610 & 1904 & 2700 & 2576 \\ 301 & 1336 & 189 & 2004 & 945 & 3006 & 441 & 4008 \\ 950 & 1232 & 1620 & 1456 & 1840 & 2128 & 2970 & 2800 \\ 247 & 2058 & 945 & 2352 & \textcircled{17} & 2940 & 1449 & 3822 \\ 1520 & 1904 & 2430 & 2128 & 2300 & 2352 & 3240 & 3024 \\ 817 & 3340 & 441 & 3674 & 1449 & 4008 & 525 & 4676 \\ 2090 & 2576 & 3240 & 2800 & 2990 & 3024 & 3780 & 3248 \end{pmatrix}$$

From $\underline{\underline{M}}_d^{(1)}$ it results that l_1 is the coefficient at position $(4,4)$. From $\underline{\underline{M}}_m^{(1)}$ and $\underline{\underline{M}}_a^{(1)}$ it can be seen, that in order to complete the computation of coefficient l_1, a single addition needs to be performed.

The third step of the analysis shows the impact of the priority weighting on the determination of the computation order. Let us focus on the coefficients with the least number of operations found at positions $(0,4)$, $(4,0)$ and at position $(2,2)$. The first two need 4 multiplications and 7 additions to be completely computed, while the last one needs 4 multiplications and 9 additions. Obviously, either the coefficient at position $(0,4)$ or $(4,0)$ is chosen as next coefficient when no priority weighting is applied, because they require 2 additions less than the coefficient at position $(2,2)$. Instead, matrix $\underline{\underline{M}}_d^{(2)}$ clearly shows that with priority weighting, the coefficient at position $(2,2)$ is preferred, although requiring a little more computation than the other two at this iteration.

$$M_d^{(2)} = M_o^{(2)} * p(i,j)$$

$$= \begin{pmatrix} ok & 508 & 301 & 1270 & 247 & 2032 & 817 & 2794 \\ 380 & 560 & 1080 & 1232 & 1610 & 1904 & 2700 & 2576 \\ 301 & 1336 & 189 & 2004 & 945 & 3006 & 441 & 4008 \\ 950 & 1232 & 1620 & 1456 & 1840 & 2128 & 2970 & 2800 \\ 247 & 2058 & 945 & 2352 & ok & 2940 & 1449 & 3822 \\ 1520 & 1904 & 2430 & 2128 & 2300 & 2352 & 3240 & 3024 \\ 817 & 3340 & 441 & 3674 & 1449 & 4008 & 525 & 4676 \\ 2090 & 2576 & 3240 & 2800 & 2990 & 3024 & 3780 & 3248 \end{pmatrix}.$$

In this way, the analysis is continued until all coefficients are sorted into list L.

APPENDIX B

Computational complexity of ME algorithms

In this appendix, the ME algorithms are compared by their computational complexity, which depends on the number of block comparisons, the block-matching criterion, the block size, the optional subsequent vector refinement, and the video material (for example size, motion and quality).

We have divided the algorithms into their basic steps and evaluated the worst-case complexity of each of those steps. Memory traffic has been omitted for simplicity, which means that we assume a large data cache is closely located to the computation unit. The overall computation cost C_{ME} for the motion estimation of a complete frame is defined by

$$C_{ME} = n_b * (n_{bc} * c_{bc} + c_r), \tag{B.1}$$

where n_b is the number of macroblocks that are processed per frame, n_{bc} is the number of block comparisons that are made (on the average) per macroblock, c_{bc} is the cost for each block comparison and c_r is the additional cost for retrieving blocks from the half-pixel grid for vector refinement.

B.1 Number of block comparisons

The main parameter in Equation (B.1) for the computational complexity of ME algorithms is the number of block compares n_{bc}. A favorable algorithm should find a suitable MV with a low number of block comparisons. The evaluated ME algorithms have different properties as indicated below.

- The computational complexity of the 2DFS and 1DFS algorithms mainly depends on the size of the search area.

- The TSS, NTSS and the simple RME algorithms evaluate a predefined number of MVs. The simple RME contains a vector-refinement step that will be discussed separately, because every algorithm can make use of this additional step.

- The BBGDS terminates if a local minimum of the error measure is found. The number of block compares depends on the number of algorithm steps that are performed to find this minimum.

Table B.1 gives an overview of the computational complexity for the different algorithms. For convenience, the search area (SA) is expected to be square. It can be seen from the table that the simple RME and the BBGDS require the lowest number of block comparisons, where the BBGDS exceeds the number of block comparisons from the simple RME with the second step. The 1DFS is rather costly, considering that commonly used search areas have $n = 16$ or $n = 32$.

Algorithm	Remark	n_{bc}	
2DFS	$n \times n$ SA	$n * n$	$= n^2$
1DFS	$n \times n$ SA	$n + (n-1) + \frac{n}{2} + (\frac{n}{2} - 1)$	$= 3n - 2$
TSS	fixed algorithm	$9 + 8 + 8$	$= 25$
NTSS	fixed algorithm	$8 + 9 + 8 + 8$	$= 33$
RME	fixed algorithm	$6 + 12$	$= 18$
BBGDS	for i steps		$= 9 + i * 5$

Table B.1: *Number of block comparisons for some ME algorithms.*

B.2 Computational complexity of a single block comparison

The computational complexity for comparing blocks with the block-matching criteria presented in Section 4.2 depends on the size of the compared blocks. Table B.2 gives an overview of the computational complexity of one block comparison, given a block size of $p \times p$ pixels. Normalization factors of the block-matching criteria are omitted (for equal block sizes).

Criteria	Operation	No. of operations	
MSE	mults	$p * p$	$= p^2$
	adds/subs	$p * p + p * p - 1$	$= 2 * p^2 - 1$
SAD	abs. value	$p * p$	$= p^2$
	adds/subs	$p * p + p * p - 1$	$= 2 * p^2 - 1$
MiniMax	abs. value	$p * p$	$= p^2$
	compares	$b * b - 1$	$= p^2 - 1$
	adds/subs	$p * p$	$= p^2$

Table B.2: *Computational complexity for different block-matching criteria.*

B.3 Vector Refinement

The refinement of a motion vector as described in Section 4.4 involves eight additional block comparisons on a half-pixel grid. This is shown in Figure B.1, where positions P refer to pixels. The pixels P_{hp} refer to the half-pixel grid. The pixels P_{hp2} indicated by crosses are interpolated from the two

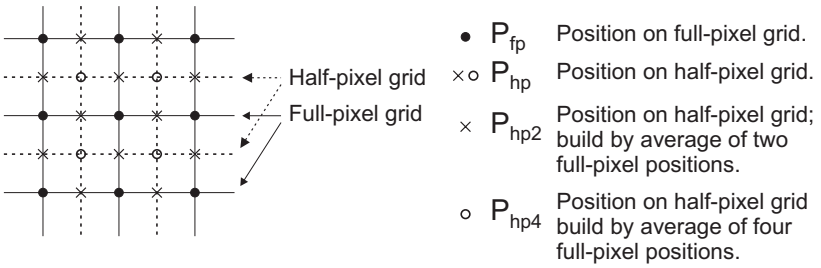

Figure B.1: *Vector refinement based on a half-pixel grid.*

surrounding full-pixel values. The middle positions P_{hp4} can be computed in two different ways. Either they are interpolated from the four surrounding full-pixel values, or from the two surrounding horizontal or vertical crosses. In the sequel, we will use the first option and average the four surrounding full-pixel grid values to obtain the desired pixel valued on the P_{hp4} positions.

The computational complexity of a single vector refinement depends on the block size. Table B.3 gives an overview of the computational complexity for $p \times p$ sized blocks.

Description	Amount	
additional block compares	8	
retrieving additional blocks (c_r)	$16 * p^2$ adds	$2 * p^2$ shifts
for blocks with pixels on P_{hp2}	$4 * (p * p * 1)$ adds	$p * p$ shifts
for block with pixels on P_{hp4}	$4 * (p * p * 3)$ adds	$p * p$ shifts

Table B.3: *Computational complexity of vector refinement per block.*

B.4 Example configurations for motion estimation

ME algorithms can be compared by combining the aforementioned options (block-matching criteria and vector refinement) into one particular configuration. For comparing different configurations, the numbers of operations are computed for the worst case based on a frame size of 720×576 pixels, the commonly used block size of 16×16 pixels and a search area of 32×32 pixels. The computation of an addition, subtraction and an absolute value is weighted as one operation and a multiplication is weighted as three operations. Shifts and compares are not counted since these operations are a fraction of a complete ME software implementation.

By referring to Equation (B.1), the above-given basis leads to $n_b = (720 \times 576)/(16 \times 16) = 1620$ macroblocks to be processed and the refinement cost equals $c_r = 16 * 16^2 = 4096$ operations. Table B.4 is based on these numbers and presents the computational complexity for some configurations (representing high, medium and low complexity) for ME of one frame (values are rounded). The abbreviation R in the table stands for refinement. It has been ignored, that off-frame search positions do not need to be processed. It can be noticed that the type of the algorithm is decisive.

Algo	Block matcher	R	$n_{bc} * n_b$	c_{bc}	$c_r * n_b$	C_{ME}
2DFS	MSE	yes	1.67 M	1279 ops	6.64 M ops	2.14 G ops
2DFS	MSE	-	1.66 M	1279 ops	-	2.12 G ops
1DFS	SAD	yes	0.17 M	767 ops	6.64 M ops	0.13 G ops
1DFS	SAD	-	0.15 M	767 ops	-	0.12 G ops
NTSS	MiniMax	yes	0.05 M	512 ops	6.64 M ops	0.04 G ops

Table B.4: *Worst-case computational complexity expressed in operations (ops) of different configurations of ME algorithms.*

B.5 Statistics of real computational effort

Table B.4 above shows worst-case computational complexity of the ME algorithms. In reality, when testing the algorithms on real video material, the average number of block comparisons is obviously lower than in the worst case. Besides the transition from worst-case to average conditions, we have applied a number of measures to further reduce the computational effort of the ME algorithms. These measures are as follows.

- The computation of an error measure can be aborted when the sub-total of the computed error exceeds a previously found minimum (only possible for positive addends). This measure was used for all algorithms.
- To increase the probability that the previous approach can be applied, MVs referring to positions around the zero vector should be computed first if possible, since these vectors represent commonly occurring motion in video sequences. This was also used for all algorithms, but mainly full-search algorithms are affected by this approach.

Table B.5 shows the statistics of the amount of operations (measured in Million Operations Per Frame (MOPF)) needed to perform ME for three different sequences. A remarkable reduction of the computational effort compared to the worst-case situation can be observed (factor 3-4 less in several cases).

Algo	R	"Voit"			"Girl"			"Teeny"		
		a	b	c	a	b	c	a	b	c
2DFS	-	585	535	105	297	306	74	1035	962	184
2DFS	yes	600	545	110	311	315	78	1065	981	194
1DFS	-	92	70	21	48	42	13	163	128	39
1DFS	yes	106	80	26	61	51	18	193	147	48
TSS	-	30	22	8	18	14	5	57	41	16
TSS	yes	45	31	13	31	23	10	87	60	25
NTSS	-	33	24	8	14	11	4	61	44	17
NTSS	yes	47	34	13	27	20	9	91	63	26
BBGDS	-	17	12	4	10	7	3	54	35	10
BBGDS	yes	31	21	9	23	16	7	84	54	19
RME	yes	38	26	11	34	23	11	40	27	12

Table B.5: *Average statistics of the amount of Million Operations Per Frame (MOPF) for different ME algorithms and video sequences ("a", "b" and "c" stand for MSE, SAD and MiniMax, respectively).*

APPENDIX C

Reduced processing resolution

Current (programmable) video hardware architectures are based on a least 8-bit data processing, which is exactly the commonly used resolution of one sample per color component of the input video. The processing of single bits is costly and not suitable for these architectures, because it would require data rearrangement that is more costly than directly processing the original resolution. However, if efficient single-bit processing is possible, it opens another dimension for scaling the computational complexity of MPEG core functions. This topic is briefly described in this appendix.

The data-rearrangement problem of bit-based processing in programmable architectures can be solved with application-specific special memory interfaces for bit-based access. Such memory interfaces may be based on serial-in-parallel-out shift registers, but this is not further elaborated. In the following, it is expected that solutions for the data-rearrangement problem of bit-based processing exist and can be used. Based on this assumption, a complexity scalable DCT and ME is presented in Section C.1 and C.2, respectively.

C.1 Scalable DCT based on bit-slices

C.1.1 Overview

In this section, a complexity scalable system is presented that computes the DCT such that the video data is processed in bit slices. The system produces

meaninful intermediate results by first transforming the most significant bits of a frame. These results are improved when more details of the remaining bits become available. This is illustrated in the following.

In the definition of the 2-D DCT as given in Equation (2.1), the pixel values $x[i,j]$ can be replaced by a summation of all n_{bits} bits, from $x[i,j]_{(0)}$ (least significant bit) to $x[i,j]_{(n_{bits}-1)}$ (most significant bit), as shown below.

$$x[i,j] = \sum_{k=0}^{n_{bits}-1} x[i,j]_{(k)} * s_k * 2^k, \tag{C.1}$$

where the factor s_k distinguishes between signed and unsigned values. This factor is defined as follows.

$$s_k = \begin{cases} (-1) & \text{if } k = n_{bits} - 1 \; and \; x \text{ is a signed value,} \\ (+1) & \text{in all other cases.} \end{cases}$$

Therefore, Equation (2.3) can be rewritten as

$$\underline{\underline{Y}} = \sum_{k=0}^{n_{bits}-1} s_k * 2^k * \left(\underline{\underline{K_N}}^{\top} * \underline{\underline{X_{(k)}}} * \underline{\underline{K_N}} \right). \tag{C.2}$$

Equation (C.2) shows that the transformation of a picture block is processed bit by bit, thus the DCT transforms 1-bit input values in each computation step. For input resolutions of n_{bits} bits, n_{bits} computation steps are performed. Each step processes a bit-slice of $N \times N$ bits for a 2-D DCT and bit-slices of N bits for a 1-D DCT.

The bit-slice based computation is a technique that was used in [75] for the implementation of a full 16×16 DCT computation. In this section, the approach is differently developed and is considered for complexity scalability.

C.1.2 Architecture

The concept of the bit-slice DCT leads to a simple hardware design that is presented in the following, using an 8×8 DCT as an example. The computation of the 8×8 DCT is separated into two 8×1 DCTs by using the row-column approach. In this case, each DCT has eight input samples, thereby limiting the set of possible bit-slices to $2^8 = 256$ bit-slices. This means that the DCT output of eight coefficients $y[.]$ depends on 256 different possible input values,

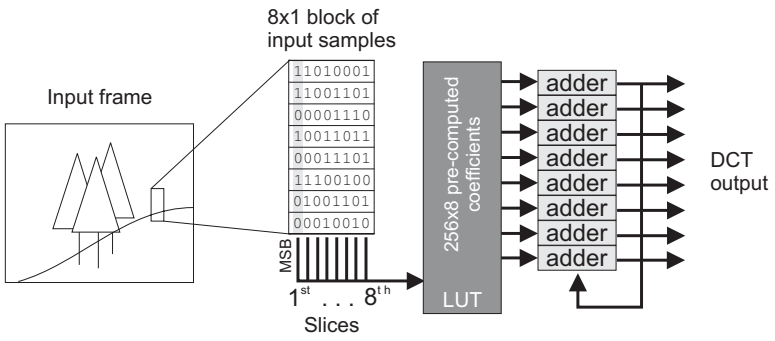

Figure C.1: *Bit-slice computation of an* 8×1 *DCT.*

which can be stored in a look-up-table (LUT). When using the LUT, the computation of the DCT reduces to a summation of LUT values, without requiring multiplications. Figure C.1 depicts the LUT-based bit-slice DCT computation. The architecture consists of three main parts, namely an interface for retrieving bit-slices from memory, a LUT and an adder stage. The implementation concept for these parts are outlined below.

Interface for retrieving bit-slices

This part can be more elegantly implemented with dedicated hardware, since standard processors are usually not prepared for efficiently accessing several values (in memory) and simultaneously retrieving single bits and subsequently combining these bits to a bit-slice. As already mentioned, this problem can be solved with parallel-in-serial-out shift registers. A more sophisticated solution as used in [75] can replace immediately shifted-out data bits by new input values, in order to obtain a continuous data flow and an optimal cache utilization.

LUT

The LUT in this example has to generate eight output values for each bit-slice (8-point DCT). The number of LUT entries is therefore $2^8 * 8 = 2048$ entries. The bit-slices taken from the eight input samples are employed as the address for accessing the values in the LUT. The size of the LUT is reduced by exploiting the redundancy found in the LUT as follows. The outputs of the LUT are the result of the 1-D DCT computation that is performed by a matrix multiplication $\underline{\underline{X_{(i)}}} * \underline{\underline{K_8}}$ (see Equation (C.2)) for $0 \leq i \leq n_{bits}$, where $X_{(i)}$ is a

bit-slice of the input-data matrix. In this computation, a row of the matrix $X_{(i)}$ is multiplied with a column of matrix K_8. Because the matrix $X_{(i)}$ contains only zeros and ones, the multiplication is reduced to a summation. Due to the periodicy of the cosine function, the arguments of the cosine matrix K_8 as defined in Equation (2.2), consist of eight different absolute values that occur as positive and negative numbers in the matrix. Since the computation has been reduced to a summation only, values regularly cancel each other, thereby mapping different input data to the same output values. It has been found that the possible mapping of input data reduced the number of required LUT entries to 71 entries only. Further details are omitted here.

Adder stage

Evidently, this part can be implemented in software. However, since no intermediate results are necessary until the desired number of bit-slices is processed, a more efficient solution can be implemented in hardware using back-coupled carry-save-adders. Such adders provide extra registers for the carry bits, thereby preventing immediate processing of these bits and thus minimizing the response time of the adder. The basic architecture of back-coupled carry-save-adders is shown in Figure C.2.

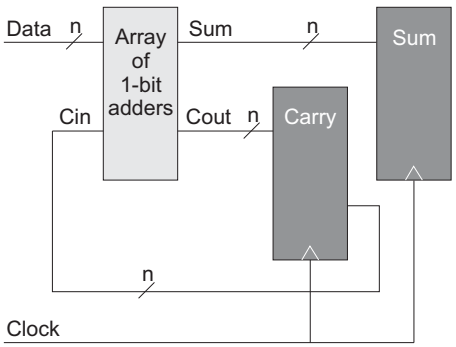

Figure C.2: *Basic architecture of back-coupled carry-save-adders.*

Implementation with pipelining

For high throughput and continuous data flow, the aforementioned parts of the bit-slice DCT computation can all be implemented with a pipelined hardware architecture, thereby allowing multiple computations in parallel. A novel architecture for high-throughput pipelining at low cost, which is applicable for

bit-sliced DCT computation, is presented in [76]. As shown in [77, 78], this architecture can also be used for ring structures, and it is therefore feasible for the adder stage shown in Figure C.1 and C.2. In a performance-oriented implementation with pipelining as suggested above, the 8×1 DCT output depends on the number n_{bits} of bit-slices processed. Eight output samples would be available with a data rate of n_{bits}/t_{pd}, where t_{pd} is the latency of one pipeline stage.

C.1.3 Experimental results

An experiment has been set-up by concatenating two 1-D DCTs to a 2-D DCT using the row-column approach. In this configuration, the two 1-D DCTs use a different bit-resolution for the input data. The first 1-D DCT uses an 8-bit unsigned integer and outputs a 12-bit signed integer format, which is subsequently processed by the second 1-D DCT. In this experiment, a perfect IDCT is used to reconstruct the picture.

Figure C.3 shows the PSNR of the reconstructed picture "Circle", where a different number of bits was processed in the first and the second 1-D DCT. The marked path shows the maximum PSNR for a given number of compu-

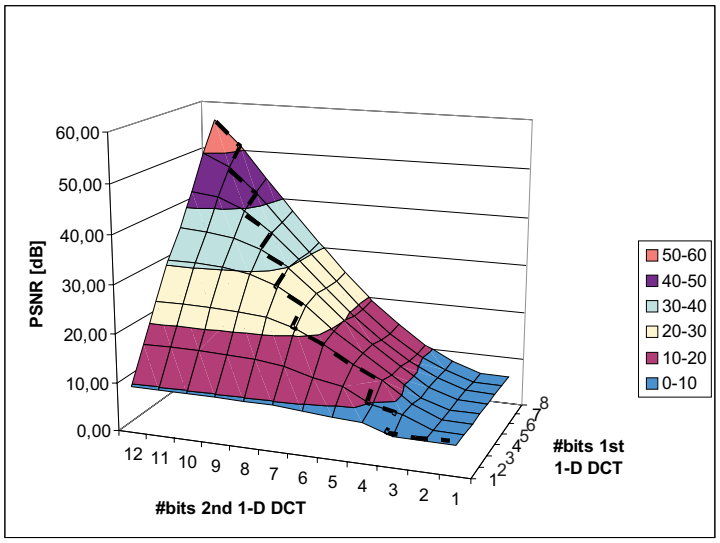

Figure C.3: *PSNR results of processing the "Circle" picture in bit-slices.*

tation steps (equal to the number of processed bit-slices). It can be seen that the number of computation steps available for a given complexity should be evenly spread over both DCTs, however with an offset of 3-4 bits additional resolution for the second 1-D DCT.

Figure C.4 shows some examples of the reconstructed pictures taken at certain points in the computation path that is shown in Figure C.3. In the figure, the index $m - n$ stands for m processed bit-slices in the first 1-D DCT and n processed slices in the second 1-D DCT. It can be seen that the content of the picture can be identified using only a few computations, where details are visible after roughly half of the overall DCT computation, when using this approach.

C.2 Bit-based computation of ME

C.2.1 Simply reduced bit-resolution of input values

In this section, we study a reduced bit-resolution of the input samples for motion estimation, in order to decrease the computational complexity, leading to e.g. faster processing or less power consumption. The following experiment indicates the change in prediction quality of the ME process. In the experiment, the 8-bit input values for the ME process are reduced in bit-resolution by simply skipping a number of their least significant bits. Figure C.5 shows the result when processing the "Voit" sequence with a 2DFS motion estimator that includes motion-vector (MV) refinement to half-pixel accuracy. The MSE was used as block-matching criterion. The bottom curve of the figure results from processing zero values (all bits skipped), which represents the absence of motion estimation. From the other curves in the figure, it can be seen that no serious video quality decrease occurs up to roughly halved input ME resolution. The dashed curve results from skipping 6 bits, thereby leaving a two-bit resolution, which results in a stronger but still acceptable decrease of the prediction quality in this experiment.

C.2.2 One-Bit Motion Estimation (1BME)

The 1BME presented in [79] is a preprocessing step for other ME algorithms. The aim of 1BME is to transform input frames, using a convolution, into a format that represents one sample with a single bit. The preprocessing is outlined in Figure C.6.

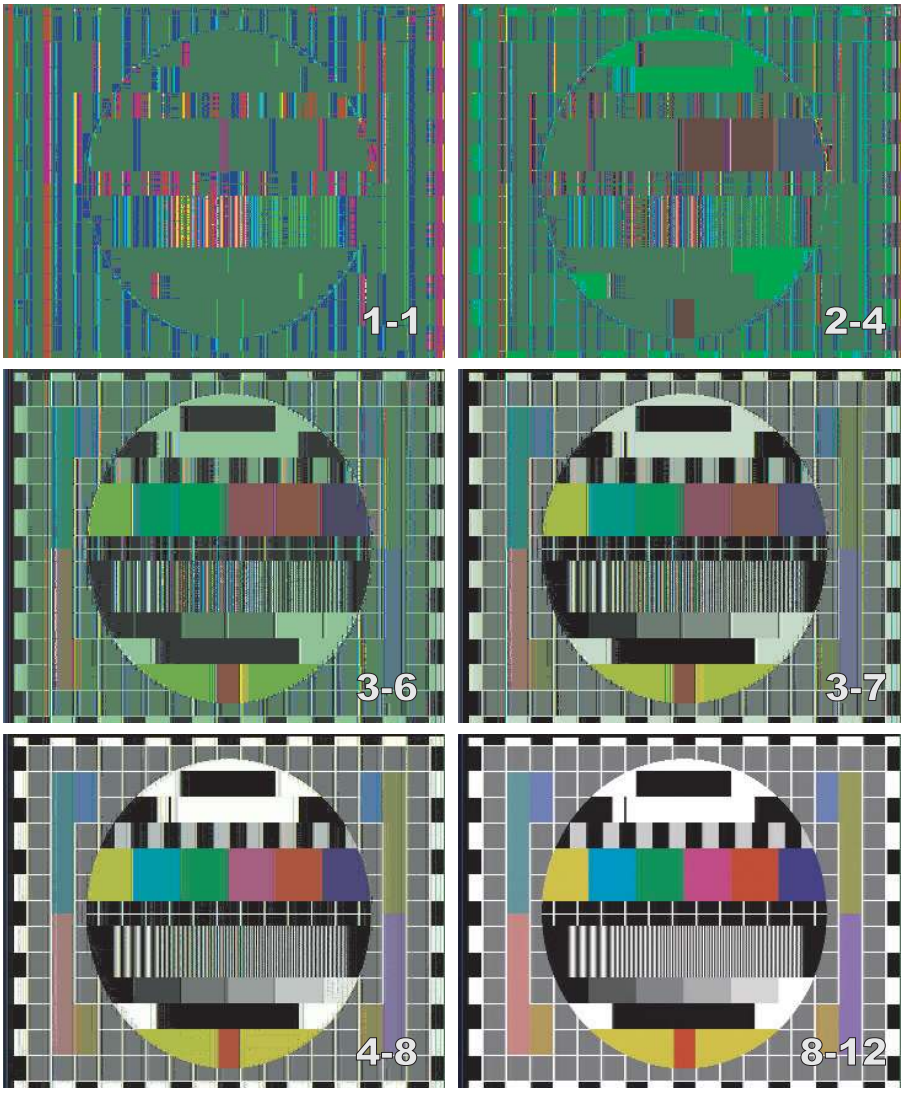

Figure C.4: *Examples of reconstructed pictures using a bit-slice DCT computation (first (second) number is the number of bit-slices in the first (second) DCT).*

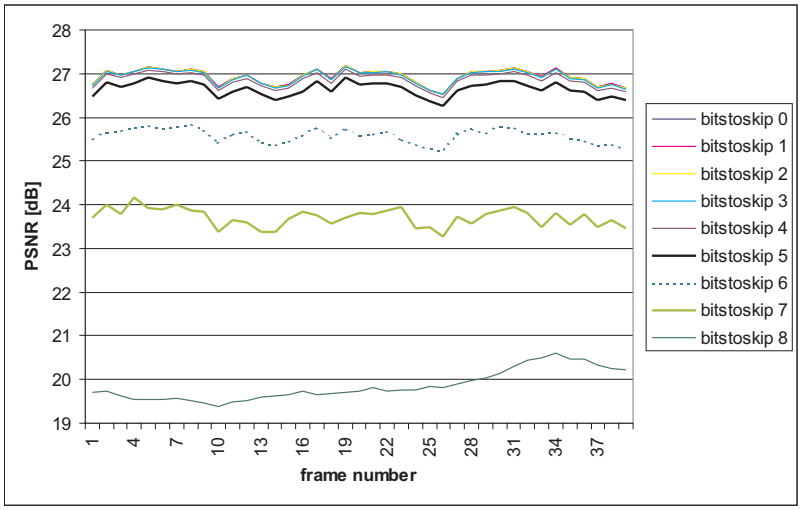

Figure C.5: *PSNR of the motion compensated frames of the "Voit" sequence when reducing the processing bit-resolution.*

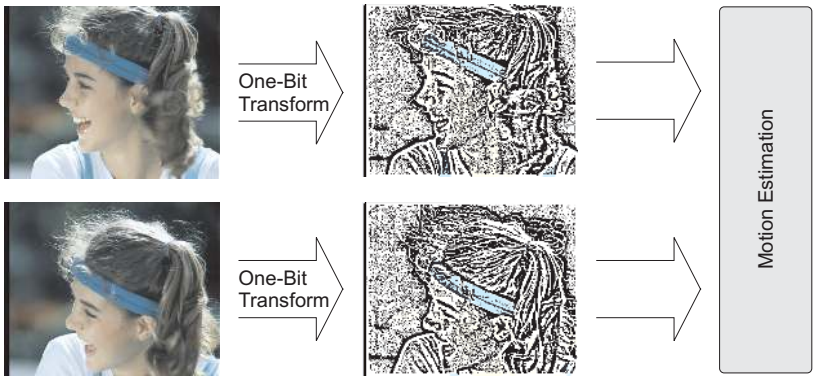

Figure C.6: *Preprocessing step of the 1BME applied to motion estimation.*

To indicate the performance of 1BME, an experiment is set up in which the ME is reduced to 1-bit processing, while the motion compensation is kept at the original resolution. Two different ME algorithms are evaluated: 2DFS and simple RME. The 2DFS is used with MSE as block-matching criterion and a refinement to half-pixel accuracy, because this gives the best quality. The search range is 32×32 pixels and the algorithm stops a vector evaluation when its intermediate acummulated error exceeds a previously found value. The simple RME uses SAD as block-matching criterion and half-pixel accuracy, because this gives a good trade-off between achieved quality and computational complexity. The computation complexity of the ME algorithm (without preprocessing) is potentially reduced by a factor 8, because 8-bit operations are replaced by 1-bit operations. It was found that the 1BME technique increases the measured number of required vector evaluations by for example 34%, when using full search. A speed-up factor of roughly between 5 and 6 is thus more realistic.

Figure C.7 shows the result of the experiment when using the "Voit" sequence. The reduction of prediction quality is acceptable in comparison to the reduced factor 5-6 in computational complexity. However, the transformation to 1-bit resolution should not involve more operations than the number of operations that can be saved. In this example, we found that this transformation per block is equally complex as approximately 9 SAD computations, which makes this approach useless for algorithms that perform less than 10 block-comparisons per macroblock.

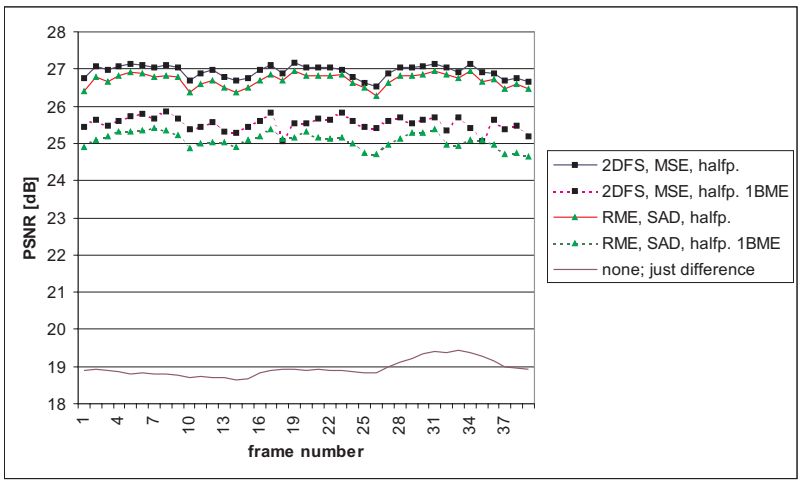

Figure C.7: *Effect of 1BME preprocessing on the prediction quality of two ME algorithms.*

APPENDIX D

Test sequences

This appendix briefly describes the test sequences that have been used for the experiments in this thesis.

D.1 “Girl”

Figure D.1: *The “Girl” sequence.*

The “Girl” sequence (see Figure D.1) has standard-definition (SD) resolution and shows two girls clapping their hands and a round picture-in-picture showing a playing girl. For motion-estimation (ME) experiments, the sequence was chosen as an example of a static scene with some slight local motion only. The background curtain moves half a pixel on the right side and the

girls' bodies show little motion. The train and the clapping hands of the girls are the only objects that move faster. For DCT experiments, the sequence was not considered.

D.2 "Voit"

Figure D.2: *The "Voit" sequence.*

The "Voit" sequence (see Figure D.2) has SD resolution and shows a scene with a car passing a gate that is closing behind it. For ME experiments, the sequence was chosen as an example of medium motion. The car and the gate in the background move in different directions. The main movement direction of the gate is along the x-axis (from left to right), while the car moves along the z-axis (from back to front), while slightly turning to the right. For DCT experiments, the sequence provides a mixture of high and low spatial activity (grass and street) and clear edges with the gate and the car.

D.3 "Teeny"

Figure D.3: *The "Teeny" sequence.*

The “Teeny” sequence (see Figure D.3) has SD resolution and shows a teenaged girl turning her head suddenly. For ME experiments, the sequence was chosen as an example of high motion. The girl turns her head very fast around the y-axis. The girl’s hair has a complex structure, which is smoothed by the fast motion. This makes it almost impossible to find reliable motion vectors. For DCT experiments, the sequence was not considered.

D.4 “Renata”

Figure D.4: *The “Renata” sequence.*

The “Renata” sequence (see Figure D.4, also SD resolution) portrays a woman walking in front of a static background. For ME experiments, the sequence provides mostly constant motion of the objects. DCT experiments were conducted to evaluate the quality of the texture details in the calendar, the wall carpet, the scarf and the woman’s face.

D.5 “Stefan”

Figure D.5: *The “Stefan” sequence.*

The "Stefan" sequence (see Figure D.5) has CIF resolution and shows a tennis scene, where the camera tracks one player. For ME experiments, the sequence was chosen as an example having a mixture of different types of motion. The sequence contains mainly camera motion, including pan and zoom. At a certain point, the camera does not move, e.g. when the player is returning a ball that came nearby. Some motion occurs when he is awaiting the next ball, and fast motion occurs in a situation where the player has to catch a ball that is further away. For DCT experiments, the sequence provides a mixture of texture details (the player and text), high and low activity (tribune and court, respectively), as well as clear edges (lines of the court).

D.6 "Foreman"

Figure D.6: *The "Foreman" sequence.*

The "Foreman" sequence (see Figure D.6) has CIF resolution and shows a foreman on a construction site. For ME experiments, the sequence provides some complex motion. The foreman has been recorded with a handheld camera, causing permanent low motion in various directions. In addition, the foreman is quickly turning his head, covering a main part of the display. At the end, the sequence contains a fast pan from the foreman to the construction site. For DCT experiments, the sequence provides some fine details (e.g. face and trees) as well as some strong edges (building in the background).

D.7 "Table tennis"

The "Table tennis" sequence (see Figure D.6) has CIF resolution and shows a scene from a table-tennis match. For ME experiments, the sequence provides a short zoom at the beginning and a scene change. Besides these two situations, the camera is static and the active players cover only a part of the

Figure D.7: *The "Table tennis" sequence.*

display, overall resulting in slow motion, except for the ball and the arms. For DCT experiments, the sequence mainly provides strong edges (lines on the table) and flat areas (the table itself). The visibility of the fine details in the background depends on the camera zoom-factor.

Summary

This thesis concentrates on designing a scalable MPEG encoding system, featuring scalable video quality and a corresponding scalable resource usage. This feature is called complexity scalability and enables MPEG encoding either on platforms having limited capabilities (computing power, memory, stand-by-time, etc.), and/or in parallel to various applications on the same platform, as motivated in Chapter 1. State-of-the-art MPEG algorithms do not provide scalability, thereby hampering e.g. low-cost solutions for portable devices and varying coding applications in multi-tasking environments. The MPEG coding standard employs two complex function blocks, i.e. the Discrete Cosine Transformation (DCT) and the Motion Estimation (ME), which were redesigned for scalability. The remaining blocks such as quantization and coding were adapted in order to cooperate fluently within our scalability concept.

The DCT transforms picture blocks into DCT-coefficient blocks in the transform domain for an efficient picture representation. Chapter 2 discusses the mathematical background of the DCT and compares a number of state-of-the-art fast DCT algorithms that have attractive properties for soft- and hardware implementations. Subsequently, two DCT algorithms are taken as case studies for developing complexity scalable DCT. Chapter 3 presents a scalable DCT computation that determines a scalability-optimized computation order for DCT coefficients, such that the amount of computed coefficients is optimized for a constrained number of computing cycles. Experiments show a clear visible improvement in picture quality (readability and sharpness) of the scalable DCT when compared with a conventional DCT implementation, if the computation effort is for example limited to half of the normal effort.

The purpose of ME is to compute motion vectors (MVs) for removing temporal redundancy via motion-compensated prediction. In Chapter 4, we study a set of fast ME algorithms. Recursive ME, employing previously found MVs

for neighboring data as an estimate for new data, forms a particularly efficient state-of-the-art solution and was therefore taken as a basis for further research. In Chapter 5, we have explored two new techniques for complexity scalability of the ME process. The first technique approximates the MPEG MV fields based on an initial low-cost ME with the video frames at the entrance of the encoder. Full-quality motion search is obtained with an optional refinement of those vector fields. Scalability in this ME system, called SMART, is obtained by varying the number of processed vector fields. SMART also applies new concepts, like multi-vector-field estimation and advanced scaling of vector fields. The second ME technique is called CARES and is based on block classification, which concentrates the computation effort to blocks containing edges that leads to accurate ME. Compared to state-of-the-art recursive ME algorithms, CARES reduces the set of motion-vector candidates that are evaluated by distributing good vectors for evaluation to other blocks, thereby preventing re-evaluation of identical MVs. Scalability is obtained by varying the classification strength, leading to a varying number of vector evaluations that are performed per block. The described techniques for ME proved to offer a large range of scalability of the ME computation process.

Finally, the developed techniques were integrated into a fully scalable MPEG encoder framework, including quantization and MPEG-2 compatible coding. The performance of the scalable encoding system is compared to regular state-of-the-art MPEG-2 encoders in Chapter 6. The scalable DCT has impacts on functions of the MPEG coding system that process DCT coefficients. These functions can be scaled accordingly to the DCT by adapting to the reduced number of computed coefficients. The scalable ME affects the number of evaluated MV candidates, but not the motion compensation or the VLC coding of MVs. The key scalability parameters that we found are therefore the number of computed DCT coefficients and the number of motion-vector evaluations.

Chapter 7 concludes that the obtained complexity scalability results in new coding algorithms, which differ from conventional MPEG encoding design. The memory usage and the processing load depend on the quality level of the processing blocks. The resulting scalable MPEG encoder offers a wide range of complexity scalability, video quality and compression rates. The obtained scalability range of our scalable encoder in terms of computational complexity is about a factor of three, whereas a non-scalable parameterized MPEG encoder has a factor of about 1-1.5. The developed scalability techniques can be readily applied in portable MPEG coding systems and may well be used in new coding standards such as MPEG-4 and H.264.

Samenvatting

Dit proefschrift behandelt het ontwerpen van een schaalbaar MPEG coderingssysteem met een schaalbare beeldkwaliteit en een daarmee corresponderend schaalbaar gebruik van rekenkracht. Dit concept wordt schaalbare complexiteit genoemd en maakt MPEG coderen mogelijk op zowel platformen met beperkte rekencapaciteit (rekenkracht, geheugen, gebruikstijd, enz.), als in combinatie met verschillende toepassingen op hetzelfde platform (zie hoofdstuk 1). De huidige MPEG algoritmen zijn niet schaalbaar, waardoor bijvoorbeeld goedkope oplossingen voor draagbare apparatuur of dynamische coderingstoepassingen in omgevingen met veel parallelle taken, worden belemmerd. De MPEG coderingsstandaard bevat twee complexe functieblokken, namelijk de Discrete Cosinus Transformatie (DCT) en de Bewegingsschatting (BS), welke beide voor schaalbaarheid opnieuw werden ontworpen. De resterende functies zoals die voor quantisatie en codering werden aangepast, zodat ze goed met de schaalbaarheid konden omgaan.

De DCT transformeert blokken uit een beeld naar blokken met DCT coëfficiënten in het transformatiedomein om een efficiënte representatie van het beeld te krijgen. Hoofdstuk 2 bespreekt de wiskundige achtergrond van de DCT en vergelijkt een aantal moderne snelle DCT algoritmen met aantrekkelijke eigenschappen voor soft- en hardwarerealisaties. Vervolgens worden twee DCT algoritmen gekozen voor het ontwikkelen van een in complexiteit schaalbare DCT. Hoofdstuk 3 presenteert een schaalbare DCT berekening gebaseerd op een optimalisatie van de berekeningsvolgorde van DCT coëfficiënten, hetgeen betekent dat de hoeveelheid berekende coëfficiënten voor een gelimiteerd aantal berekeningscycli wordt geoptimaliseerd. Experimenten met een schaalbare DCT geven een duidelijk zichtbare verbetering in beeldkwaliteit (leesbaarheid en duidelijkheid) vergeleken met een conventionele DCT uitvoering; dit geldt wanneer de berekeningsinspanning tot bijvoorbeeld de helft van de nominale (niet schaalbare) inspanning wordt beperkt.

Het doel van bewegingsschatting (BS) is om bewegingsvectoren (BV) te berekenen voor het verwijderen van redundantie in de tijdsrichting met bewegingsgecompenseerde predictie. Hoofdstuk 4 bestudeert een aantal snelle BS algoritmen. De recursieve vorm van BS, die eerder gevonden bewegingsvectoren voor naburige datablokken als een schatting voor nieuwe datablokken gebruikt, is een moderne efficiënte oplossing en werd daarom als basis voor verder onderzoek genomen. In hoofdstuk 5 zijn twee nieuwe technieken onderzocht voor een in complexiteit schaalbare bewegingsschatting. De eerste techniek benadert de MPEG BV velden, gebaseerd op een eerste goedkope bewegingsschatting die gebruik maakt van videobeelden aan de ingang van de encoder. De volledige kwaliteit van de bewegingsschatting wordt verkregen met een aanvullende facultatieve verbetering van die vectorvelden. Schaalbaarheid in dit BS systeem, SMART genoemd, wordt door het variëren van het aantal bewerkte vectorvelden bereikt. SMART past ook nieuwe begrippen toe, zoals veelvoudige vectorveld schatting en verbeterde schaalbaarheid van vectorvelden. De tweede BS techniek wordt CARES genoemd en is gebaseerd op een blokrangschikking, die de berekeningsinspanning concentreert op blokken die sterke signaalovergangen bevatten, wat tot een nauwkeurige BS leidt. In vergelijking met recente recursieve BS algoritmen vermindert CARES het aantal bewerkte kandidaat bewegingsvectoren. Dit komt door het verdelen van goede vectoren aan andere blokken voor bewerking, zodat een herhaalde bewerking van identieke bewegingsvectoren wordt vermeden. Schaalbaarheid wordt bereikt door de mate waarin blokken veranderlijk worden gerangschikt; dit leidt tot een variabel aantal vectorbewerkingen die per blok wordt verricht. De beschreven technieken voor BS hebben bewezen dat een groot bereik in schaalbaarheid van het BS proces wordt verkregen.

Tenslotte worden de ontwikkelde nieuwe technieken in een compleet schaalbaar MPEG coderingssysteem geïntegreerd, inclusief quantisatie en codering die compatibel is met MPEG-2. De performance van het schaalbare coderingssysteem wordt in hoofdstuk 6 met recente MPEG-2 encoders vergeleken. De schaalbare DCT heeft vooral invloed op de functies van het MPEG coderingssysteem die DCT coëfficiënten bewerken. Deze functies kunnen net als de DCT aangepast worden aan het gereduceerde aantal berekende DCT coëfficiënten. De schaalbare bewegingsschatting verandert het aantal bewerkte kandidaat bewegingsvectoren, maar niet de bewegingscompensatie of het VLC coderen van bewegingsvectoren. De belangrijkste gevonden parameters voor schaalbaarheid zijn daarom: het aantal berekende DCT coëfficiënten en het aantal bewerkte bewegingsvectoren.

Hoofdstuk 7 concludeert dat de bereikbare schaalbaarheid in complexiteit resulteert in het ontwerpen van nieuwe coderingsalgoritmen, die van conventionele ontwerpen voor MPEG codering verschillen. Het geheugengebruik en de benodigde rekenkracht hangen af van de kwaliteitsinstelling van de signaalbewerkende functies. De resulterende schaalbare MPEG codering biedt een groot bereik van schaalbaarheid in complexiteit, beeldkwaliteit en compressiefactoren aan. Het schaalbare bereik van het voorgestelde coderingssysteem uitgedrukt in rekencomplexiteit is meer dan een factor drie, terwijl een niet-schaalbare MPEG codering met variabele parameterinstellingen een factor 1-1,5 heeft. De ontwikkelde technieken voor schaalbaarheid kunnen direct in mobiele MPEG coderingssystemen worden toegepast. Daarnaast zijn ze goed bruikbaar voor nieuwe coderingsstandaarden zoals MPEG-4 en H. 264.

Zusammenfassung

Diese Arbeit befaßt sich mit der Entwicklung eines skalierbaren MPEG-Codierungssystems, das sich durch skalierbare Bildqualität und einem entsprechenden skalierbaren Berechnungsaufwand auszeichnet. Dieses Konzept wird skalierbare Komplexität genannt und ermöglicht die MPEG-Codierung auf Plattformen mit begrenztem Leistungsvermögen (Rechenleistung, Speicher, Bereitschaftszeit, usw.), und/oder in Kombination mit verschiedenen Anwendungen, die auf der gleichen Plattform laufen (siehe Kapitel 1). Heutige MPEG-Algorithmen sind nicht skalierbar und behindern dadurch z.B. die Entwicklung preiswerter Lösungen für mobile Geräte oder für dynamische Codierungsanwendungen in Systemen, die viele Aufgaben parallel ausführen. Der MPEG-Codierungsstandard beinhaltet zwei komplexe Funktionsblöcke, nämlich die Diskrete Cosinus Transformation (DCT) und die Bewegungsschätzung (BS), die beide zu Skalierungszwecken neu entworfen wurden. Die restlichen Module, wie zum Beispiel die Quantisierung und die Codierung, wurden angepaßt, damit sie gut in ein skalierbares System passen.

Die DCT transformiert Bildblöcke in Blöcke aus DCT-Koeffizienten, welche eine effiziente Repräsentation eines Bildes ermöglichen. Kapitel 2 stellt den mathematischen Hintergrund der DCT dar und vergleicht eine Auswahl moderner Algorithmen zur schnellen DCT-Berechnung, die interessante Eigenschaften für Software- und Hardwarerealisierungen haben. Anschließend werden zwei DCT-Algorithmen für die Entwicklung einer skalierbaren DCT ausgewählt. In Kapitel 3 wird eine skalierbare DCT-Berechnung präsentiert, die, basierend auf der Anzahl von Operationen, eine optimierte Berechnungsreihenfolge für DCT-Koeffizienten ermittelt. Daraus folgt eine optimierte Anzahl von berechneten Koeffizienten, die für eine eingeschränkte Anzahl von Berechnungsschritten erreicht wird. Die skalierbare DCT zeigt in Experimenten im Vergleich mit einer konventionellen DCT-Berechnung eine deutlich sichtbare Verbesserung der Bildqualität (Leserlichkeit und Bildschärfe),

wenn der nominale (nicht skalierbare) Berechnungsaufwand beispielsweise halbiert wird.

Das Ziel der Bewegungsschätzung (BS) ist es Bewegungsvektoren (BV) zu berechnen, mit deren Hilfe bewegungskompensierte Bildvorhersagen erstellt werden, die zur Entfernung der in Videosequenzen enthaltenen zeitlichen Redundanz dienen. In Kapitel 4 wird eine Auswahl von schnellen BS-Algorithmen verglichen. Die rekursive Form der BS, welche bereits vorher gefundene Bewegungsvektoren von angrenzenden Datenblöcken als eine Vektorschätzung für neue Datenblöcke verwendet, ist eine moderne und effiziente Lösung und formte daher die Basis für weitere Untersuchungen. In Kapitel 5 werden zwei neue Verfahren zur skalierbaren Bewegungsschätzung erarbeitet. Das erste Verfahren nähert die MPEG-BV-Felder basierend auf einer ersten preiswerten Bewegungsschätzung an, welche die Videobilder am Encodereingang verwendet. Die vollständige Qualität der Bewegungsschätzung wird mit einer optionalen Verfeinerung der Vektorfelder erreicht. Die Skalierbarkeit in diesem BS-System, genannt SMART, wird durch eine veränderliche Anzahl der verarbeiteten Vektorfelder erreicht. SMART verwendet dabei auch neue Verfahren, wie eine auf mehreren Feldern basierende Vektorfeldschätzung und eine verbesserte Skalierung von Vektorfeldern. Das zweite Verfahren zur skalierbaren Bewegungsschätzung wird CARES genannt und basiert auf Blockklassifizierung, die den Berechnungsaufwand auf Blöcke mit starken Signalübergängen konzentriert und damit zu einer genaueren Bewegungsschätzung führt. Verglichen mit modernen rekursiven BS-Algorithmen, verringert CARES die Anzahl der verarbeiteten Vektorkandidaten durch die Weitergabe von guten Vektoren zur Verarbeitung an andere Blöcke, wodurch eine wiederholte Verarbeitung von identischen Vektoren verhindert wird. Die Skalierbarkeit wird durch die Änderung der Klassifizierungsstärke erreicht, was zu einer variablen Anzahl von Vektorkandidaten führt, die pro Datenblock verarbeitet werden. Es wurde gezeigt, daß die beschriebenen Verfahren zur Bewegungsschätzung eine weitreichende Skalierung des BS-Prozesses ermöglichen.

Abschließend wurden die neu entwickelten Verfahren, inklusive der Quantisierung und der MPEG-2 kompatiblen Codierung, in ein vollständig skalierbares MPEG-2-Codierungssystem integriert. Die Leistung des skalierbaren Codierungssystems wird in Kapitel 6 mit modernen MPEG-2-Encodern verglichen. Die skalierbare DCT zeigt vor allem Auswirkungen auf jene Funktionen des MPEG-Codierungssystems, die DCT-Koeffizienten verarbeiten. Diese Funktionen können ähnlich wie die DCT an eine verringerte Anzahl von be-

rechneten Koeffizienten angepaßt werden. Die skalierbare Bewegungsschätzung verändert die Anzahl der verarbeiteten BV-Kandidaten, nicht aber die Bewegungskompensierung oder die VLC-Codierung der Bewegungsvektoren. Die wichtigsten gefundenen Parameter zur Skalierbarkeit sind daher: die Anzahl der berechneten DCT Koeffizienten und die Anzahl der verarbeiteten BV-Kandidaten.

In Kapitel 7 wird die Schlußfolgerung gezogen, daß die erreichte Skalierbarkeit der Komplexität neue Codierungsalgorithmen hervorbringt, die vom konventionellen Entwurf der MPEG-Codierung abweichen. Die Speicherverwendung und der notwendige Berechnungsaufwand hängen von den Qualitätseinstellungen der signal-verarbeitenden Module ab. Die entwickelte skalierbare MPEG-Codierung ermöglicht eine weitreichende Skalierbarkeit in punkto Komplexität, Bildqualität und Kompressionsraten. In Hinsicht auf die Komplexität des Berechnungsaufwands erzielt der Skalierungsbereich des vorgestellten skalierbaren Codierungssystems einen Faktor von ungefähr drei, während eine nicht-skalierbare MPEG-Codierung mit variablen Parametereinstellungen einen Faktor von ungefähr 1-1,5 hat. Die entwickelten Verfahren zur Skalierbarkeit können direkt in mobilen MPEG-Codierungssystemen verwendet werden. Weiterhin sind die Verfahren für neue Codierungsstandards, wie beispielsweise MPEG-4 und H.264, geeignet.

Acknowledgments

This thesis is based on research that I performed first at the University of Mannheim, Germany, and later at the Eindhoven University of Technology, The Netherlands. The conducted research was part of a larger cooperation framework with Philips Research Labs. in Eindhoven. Because of the good connections between my promoter Peter de With and Philips, I had the opportunity to perform scientific research on industrially relevant topics. Thanks to Peter for his endless support, encouragement and enthusiasm during all these years.

The research project I was involved in at Philips was supervised by my copromoter Christian Hentschel, whom I owe thanks not only for showing confidence in me by contributing to the major funding of my research, but also for providing fruitful discussions on scalable systems and valuable recommendations on other topics. Thanks to the University of Mannheim and the Philips Research management for their funding.

I thank my second promoter Inald Lagendijk for the new insights he gave me about my work and for his attempt for speeding up the review process.

Special thanks to Dirk Farin, who joined the University of Mannheim shortly after me, for all his help, input and regular fruitful discussions on video coding issues. During the moving to Eindhoven, many of my colleagues and friends in Mannheim and Eindhoven gave a hand that allowed a quick settle down. I really appreciate their help that made things easier. Thank goes to my parents who made my study at the Technical University of Darmstadt possible and thereby enabled me to work in research as I'm doing now. Big hugs to my family for their support and the missed hours I spent on working on this thesis and could not share with them.

Biography

Stephan Mietens was born in Frankfurt/Main, Germany, in 1972. He graduated in computer science from the Technical University of Darmstadt, Germany, in 1998 on the topic of "Asynchronous VLSI design". Subsequently, he joined the University of Mannheim, where he started his research on "Flexible Video Coding and Architectures" in cooperation with Philips Research Laboratories in Eindhoven, The Netherlands. He joined the Eindhoven University of Technology in Eindhoven, The Netherlands, in 2000, where worked towards a Ph.D. degree on "Scalable Video Systems". In 2003 he joined the Philips Research Laboratories, where he is involved in projects developing new video coding techniques, and in the same year, he became project leader of one of these projects.

STELLINGEN

behorende bij het proefschrift

Complexity Scalable MPEG Encoding

door

Stephan Oliver Mietens

I

The number of computed DCT coefficients is a key parameter for MPEG scalability, because it controls the computational complexity of almost all coding functions in the prediction loop of an encoder.

Chapter 6 of this thesis.

II

The number of evaluated motion vectors is a key parameter for MPEG scalability, because its number is varied at different abstraction levels of motion estimation.

Chapter 5 of this thesis.

III

As opposed to the full DCT computation where the actually used algorithm has a minor influence on the video quality, the underlying fast DCT algorithm for scalable DCT computation should be carefully selected for optimizing the obtained quality under computation limitations.

Chapter 3 of this thesis.

IV

Despite the research effort in motion estimation during the last decades, it was not found that approximations of motion-vector fields at the GOP-structure level, as performed by SMART, enables a broad range for trading off motion-compensated frame prediction quality against the number of performed vector evaluations.

Chapter 5 of this thesis.

V

Whereas modern recursive motion-estimation algorithms base their prediction on motion vectors of previously processed blocks, the CARES algorithm discloses the selected motion vector of the actual block in the opposite way to previously processed (and near future) blocks, which leads to a further reduced number of vector evaluations.

Chapter 5 of this thesis.

VI

The reason why the MPEG scalability proposals as introduced with the MPEG-2 specification have not been applied in practical systems, can be found in the additional computational complexity required when performing video coding according to these proposals.

VII

When taking the peak-signal-to-noise ratio (PSNR) as quality measure, it should be considered that the relative visual improvement of the picture quality per unity dB of gain in PSNR is inversely proportional to the absolute measured PSNR value. This means that a unity dB gain in PSNR adds relatively more to the visual quality at the lower part of a quality range than at its higher part.

Chapter 3 of this thesis.

VIII

The use of complexity scalable algorithms for streaming multimedia data over networks will improve the quality for noisy channels and the operation robustness, which leads to an improved uptime of the network.

IX

Efficient video compression over time results from both adapting the video coding standards to the video content and vice versa.

X

If a picture says more than a 1000 words, a film displaying at 24 pictures per second says more than 24,000 words per second.

XI

Although dust and mechanical computer mouses are enemies, an optical mouse is probably the only electronic device that benefits from dust on the surface on which it is used.

XII

When comparing the research efforts of computer scientists in minimizing redundant memory accesses in algorithmic design and the required high amount of redundant repetitions during the education of children, it should not be concluded that education of children is an unsuitable task for computer scientists.

XIII

With the widespread use of language translation programs, the work of professional translators contains more verification than translation.

XIV

The basic MPEG-2 coding concept applying I-, P- and B-frames, can be similarly applied to the initial learning of a foreign language.