Motion compensated blocking artefact repair on low bit rate block transform coded video

Coezijn, E.R.E.

Award date:
2005

TU/e technische universiteit eindhoven

Faculty of Electrical Engineering
Section Design Technology For Electronic Systems (ICS/ES)
ICS-ES 854

Master's Thesis

Motion compensated blocking artefact repair on low bit rate block transform coded video

E.R.E. Coezijn

Coach:	ing. E. Funke (Philips Research)
	ing. R.J. Schutten (Philips Research)
Supervisor:	prof.dr.ir. G. de Haan
Date:	June 2005

Motion compensated blocking artefact repair on low bit rate block transform coded video

E.R.E. Coezijn

April 2005

Master's thesis

Koninklijke Philips Electronics N.V.
Philips Consumer Electronics
BG Connected Displays - Innovation Lab
Supervisors: ing. E.P. Funke, ing. R.J. Schutten

Eindhoven University of Technology (TU/e)
Department of Electrical Engineering (E)
Information and Communication Systems group (ICS)
Electronic Systems research chair (ES)
Supervisor: prof. dr. ir. G. de Haan
Student ID: 400493

Abstract

Digital video is broadcast in an encoded form. For standard definition video, acceptable quality can be obtained using a bit rate of 6 megabits per second or higher. However, bandwidth constraints often lead broadcasters to reduce the bit rate. And especially for Internet Video, it is not uncommon to have bit rates as low as 300 kilobits per second.

Low bit rates lead to degradation in picture quality, commonly called in the literature *coding artefacts*. These include the blocking artefact, ringing, mosquito noise, etc. Philips Electronics N.V. develops algorithms to repair these artefacts, especially for the higher bit rates.

This thesis gives an overview of existing approaches for artefact repair to improve the quality of low bit rate Internet Video by reducing the blocking artefact especially, the most annoying visible degradation. Since prior work on this topic was mainly based on spatial post-processing of still images, special interest goes to algorithms that model the process of motion compensation (MC). Several MC approaches have been proposed and assessed both objectively and subjectively.

Conclusions

- Many existing algorithms model the process of block discrete cosine transform coding and quantization for still images, but do not take into account the process of motion compensation (MC) when extending traditional algorithms to digital video. Amongst existing algorithms, spatial adaptive filters are the most popular in the industry.
- MC methods are able to reduce the blocking artefact. Though compared to existing methods, whether or not using quantization information from the bit stream, the unconstrained MC methods developed in this study on average yield worse results than existing non-MC methods.
- The result of a MC filter strongly depends on the reliability of the motion vectors. With reliable vectors, an improvement in image fidelity as well as blocking artefact reduction can be achieved comparable to that of existing methods. Turning off the processing for non-reliable vectors preserves the blocking artefact. In those cases, reliable fall-back processing improves the result. Considering the complexity of MC methods, less complex non-motion compensated algorithms still provide the best alternative for blocking artefact repair. However, the fall-back median approach does give better subjective results in terms of ringing artefact and mosquito noise.
- The Peak Signal-to-Noise Ratio (PSNR) metric guards the post-processed image fidelity and has to be observed next to the General Block Impairment Metric (GBIM) metric. Subjective evaluation remains necessary for the assessment of algorithms on picture quality improvement.

Keywords

anisotropic diffusion, artefact repair, artifact repair, block-based transform coding, constrained least squares, corner outlier, block-edge impairment, de-blocking, compression, digital video, discrete cosine transform, grid noise, image enhancement, image restoration, maximum a posteriori, motion compensation, MPEG, post-processing, projection onto convex sets, staircase noise.

Preface

This thesis is the result of my graduation project, proposed by the Consumer Electronics division of the Royal Philips Electronics company, to obtain the degree Master of Science in Information Technology from the Department of Electrical Engineering at the Eindhoven University of Technology.

Innovations in digital video technology made digital video very popular among consumers through multimedia products. To make these innovations economically profitable, digital video is broadcast and stored in an encoded form that requires less information than the original, at the cost of picture quality. Post-processing of digital video, the subject of this report, is an acceptable means to improve visual perception of a distorted signal. Many implementations are available in hardware and as software plug-ins. Those who are interested in and want to keep up with the latest technologies, I recommend nternet forums like Neuron2's Video Processing Forum (`neuron2.net`) and Doom9's Forum (`forum.doom9.org`) as very interesting sources.

The research field of digital video processing involves many topics in electrical engineering and information technology. I wrote this document assuming that the reader is familiar with basic issues in electrical engineering, like linear algebra (vector algebra, matrix algebra, orthogonal vector spaces), information theory (entropy, information, bits, the source coding theorem, Pulse Code Modulation, variable length codes), and signal processing (Fourier transform, Finite and Infinite Impulse Response filters, sampling theorem). Knowledge of the image formation process, digital video, and digital video processing is desirable. The tutorials and resources on these topics in the literature and at the Internet are manifold. While writing this document, it was very tempting trying to be complete and to go into specific detail. Instead, I tried to be brief, providing many references to additional literature in the text.

It is common practice to use mathematics as a tool to describe algorithms. Though I agree that mathematics is helpful for an implementational approach, it is not always easy to grasp the author's idea visually. Therefore, I added illustrations where the mathematics might be unclear.

I owe gratitude to Gerard de Haan, professor at the Department of Electrical Engineering of the Eindhoven University of Technology and Research Fellow of the Video Processing and Visual Perception group at the Philips Research Laboratories in Eindhoven, who introduced and interested me in the field of video processing, put me in contact with Philips Consumer Electronics, and guided me through this project. Furthermore, I would like to thank Philips Consumer Electronics for giving me the opportunity to perform this project at their Connected Displays Innovation Laboratory in Eindhoven. Specially, I thank Age van Dalfsen, Eric Funke, Ingrid Heynderickx, Tanya Kwaaitaal, and Robert Jan Schutten for their support, comments, and reviews, and the many other people of the Innovation Lab and the Video Processing and Visual Perception group, being very helpful in providing me information, software, and other means.

Etienne Coezijn,
Eindhoven, April 2005.

Contents

Chapter 1

Introduction

Assisting the human visual perception, images and image sequences (i.e. *video*) are powerful media for communication, information transfer, and artistic expression. The significance of images since the very beginning of human history pushed the innovations in video technology.

As silicon technology improved, *digital* storage and transmission of image sequences, or *digital video*, became a rapidly developing technology [55]. Digital video has many advantages over the mature analogue techniques. For example, digital video is easier to store and to transmit, and can be duplicated without loss of quality, as opposed to analogue video. Moreover, digital video has the ability to provide pictures with better quality, which can more easily be accessed and manipulated. Other means for transmission and storage became available, enabling new applications such as video conferencing, video telephony, video-on-demand, Digital Video Broadcast (DVB), and Internet television (TV over IP). One of the most popular applications of digital video are those in *multimedia* products such as the multimedia PC or TV, which integrate digital video interactively with other media like sound, animation, graphics, pictures, and text.

Innovations in digital video technology lead to the use of infrastructures (Digital Video Broadcast, Internet television, etc.) and storage media (Compact Disks, Digital Versatile Disks, etc.) for digital transmission and storage of image sequences, that made digital video very popular among consumers through multimedia products. Recent innovations, for example high-definition DVD (HD-DVD) and Blu-ray disks, meet the growing demand for capacity. Other innovations, like Advanced Video Coding (AVC), aim at more efficient coding techniques while preserving the perceived picture quality. However, bandwidth and storage capacity are scarce, and broadcasters try to transmit as many video channels per multiplex as possible to make these techniques economically profitable. Therefore, digital video is broadcast and stored in an encoded form that requires less information (bits) than the original, at the cost of picture quality. For standard definition video, acceptable quality can be obtained using a bit rate of 6 megabits per second ($6Mbit/s$) or higher [102]. Especially for Internet Video, it is not uncommon to have bit rates as low as 300 kilobits per second ($300kbit/s$).

Severe bit rate reduction introduces visual degradation of picture quality, commonly called in the literature *coding artefacts*. These include blocking artefacts, ringing noise, mosquito noise, etc.

Philips Electronics develops algorithms to repair these artefacts, especially for the higher bit rates. With the advent of Internet Video, low bit rate video enters multimedia devices and with the current high-resolution display systems, these artefacts become even more visible. Since picture quality is an important selling point for multimedia products, there is a strong need for methods to reduce these impairments.

(a) (b)

Figure 1.1: A 352 × 288 sample frame from a tennis game sequence (a), MPEG-2 coded at 500*kbps* and decoded. (b) is a detail of the tennis player's face, showing extreme blocking artefact due to the fast moving tennis player and the panning background.

1.1 Scope and outline

Common compression techniques are founded on a coding scheme that divides the image into blocks of typically 8 by 8 samples. The lossy coding of these blocks causes the loss of the interdependency between adjacent blocks, that manifests as discontinuities along the block borders (*blocking*). The *human visual system* (HVS) is very susceptible to these discontinuities. Especially fast moving sequences suffer from this annoying *blocking artefact*. Efforts have been made to reduce the blocking effect at the encoder side, for example, using another encoder scheme with overlapped blocks (the *lapped orthogonal transform* [54]), or, more recently, using a reconstruction filter in both encoder and decoder that smoothes the image along the block boundaries (in Advanced Video Coding [78]). However, most compression schemes currently in use are based on a non-overlapping block coding scheme. Post-processing of the decoded video stream gives acceptable results [80]. This project focuses on the removal of the most visible degradation for low bit rate, block transform coded video, the *blocking artefact*. Figure 1.1 shows an extreme example of the blocking artefact. Most algorithms for de-blocking are designed and evaluated for greyscale video only, as done throughout this project. These algorithms can easily be modified for coloured sequences. Furthermore, we will limit to de-interlaced or progressive-scanned sequences, since interlaced video complicates video processing [23, page 145]. This report first gives an overview of existing approaches for de-blocking found in the literature and in the industry. Since prior work on this topic was mainly based on spatial post-processing of still (two-dimensional) images, special interest goes to algorithms that operate in the temporal domain (a third dimension) as well. We propose several methods that model the process of motion estimation and compensation used in digital video coding. The motion compensated approaches are then assessed both objectively and subjectively.

This report is organized as follows. Chapter 2 is an introductory text on lossy video coding and associated artefacts due to the quantization process. In this chapter, the necessary mathematics is defined for the remainder of this report. Chapter 3 is an overview of different spatial and temporal algorithms existing in the literature, accompanied by some examples. In Chapter 4, some motion compensated techniques are proposed that are evaluated in Chapters 5 and 6.

1.2 Mathematical conventions

Mathematics is a powerful means to show algorithms in an implementational approach. We will use some conventions on mathematical notations that are more appropriate for image and video processing. The scope of these conventions encompasses the entire report, unless explicitly specified otherwise.

Mathematical sets are written in calligraphic capitals $\mathcal{A}$, with the size of the set (the number of elements), denoted as $|\mathcal{A}|$. Vectors are always column vectors printed in lowercase, with their respective elements starting with index 0, such that an N point vector $\vec{v}$ is defined as

$$\vec{v} = (v_0, v_1, \ldots, v_{N-1})^T, \tag{1.1}$$

with the transpose operator denoted with the superscript ^{T}.

Digital video deals with a finite, three-dimensional space-time lattice that can be addressed by integer coordinates. The generic variables x, y and i, j are used as indices in the spatial domain (and as we see later, in the wavelet domain), whereas u and v are used in the transform domain. An integer n describes the position in the temporal domain.

Variables that are subject to modification are indicated with a hat: $\hat{F}$.

Functions have their normal mathematical meaning. We explicitly define the *sign* function:

$$\text{sign}(x) = \begin{cases} -1, \text{ if } x < 0, \\ 0 \ \ , \text{ if } x = 0, \text{ and} \\ 1 \ \ , \text{ if } x > 0. \end{cases} \tag{1.2}$$

The *clip*$[x]$ function clips x between two values $a < b$:

$$\text{clip}[x]_a^b = \begin{cases} a, \text{ if } x < a, \\ x, \text{ if } a \leq x \leq b, \text{ and} \\ b, \text{ if } x > b. \end{cases} \tag{1.3}$$

We further define the *argmax* function,

$$\underset{\vec{v}}{\text{argmax}}\{f(\vec{v})\}, \tag{1.4}$$

which finds the vector $\vec{v}$ that maximizes the vector function $f(\vec{v})$. The *argmin* function is defined analogous. Lastly, we define the 3-point *median* filter which selects the centre value of the ordered set of three (the least extreme value):

$$\text{median}\,\{a, b, c\} = \begin{cases} a, \text{ if } b < a < c \text{ or } c < a < b, \\ b, \text{ if } a < b < c \text{ or } c < b < a, \\ c, \text{ otherwise.} \end{cases} \tag{1.5}$$

Chapter 2

Video coding and artefacts

This chapter provides an introduction to digital video, compression techniques and their associated artefacts. Special interest goes to the MPEG standards, popular standards used in for example DVDs and Internet Video. Moreover, MPEG is the basis for many video codecs currently in use, suffering to certain extend from the same artefacts.

2.1 Digital video principles

A digital video stream can be thought of as a chronological ordered sequence of images or *frames*, sampled in time. Each frame is a mapping of a three-dimensional real world scene onto a finite two-dimensional grid. A point on this grid describes a *pixel*, or picture element, which is the basic element of a picture, a digital representation of the continuous real world. Note that a pixel has no dimensions, although we represent pixels as squares in figures. If we assume an orthogonal, equidistant sampling grid, normalized to the sampling distance in both horizontal and vertical directions, we can address a pixel by its integer position

$$\vec{x} = \begin{pmatrix} x \\ y \end{pmatrix}, \tag{2.1}$$

where x and y denote the horizontal and the vertical offset respectively, relative to the image origin, which we will define at the upper-left corner of an image (see Figure 2.1). An image with N columns and M rows is a matrix of samples defined by the set $\mathfrak{I}$ of pixel positions

$$\mathfrak{I} = \left\{ \vec{x} \in \mathbb{N}^2 \mid 0 \leq x < N \wedge 0 \leq y < M \right\}. \tag{2.2}$$

Each pixel with spatial position $\vec{x}$ and temporal position n is associated with at least one sample value $F(\vec{x}, n)$, limited in both time and area, of a physical quantity called *luminance*[1] [25]. Luminance is the amount of incident luminous energy (i.e. *light*) per second at a surface.

In colour images, a pixel consists of multiple samples, each being a vector coordinate in a *colour space*. Any colour can be synthesized by a linear combination of these vectors called *colour primaries*. Since the human eye has receptors for Red, Green, and Blue spectral response, these three colours are often used as primaries (*RGB colour space*). Another colour space more suitable for digital video, as we will see later, is known as *YCrCb*, where Y represents the luminance, and *Cr* and *Cb* are the *colour difference signals* for red and blue *chrominance* respectively. A colour image in digital video consists of three two-dimensional matrices, one for luminance, and two for chrominance. Unless explicitly specified otherwise, we will associate the term pixel with a single sample of the luminance component.

[1] 'Luminance' is the quantity related to the visible part of the electromagnetic spectrum. Generally, the term 'irradiance' refers to the entire electromagnetic spectrum.

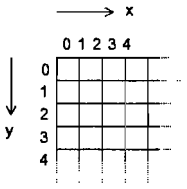


Figure 2.1: The integer pixel grid, its origin in the upper-left corner of the image. Though pixels are scalars, they are represented as squares, bounded by the grid lines.

Conventionally, broadcasted picture sequences (*television*) were transmitted in two main formats, one at a resolution of 625 scanning lines (576 active display lines) at $50Hz$ (*PAL* or *SECAM*[2]), and one at a resolution of 525 scanning lines (480 active display lines) at $60Hz$ (*NTSC*[3]). These two main television formats are referred to as *standard definition (SD) television*. Early broadcast limitations forced broadcasters to transmit pictures at a rate of 25 respectively 30 frames per second, potentially causing large area flicker. To avoid this, both SD formats were and still are broadcast *interlaced*, i.e. even and odd lines of the image are transmitted as two temporally separated *fields* sequentially. Interlaced video, however, complicates video processing [23, page 145]. Therefore, we will limit to *progressive* (frame-based) and *de-interlaced* video. The process of de-interlacing is a key technology in video processing. A complete overview of de-interlacing methods can be found in [23, pages 145–174].

The high sampling rate of the video camera results in high rate bit streams. If no compression were applied, raw, 8-bit SD video data would require a bandwidth of 720 pixels $\times$ 576 lines $\times$ 25 frames per second $\times$ 3 colour primaries $\times$ 8 bits per sample = $248Mbit/s$, equivalent to $112GB$ (Gigabytes) for an hour of video, far too demanding for common transmission channels, magnetic and optical storage devices such as Compact Disks (CDs), Digital Versatile Disks (DVDs), and Blu-ray Disks (BDs), which hold up to an approximate maximum of $0.7GB$, $9.4GB$, and $54GB$ respectively. These limited bandwidth and capacity constraints force bit rates to be reduced in order to make digital video economically profitable. Bit rate reduction techniques, or *compression techniques*, are means to use the transmission channel bandwidth or storage device capacity more efficiently by using a format that requires fewer bits than the original. Naturally, a video signal contains a lot of redundant (mutually dependent) information. It is the purpose of a compressor or *encoder* to remove as much as redundancy without visual degradation of picture quality. Typically, both *lossless* and *lossy compression* are used to minimize bit rates. Lossless compression is an invertible process that involves a compressor to encode raw video data to a data format that requires fewer bits, from which a *decoder* or de-compressor can reconstruct an exact replica of the original data. This as opposed to lossy compression, where the data is approximated based on the spatial and temporal redundancy. The lossy compression techniques introduce visual degradation of the image quality, called *coding artefacts*. For a better understanding of the origin of artefacts, the next sections give a brief overview of compression techniques commonly used in video coding.

Note that next to compression techniques, also *channel coding techniques* are required for reliable transmission over a physical, lossy channel. These techniques include encryption, error control, packetizing, multiplexing, and modulation. Transmission over a lossy channel can cause *transmission artefacts*, which are out of the scope of this work. Channel coding adds to the bandwidth required for transmission, emphasizing the need for compression.

[2]PAL (Phase Alternation Line) and SECAM (Système Électronique Couleur Avec Mémoire) are European television standards.

[3]NTCS (National Television Systems Committee) is the United States television standard of the Electronic Industries Association

2.2 Compression techniques

There are several ways to categorize compression techniques. We distinguished lossless and lossy compression schemes earlier. Another classification distinguishes between *entropy coding* and *source coding* [91]. Entropy is a measure, initially developed for thermodynamics, used in information theory, for the amount of uncertainty in a signal. Entropy coding does not use the characteristics of the source, but instead it uses the statistics of the data itself. Entropy coding is lossless. Source coding can be either lossless or lossy. The encoder uses its prior knowledge of the source to remove redundancy. Manning [55] proposes the following classification, which we will adopt here as well: *general-purpose compression* that can be applied to any kind of data, *inter frame compression*, using spatial redundancy in still images, and *intra frame compression*, using temporal redundancy in image sequences.

2.2.1 General-purpose compression techniques

General-purpose compression techniques exploit the fact that most data has redundant *symbols*. The symbols in a video stream are in fact the pixel sample values mentioned earlier. The techniques discussed below, used in video compression, are in general lossless.

Run length encoding

Run length encoding is useful where long runs of identical symbols occur. These runs can be replaced by pairs of numbers each holding the repetition count or *run length* and the symbol value. Though natural video does generally not contain long runs of identical samples in the spatial domain, a variation on this technique for the compression of long runs of zero values called *zero run length coding*, is very useful in the frequency domain, as clarified later this chapter. Run length encoding is a kind of *entropy coding*.

Differential or relative encoding

When data samples are slowly varying, one can code the difference with respect to the previous value instead of the values themselves (source coding). Difference values (a sign and a magnitude) often have a smaller range and can be assigned fewer code bits than the original sample value. This technique is especially suited for video compression, where there is a lot spatial and temporal consistency. The technique is also referred to as *predictive* or *relative coding*. When the difference values exceed the maximum range, the method becomes lossy.

Huffman coding

Huffman coding is an entropy coding technique. A *statistical modelling* algorithm analyses the data and assigns probabilities to the symbols. Next, the encoder produces variable length codes (VLCs) for these probabilities such that fewer code bits are assigned to more likely symbols and more code bits are assigned to less likely symbols. The Huffman coding is optimal when the symbol probabilities are integer powers of two.

Other compression techniques

There are many other, less used data compression techniques. For example, *arithmetic coding* is a more efficient and complex scheme similar to Huffman coding, that is also optimal for symbol probabilities not equal to integer powers of two. The *Advanced Video Coding* (AVC) standard defines *Context-based Adaptive Binary Arithmetic Coding* (CABAC), which chooses a *context model* depending on the probability distribution of the data [76].

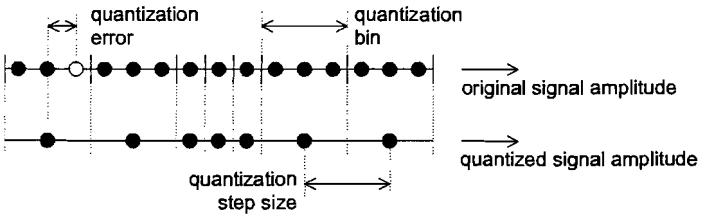


Figure 2.2: Quantization example. The line above represents a sample space as a line with possible sample values and their quantization bins. The open dot is a sample at a certain spatial and temporal position. The line below shows the same sample space with the quantized samples, typically in the middle of the corresponding bin above. In this example, the quantization step sizes are not equal, making the quantization non-linear.

2.2.2 Intra frame compression

Intra frame coding is developed to compress still image data. It uses the *spatial redundancy* in a signal. The techniques are generally lossy and use the characteristics of the source.

Sub-sampling

Sub-sampling is the process of reducing the bit rate by simply discarding information. Digital video formats exploit the fact that the *human visual system* (HVS) is less sensitive for detail in chrominance than in luminance. Video compression schemes use this *psychovisual redundancy* to drop every other chrominance sample, both in horizontal and vertical direction [29].

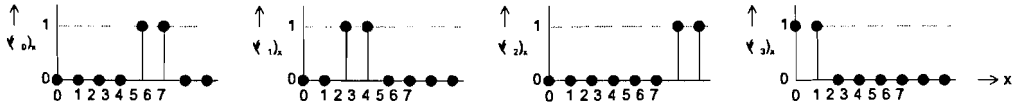
Quantization

Intrinsic to digital video is the quantization process. In order to describe a physical quantity with a finite number of binary digits (*bits*), a range of continuous values has to be mapped onto a finite number of discrete quantization levels. This round off process gives rise to errors in the form of quantization noise[4]. The image acquisition process itself is, therefore, a kind of lossy coding. Instead of reducing the number of samples with sub-sampling, an encoder can choose to reduce the number of bits per pixel, thus reducing the number of quantization levels and increasing the quantization step sizes. Figure 2.2 shows this effect. If the quantization steps are large enough for the quantization noise to be annoying, the quantization is called coarse. Quantization, or *bit depth reduction*, is, of course, lossy. From a mathematical point of view, the quantization process is a many-to-one mapping.

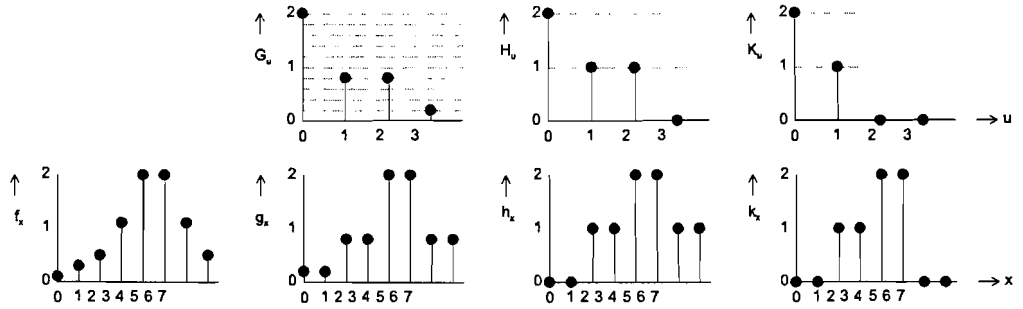
Transform coding

Transform coding removes the correlation in a signal and packs the signal energy in as few transform coefficients as possible. To understand the theory behind transform coding, we will observe the one-dimensional case first. Transform coding is the rotation of a set of sample vectors with length K from the original K-dimensional space to a set of coefficient vectors of length L in another L-dimensional transform space, with $L \leq K$. If $L < K$, the compression is obvious. One can choose L basis vectors of the L-dimensional transform space such that the first basis vector (the first *principal component*) contributes the most to the variance (i.e. the signal energy) of all sample vectors in the set, the second basis vector (the second principal component) contributes the second most, etc. In this way, later principal components are less important and can be discarded, so achieving compression. The optimal set of basis vectors, found with a method called *Principal*

[4]Quantization noise is just one of the many disturbances that occur in the image acquisition process. Van der Heijden [25, pages 52–53] enumerates many other noise sources.



(a) Example basis vectors



(b) Example signal vectors with their corresponding transform signals above, showing the quantization levels

Figure 2.3: Transform coding example. $f(x)$ is the original signal, $g(x)$ is the reconstruction from the inverse transform of $G(u)$, $h(x)$ is the reconstruction of $H(u)$ after quantization of $G(u)$, and $k(x)$ is the reconstruction of $K(u)$ after coarse quantization of $G(u)$ (see text).

Component Analysis (PCA) [82], are in fact the *eigenvectors* of the covariance matrix[5], ordered by their corresponding *eigenvalues* in descending order. The corresponding transformation is also known as the *Karhunen-Loève transform* or *Hotelling transform*. The calculation of the eigenvectors is computationally expensive and, therefore, more general basis vectors such as cosine waves (*discrete cosine transform* or DCT), both sine and cosine waves (*discrete Fourier transform* or DFT), wavelets (*discrete wavelet transform* or DWT), or square waves (*Walsh-Hadamard transform*) are used for transform coding. The sample vectors with length K have to be long enough to achieve acceptable compression and they have to be short enough to use the spatial redundancy. Instead of one-dimensional vectors, two-dimensional blocks are used for images. Common block sizes in transform coding are 4×4 and 8×8 pixels.

Transform coding, that is in itself lossless, is usually followed by quantization of the transform coefficients. To illustrate the concept of transform coding with quantization, consider a set $\{\vec{v}_0, \vec{v}_1, \vec{v}_2, \vec{v}_3\}$ ($L = 4$) of basis functions in Figure 2.3(a) and a sample vector $\vec{f}$ (with $K = 8$) in Figure 2.3(b) to be encoded. The original signal $\vec{f}$ can be approximated by a linear combination of the basis vectors: 2 times the vector $\vec{v}_0$, $0.75 \cdot \vec{v}_1$, $0.75 \cdot \vec{v}_2$, and $0.25 \cdot \vec{v}_3$. The results of transform is the vector of coefficients $\vec{G} = (2,\ 0.75,\ 0.75,\ 0.25)^T$. The coefficients can, for example, be represented by a number in the range from 0 to 2 with equal quantization step sizes of 0.25, making 9 quantization bins in total as shown in the figure. The encoder has to encode for 9 different values. Reconstructing the original signal given the vector of coefficients and the basis vectors (inverse transform) results in the signal $\vec{g}$. Now suppose a quantization scheme with only three quantization bins, 0, 1 and 2, achieving a compression of a factor three. The vector of rounded coefficients is now $\vec{H} = (2,\ 1,\ 1,\ 0)^T$, yielding the reconstruction $\vec{h}$. Since the later components contribute less to the signal vector's energy, the coefficients for these components can be quantized more, or even be zeroed (which is the ultimate form of quantization leaving just one quantization bin which is represented by the value zero). Coarse quantization of the vector, discarding

[5]The covariance matrix is a matrix describing the variances between each pair of (random) variables, in this case the K sample values, for all available observations, i.e. all vectors in the set of sample vectors.

the third and the fourth coefficient (thus achieving another compression factor of two), gives the vector $\vec{K} = (2,\ 1,\ 0,\ 0)^T$ with the reconstructed signal $\vec{k}$. Each step, transform, quantization, and coarse quantization introduces a further deviation from the original signal.

Other intra frame compression techniques

Other techniques have been investigated to remove spatial redundancy in an image. *Vector quantization* replaces a vector, or the in two-dimensional case, a block of image data with its best match to a predefined (source coding) or dynamically constructed (entropy coding) code-book of images. Another method called *fractal coding* mathematically describes an image in terms of translation, scaling, rotation, and mirroring of repeating (self-similar) patterns that occur in a natural image. Fractal encoding is extremely computationally expensive, but can yield high compression rates. *Intra-prediction coding* is a technique used in AVC that uses the information of previously decoded blocks of samples to predict the current block of samples, given a *prediction mode* [77]. The prediction mode indicates the direction of the prediction such that the decoder can calculate the interpolated pixel values as a weighted average of the previously decoded pixels. The compression algorithms mentioned here return an estimation of the original data, making them lossy.

2.2.3 Inter frame compression

Inter frame coding profits from the *temporal redundancy* in consecutive images. These source coding techniques are in general lossy. Because there is high consistency between pixels at the same spatial position of two temporal consecutive frames, differential coding is one way to code the differences between the two. However, if there is a lot of motion between frames, for example during a tilt, pan or zoom of the camera, all pixels require update, making the technique inefficient. Other techniques are discussed below.

Sub-sampling

Like sub-sampling in the spatial domain, one can also apply sub-sampling in the temporal domain. Simply removing pictures from the stream reduces the bit rate. The missing pictures can be restored at the decoder by interpolation (*Picture rate conversion*, [23, pages 99–144]). The interlacing process mentioned earlier is in fact a kind of sub-sampling in both the spatial and the temporal domain.

Motion estimation

Motion estimation is a lossy technique that estimates (at encoder side) and predicts (at decoder side) an image by reusing data from past and, if bi-directional estimation is used, future reference images. Note that bi-directional motion estimation requires a future reference image to be available which implies that the images have to be transmitted in a different order than the order in which they will be displayed. Motion estimation can be pixel based, (hierarchical) block based, or object based [23]. The vector that describes the displacement of a pixel, block or object from a spatial position at a given temporal location to another spatial and temporal location is called *displacement vector* or *motion vector*. For a pixel, the motion vector is defined mathematically as:

$$\vec{D}(\vec{x}, n) = \begin{pmatrix} D_x(\vec{x}, n) \\ D_y(\vec{x}, n) \end{pmatrix}, \tag{2.3}$$

where $\vec{x}$ is the spatial position on an integer pixel grid and n is the temporal position for which the vector is valid. D_x and D_y are the displacements in the horizontal and vertical direction respectively, typically with sub-pixel accuracy, i.e. $D_x, D_y \in \mathbb{Q}$. We assume the frame rate to be constant throughout a sequence, so we can assign integer temporal positions to each consecutive frame, normalizing the time by the sampling time.

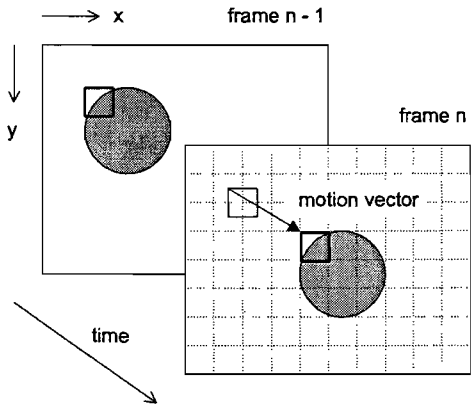


Figure 2.4: Block-based motion estimation example. The figure shows the current frame n and its previous frame n - *1* with an object (grey) moving in a static background (white). The motion estimation block grid is shown as dotted lines. The current frame shows the displacement of a motion vector block with respect to the previous frame.

Throughout this report, we will use the convention that all vector quantities are positive, both spatially and temporally, for the (three-dimensional) coordinate system depicted in Figure 2.4. Because the prediction will generally not be the same as the original, the technique is lossy. The difference between the prediction and the original is called the *prediction error* ε, defined for each pixel as:

$$\varepsilon(\vec{x}, n) = F_o(\vec{x}, n) - F_p(\vec{x}, n), \tag{2.4}$$

where F_o is the original sample value and F_p is the sample value of the motion compensated prediction. If the motion compensated prediction is based on reference frame m with sample value F, this prediction is defined as

$$F_p(\vec{x}, n) = F(\vec{x}_m, m), \ \vec{x}_m = \vec{x} - (n - m) \cdot \vec{D}(\vec{x}, n). \tag{2.5}$$

Since the vector $\vec{x}_m$ will generally not be on the integer pixel grid, we will use *bilinear interpolation* to estimate the sample value for non-integer positions. For that purpose, let us define

$$\begin{aligned} &\vec{x}_m = \begin{pmatrix} x_{int} + x_{frac} \\ y_{int} + y_{frac} \end{pmatrix}, \text{ with} \\ &x_{int} = \lfloor x_m \rfloor, \ y_{int} = \lfloor y_m \rfloor, \ x_{frac} = x_m - x_{int}, \text{ and } y_{frac} = y_m - y_{int}, \end{aligned} \tag{2.6}$$

then the bilinear interpolated value is given by

$$\begin{aligned} F_p(\vec{x}, n) = {} & (1 - x_{frac}) \cdot (1 - y_{frac}) \cdot F((x_{int}, \ y_{int})^T, m) + \\ & x_{frac} \cdot (1 - y_{frac}) \cdot F((x_{int} + 1, y_{int})^T, m) + \\ & (1 - x_{frac}) \cdot y_{frac} \cdot F((x_{int}, \ y_{int} + 1)^T, m) + \\ & x_{frac} \cdot y_{frac} \cdot F((x_{int} + 1, y_{int} + 1)^T, m). \end{aligned} \tag{2.7}$$

2.3 Compression standards

In the development of common compression methods for digital video applications, standardization is a very important issue where it comes to interoperability of these applications [88]. In 1982, the *International Organization for Standardization* (ISO) and the *International Electrotechnical Commission* (IEC) started a working group for the development of digital (multiplexed audio and) video standards for compression, decompression, processing, and representation, the *Moving*

Pictures Experts Group (MPEG) [2]. The work of the group resulted in several standards for video compression, based on the JPEG standard for photographic images and the H.261 standard for video conferencing [59]. MPEG-1 (ISO/IEC 11172) is a standard accepted in 1991 and intended for the efficient storage of progressive SD video with VHS (Video Home System) picture quality at bit rates up to $1.5Mbit/s$. Video CD (VCD) is an application of MPEG-1 for encoding movies on two CDs. MPEG-2 (ISO/IEC 13818) supports interlaced video at different spatial resolutions including high-definition television (HDTV), making it suitable for digital broadcast television. Its application can be found in digital television set top boxes, Super Video CD (SVCD), and DVDs. The standard was accepted in 1994. MPEG-4 (ISO/IEC 14496) supports coding for individual objects, achieving high compression. It is intended for Internet and multimedia applications. It is suitable for both low and high bit rates and became an International Standard in 1999. A new state-of-the-art codec, officially called the *Advanced Video Coding* standard (AVC), was finalized in May 2003 by the Video Coding Experts Group *VCEG* from the International Telecommunication Union (ITU) and the ISO/IEC MPEG group, joint in the *Joint Video Team* (JVT). AVC is also known under its project name, H.264, H.26L, ISO/IEC MPEG-4 Advanced Video Coding, or MPEG-4 Part 10 [24]. AVC is not yet widely used because it suffers from lack of interoperability: it comes in different *container formats*[6].

For MPEG-2, the most popular standard used for DVDs, a bandwidth of 3 to $8Mbit/s$ is available [109]. Streaming Internet Video has to settle for less bandwidth, setting developers of coder-decoder schemes (*codecs*) the aim to deliver DVD quality for less than a single megabit per second. Many of such codecs are available, all promising DVD quality. Recent propriety codecs like On2 Truemotion VP7, RealVideo 10, Sorenson Video 3 Pro, and Windows Media Video 9 can achieve AVC video quality or better. Other codecs like Dicas mpegable, DivX 5.2.1, Apple QuickTime MPEG-4, Sorenson MPEG-4 Pro, and XviD are conformable to the MPEG-4 *Simple profile* standard. Just like MPEG-2, *profiles* and *levels* are defined that restrict the bit stream's syntax and semantics respectively, so decoders can limit to a range of specific bit streams. MPEG-4 (Advanced) Simple Profile (MPEG-4 ASP) and MPEG-2 Main Profile at Main Level (MPEG-2 MP@ML) are the most popular combination of constraints. The coding system described by the MPEG standards share the same principles with many codecs in use.

2.3.1 MPEG coding

This section introduces the MPEG video coding system, which is the basis for many DCT-based codecs. MPEG compression is a hybrid technique in that it is mainly based on two compression techniques discussed earlier: spatial redundancy is reduced by a block DCT-based (BDCT) coding system and temporal redundancy is minimized using block-based motion estimation. For specific details, refer to the standards of MPEG-2 [26] or MPEG-4 [27].

Figure 2.5 shows the global structure of the MPEG-2 MP@ML hierarchy, together with some relevant elements, discussed in this report. Note that most layers begin with a start code that marks the beginning of a specific syntactic sequence of layer-specific parameters, including image size, frame rate, quantization parameters, and required buffer sizes. Furthermore, start codes provide an entry point for the decoder for random access or for recovery after bit stream errors. An MPEG video sequence is divided into *groups of pictures*[7], that provide entry points for the decoder at a random point, given the *time stamp* that is encoded within each group. Each group holds an intra coded picture, or *I-picture*, that acts as a reference frame, or key frame, from which frames can be predicted using block-based motion compensation. Next to I-pictures, an MPEG stream contains forward predictive coded pictures (*P-pictures*) and bi-directionally predictive coded pictures (*B-pictures*). P-pictures are predicted with reference to previous decoded I or P-pictures. B-pictures can contain references to previous I or P-pictures, future I or P-pictures, or both. In case that both forward and backward prediction is present, the reconstructed motion estimated block is simply the pointwise average of the two.

[6] A container format is a wrapper around the compressed data that specifies how audio, video, metadata, etc. are interleaved in a bit stream.

[7] Another word for 'picture' in Recommendations and International Standards is 'Video Object Plane' (VOP).

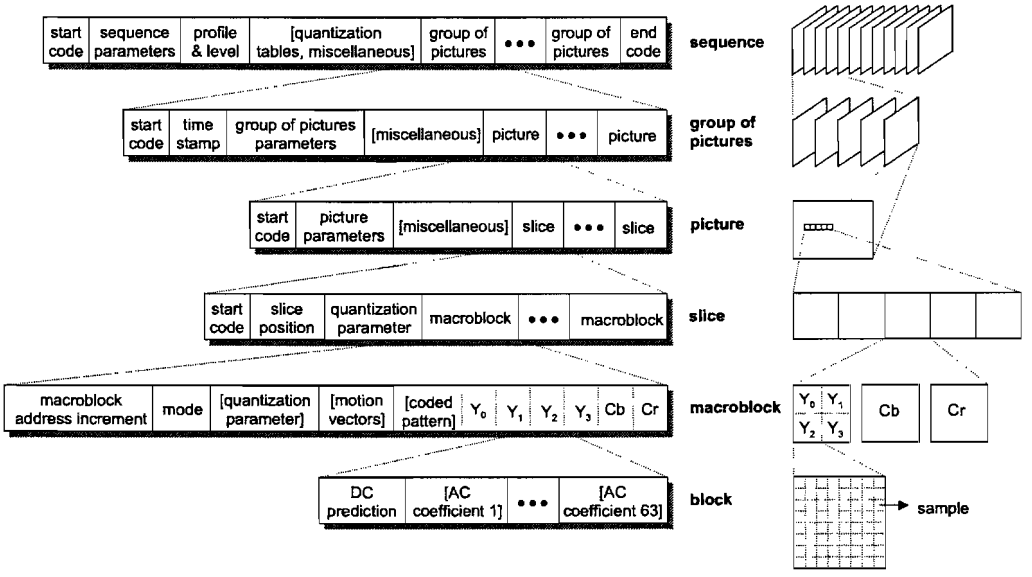


Figure 2.5: Hierarchical structure of an MPEG-2 video sequence, from top to bottom. The structures on the left show the bit stream layers, the drawings on the right visualize the corresponding elements of the physical video stream. Elements between square brackets are optional, depending on parameters.

A single picture is divided into rows of 16 × 16 non-overlapping sample blocks called *macroblocks*, that are organized in non-overlapping *slices* in order to spatially localize errors due to lossy, packetized transmission. Figure 2.6 shows a picture sequence, each picture divided into macroblocks. Depending on the type of picture, there exist several macroblock *modes*. Forward or backward predicted macroblocks contain motion vectors to past or future reference frames respectively. Each 16 × 16 macroblock consists of four 8 × 8 luminance *blocks* and optionally several chrominance blocks, depending on the chrominance sub-sampling scheme used (see Figure 2.5). As we only observe the luminance component, we define a block $\mathcal{B}_{\vec{x}_b}$ as the set of 8 × 8 pixel positions, characterized by the position of its upper left pixel $\vec{x}_b$, as

$$\forall\ \vec{x}_b = \begin{pmatrix} x_b \\ y_b \end{pmatrix} \in \mathcal{I},\ \exists\ \mathcal{B}_{\vec{x}_b} = \{\vec{x} \in \mathcal{I} \mid x_b \leq x < x_b + 8 \wedge y_b \leq y < y_b + 8\}, \tag{2.8}$$

with the luminance sample values for each block

$$b_{\vec{x}_b,n}(x - x_b, y - y_b) = F(\vec{x}, n),\ \forall \vec{x} \in \mathcal{B}_{\vec{x}_b}. \tag{2.9}$$

The non-overlapping coding blocks are located at pixel positions that are integer multiples of 8. We define this set of pixels formally as

$$\mathcal{B} = \{\vec{x} \in \mathcal{I} \mid x \bmod 8 = 0 \wedge y \bmod 8 = 0\}, \tag{2.10}$$

so that $x_b \in \mathcal{B}$. Block-based motion estimation in MPEG-2 or MPEG-4 is performed on each macroblock or each block. The motion vector for a block $\vec{D}(\vec{x}_b, n)$ is identical for all pixels in that block, so each pixel has an identical displacement:

$$\vec{D}(\vec{x}, n) = \vec{D}(\vec{x}_b, n),\ \forall \vec{x} \in \mathcal{B}_{\vec{x}_b}. \tag{2.11}$$

The same can be defined for macroblock-based motion estimation. Because the motion compensated prediction is generally not the same as the original, the encoder can choose to code the residual errors ε (cf. Equation 2.4). Residual coding is a kind of differential coding.

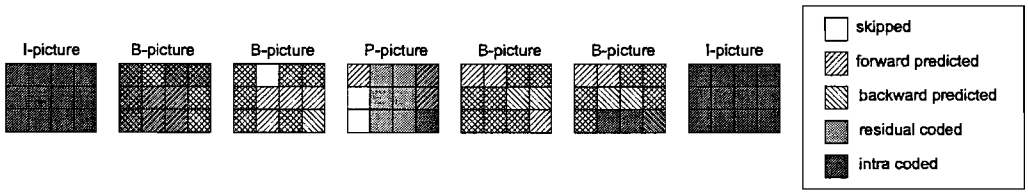


Figure 2.6: Display order sequence of pictures showing macroblocks with different modes.

2.3.2 Block discrete cosine transform

Intra coded blocks and residual coded blocks are transform coded by the *two-dimensional discrete cosine transform* in order to concentrate the signal energy in the transform coefficients associated with low-frequency basis images. The one-dimensional DCT of an N-dimensional sample vector $\vec{f}$ is in fact the real part of the Discrete Fourier Transform. The N elements of the coefficient vector $\vec{F}$ can be found with the equation:

$$F_u = \alpha(u)\sqrt{\frac{2}{N}}\sum_{x=0}^{N-1} f_x p_u(x), \text{ with } u, x, N \in \mathbb{N}, \tag{2.12}$$

where $\alpha(u)$ is a scaling factor to normalize the basis waveforms

$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{2}}, \text{ for } u = 0, \\ 1 \quad , \text{ for } u \neq 0, \end{cases} \tag{2.13}$$

and $p_u(x)$ are the orthogonal DCT basis functions

$$p_u(x) = \cos\left(\frac{(2x+1)u\pi}{2N}\right), \tag{2.14}$$

for $N = 8$ shown in Figure 2.7. The reconstruction of the N signal sample vector $\vec{f}$ from the transform coefficients vector $\vec{F}$ follows from the *Inverse Discrete Cosine Transform* (IDCT):

$$f_x = \sqrt{\frac{2}{N}}\sum_{u=0}^{N-1} \alpha(u) F_u p_u(x) = \sqrt{\frac{1}{N}} F_0 + \sqrt{\frac{2}{N}}\sum_{u=1}^{N-1} F_u p_u(x). \tag{2.15}$$

Equation 2.15 shows the transform coefficients split up into a *DC component* F_0 and *AC components* F_u for $u \neq 0$. The two-dimensional transform and inverse transform for a block $b_{\vec{x}_b,n}$ with N columns and M rows simply follow from the one-dimensional case and are defined in Equations 2.16 and 2.17, dropping the dependencies on $\vec{x}$ and n.

$$B(u,v) = \alpha(u)\alpha(v)\sqrt{\frac{2}{N}}\sqrt{\frac{2}{M}}\sum_{x=0}^{N-1}\sum_{y=0}^{M-1} b(x,y)\cos\left(\frac{(2x+1)u\pi}{2N}\right)\cos\left(\frac{(2y+1)v\pi}{2M}\right). \tag{2.16}$$

$$b(x,y) = \sqrt{\frac{2}{N}}\sqrt{\frac{2}{M}}\sum_{u=0}^{N-1}\sum_{v=0}^{M-1} \alpha(u)\alpha(v) B(u,v)\cos\left(\frac{(2x+1)u\pi}{2N}\right)\cos\left(\frac{(2y+1)v\pi}{2M}\right). \tag{2.17}$$

Figure 2.8(a) shows the basis functions for the 8×8 (i.e. $N = 8, M = 8$) DCT for all values of u and v. Note the similarity between these two-dimensional DCT basis images and the one-dimensional functions (Figure 2.7). The top-leftmost coefficient $B(0,0)$ is called the *DC coefficient*, the other 63 coefficients are *AC coefficients*.

The DCT of an 8×8 block $b_{\vec{x}_b,n}(i,j)$ of data results in a matrix of 8×8 DCT coefficients $B_{\vec{x}_b,n}(u,v)$ from which the original block can be reconstructed via IDCT. When we arrange the

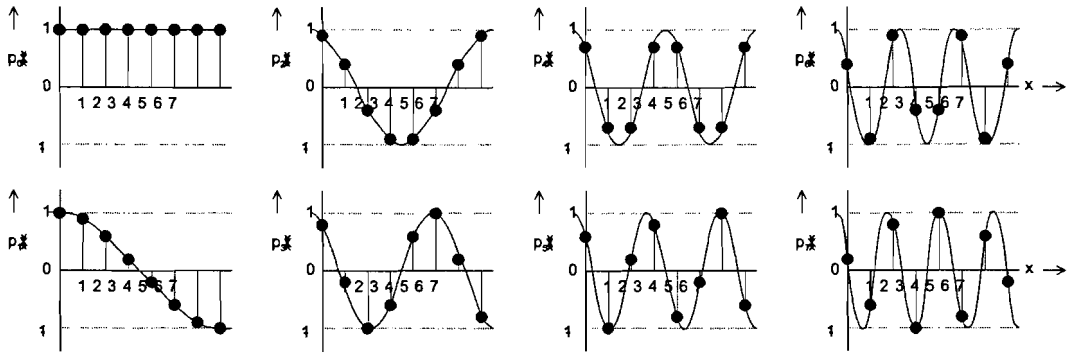
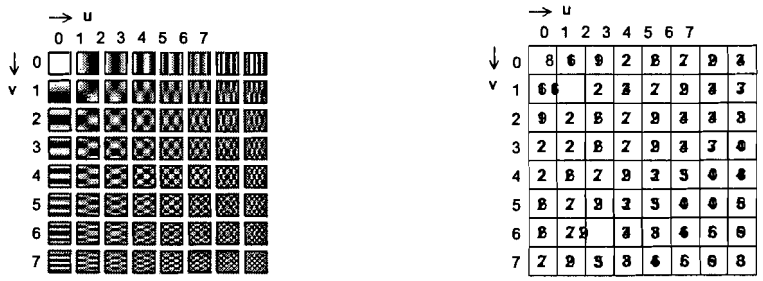


Figure 2.7: DCT basis functions $p_u(x)$ of Equation 2.14 with $N = 8$, shown as dots. The solid line shows the corresponding continuous cosine waves.



(a) DCT basis images

(b) quantization matrix

Figure 2.8: Figure (a) shows the two-dimensional DCT basis functions of Equations 2.16 and 2.17 with $N = 8$ and $M = 8$, represented as images. Full white represents the value 1, full black the value -1. Each individual image has a coordinate system similar to the one defined in Figure 2.1. Figure (b) shows the default quantization matrix used in MPEG-2 for intra coded blocks.

DCT matrix with u and v like in Figure 2.8(a), the coefficients are ordered by increasing frequency from the top-leftmost coefficient until the bottom-rightmost coefficient. The DCT tends to concentrate the energy of the signals into the top-left corner of the coefficient matrix, leaving the rest of the coefficients near zero. Furthermore, the human visual system is less perceptible for fine spatial detail, i.e. the higher frequencies [88]. A first compression step is to remove psychovisual redundancy by quantizing those frequencies more severe that are less visible for the human eye. Therefore, each transform coefficient is divided by the corresponding entry in a quantization matrix, resulting in the higher frequencies to be coarser quantized than the lower frequencies, allowing fewer quantization levels and more compression for those coefficients. A default matrix for MPEG-2 intra coded blocks can be found in Figure 2.8(b), though the MPEG standards allow for custom defined matrices to be encoded in the bit stream.

Typically, when traversing the sparse coefficient matrix in *zigzag* direction, as demonstrated in Figure 2.9, the result is a linear AC coefficient vector with long runs of zeros. A zero-run-length-decoder translates the AC coefficient vector to an array of two-dimensional vectors, each vector indicating a non-zero value and the preceding number of zeros. These vectors are variable length coded and then transmitted. The more zeros occur, the longer the zero-runs and the smaller the variable length codes. DC coefficients are encoded somewhat different. Exploiting the fact that spatial consistency between DC coefficients of neighbouring blocks is high, the DC coefficient is differentially decoded with respect to the previous decoded block.

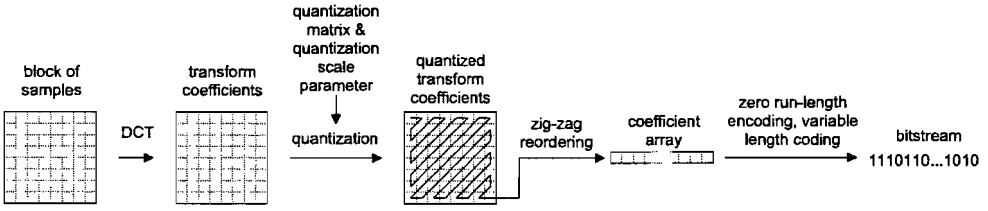


Figure 2.9: Block encoding process (see text).

To achieve more compression, the DCT coefficients can be even coarser quantized. By dividing the DCT coefficients by a *quantization scale parameter*, which we further refer to as *qscale*, more coefficients become zero and more compression is achieved. A second effect of coarser quantization is that more likely values occur, resulting in variable length codes with fewer bits. Appendix A shows a block coding example for a reference MPEG-2 implementation. The quantization scale parameter is as such a means to control the bit rate of the encoded stream. The decoder can, when the video buffer defined in the sequence header is about to underflow or overflow, adjust the quantization scale parameter. Some MPEG-encoders first calculate the video statistics in order to adjust the quantization scale parameter with fewer visual consequences (*two pass encoding*). We assume linear quantization, which means that *qscale* is the same as the quantization step size.

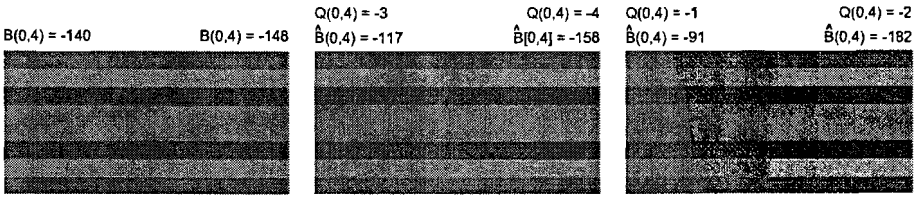
The concept of adjusting the quantization parameter while encoding is called *variable bit rate* coding. The bit rate can then be adjusted to keep constant quality of video. For Internet Video, the bandwidth is constrained, causing the encoder to code at constant bit rate while the quality varies from picture to picture.

2.4 Coding artefacts

Prior to discussing existing artefact repair techniques, we will discuss the typical artefacts that come with quantization in the DCT domain. A large *qscale* introduces larger quantization steps and more quantization noise. This is the cause for coding artefacts like the blocking artefact and the ringing artefact. Artefacts can be caused by the image acquisition process, compression, transmission, and display [1, 25]. This section only concentrates on compression artefacts, associated with quantization in the DCT domain.

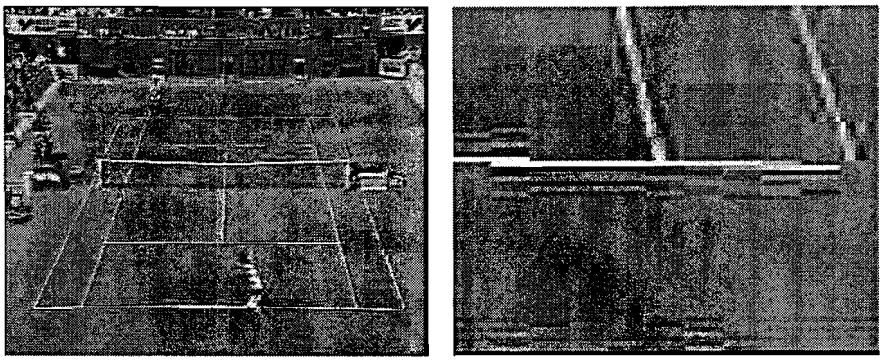
2.4.1 Blocking artefact

The blocking artefact (or *grid noise* [74]) is an effect that is caused by all block-based coding techniques. It is the most visible image degradation of all artefacts [40, 92]. Due to coarse quantization, the correlation between the blocks is lost. Especially mismatched low-frequency components contribute to the blocking effects. The coarse quantization causes the low-frequency DCT coefficients of neighbouring blocks to be put in different bins, introducing discontinuity along the block boundaries. This effect is illustrated in Figure 2.10. Figure 2.10(a) shows two adjacent 8×8 blocks with their 8-bit sample values, constant within a block in horizontal direction. The blocks are in fact two-dimensional DCT basis images for $u = 0$ and $v = 4$. The sample values are chosen such that there is an unnoticeable difference between the original blocks. Figure 2.10(b) demonstrates what the values of the DCT coefficients become after quantization and de-quantization with $qscale = 24$. Quantization and de-quantization with $qscale = 56$ results in a reconstruction shown in Figure 2.10(c), clearly showing the block boundary. The coarser the quantization is, the more noticeable the discontinuities become. The blocking artefact can also occur when the motion vectors are corrupted such that the motion compensated prediction mismatches the surrounding blocks. In the extreme case that the bit rate does not allow for AC coefficients to be coded, the blocking artefact is very severe. This is the case with fast motion sequences where nearly all macroblocks require update or at very low bit rates, so that the available



(a) original (b) $qscale = 24$ (c) $qscale = 56$

Figure 2.10: Figure (a) shows two adjacent 8×8 blocks with their DCT coefficients $B(0,4)$. (b) is the reconstruction with $qscale = 24$ and (c) is the reconstruction with $qscale = 56$ with quantized DCT coefficients $Q(0,4)$ and reconstructed coefficients $\hat{B}(0,4)$.



(a) (b)

Figure 2.11: Frame no. 85 (I-frame) (a) of the *tennis* sequence, MPEG-2 encoded at $500kbits/s$, and an image detail (b) showing both blocking and ringing effect.

number of bits has to be divided among those macroblocks. Figure 2.11 shows large blocks with few or no AC coefficients.

The visibility of the blocking artefact depends on the magnitude of the discontinuity, the position of the blocks, the local contrast in the image, the spatial frequency content, the presence of motion, the viewing distance [45], etcetera. Especially in smooth regions, the human visual system is very perceptive for inconsistencies along block borders. Typically, I-frames suffer the most from the blocking artefact. I-frames have a regular 8×8 square pattern that is not interrupted by motion compensated predicted blocks. If the residual coding mechanism provides insufficient compensation, the grid can propagate through P and B-frames as well, resulting in a shifted block grid as illustrated in figure 2.4. This propagation of the block grid is referred by Jung [32] as *grid shift artefact*, and by Wu *et al.* [92] as *false edges*.

2.4.2 Ringing artefact

The ringing artefact is visible for all sub-band and wavelet compressed video schemes, including MPEG-2. It appears as repeating ghost edges along sharp edges in the original sequence (Figure 2.11), or as a texture deviation in high textured areas. The artefact is the result of absence of high frequency components, also known as the *Gibbs effect*. Ringing occurs within the length of the filter's impulse response, i.e. within an 8×8 block. Appendix A shows a coding example of an edge showing ringing after reconstruction. The ringing artefact is visible for all compression

techniques involving quantization in the frequency domain.

2.4.3 Blurring

Blurring is another artefact resulting from the absence of high frequencies in low bit rate MPEG video. Sharp edges and colours are blurred and high textures are smoothed (*texture deviation*). This smoothing effect can also be observed in Figure 2.11. The effect depends strongly on the viewing distance: when the distance is large enough for the missing frequencies to fall outside the pass-band of the human visual system, the artefact is not annoying anymore.

2.4.4 Mosquito noise

Mosquito noise is a typical temporal artefact that shows up as fluctuations in luminance and chrominance values in blocks at the boundary of objects moving with respect to other objects or the background [10], also known as *occlusion blocks*. Moreover, blocks with the same image content can be coded with different quantization scales, resulting in small fluctuations over time. Though the fluctuations are small and the area is confined to the block area, the artefact can be considered annoying. Mosquito noise manifests itself also as *flickering* [80] (or *stationary area granular noise* [92]) in the background.

2.4.5 Other artefacts

Some other less visible artefacts resulting from coarse quantization can be found in Internet Video. We will mention them shortly. *DCT basis image blocks* appear when the signal energy is concentrated in a single basis image, showing the particular image on top of the DC component. *Staircase effect* or *corner outliers* occur when edges are misaligned at the block boundary, or when edges are not coded for in a block intersecting an edge. The latter artefacts are often associated with the blocking artefact as well. Though some publications treat them as different artefacts, they have the same origin. *Colour bleeding* is the blurring effect of coarsely quantified chrominance blocks, such that the colour smears across an edge that is clearly defined in the luminance blocks. If a motion vector in a motion area is zero, it appears static with respect to its surrounding blocks, experienced as the *dirty window effect.*

Other kinds of coding artefacts exist in MPEG-like coding schemes, that are not caused by quantization of the transform coefficients. An example is *motion jerkiness*, the perception of discontinuous motion due to temporal sub-sampling. Another artefact, the *down-sampling effect*, causes spatial and temporal business due to down-scaling of the image. Object based transform schemes use mathematical descriptions of objects for image synthesis. When crucial features are changed, *geometrical deformation* (of e.g. a human face) appears. The MPEG-2 standard allows for both interlaced and progressive macroblocks within the same picture. When not well de-interlaced, blocks show information at another temporal position every even or odd line, within a macroblock. The artefact is visible as jagged edges.

The focus of this project lies on the blocking artefact, the most visible of all coding artefacts. Numerous algorithms have been proposed in the last decades. The following chapter gives an overview of existing approaches for de-blocking.

Chapter 3

Blocking artefact repair

At some point in the current video broadcast chain, video compression is applied to reduce bandwidth or storage size. As shown in the previous chapter, severe quantization in the frequency domain gives rise to annoying distortion or artefacts. Post-processing of the decoded image sequence is an widely accepted technique to achieve better perceived picture quality [80]. Furthermore, modern consumer vision products like televisions and PCs use image enhancement and restoration techniques to improve the objective and subjective picture quality [23, page 21]. These algorithms, for example sharpness improvement and contrast improvement techniques, make the coding artefacts more visible [102], emphasizing the need for artefact repair. This chapter provides an overview of blocking artefact repair algorithms existing in the literature and in the industry.

3.1 Objective detection and measurement

When fighting the blocking artefact, knowledge of the position of the block coding grid is of great help. For digital television, this knowledge is directly available from the bit stream. However, when a classic analogue video signal enters for example a television set, one cannot assume the coding grid to be an 8×8 grid with its position exactly in the upper left corner of the image [13, 44]. The original signal has been subject to scaling, sampling, image processing, and conversion operations that result in a shifted, scaled coding grid at the television input. For these signals, a first step in the post-processing chain is the detection of the coding grid. The video post-processing chain is depicted in Figure 3.1. Philips Research Laboratories developed several effective algorithms for *block grid detection*, such as DADAR (Digital Artifact Detection And Repair) [13], BAM (Blocking Artifact Meter) [44], DATES (Digital Artifact Estimator), BGD (Block Grid Detector), and, just recently, BEACoN, that return accurate information of the coding grid size, the offset with respect to the image origin, as well as a measurement for the severeness of the blocking artefact. Since we are focusing on low bit rate Internet Video, we assume the coding parameters to be directly available from the decoder. All the algorithms in this report assume an 8×8 coding block grid, and image dimensions that are integer multiples of 8.

For optimal blocking artefact reduction, we need a metric to measure the severeness of the artefact. Furthermore, we need a metric to compare different algorithms. Another term for blocking artefact is grid noise. The word *noise* suggest a noise-like measurement of the grid noise level.

Widely used benchmark metrics for the comparison of the artefact reduction algorithm's effectiveness are the *Mean Squared Error* (MSE) and the *Peak Signal-to-Noise Ratio* (PSNR), defined for the n^{th} image as:

$$MSE(n) = \frac{1}{M \cdot N} \sum_{\vec{x} \in \mathfrak{I}} \left(\hat{F}(\vec{x}, n) - F_o(\vec{x}, n) \right)^2 \text{ and} \tag{3.1}$$

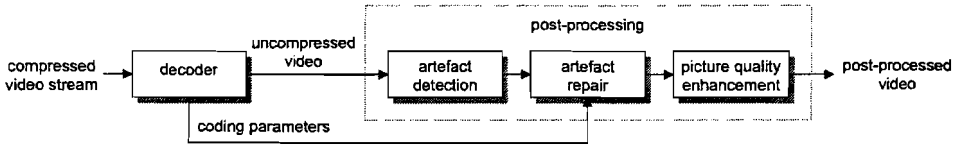


Figure 3.1: The video post-processing chain.

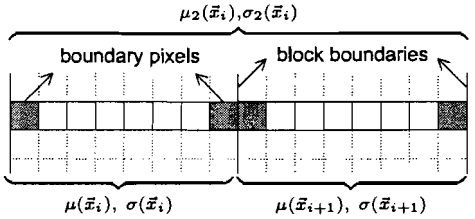


Figure 3.2: Two image segments used for block impairment metric calculation.

$$PSNR(n) = 10\log_{10}\left(\frac{F_{pp}^2}{MSE(n)}\right)[dB], \tag{3.2}$$

where $F_o(\vec{x},n)$ is the original signal and $\hat{F}(\vec{x},n)$ is the reconstructed (post-processed) signal. F_{pp} is the peak-to-peak amplitude of the signal (for an 8-bit signal, $F_{pp}=255$). Comparable to the n^{th} image, we can define the MSE and PSNR for an entire sequence of images, summing over all pixels in all images available in the sequence. The MSE and PSNR indicators tell how much the original sequence differs from the post-processed sequence and as such, they indicate the post-processed image's fidelity.

However, these traditional metrics have a poor relation with the human visual perception of coding impairments. Another drawback of the MSE and the PSNR is that they require the original sequence as a reference. A metric showing a strong consistency with subjective evaluations was introduced by Wu *et al.* [93], who introduced the General Block Impairment Metric (GBIM) for the most annoying artefact, the blocking artefact, especially. Let us first define the set $\mathcal{H}$ of leftmost pixels of all horizontal 8×1 image segments like illustrated in Figure 3.2:

$$\mathcal{H} = \{\vec{x}\in\mathcal{I} \mid x \bmod 8 = 0\}. \tag{3.3}$$

We shall refer to this set often for the de-blocking algorithms in the remainder of this report. A similar set exists for the vertical 1×8 image segments. GBIM calculates the one-dimensional means $\mu(\vec{x}_i)$, $\mu(\vec{x}_{i+1})$ and standard deviations $\sigma(\vec{x}_i)$, $\sigma(\vec{x}_{i+1})$ of the luminance sample values on both sides of a DCT block boundary (Figure 3.2):

$$\begin{gathered}\forall\ \vec{x}_i = \begin{pmatrix}x_i\\ y_i\end{pmatrix}\in\mathcal{H},\\ \exists\ \sigma(\vec{x}_i) = \sqrt{\frac{1}{8}\sum_{k=0}^{7}\left(F\left(\begin{pmatrix}x_i+k\\ y_i\end{pmatrix},n\right)-\mu(\vec{x}_i)\right)^2},\ \mu(\vec{x}_i) = \frac{1}{8}\sum_{k=0}^{7}F\left(x_i\vec{+}ky_i,n\right).\end{gathered} \tag{3.4}$$

Now define the mean $\mu_2(\vec{x}_i) = \frac{1}{2}(\mu(\vec{x}_i)+\mu(\vec{x}_{i+1}))$ and mean standard deviation $\sigma_2(\vec{x}_i) = \frac{1}{2}(\sigma(\vec{x}_i)+\sigma(\vec{x}_{i+1}))$ of two adjacent 8×1 block segments as depicted in Figure 3.2, then we can calculate the sum of weighted absolute differences between each pair of adjacent boundary pixels to get a metric for the blocking artefact M_h:

$$M_h = \sum_{\vec{x}_i\in\mathcal{H},x_i\neq N-8} w(\vec{x}_i)\left|F\left(\begin{pmatrix}x_i+7\\ y_i\end{pmatrix},n\right)-F\left(\begin{pmatrix}x_i+8\\ y_i\end{pmatrix},n\right)\right|. \tag{3.5}$$

The weighting factors $w(\vec{x}_i)$ account for the HVS masking of the blocking artefact in dark and bright areas:

$$w(\vec{x}_i) = \begin{cases} \frac{\ln\left(1+\sqrt{F_{pp}-\zeta}\right)}{\ln\left(1+\sqrt{\zeta}\right)} \ln\left(1+\frac{\sqrt{\mu_2(\vec{x}_i)}}{1+\sigma_2(\vec{x}_i)}\right), & \text{if } \mu_2(\vec{x}_i) \leq \zeta, \\ \ln\left(1+\frac{\sqrt{F_{pp}-\mu_2(\vec{x}_i)}}{1+\sigma_2(\vec{x}_i)}\right) & , \text{if } \mu_2(\vec{x}_i) > \zeta, \end{cases} \tag{3.6}$$

where ζ is the average luminance value for which the weighting function has its maximum. In [93], $\zeta = 81$. The same calculation can be done for the non-edge pixels:

$$E_h = \frac{1}{7}\sum_{k=0}^{6} \sum_{\vec{x}_i \in \mathcal{H}, x_i \neq N-8} w(\vec{x}_i) \left| F\left(\begin{pmatrix} x_i+k \\ y_i \end{pmatrix}, n\right) - F\left(\begin{pmatrix} x_i+k+1 \\ y_i \end{pmatrix}, n\right)\right|. \tag{3.7}$$

Normalizing M_h by E_h yields a metric for the horizontal blocking artefact (i.e. the vertical edges) $M_{hGBIM} = M_h/E_h$. The same metric can be found for the vertical artefact $M_{vGBIM} = M_v/E_v$. The average of the two is the GBIM metric, $M_{GBIM} = \frac{1}{2}(M_{hGBIM} + M_{vGBIM})$. GBIM gives an estimate of the severeness of the blocking artefact without the presence of the original sequence.

Furthermore, some propriety metrics are available, like the JNDmetrix^{TM}(Just Noticeable Difference) metric from the Sarnoff company [109]. It is based on a model of the human visual system that was obtained during numerous subjective quality assessments. The unit of 1 JND corresponds to a probability of 75 % that a viewer notices a difference between the original and the reconstructed sequence. The technology is, however, only available under licence. Note that the MSE, PSNR, and JND metrics require the original sequence to be available, as opposed to the GBIM metric.

Now that we defined the GBIM metric, we will give an overview of existing de-blocking methods in the literature. One can distinguish three main approaches for artefact repair:

- Adaptive filtering tries to smooth the block boundaries without affecting the natural edges in the image.
- Set theoretic methods like Projection Onto Convex Sets (POCS) and Constrained Least Squares (CLS) try to narrow down the set of solutions by setting constraints.
- Statistical estimation methods like Maximum A Posteriori (MAP) probability based restoration and anisotropic diffusion.

The first category of techniques are *image enhancement* techniques, meant to improve the subjective picture quality. The latter two are *image restoration* approaches, trying to recover the original sequence given the distorted image and its properties. The algorithms found in the literature and in the industry are manifold, yet they share some basic approaches. In the following sections, we will give an overview of the different approaches including some examples. The overview is not meant to be exhaustive, but instead it enumerates the approaches that are illustrative for the methods found in the literature and applied in the industry.

3.2 Adaptive filtering

This section provides an overview of heuristic, adaptive filter approaches. Non-adaptive methods often result in excessive blurring [74]. For example, one can apply an approximately 3×3 Gaussian filter at the block boundary pixels to reduce the blocking artefact [75]. Gaussian filters are a very popular choice for low-pass filters. Theoretically, the Gaussian filter is defined as:

$$\hat{F}(\vec{x}, n) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} \frac{1}{2\pi\sigma_x^2\sigma_y^2} F\left(\begin{pmatrix} x+i \\ y+j \end{pmatrix}, n\right) \exp\left(\frac{i^2}{\sigma_x^2} + \frac{j^2}{\sigma_y^2}\right), \tag{3.8}$$

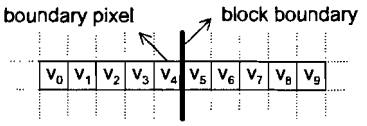


Figure 3.3: Pixels commonly used for one-dimensional filter support.

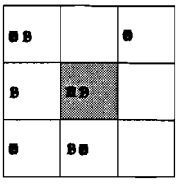


Figure 3.4: 3×3 Gaussian filter weights with $\sigma_x = \sigma_y = 1.0$, for the centre pixel to be filtered.

Figure 3.5: Sobel filter kernels for vertical edges (a) and horizontal edges (b), for the centre pixel to be evaluated (grey).

with variances σ_x^2 and σ_y^2 of the Gaussian along the horizontal and the vertical axes respectively. Figure 3.4 shows an example kernel used in [75]. Another popular class of filters consists of those who apply one-dimensional low-pass filtering orthogonal to the block edge. Such a filter alters the pixel value near the block boundary by a weighted average of the pixels in the support size. The support size is in general small in high-detail areas to avoid over-smoothing. The size can be larger in smooth areas. The adaptation of the filter support and its weights is the topic of the next subsection. A commonly used pixel support for one-dimensional filters is depicted in Figure 3.3. In the remainder of this report, we will refer to pixel v_i as both its position in the figure and its luminance value v_i. The context determines which of the two is meant.

3.2.1 Spatial adaptive filtering

Filtering without considering local spatial statistics causes the loss of natural high frequencies. Spatial adaptive filtering was proposed to overcome such blurring. Adaptive filters have coefficients that vary depending on the local content. Generally, adaptive filtering requires a step of classification and/or edge detection, followed by linear or non-linear filtering. Classification is often based on first-order (mean) and second-order (variance) statistics.

Filtering based on local spatial characteristics

For example, Ramamurthi *et al.* [74] introduced a generic filter for the removal of the blocking artefact and the staircase effect, that uses local statistics like mean and variance to differentiate between monotone and edge blocks. Monotone blocks contain little spatial detail, exhibiting low variance. Edge blocks have a higher variance, revealing the presence of one of more edges. The filter further classifies edge blocks into four classes for four different orientations. Next, two-dimensional filtering is applied for monotone blocks, and one-dimensional, directional filtering is applied for edge blocks. Spatial median filters were found to be ineffective in reducing the blocking artefact. They are more appropriate for random and spot-like noise.

An example of adaptive filters combined with an edge detector can be found in [42]. For edge detection, the *Sobel* filter kernels are among the most popular found in the literature. These filters calculate the gradient $\nabla F(\vec{x}, n)$ of the n^{th} image at pixel position $\vec{x}$ while they smooth in the orthogonal direction, making them less sensitive to noise. If the Sobel filter kernels $S_x(\vec{x})$ and $S_y(\vec{x})$ are defined for $-1 \leq x \leq 1$ and $-1 \leq y \leq 1$ as in Figure 3.5(a) and 3.5(b) respectively, the horizontal and the vertical *gradient* components $G_x(x,\vec{n})$ and $G_y(x,\vec{n})$ of the n^{th} image are

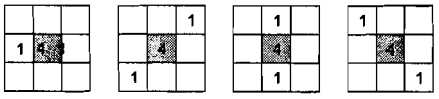


Figure 3.6: Directional filters for the centre pixel to be filtered.

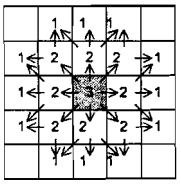


Figure 3.7: Filter kernel used for adaptive filtering.

obtained by convolution

$$\nabla F(\vec{x},n) = \begin{pmatrix} G_x(\vec{x},n) \\ G_y(\vec{x},n) \end{pmatrix}, G_x(\vec{x},n) = F(\vec{x},n) * S_x(\vec{x}), \text{ and } G_y(\vec{x},n) = F(\vec{x},n) * S_y(\vec{x}). \quad (3.9)$$

The approximated *gradient magnitude* $M(\vec{x},n)$ and the *gradient angle* $\theta(\vec{x},n)$ are then defined by

$$M(\vec{x},n) = \sqrt{G_x^2(\vec{x},n) + G_y^2(\vec{x},n)} \approx |G_x(\vec{x},n)| + |G_y(\vec{x},n)| \text{ and} \quad (3.10a)$$

$$\theta(\vec{x},n) = \arctan \frac{G_y(\vec{x},n)}{G_x(\vec{x},n)}. \quad (3.10b)$$

One can locate the edge pixels in an image by thresholding the gradient magnitude. Edge and non-edge pixels are treated separately in [42]:

- First step of the algorithm is to reduce staircase noise in edge areas by filtering the edge pixels with a one-dimensional filter in the direction parallel to the direction of the edge. Example one-dimensional directional filters for the quantized angles $0°, \pm 45°, \pm 90°, \pm 135°$ or $180°$ can be found in Figure 3.6. An implementation of such a filter was registered in [20]. In [96] a similar approach is described with median (non-linear) filters.
- For non-edge areas, the grid noise is reduced by a signal adaptive filter. The algorithm classifies a pixel *within* a block as a local edge pixel if the gradient magnitude exceeds a certain threshold that depends on the local mean and variance in that block. A two-dimensional filter kernel smoothes those pixels that are not classified as edge pixels. The pixel weights for the filter aperture can be found in Figure 3.7. If one or more pixels in the aperture are classified as edge pixels, those pixels including its successors are excluded from the aperture. The successors are indicated by the direction of the arrows. If there is not a single edge pixel in the filter aperture, the weights assigned are all one. Filters like these are also used for the removal of the ringing artefact.

Another simple adaptive filter that takes into account the local characteristics was introduced by Kim *et al.* [37]. A pixel is filtered with a 3×3 Gaussian kernel with variance $\sigma^2 = \sigma_Z^2 / \sigma_S^2$, where σ_Z^2 bounds the amount of smoothing and σ_S^2 is the local variance. Triantafyllidis *et al.* [86] use another adaptive Gaussian filter, where the variances σ_x and σ_y of the Gaussian in Equation 3.8 depend on the smoothed image gradient (the *windowed second moment matrix*). These filters are non-linear in the sense that their smoothing strength depend on the local inter-pixel differences. Generally, filters of which the weights can be written as a ratio between two polynomials are called *rational filters*. The nominators have a low-pass characteristic, the denominators are a function of the local inter-pixel differences.

Derviaux *et al.* [11] incorporated some characteristics of the HSV. They propose a one-dimensional FIR filter to be applied perpendicular to the block edge, its coefficients depending on the visibility v of the impairment (see Figure 3.8):

$$v = \frac{C_m}{1 + A}, \quad (3.11)$$

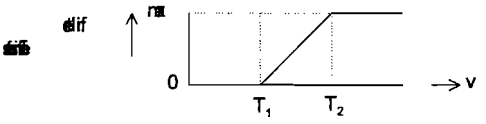


Figure 3.8: Adaptation of the smoothing filter coefficients to the visibility v of the artefact. T_1 and T_2 are predetermined thresholds.

where C_m is the local mean contrast based on the local contrast $C = \frac{1}{2}v_3 - \frac{2}{3}v_4 + \frac{2}{3}v_5 - \frac{1}{2}v_6$ along the block edge (cf. Figure 3.3) and weighted by the local mean luminance. Parameter A represents the spatial masking of the HVS, based on a weighted sum of AC coefficients. Another characteristic of the HVS is that high-motion objects are harder to track. For high-motion objects, they propose a temporal filter discussed further on in this section.

Multistage median filters

Liu *et al.* [49] use multistage median filters to preserve edges. For the different directions of edge pixels in Figure 3.6, a one-dimensional 5-point median operation is performed on the centre pixel and its four closest neighbours. Let us denote those results as $F_{0°}(\vec{x},n), F_{45°}(\vec{x},n), F_{90°}(\vec{x},n)$ and $F_{135°}(\vec{x},n)$. Define the maximum and the minimum of those median values as

$$F_{max}(\vec{x},n) = \max\{F_{0°}(\vec{x},n), F_{45°}(\vec{x},n), F_{90°}(\vec{x},n), F_{135°}(\vec{x},n)\}, \tag{3.12a}$$

$$F_{min}(\vec{x},n) = \min\{F_{0°}(\vec{x},n), F_{45°}(\vec{x},n), F_{90°}(\vec{x},n), F_{135°}(\vec{x},n)\}, \tag{3.12b}$$

then the filtered output of the multistage median filter is given by

$$\hat{F}(\vec{x},n) = \text{median}\{F(\vec{x},n), F_{max}(\vec{x},n), F_{min}(\vec{x},n)\}, \tag{3.13}$$

for edge pixels. Non-edge pixels are two-stage median filtered (5×5 median of 5×5 medians) for smooth area's, and 3×3 median filtered for non-smooth areas with motion.

Classification based on DCT coefficients

Some methods make a classification based on the DCT coefficients and then perform spatial filtering. A simple algorithm [8] filters the edge pixels, pixels v_4 and v_5 in Figure 3.3, to compensate for an artificial discontinuity $d = v_4 - v_5$:

$$\hat{v}_4 = \begin{cases} v_4 - \alpha d, \text{ for } |d| \le t, \\ v_4 - \alpha t, \text{ for } d > t, \\ v_4 + \alpha t, \text{ for } d < -t, \end{cases} \quad \text{and } \hat{v}_5 = \begin{cases} v_5 + \alpha d, \text{ for } |d| \le t, \\ v_5 + \alpha t, \text{ for } d > t, \\ v_5 - \alpha t, \text{ for } d < -t, \end{cases} \tag{3.14}$$

$\alpha = \frac{t-v}{2t}$ is a proportionality constant based on the visibility threshold v (≈ 0.025 the mean image luminance) and t is a threshold based on the probability distribution $p_{u,v}(y)$ of the DCT coefficients $B(u,v)$. The distribution can be modelled as the Gaussian distribution, with mean and variance directly calculated from the decoded DCT coefficients $B(u,v)$:

$$t = 2\sqrt{\frac{1}{64}\sum_{u=0}^{7}\sum_{v=0}^{7}\frac{\int_{B(u,v)-\frac{qstep}{2}}^{B(u,v)+\frac{qstep}{2}}(y-B(u,v))^2 p_{u,v}(y)dy}{\int_{B(u,v)-\frac{qstep}{2}}^{B(u,v)+\frac{qstep}{2}} p_{u,v}(y)dy}}. \tag{3.15}$$

To avoid discontinuities caused by the modified pixels, the algorithm is applied iteratively to the pixel pairs v_3, v_4 and v_5, v_6 and so forth, until the centre of the block is reached.

A combination of edge detection and classification in the DCT domain is also reported. Park *et al.* [70] de-block four neighbouring low-frequency blocks (i.e. blocks for which all pixels other

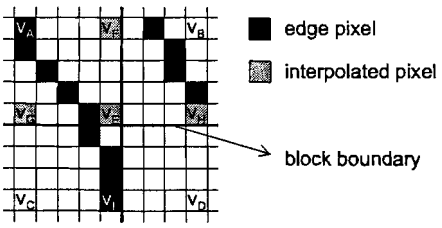


Figure 3.9: Interpolation of pixel values.

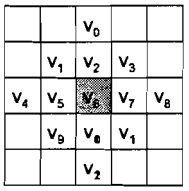


Figure 3.10: Filter aperture used for content adaptive filtering. Pixel v_6 is the centre pixel to be evaluated.

than $B(0,0), B(0,1)$ and $B(1,0)$ are zero) by iteratively interpolating *non-edge* pixels only. The algorithm is illustrated in Figure 3.9. At the first iteration, given pixels v_A, v_B, v_C and v_D, pixels v_E is filtered by the average of pixel v_A, v_B, v_C and v_D. Since v_A is an edge pixel, the pixel does not participate in the averaging process, resulting in the pixel value $v_E = \frac{1}{3}(v_B + v_C + v_D)$. The second iteration calculates pixel values v_F, v_G, and v_H. Pixel v_I is an edge pixel and is, therefore, not interpolated. The iteration continues until all pixels have been evaluated.

Region growing

Meier *et al.* [56] choose an iterative region-based approach. An adaptive segmentation algorithm calculates the means of regions of pixels along the block boundary and merges them if they are close to each other. Then a 3×3 Gaussian filter is applied on those pixels of which the 8 neighbouring pixels belong to the same region. After filtering, the *quantization constraint* (which shall be dealt with in the next section) is applied and the process is repeated until convergence is reached.

Filtering based on training sets

Zhao *et al.* [107] proposed a content adaptive image de-blocking method. Therefore, observe the decoded pixel v_6 to be filtered and its filter aperture in Figure 3.10. Each pixel v_6 belongs to a class c of pixels with a similar local pattern. The class number c is depends on the surrounding decoded pixels v_0 to v_{12} and is calculated by Adaptive Dynamic Range Coding (ADRC):

$$c = \sum_{k=0}^{12} 2^k \delta(v_k), \text{ with } \delta(v_k) = \begin{cases} 0, \text{ if } v_k < \mu, \\ 1, \text{ if } v_k \geq \mu, \end{cases} \tag{3.16}$$

where μ is the average pixel value in the aperture. Each centre pixel value $v_6 \in \mathbb{J}$ is then replaced by a weighted average of the samples within the filter aperture of that pixel:

$$\hat{v_6} = \sum_{k=0}^{12} w_{k,c} \cdot v_k. \tag{3.17}$$

The weights $w_{k,c}$ for each pixel k in the filter aperture depend on the class c and can be found in a look-up table. The weights for each class are obtained from a training set such that they minimize the summed squared errors in the aperture between the original and the corresponding encoded and decoded pixels from that class. The results can be improved by taking into account the relative position of the pixel with respect to the block grid, involving multiple look-up tables for each relative position.

The latter technique uses a database obtained by a large training set. Good results have also been reported with the use of neural networks. The technique of neural networks in [72] corrects the value of the boundary pixels v_4 and v_5 in Figure 3.24 by the output $\vec{y} = (y_1, y_2)^T$ of a Multi-Layer Perceptron (MLP) neutral network

$$\begin{aligned} \hat{v}_4 &= v_4 + y_1, \\ \hat{v}_5 &= v_5 + y_2, \end{aligned} \tag{3.18}$$

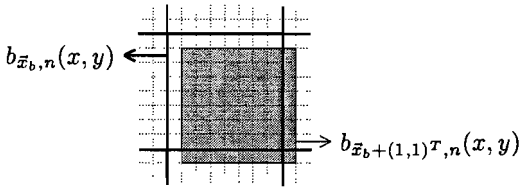


Figure 3.11: A block and a diagonally shifted block.

given the input vector $\vec{x} = (x_1, x_2, x_3)^T$ with $x_1 = v_4 - v_3$, $x_2 = v_5 - v_4$, and $x_3 = v_6 - v_5$.

3.2.2 Filtering in the frequency domain

Yang *et al.* [98] used the *Lapped Orthogonal Transform* (LOT) as a starting point of an algorithm. The one-dimensional DCT is performed on the pixels across the block boundary, pixels v_1 to v_8 in Figure 3.3. They assumed the blocking artefacts are caused by the first two odd-symmetric coefficients F_1 and F_3 (cf. $p_1(x)$ and $p_3(x)$ of Figure 2.7). By updating those coefficients according to

$$\hat{F}_1 = \begin{cases} \frac{1}{2}F_1, \text{ if } |F_1| < \epsilon \\ F_1 \text{ , otherwise,} \end{cases} \text{ and } \hat{F}_3 = \begin{cases} \frac{1}{2}F_3, \text{ if } |F_3| < \frac{\epsilon}{2} \\ F_3 \text{ , otherwise,} \end{cases}, \tag{3.19}$$

where ϵ is a small experimentally determined energy threshold, the blocking artefact can be reduced. Coefficients higher than the threshold are likely to result from natural edges and remain unmodified. A similar algorithm is discussed later in the theory on projections onto convex sets.

Chen *et al.* [4] introduce a filter in the frequency domain that uses the sensitivity of the HVS. Therefore, they define the activity $A_{\vec{x},n}$ of a block $\mathcal{B}_{\vec{x}_b}$, $x_b \in \mathcal{B}$ given the BDCT $B_{\vec{x},n}$ as:

$$A_{\vec{x}_b,n} = \frac{1}{B_{\vec{x}_b}(0,0)} \sqrt{\sum_{u=0}^{7}\sum_{v=0}^{7} \left(B^2_{\vec{x}_b}(v,u) - B^2_{\vec{x}_b,n}(0,0) \right)}. \tag{3.20}$$

Furthermore, consider the set of BDCT matrices of the shifted blocks $B_{\vec{x}+(x,y)^T,n}$ of their respective shifted blocks $\mathcal{B}_{\vec{x}_b+(x,y)^T}$ (cf. Figure 3.11). If the activity is lower than a experimentally determined threshold, the coefficients are filtered by a weighted average of the BDCT matrices of the shifted variants in a 5×5 neighbourhood

$$B_{\vec{x}_b,n}(v,u) = \frac{1}{25} \sum_{x=-2}^{2} \sum_{y=-2}^{2} B_{\vec{x}_b+(x,y)^T,n}(v,u), \tag{3.21}$$

otherwise by a weighted average of the BDCT matrices of the shifted variants in a 3×3 neighbourhood

$$B_{\vec{x}_b,n}(v,u) = \frac{1}{11} \sum_{x=-1}^{1} \sum_{y=-1}^{1} B_{\vec{x}_b+(x,y)^T,n}(v,u) + 2 \cdot B_{\vec{x}_b,n}(v,u), \tag{3.22}$$

where the original coefficient weighs three times.

Nosratinia [64] calculates the BDCT on 64 shifts $F_s(\vec{x},n) = F(\vec{x} + \vec{x}_s, n)$, for $0 \leq x_s < 8$ and $0 \leq y_s < 8$, of the decoded image. Then he performs the same quantization and inverse quantization operation as the MPEG encoder and decoder on those shifts. Over the resulting 64 images, the average value for each pixel form the filtered result.

Other DCT filtering algorithms are often combined with the theory of projection onto convex sets, discussed further on in this report.

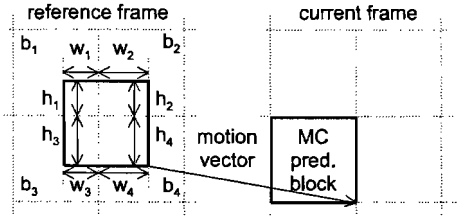


Figure 3.12: DC calculation from a reference frame, used in DC sequencing.

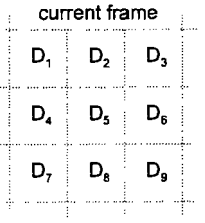


Figure 3.13: DC image neighbourhood used for the calculation of AC coefficients of the block corresponding with the centre DC sample.

AC prediction

Shin *et al.* [81] use a technique called *DC sequencing* to predict corrupted AC coefficients. The DC image $D(\vec{x}, n)$ with $0 \leq x < \frac{N}{8}$ and $0 \leq y < \frac{M}{8}$ is an $\frac{M}{8} \times \frac{N}{8}$ image containing the DC coefficients of the n^{th} image obtained from the decoder. For I-pictures, the DC-image is obvious: $D(y_b/8, x_b/8)^T, n) = B_{\vec{x}_b,n}(0,0)$ for all blocks $\mathcal{B}_{\vec{x}_b} \in \mathcal{I}$. For P and B-pictures, the DC values are calculated from the reference frames. Therefore, observe a generic reference frame in Figure 3.12 and the four blocks b_1, b_2, b_3, b_4 with their respective BDCTs B_1, B_2, B_3, B_4 from which the block in the current frame is predicted. Then the estimated DC value for the motion compensated prediction is calculated from:

$$B_{\vec{x},n}(0,0) = \sum_{i=1}^{4} \frac{w_i h_i}{64} B_i(0,0), \tag{3.23}$$

where w_i and h_i are the widths and the heights, respectively, of the overlap of the motion compensated prediction with the respective reference blocks, in number of pixels (cf. Figure 3.12). For bi-directionally predicted blocks, the estimated DC values are simply averaged. Next step in the process is the filtering of the BDCT coefficients for each block in the video sequence. If one or more of the first five AC coefficients in zigzag order are zero, indicating block impairment, AC prediction is performed based on the DC values in the DC image neighbourhood shown in Figure 3.13:

$$\begin{aligned}
B_{\vec{x},n}(0,1) &= 1.13884 \cdot \tfrac{1}{8}(D_4 - D_6), \\
B_{\vec{x},n}(1,0) &= 1.13884 \cdot \tfrac{1}{8}(D_2 - D_8), \\
B_{\vec{x},n}(0,2) &= 0.27881 \cdot \tfrac{1}{8}(D_4 + D_6 - 2D_5), \\
B_{\vec{x},n}(2,0) &= 0.27881 \cdot \tfrac{1}{8}(D_2 + D_8 - 2D_5), \\
B_{\vec{x},n}(1,1) &= 0.16213 \cdot \tfrac{1}{8}(D_1 + D_9 - D_3 - D_7).
\end{aligned} \tag{3.24}$$

The prediction in Equation 3.24 can be adaptively changed if the block to be filtered contains an edge, which is further left undocumented. An implementation using AC prediction can be found in [35].

3.2.3 Filtering in the wavelet domain

Recently, filtering in the sub-band domain has become a popular solution to tackle the blocking artefact. Donoho [12] proposed a noise reduction technique that suppresses noise by limiting the coefficients in the Discrete Wavelet Transform (DWT) domain, which showed to be nearly optimal for white Gaussian noise reduction [6]. The two-dimensional DWT decomposition splits an image into several complementary sub-band images. A popular realization of the *two-dimensional wavelet transform* at scale j recursively splits a low-pass image $S_{2^{j-1}}(\vec{x}, n)$ into two high-pass images for the horizontal $W^1_{2^j}(\vec{x}, n)$ and the vertical $W^2_{2^j}(\vec{x}, n)$ frequencies (the *detailed* images) and a

complementary low-pass image S_{2^j} (the *coarse* or *smooth* image). An implementation with digital filters was proposed by Mallat [53] using high-pass G_j and low-pass filters H_j for all $\vec{x} \in \mathcal{I}$:

$$W^1_{2^j}(\vec{x}, n) = S_{2^{j-1}}(\vec{x}, n) * G_{j-1}, \tag{3.25a}$$

$$W^2_{2^j}(\vec{x}, n) = S_{2^{j-1}}(\vec{x}, n) * G^T_{j-1}, \tag{3.25b}$$

$$S_{2^j}(\vec{x}, n) = S_{2^{j-1}}(\vec{x}, n) * H_{j-1} * H^T_{j-1}, \tag{3.25c}$$

where $S_{2^0}(\vec{x}, n) = F(\vec{x}, n)$ is the original image and the j^{th} filter is dilated by a factor 2^j (forming a diadic sequence). After each iteration, the two sub-band images and the smooth image contain redundant data and are typically down-sampled (*decimated*) by factor two. However, for singularity (edge) analysis we do not down-sample the images, so that the transform coefficients spatially correspond to the pixels in the original image, keeping the redundant data and making the transform *over-complete*. Moreover, filtering of the un-decimated transform coefficients outperforms filtering of the decimated transform coefficients in noise reduction [6]. The filtered image can be reconstructed from the modified transform coefficients with the iteration

$$\begin{aligned} S_{2^{j-1}}(\vec{x}, n) = {} & W^1_{2^j}(\vec{x}, n) * K_{j-1} * L^T_{j-1} + W^2_{2^j}(\vec{x}, n) * L_{j-1} * K^T_{j-1} + \\ & S_{2^j}(\vec{x}, n) * \bar{H}_{j-1} * \bar{H}^T_{j-1}, \end{aligned} \tag{3.26}$$

where K_j and L_j are high-pass reconstruction filters and $\bar{H}_j$ is the space-reversed of H_j, defined in [53]. Figure 3.14 shows the two-dimensional over-complete wavelet transform on a blocky image.

Soft thresholding

Gopinath *et al.* [18] applied the concept of noise reduction in the wavelet domain to smooth the blocking artefact. Therefore, the coefficients of the horizontal and vertical high-pass images are modified by *soft thresholding* the coefficients according to

$$W'_{2^j}(\vec{x}, n) = \begin{cases} W_{2^j}(\vec{x}, n) - T, & \text{for } W_{2^j}(\vec{x}, n) > T, \\ 0 \qquad\qquad\quad , & \text{for } -T \leq W_{2^j}(\vec{x}, n) \leq T, \\ W_{2^j}(\vec{x}, n) + T, & \text{for } W_{2^j}(\vec{x}, n) < -T, \end{cases} \tag{3.27}$$

where $T = \delta\sqrt{2 \ln N}$ is a threshold depending on the square image size $N \times N$ and an estimation of the granular noise variance defined by:

$$\delta = \frac{1}{L} \sum_{k=1}^{L} \sigma_j^2, \tag{3.28}$$

with L is the number of high-frequency sub-bands and σ_j^2 is the variance per 8×8 block in the j^{th} sub-band.

Another thresholding method has been proposed by Xiong *et al.* [95]. We will demonstrate their method for the horizontal artefacts only. Define the set $\mathcal{I}_0$ as the set of high-pass transform coefficients that affect the horizontal blocking artefact (the vertical edges). This is the same set we defined earlier as $\mathcal{H}$ in Equation 3.3. Then for all $\vec{x} \in \mathcal{I}_0$, the high-pass transform coefficients of the first scale are low-pass filtered:

$$\hat{W}^1_{2^1}(\vec{x}, n) = \frac{W^1_{2^1}(\vec{x} - (1,0)^T, n) + W^1_{2^1}(\vec{x}, n) + W^1_{2^1}F(\vec{x} + (1,0)^T, n)}{3}. \tag{3.29}$$

Furthermore, the high-pass coefficients for all $\vec{x} \in \mathcal{I}_0$ at scales $j = 1$ and $j = 2$ are zeroed if the *cross-scale high-pass coefficients* exceed a certain threshold T, or otherwise remain untouched, as in Equation 3.30.

$$\hat{W}^1_{2^j}(\vec{x}, n) = \begin{cases} 0 \qquad\qquad\quad , & \text{for } W^1_{2^1}(\vec{x}, n) * W^1_{2^2}(\vec{x}, n) + W^2_{2^1}(\vec{x}, n) * W^2_{2^2}(\vec{x}, n) > T, \\ W^1_{2^j}(\vec{x}, n), & \text{otherwise.} \end{cases} \tag{3.30}$$

The smooth image at the second scale is further low-pass filtered if the cross-scale high-pass coefficients lie below that threshold T ($\forall \vec{x} \in \mathcal{I}_0$). The threshold T is estimated as the mean squared inter-pixel difference between all horizontal and vertical boundary pixels.

(a) S_{2^0} (b) M_{2^1} (c) M_{2^2}

(d) S_{2^1} (e) $W^1_{2^1}$ (f) $W^2_{2^1}$

(g) S_{2^2} (h) $W^1_{2^2}$ (i) $W^2_{2^2}$

(j) S_{2^3} (k) $W^1_{2^3}$ (l) $W^2_{2^3}$

Figure 3.14: Over-complete two-dimensional discrete wavelet transform of frame 481 of the *tennis* sequence, up to the third scale. The modulus of the first scale is M_{2^1} is squared to make the block edges more visible.

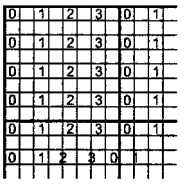


Figure 3.15: $W^1_{2^1}(\vec{x}, n)$ showing the pixel site for each sub-set. The bold lines are the block boundaries, the thin lines are the pixel boundaries.

Linear minimum mean squared error

Choi *et al.* [6] extend the idea above with *Linear Minimum Mean Squared Error* (LMMSE) filters. Observe the high-pass image for the horizontal frequency of the first scale $W^1_{2^j}F(\vec{x}, n)$ in Figure 3.14. Now define the subsets of sub-sampled pixel positions

$$\mathfrak{I}_k = \{\vec{x} \in \mathbb{N}^2 \mid 0 \leq x < N \wedge 0 \leq y < M \ \wedge x \bmod 8 = 2k \ \wedge y \bmod 2 = 0\}, \tag{3.31}$$

for $k = 0, 1, 2, 3$, as illustrated in Figure 3.15, with their respective variances

$$\sigma_k^2 = \frac{1}{|\mathfrak{I}_k|} \sum_{\vec{x} \in \mathfrak{I}_k} \left(W^1_{2^1}(\vec{x}, n)\right)^2, \tag{3.32}$$

assuming zero mean. The variance σ_k^2 for $k = 0$ is significantly higher than for $k = 1, 2, 3$, caused by the blocking artefact. The blocking artefact noise is estimated by $\sigma_b^2 = \sigma_0^2 - \frac{1}{3}(\sigma_1^2 + \sigma_2^2 + \sigma_3^2)$. This yields a LMMSE estimate of the transform coefficients that are responsible for the blocking artefact:

$$\hat{W}^1_{2^1}(\vec{x}, n) = \frac{\sigma_g^2}{\sigma_g^2 + \sigma_b^2} W^1_{2^1}(\vec{x}, n) \ \forall \vec{x} \in \mathfrak{I}_0. \tag{3.33}$$

σ_g^2 is the local variance in a local window in the set $\mathfrak{I}_1 \cup \mathfrak{I}_2 \cup \mathfrak{I}_3$. The same is done for the vertical artefact prior to inverse transformation.

Wavelet transform modulus maxima

By observing the modulus maxima across different wavelet scales, step edges like the blocking artefact can be recognized by a typical decay of the modulus across the wavelet scales [48]. The *wavelet modulus* of the j^{th} scale is defined as:

$$M_{2^j}(\vec{x}, n) = \sqrt{|W^1_{2^j}(\vec{x}, n)|^2 + |W^2_{2^j}(\vec{x}, n)|^2}. \tag{3.34}$$

Figure 3.16 shows the decomposition of a one-dimensional signal with three typical features at three wavelet scales. The ratios of the modulus maxima of a step edge are $M_{2^2}/M_{2^1} = 0.75$ and $M_{2^3}/M_{2^1} = 0.6875$, and the ratios of the modulus maxima of an impulse are $M_{2^2}/M_{2^1} = 0.375$ and $M_{2^3}/M_{2^1} = 0.171875$. Liew *et al.* decided to define the three thresholds for the three wavelet scales to be the average of the wavelet modulus maxima responses for a step discontinuity and an impulse:

$$T_{2^2} = 0.3164 \cdot T_{2^1}, \tag{3.35}$$

$$T_{2^3} = 0.1846 \cdot T_{2^1}, \tag{3.36}$$

with T_{2^1} the root-mean-square of all inter-pixel differences in the image. If the modulus $M_{2^j}(\vec{x}, n)$ is below the threshold T_{2^j}, then the coefficients $W^1_{2^j}(\vec{x}, n)$ and $W^2_{2^j}(\vec{x}, n)$ are set to zero for the scales $j = 1, 2, 3$, prior to inverse transformation.

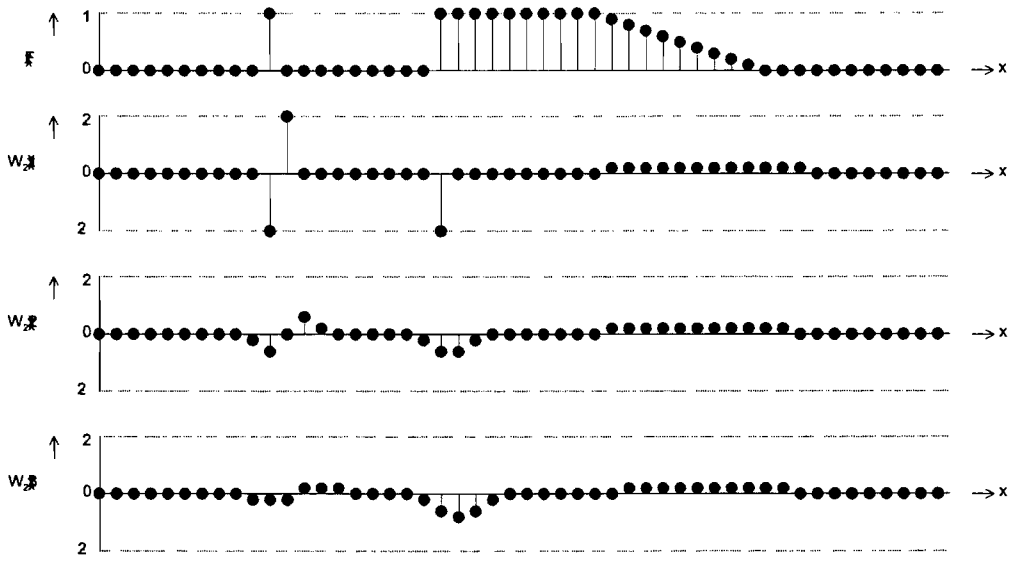


Figure 3.16: A signal $F(x)$ and its corresponding un-decimated high-pass wavelet decompositions at three scales W_{2^k}. From left to right, the signal shows an impulse, a step edge, and a ramp.

Another method using the modulus maxima was introduced by Fan *et al.* [14], who directly smooth the wavelet modulus image. The reconstruction of the detailed images from the wavelet modulus maxima can be done with the *wavelet phase* function, defined as

$$A_{2^j}(\vec{x},n) = \arctan \frac{W^1_{2^j}(\vec{x},n)}{W^2_{2^j}(\vec{x},n)}. \tag{3.37}$$

Numerous other methods can be found in the literature. Nosratinia [63] for example, performs wavelet encoding and decoding according to the JPEG-2000 standard on several shifts of the image, and then averages over all results. The result, however, is slightly blurred. Wavelet filtering is also effective in removing the ringing artefact.

3.2.4 Temporal filtering

In [96], a method is introduced for the reduction of noise in general, including grid noise. The output of the filter is an Infinite Impulse Response (IIR) filter that calculates a weighted average of the current frame and its motion compensated prediction from the previous filtered frame:

$$\hat{F}(\vec{x},n) = \alpha F(\vec{x},n) + (1-\alpha)\hat{F}(\vec{x} - \vec{D}(\vec{x},n), n-1). \tag{3.38}$$

Best results were found with $\alpha = 0.1$ to 0.5 for $D(\vec{x},n) = \vec{0}$, and $\alpha = 1$ for motion blocks.

Derviaux *et al.* [11] proposed a similar method based on a three-frame motion compensated prediction, using the motion vectors from the MPEG-stream:

$$\hat{F}(\vec{x},n) = \alpha F(\vec{x} - \vec{D}(\vec{x},n), n-1) + (1-\alpha-\beta)F(\vec{x},n) + \beta F(\vec{x} + \vec{D}(\vec{x},n), n+1), \tag{3.39}$$

with the weighting factors α and β depending on the temporal distance between the motion compensated frame and the reference frames. Liu *et al.* [49] chooses these parameters as $\alpha = \beta = \frac{1}{3}$, if both motion compensated predictions are reliable[1]. Note that the reference frames in Equation 3.39 are the previous and the next frame in the sequence, while the reference frames in [11] are those from the MPEG stream.

[1] In Chapter 4 we will define some reliability metrics.

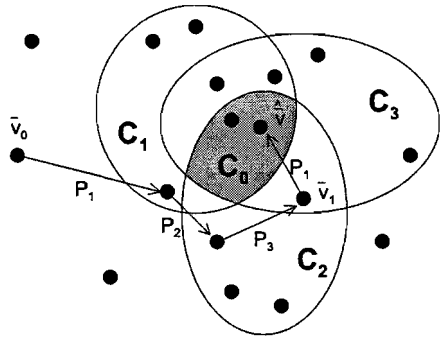


Figure 3.17: Two-dimensional Hilbert space with three constraint sets and the solution space $\mathcal{C}_0$. The dots represent the vectors, and the norm function is simply the Euclidian distance between the vectors. The arrows denote the projection operators. Here, the initial vector lies outside all constraint sets, which is generally not the case for image restoration.

3.2.5 Other domains

Some less common techniques can be found in the literature, for other domains than mentioned above. For example, filtering in the Hermite transform domain [61] or low-pass filtering of the coefficients in the cepstral[2] domain [5] can be found.

3.3 Projection onto convex sets

Quantization is the process of assigning ranges of DCT coefficient values to several quantization bins. At the decoder side, the reconstructed value is typically the value in the middle of the corresponding bin, as illustrated in Figure 2.2. Mathematically, this is equivalent to mapping a range of coefficient values onto a single coefficient value. This implies that the quantization function is not injective and has no inverse function; an input value does not have a unique output value, so many combinations of input sequences yield the same output sequence. The decoded sequence is just one out of many solutions, probably not the original sequence. Let $\mathcal{A}$ be a set of all sequences that leads to the same encoded signal. This implies that the set $\mathcal{A}$ is the set of all possible solutions from that encoded signal. Now let $\mathcal{B}$ be the set of all sequences without artefacts, then our target is to choose a solution from the intersection $\mathcal{A} \cap \mathcal{B}$. *Projection onto convex sets* (POCS) is a *set-theoretic method* that defines constraint sets with the purpose to narrow down the number of solutions in the solution space in order to estimate the original sequence, and is as such a restoration technique. POCS is a mathematical information recovery technique, introduced in the field of image processing by Youla *et al.* [104, 105]. Let $\vec{v} = (v_0, v_1, \ldots, v_{(M \cdot N-1)})$ be a vector of length $M \cdot N$ in Hilbert space[3] $\mathcal{S}$, and let $\vec{v}_0$ represent the initial decoded image samples $F(\vec{x}, n)$:

$$\vec{v}_0 = \{\vec{v} \in \mathbb{R}^{M \cdot N} | v_{(y \cdot N + x)} = F(\vec{x}, n) \; \forall \; 0 \leq x < N \wedge 0 \leq y < M\}. \tag{3.40}$$

By iteratively projecting this initial image onto a number, say m, closed convex sets $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_m \in \mathcal{S}$ with their respective projection operators $P_1, P_2, \ldots, P_m$, then the iteration

$$\vec{v}_{i+1} = P_m P_{m-1} \ldots P_1 \vec{v}_i \tag{3.41}$$

converges towards a solution $\hat{\vec{v}}$

$$\hat{\vec{v}} = \lim_{i \to \infty} \vec{v}_i \tag{3.42}$$

[2]The cepstrum is the inverse Fourier transform of the logarithm of the forward Fourier transform.

[3]A Hilbert space is a vector space for which the inner product (norm) is defined which satisfies certain requirements.

in the solution space $\mathcal{C}_0$

$$\mathcal{C}_0 = \bigcap_{i=1}^{m} \mathcal{C}_i \neq \emptyset, \tag{3.43}$$

which is the non-empty intersection of all m sets. The projection $P_i\vec{v}$ onto constraint set $\mathcal{C}_i$ is the closest (in terms of the norm function) element in $\mathcal{C}$ given $\vec{v}$:

$$\|\vec{v} - P_i\vec{v}\| = \min_{\vec{g}\in\mathcal{C}_i} \|\vec{v} - \vec{g}\|. \tag{3.44}$$

In practice, the iteration stops when the distance reaches a certain minimum. The process is clarified in Figure 3.17. Since the solution set $\mathcal{C}_0$ is generally contains more than one vector, the solution $\hat{\vec{v}}$ depends on the initial vector $\vec{v}_0$ and the order of the projections [100]. The challenge for set-theoretic methods is to find the constraint sets $\mathcal{C}_i$. This is the topic in many papers. The next subsections give some constraints found in the literature.

3.3.1 Intensity constraint

The intensity constraint simply projects a filtered sample onto the range of available sample values, given by:

$$P : \mathbb{Z} \to \mathbb{N}, \hat{F}(\vec{x}, n) = \text{clip}\left[F(\vec{x}, n)\right]_0^{F_{pp}}. \tag{3.45}$$

3.3.2 Quantization constraint

A second constraint claims that every processed DCT coefficient $\hat{B}_{\vec{x}_b,n}(u,v)$ must lie in the quantization bin of the original decoded DCT coefficient. Introduced by Zakhor [106], the quantization constraint is always used in POCS-based methods. For each block $B_{\vec{x}_b}$, the de-quantized two-dimensional DCT coefficient lies between the quantization bin boundaries (cf. Equation A.2):

$$B_{\vec{x}_b,n}(u,v) - \frac{qscale}{2} \leq \hat{B}_{\vec{x}_b,n}(u,v) \leq B_{\vec{x}_b,n}(u,v) + \frac{qscale}{2}. \tag{3.46}$$

The different rounding operations and the quantization matrix $quant_{v,u}$ are ignored for convenience. The corresponding projection operator is simply the clip function, clipping the de-quantized coefficients to the closest value in a valid range.

3.3.3 Smoothness constraints

Zakhor [106] was one of the first to use constraints. Zakhor proposed a *band-limitation constraint* by filtering the entire image with a low-pass 3×3 FIR filter, assuming that the block discontinuities cause frequencies not present in the original image. Unfortunately, this method over-smoothed the image when applied iterative. Though its design was POCS-based, the proposed FIR operator was not a projection operator.

Smoothing in the spatial domain

Another smoothness constraint to smooth the boundaries between blocks has been introduced by Yang [99]. The constraint chooses the images for which the sum S of squared boundary pixel differences is smaller than a certain threshold E. Observe Figure 3.2 and define the set of leftmost pixel positions of 8×1 image segments as $\mathcal{H}$ as in Equation 3.3, then the constraint is given by:

$$\mathcal{C} = \{F(\vec{x}, n) \in \mathcal{I} \mid S \leq E\}, \text{ with} \tag{3.47}$$

$$S = \sum_{\vec{x}\in H} \sqrt{\left(F\left(\begin{pmatrix} x+7 \\ y \end{pmatrix}, n\right) - F\left(\begin{pmatrix} x+8 \\ y \end{pmatrix}, n\right)\right)^2}. \tag{3.48}$$

The image is smoothed by the projection ($\forall \vec{x} \in \mathcal{H}$):

$$F_{i+1}\left(\begin{pmatrix} x+i \\ y \end{pmatrix}, n\right) = \begin{cases} (1-\alpha)F_i\left(\begin{pmatrix} x+i \\ y \end{pmatrix}, n\right) + \alpha F_i\left(\begin{pmatrix} x+i-1 \\ y \end{pmatrix}, n\right), & \text{for } i = 0, \\ \alpha F_i\left(\begin{pmatrix} x+i \\ y \end{pmatrix}, n\right) + (1-\alpha)F_i\left(\begin{pmatrix} x+i+1 \\ y \end{pmatrix}, n\right), & \text{for } i = 7, \\ F_i\left(\begin{pmatrix} x+i \\ y \end{pmatrix}, n\right) & \text{for } 0 < i < 7, \end{cases} \tag{3.49}$$

with $\alpha = \frac{1}{2}(E/S + 1)$. The constraint uses a GBIM-like metric. In fact, the GBIM metric is based on this constraint. A similar algorithm was registered by Zhou [108]. Yang *et al.* [100] proposed a better constraint set, incorporating the weight of Equation 3.6 to the sum S. Later work from Yang *et al.* [101] extended the constraint sets with directional smoothness constraints for the diagonal directions, in order to reduce the ringing artefact as well. For low bit rate video, however, the block discontinuity is not restricted to the boundary pixels, giving poor results on highly compressed sequences.

Smoothing in the frequency domain

A third smoothness constraint was proposed by Paek *et al.* [67]. Observe the two neighbouring 8×1 block segments in Figure 3.2 again. Paek proposed to cut off high frequencies in the DCT domain, in order to remove any discontinuity between blocks. Therefore, the 8-point DCTs are calculated from both the left and the right segment, as well as the 16-point DCT of both segments. The assumption is that the frequency characteristics of two adjacent blocks are highly correlated. If high-frequency components exist in the 16-point DCT coefficients, which are not present in both 8-point DCT coefficients, these are likely to result from the blocking artefact. By zeroing these coefficients before performing the 16-point IDCT, the discontinuity can be removed. Paek observed that only odd-numbered coefficients were responsible for the discontinuities. Let $\vec{u} = (u_0, u_1, \ldots, u_7)$ and $\vec{v} = (v_0, v_1, \ldots, v_7)$ represent the sample values from the left and the right block respectively. Furthermore, let $\vec{w} = (u_0, \ldots, u_7, v0, \ldots, v_7)$ be the concatenation of both segments, and let $\vec{U}$, $\vec{V}$, and $\vec{W}$ be the one-dimensional DCTs of the corresponding segments. Then the non-zero operator NZ returns the location of the last non-zero coefficient in the DCT vector:

$$NZ(\vec{X}) = \max\{0 \leq i < 8 \,|\, X_i \neq 0\}. \tag{3.50}$$

The 16-point DCT $\vec{W}$ is modified to $\hat{\vec{W}}$ according to:

$$\hat{W}_i = \begin{cases} 0 & \text{, for } i \bmod 2 = 1 \wedge i > (2 \cdot \max\{NZ(\vec{U}), NZ(\vec{V})\} + 2), \\ W_i & \text{, otherwise.} \end{cases} \tag{3.51}$$

The filter is applied in both horizontal and vertical direction. Gesnot [16, 58] implemented and extended this idea with a simple edge detector to prevent post-processing in high textured areas. In later work [67], Paek *et al.* excluded those segments from processing which have larger inter-pixel differences than the boundary pixel difference.

Kim *et al.* [38] proposed a similar algorithm, considering a coding block and a diagonally shifted variant as in Figure 3.11. Both blocks are transformed to the two-dimensional DCT domain. Let the coefficient vector $\vec{U} = (u_0, u_1, \ldots, u_{63})$ be the zigzag reordered coefficient vector of the original block, and $\vec{V}$ the same coefficient vector of the shifted block. A non-zero function similar to the one defined in Equation 3.50 determines the last non-zero coefficient in the vector $\vec{U}$. The high frequencies of the shifted block are zeroed prior to inverse transformation, according to:

$$\hat{V}_i = \begin{cases} 0 & \text{, if } i > NZ(\vec{U}) + p, \\ V_i & \text{, otherwise,} \end{cases} \tag{3.52}$$

where p is a factor representing how many DCT coefficients contribute to the blocking artefact, depending on the local statistics and the properties of the HVS:

$$p = \lfloor \eta \sqrt{\frac{V_1 + V_2 + V_3 + V_4 + V_5}{V_0 + 1}} + 0.5 \rfloor, \text{ with } \eta = 10. \tag{3.53}$$

3.3.4 Multi-frame constraints

The use of the temporal dimension in constraint sets was first explored by Gunturk *et al.* [19]. They introduced a POCS-based de-blocking algorithm over multiple frames using accurate motion estimation. A *multi-frame constraint set* is proposed using the quantization bounds available from the bit stream. To make the technique clear, consider Figure 2.4 again and the 8×8 coding block in frame n. Its best match in the previous frame $n-1$ or, more generic, its best match in any reference frame m, is generally not aligned with the block grid, so it may contain the blocking artefact, as illustrated in the figure. From the MPEG-bit stream, the quantization parameter for the transform coefficients of the block in the current frame n and the blocks in the reference frame m are known. Using these quantization constraints on the best-match block in the reference frame m, one can reduce the blocking artefact for this particular block. For the method to work correctly, the motion vectors have to be reliable. It has been concluded that motion vectors from the MPEG stream are inaccurate.

The method of POCS is also a technique to obtain a high-resolution image from several low-resolution input images (*superresolution*) and as a means for *error concealment*, to recover from errors due to lossy transmission. Transmission artefacts are outside the scope of this project.

3.4 Constrained least square regularization

The *Constrained Least Square* (CLS) regularization is an image recovery technique that finds the optimum of two contradictory constraints [80]. Let an image $F(\vec{x}, n)$ defined by a column vector $\vec{f}$ that concatenates all rows of the image:

$$\vec{f} \in \mathbb{R}^{M \cdot N \times 1}, f_{N \cdot y + x} = F(\vec{x}, n), \forall \vec{x} \in \mathfrak{I}. \tag{3.54}$$

In the reminder of this section, we will use this mapping next to each other. The decoded image $\vec{f_0}$ can be modelled with the *image degradation model*:

$$\vec{f_0} = \vec{f} + \vec{e}, \tag{3.55}$$

where $\vec{f}$ is the original image we try to estimate and $\vec{e} \in \mathbb{R}^{M \cdot N \times 1}$ is a noise vector. Let us define the set $\mathcal{C}_{\epsilon_1}$ of all images $\vec{f}$ that are close to the decoded image $\vec{f_0}$

$$\mathcal{C}_{\epsilon_1} = \{\vec{f} \mid \|\vec{f} - \vec{f_0}\|^2 \leq \epsilon_1^2\}, \tag{3.56}$$

and the set $\mathcal{C}_{\epsilon_2}$ of all images $\vec{f}$ that have a certain smoothness

$$\mathcal{C}_{\epsilon_2} = \{\vec{f} \mid \|\mathbf{S}\vec{f}\|^2 \leq \epsilon_2^2\}, \tag{3.57}$$

where $\mathbf{S}$ is a $MN \times MN$ matrix called the *regularization operator*, usually implemented as a Laplacian high-pass filter:

$$F_{filtered}(\vec{x}, n) = \left(F(\vec{x}, n) - F\left(\vec{x} - (1,0)^T, n\right)\right) + \left(F(\vec{x}, n) - F\left(\vec{x} - (0,1)^T, n\right)\right), \tag{3.58}$$

such that

$$\|\mathbf{S}\vec{f}\| = \sum_{\vec{x} \in \mathfrak{I}} \left(F(\vec{x}, n) - F\left(\vec{x} - (1,0)^T, n\right)\right)^2 + \left(F(\vec{x}, n) - F\left(\vec{x} - (0,1)^T, n\right)\right)^2. \tag{3.59}$$

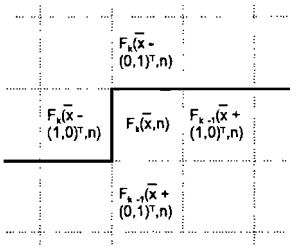


Figure 3.18: Filter support of the CLS iteration. Pixels above the bold line are already updated.

ϵ_1^2 is an upper bound for the allowed error and ϵ_2^2 is the upper bound for the smoothness. The CLS solution for the restored image is the minimum of the two contradictory constraints joint in a *cost function*:

$$J(\vec{f}) = \|\vec{f} - \vec{f}_0\|^2 + \mu\|\mathbf{S}\vec{f}\|^2, \tag{3.60}$$

with the *regularization parameter* that controls the smoothness of the result:

$$\mu = \left(\frac{\epsilon_1}{\epsilon_2}\right)^2. \tag{3.61}$$

The regularization parameter depends on the noise variance: the higher μ, the smoother the resulting image is. Several choices for μ are given in [15]. A mathematical solution to the minimization problem in the DCT domain was found by Yang *et al.* [99] as an iteration. Therefore, divide an image as defined in Equation 3.54 into 8×8 sample blocks and perform the 8×8 DCT on each of them. Then reorder the coefficient image in the same way as $\vec{f}$ in the Equation 3.54 to get the vector $\vec{F}$ of the coefficient image. Given that the DCT coefficients are linear combinations of the samples in the corresponding coding block, there exists a $MN \times MN$ matrix $\mathbf{B}$ that transforms $\vec{f}$ into $\vec{F}$ by a matrix multiplication:

$$\vec{F} = \mathbf{B}\vec{f}. \tag{3.62}$$

The rows of the DCT matrix form an orthogonal basis, and so do the rows of the transform matrix $\mathbf{B}$. For an orthogonal matrix, the inverse is simply its transpose, so we can write for Yang's iterative solution:

$$\vec{F}_k = \vec{F}_{k-1} + \beta\left(\vec{F}_0 - \left(\mathbf{I} + \mu\mathbf{B}\mathbf{S}^T\mathbf{S}\mathbf{B}^T\right)\vec{F}_{k-1}\right), \tag{3.63}$$

with $\mathbf{I}$ the $MN \times MN$ identity matrix, $\vec{F}_0$ the DCT transformed image of the initial decoded image $\vec{f}_0$, and the *relaxation parameter* β somewhere between

$$0 < \beta < \frac{2}{\|\mathbf{I} + \mu\mathbf{S}^T\mathbf{S}\|}. \tag{3.64}$$

If the previous DCT coefficients $\vec{F}_{k-1}$ are first projected onto the quantization constraint in Equation 3.46, the results improve considerably. Crouse *et al.* [9] modified the regularization operator $\|\mathbf{S}\vec{f}\|$ (Equation 3.59) such that is only adds the clipped inter-pixel differences of the boundary pixels.

A practical implementation using small filters was introduced by Kaup [34]. The method was designed to reduce both the blocking and the ringing artefact. Minimizing the cost function leads to an approximate iterative solution, an averaging filter that is applied from left to right and top to bottom:

$$\begin{aligned} F_k(\vec{x}, n) = &\frac{1}{1 + j\mu}\left(F_0(\vec{x}, n) + \mu\left(F_k\left(\vec{x} - (1,0)^T, n\right) + F_k\left(\vec{x} - (0,1)^T, n\right) + \right.\right. \\ &\left.\left. F_{k-1}\left(\vec{x} + (1,0)^T, n\right) + F_{k-1}\left(\vec{x} + (0,1)^T, n\right)\right)\right), \end{aligned} \tag{3.65}$$

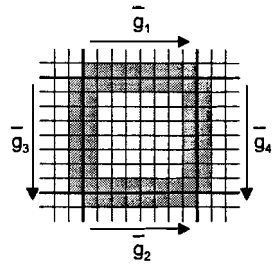


Figure 3.19: Pixel support for the calculation of the squared block boundary discontinuity.

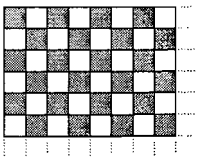


Figure 3.20: Subsets of coding blocks used by the least square block discontinuity iteration.

with $j = 4$. Figure 3.18 shows the filter support of the averaging filter. Mayor drawback is that the filter does not differentiate between natural edges and block edges. To solve this, the filter support can be adapted so that it only averages over those pixels which differ less than a certain threshold T from the centre pixel in order not to smooth natural edges (sigma filter), *or* over those pixels which are situated in another coding block than the centre pixel (to reduce the blocking effect). The parameter j in Equation 3.65 changes correspondingly with the number of pixels supported in the kernel. The threshold T is actually the *qscale* parameter for intra-coded images, twice the *qscale* parameter for predicted images. The best results were found with μ lying in the range from 0.125 to 0.25.

Least square block discontinuity

Jeon *et al.* [30] developed an iterative technique to filter the DCT coefficients. Therefore, define the difference signal vectors $\vec{g}_1, \vec{g}_2, \vec{g}_3, \vec{g}_4 \in \mathbb{R}^{8\times 1}$:

$$\begin{aligned}
(g_1)_x &= \hat{F}\left(\begin{pmatrix} x_b + x \\ y_b - 1 \end{pmatrix}, n\right) - F\left(\begin{pmatrix} x_b + x \\ y_b \end{pmatrix}, n\right), \text{ for } x = 0, \cdots, 7, \\
(g_2)_x &= \hat{F}\left(\begin{pmatrix} x_b + x \\ y_b + 8 \end{pmatrix}, n\right) - F\left(\begin{pmatrix} x_b + x \\ y_b + 7 \end{pmatrix}, n\right), \text{ for } x = 0, \cdots, 7, \\
(g_3)_y &= \hat{F}\left(\begin{pmatrix} x_b - 1 \\ y_b + y \end{pmatrix}, n\right) - F\left(\begin{pmatrix} x_b \\ y_b + y \end{pmatrix}, n\right), \text{ for } y = 0, \cdots, 7, \\
(g_4)_y &= \hat{F}\left(\begin{pmatrix} x_b + 8 \\ y_b + y \end{pmatrix}, n\right) - F\left(\begin{pmatrix} x_b + 7 \\ y_b + y \end{pmatrix}, n\right), \text{ for } y = 0, \cdots, 7,
\end{aligned} \tag{3.66}$$

where $F(\vec{x}, n)$ is the decoded image and $\hat{F}(\vec{x}, n)$ the de-blocked estimate of the original. Figure 3.19 shows the pixels involved in the calculation. Initializing the estimate with the decoded image, the algorithm aims at minimizing the square block discontinuity D for each block $b_{\vec{x}_b,n}$ in the n^{th} frame,

$$D = \sum_{i=1}^{4} \vec{g}_i \bullet \vec{g}_i, \tag{3.67}$$

for $F(\vec{x}, n) = \hat{F}(\vec{x}, n)$. For each block of samples $b_{\vec{x}_b,n}$, the de-blocked estimation $\hat{b}_{\vec{x}_b,n}$ can be found by adding a compensation signal $\Delta b_{\vec{x}_b,n}$ that minimizes D:

$$\hat{b}_{\vec{x}_b,n} = b_{\vec{x}_b,n} + \Delta b_{\vec{x}_b,n}. \tag{3.68}$$

The minimization is performed in the DCT domain. Using the properties of the HVS with respect to the artefact visibility, it is sufficient to compensate only the lower DCT coefficients. The

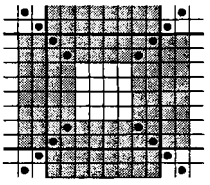


Figure 3.21: Pixel support for the calculation of the mean squared difference of slope (grey) and for the $MSDS_2$ across the boundary (thick lines).

authors use the four lowest DCT coefficients for compensation, with the following iteration applied alternatively on each subset of 8×8 blocks, depicted in Figure 3.20:

$$\begin{aligned}
B(0,0) &= \sqrt{8}\frac{(G_1)_0 + (G_2)_0 + (G_3)_0 + (G_4)_0}{4}, \\
B(0,1) &= \sqrt{8}\frac{(G_1)_1 + (G_2)_1 + (G_3)_0 - (G_4)_0}{2 + 4\cos^2(\pi/16)}, \\
B(1,0) &= \sqrt{8}\frac{(G_1)_0 + (G_2)_0 + (G_3)_1 - (G_4)_1}{2 + 4\cos^2(\pi/16)}, \\
B(0,1) &= \sqrt{8}\frac{(G_1)_1 - (G_2)_1 + (G_3)_1 - (G_4)_1}{4\sqrt{2}\cos(\pi/16)},
\end{aligned} \tag{3.69}$$

where $\vec{G}_i$ is the 8-point one-dimensional DCT of $\vec{g}_i$. The coefficients are updated under the quantization constraint.

The mean squared difference of slope

The *Mean Squared Difference of Slope* (MSDS) is a minimization technique that, as opposed to the previous metric, not tries to minimize the difference between the pixel values across the block boundaries, but instead it minimizes the difference between the *derivatives* across the block boundaries. The metric was presented by Minami *et al.* [57]. For the pixels in Figure 3.3, the squared difference of slope is defined as

$$SDS = \left((v_5 - v_4) - \left(\frac{v_4 - v_3}{2} + \frac{v_6 - v_5}{2}\right)\right)^2. \tag{3.70}$$

The $MSDS$ of a block $b_{\vec{x}_b,n}$ is then defined as the sum of all for all squared differences along the block borders as illustrated in Figure 3.21. The goal of the approach is to modify the pixels in block $b_{\vec{x}_b,n}$ such that it minimizes the $MSDS$.

In [87], a second metric $MSDS_2$ has been added to the original MSDS, that includes the squared difference of slope for the corner pixels (the pixels indicated by a dot in Figure 3.21), with the aim of reducing corner outliers. The problem of minimization of the total MSDS, $MSDS_t = MSDS + MSDS_2$, is investigated in the DCT domain and has a rather complex solution:

$$\begin{aligned}
\hat{B}_{\vec{x}_b,n}(u,v) = &\left(\left(B_{\vec{x}_b+(0,8)^T,n}(u,v) + B_{\vec{x}_b+(0,-8)^T,n}(u,v)\right)(-1)^u q_u^2 + \right. \\
&\left(B_{\vec{x}_b+(8,0)^T,n}(u,v) + B_{\vec{x}_b+(-8,0)^T,n}(u,v)\right)(-1)^v q_v^2 + \\
&\left(B_{\vec{x}_b+(-8,-8)^T,n}(u,v) + B_{\vec{x}_b+(8,-8)^T,n}(u,v) + \right. \\
&\left.\left. B_{\vec{x}_b+(-8,8)^T,n}(u,v) + B_{\vec{x}_b+(8,8)^T,n}(u,v)\right) R\right) / \left(2(q_u^2 + q_v^2) + 4R\right),
\end{aligned} \tag{3.71a}$$

$$R = 9p_u^2(0)p_v^2(0) - 6p_u(0)p_v(0)p_u(1)p_v(1) - p_u^2(1)p_v^2(1), \tag{3.71b}$$

$$q_u = 3p_u(0) - p_u(1). \tag{3.71c}$$

$p_u(x)$ is defined in Equation 2.14. The modified coefficients are projected onto the quantization constraint.

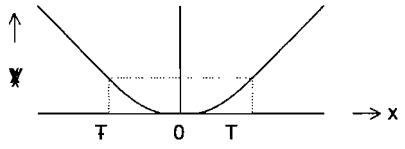


Figure 3.22: The Huber minimax function, parabolic for $|x| \leq T$, linear elsewhere.

3.4.1 Temporal regularization

Yao *et al.* [103] extend the idea of regularization in the spatial domain to the temporal domain, such that Equation 3.60 changes to:

$$J(\vec{f}) = \|\vec{f} - \vec{f}_0\|^2 + \mu\|\mathbf{S}\vec{f}\|^2 + \gamma\|\vec{f} - \vec{f}_{mc}\|, \tag{3.72}$$

with regularization parameter γ and where $\vec{f}_{mc}$ is a motion compensated prediction of $\vec{f}$ based on any reference frame. The observation leads to an iterative solution, described in their paper.

3.5 Maximum a posteriori probability based restoration

The *Maximum A Posteriori* (MAP) based restoration technique maximizes the probability P that the reconstructed image is the original image, given the decoded image and an image model. Define an image $\vec{f}$ again as in Equation 3.54. Then the problem is for all $\vec{f} \in \mathbb{R}^{M \cdot N \times 1}$ to find the maximum

$$\begin{aligned} \vec{f}_{MAP} &= \underset{\vec{f}}{\operatorname{argmax}}\{P(\vec{f}|\vec{f}_0)\} \overset{\{\text{Bayes}\}}{=} \underset{\vec{f}}{\operatorname{argmax}} \left\{ \frac{P(\vec{f}_0|\vec{f})P(\vec{f})}{P(\vec{f}_0)} \right\} \overset{\{\text{log-likelihood}\}}{=} \\ &\underset{\vec{f}}{\operatorname{argmax}} \left\{ \log P(\vec{f}_0|\vec{f}) + \log P(\vec{f}) - \log P(\vec{f}_0) \right\}, \end{aligned} \tag{3.73}$$

where $\vec{f}_0$ is the decoded image. Because $\vec{f}_0$ doe not dependent on the chosen solution, the probability $\log P(\vec{f}_0)$ can be dropped. Furthermore, let $\mathcal{F}$ be the set of possible decoded images given the decoded image $\vec{f}_0$ (as in the POCS theory), then the probability that the decoded image occurs given an input image from the set $\mathcal{F}$ is always unity, so the probability $P(\vec{f}_0|\vec{f})$ can also be dropped, leaving the problem:

$$\vec{f}_{MAP} = \underset{\vec{f} \in \mathcal{F}}{\operatorname{argmax}}\{\log P(\vec{f})\} = \underset{\vec{f} \in \mathcal{F}}{\operatorname{argmin}}\{-\log P(\vec{f})\}. \tag{3.74}$$

The probability $P(\vec{f})$ depends on a stochastic image model. A good model is the Markov Random Field (MRF) model [66]. MRFs are modelled with the *Gibbs* distribution given by

$$P(\vec{f}) = \frac{1}{Z} \exp\left(-\sum_{\mathcal{C}} V_{\mathcal{C}}(\vec{f}) \right), \tag{3.75}$$

Z being a normalization constant, $\mathcal{C}$ is any group (clique) of data, and $V_{\mathcal{C}}(\vec{f})$ is a potential function of clique $\mathcal{C}$ in $\vec{f}$. The choice of the potential function is essential. A comparison between potential functions used in signal and image recovery can be found in [62]. Though yielding slightly blurred, a common potential function used in the literature is the *Huber minimax function* (see Figure 3.22):

$$V(x) = \begin{cases} x^2 & , \text{for } |x| \leq T, \\ T^2 + 2T(|x| - T), & \text{for } |x| > T. \end{cases} \tag{3.76}$$

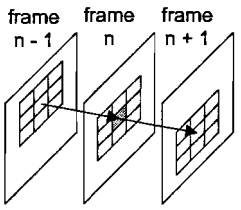


Figure 3.23: A pixel (grey) and its 26-neighbourhood for the motion compensated temporal MAP filter, showing the motion trajectory.

O'Rourke *et al.* [66] define the clique $\mathcal{C}_{\vec{x}_i}$ of a pixel $\vec{x}_i$ as all pixels in a 8-connected neighbourhood of that pixel,

$$\mathcal{C}_{\vec{x}_i} = \{\vec{x} \mid x_i - 1 \leq x \leq x_i + 1 \wedge y_i - 1 \leq y \leq y_i + 1 \ \wedge (x \neq x_i \vee y \neq y_i)\}, \tag{3.77}$$

so the minimization problem reduces to

$$\vec{f}_{MAP} = \underset{\vec{f} \in \mathcal{F}}{\operatorname{argmin}} \left\{ \sum_{\vec{x}_i \in \mathcal{I}} \sum_{\vec{x}_j \in \mathcal{C}_{\vec{x}_i}} V(F(\vec{x}_i, n) - F(\vec{x}_j, n)) \right\}. \tag{3.78}$$

The solution of this problem can be found by initializing the iteration with the decoded image $\vec{f}_0$, iterating towards a solution using the *gradient projection* method to find the steepest descent to the minimum:

$$\vec{f}_{k+1} = \vec{f}_k + \alpha \nabla \vec{f}_k, \tag{3.79}$$

with step size α and gradient $\nabla \vec{f}_k$ defined in [66]. Since the iteration might result in a vector outside $\mathcal{F}$, the vector is projected onto $\mathcal{F}$ in the DCT domain, using the quantization constraint, before entering the next iteration. In experiments, T is chosen at 1.0. The same algorithm can be applied in the decimated wavelet domain [46] to reduce computational speed. In an improvement of the above algorithm by Luo *et al.* [51] they distinguish between boundary and non-boundary pixels by choosing different thresholds T for boundary and non-boundary pixels. Furthermore, the iteration is done locally per pixel so convergence is reached faster (*iterative conditional mode*). Another method to reach a MAP solution is used by Özcelik *et al.* [65] using a technique called *mean field annealing.*

3.5.1 Temporal extension

The MAP filtering method can also be extended to a motion compensated temporal MAP-based (MC-TMAP) method [47]. Therefore, the 8-neighbourhood of Equation 3.77 is enlarged to a motion compensated 26-neighbourhood as depicted in Figure 3.23.

3.6 Anisotropic diffusion

Perona *et al.* [71] were the first to introduce the theory of heat conduction (which occurs by diffusion) into the field of image processing to remove speckle noise. The heat conduction equation for two dimensions gives the temperature $T_t(\vec{x})$ at *diffusion* time t and spatial position $\vec{x}$ and is given by the partial differential equation

$$\frac{\partial T_t(\vec{x})}{\partial t} = c \nabla \bullet (\nabla T_t(\vec{x})), \ \nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y} \right)^T, \tag{3.80}$$

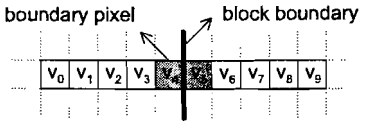


Figure 3.24: Filter support for the two-mode filter.

with the gradient operator ∇ and diffusion coefficient c. Now translate temperature to luminance in image n, $T_t(\vec{x}) = F_t(\vec{x}, n)$ with x and y as pixel coordinates, then for initial condition $F_t(\vec{x}, n)|_{t=0} = F_0(\vec{x}, n)$ (the original decoded image), the solution is the Gaussian blurred image, where the blurring becomes heavier as more diffusion time elapses. However, the blurring is isotropic (rotationally invariant), resulting in edges to become blurred too. The solution given in [71] is to incorporate a *diffusion coefficient* $c(\vec{x}, n)$ to make the diffusion spatially and directionally dependent, changing Equation 3.80 into

$$\frac{\partial F_t(\vec{x}, n)}{\partial t} = \nabla \bullet (c(\vec{x}, t)\nabla F_t(\vec{x}, n)) = c(\vec{x}, t)\nabla \bullet (\nabla F_t(\vec{x}, n)) + \nabla c(\vec{x}, t)\nabla F_t(\vec{x}, n), \tag{3.81}$$

which reduces to the original equation if the diffusion coefficient $c(\vec{x}, t) = c$ is a constant. The diffusion coefficient must be equal to 1 in smooth areas and 0 in areas where smoothing is not desired. An example of such a function is

$$c(\vec{x}, t) = \frac{1}{1 + \left(\frac{\|\nabla F(\vec{x}), n)\|}{\lambda}\right)^2}, \tag{3.82}$$

where λ is the gradient below which blurring occurs and above which enhancement occurs. In [97], λ is set to a multiple of the average gradient across the block edge. Furthermore, deviation from the original image is penalized by biasing the diffusion equation

$$\frac{\partial F_t(\vec{x}, n)}{\partial t} = \sigma(F_t(\vec{x}, n) - F_0(\vec{x}, n)) + \nabla \bullet (c(\vec{x}, t)\nabla F_t(\vec{x}, n)), \tag{3.83}$$

where σ is proportional with the MSDS and the number of non-zero AC coefficients in the coding block.

The two-dimensional equations can be easily extended to three-dimensional equations, adding the temporal dimension to the definition of the gradient [41]. This non-motion compensated method reduces also the temporal flickering.

3.7 Hybrid algorithms

Methods found in the literature often use a combination of de-blocking approaches in the previous sections. A widely used MPEG post-processing algorithm found in many multimedia PC applications video coders and encoders (AviSynth, DivX, FFDirectShow, XviD, amongst others) and players (MPlayer, VideoLan Client, etc.) is a combination of a de-blocking filter and a de-ringing filter that can be applied individually or both. The filters are recommended as standard post-processing in the MPEG-4 standard [27]. The de-blocking filter is often used as a reference and will be explained next.

3.7.1 MPEG-4 two-mode de-blocking

For an explanation of the filter, again observe Figure 3.24 with the vector $\vec{v} = (v_0, \ldots, v_9)^T$ of sample values. The first step of the de-blocking process is to determine which of the two modes to apply, *smooth region mode* or *default mode*. The smooth region mode is chosen if the flatness measure $S(\vec{v})$ is equal to or greater than a certain flatness threshold $T_2 = 6$. The flatness measure

counts how many samples in the vector $\vec{v}$ have an inter-pixel difference less than or equal to a threshold $T_1 = 2$:

$$S(\vec{v}) = \sum_{i=0}^{8} \phi(v_i - v_{i+1}), \text{ with } \phi(v_i - v_{i+1}) = \begin{cases} 1, \text{ if } |v_i - v_{i+1}| \leq T_1, \\ 0, \text{ if } |v_i - v_{i+1}| > T_1. \end{cases} \tag{3.84}$$

In *smooth region mode*, no filtering ($\hat{\vec{v}} = \vec{v}$) is applied if $v_{max} - v_{min} \geq 2 \cdot qscale$, with $v_{max} = \max\{v_i \mid 0 \leq i < 8\}$, $v_{min} = \min\{v_i \mid 0 \leq i < 8\}$ and $qscale$ is the quantization scale of the block containing pixel v_5. Otherwise, strong filtering with a 9-tap filter is applied. The filtered vector is then calculated with:

$$\hat{v}_n = \frac{1}{16} \sum_{j=-4}^{4} b_k \cdot p_{n+k}, 1 \leq n \leq 8, \text{ where} \tag{3.85a}$$

$$p_0 = \begin{cases} v_0, \text{ if } |v_1 - v_0| < qscale, \\ v_1, \text{ otherwise.} \end{cases} \tag{3.85b}$$

$$p_9 = \begin{cases} v_9, \text{ if } |v_9 - v_8| < qscale, \\ v_8, \text{ otherwise.} \end{cases} \tag{3.85c}$$

$$p_m = \begin{cases} p_0 \text{ , if } m < 1, \\ v_m, \text{ if } 1 \leq m \leq 8, \\ p_9 \text{ , if } m > 8. \end{cases} \tag{3.85d}$$

$$\vec{b} = (b_{-4}, \ldots, b_4)^T = (1, 1, 2, 2, 4, 2, 2, 1, 1)^T. \tag{3.85e}$$

In *default mode*, only the boundary pixels are affected. To analyse the local statistics, an approximation of the 4-point DCT is applied on the pixels left to the block boundary $\vec{v}_L = (v_1, v_2, v_3, v_4)^T$, right to the block boundary $\vec{v}_R = (v_5, v_6, v_7, v_8)^T$, and *at* the block boundary $\vec{v}_C = (v_3, v_4, v_5, v_6)^T$, for wave number 3, the wave form most contributing to the blocking artefact (cf. Equation 2.12 with $N = 4$ and $u = 3$). This yields the transform coefficients:

$$a_L = \frac{1}{8}(2 \quad -5 \quad 5 \quad -2)\vec{v}_L, \tag{3.86a}$$

$$a_R = \frac{1}{8}(2 \quad -5 \quad 5 \quad -2)\vec{v}_R, \tag{3.86b}$$

$$a_C = \frac{1}{8}(2 \quad -5 \quad 5 \quad -2)\vec{v}_C. \tag{3.86c}$$

If $|a_C| \geq qscale$, the boundary pixels remain unaffected. If not, the pixels are updated by scaling the transform coefficient, smoothing flatter areas more than complex areas:

$$\hat{v}_4 = v_4 - d \text{ and } \hat{v}_5 = v_5 + d, \text{ where} \tag{3.87}$$

$$d = \text{clip}\left[\frac{5}{8}(\hat{a}_C - a_C)\right]_0^{\frac{v_4 - v_5}{2}} \text{ and } \hat{a}_C = \text{sign}(a_C) \cdot \min(|a_L|, |a_C|, |a_R|). \tag{3.88}$$

The reference implementation was presented by Kim *et al.* [36]. Note that for low bit rate Internet Video, the low pass filter is chosen more frequently. Cahill *et al.* [3] distinguishes a third mode, an *intermediate mode* for which the boundary pixels are filtered with a 3×3 Gaussian filter if $|v_6 - v_7| < qscale$.

Methods found in the industry are often based on this two-mode filtering scheme because of its ease of implementation (e.g. [33, 35, 43]). For example, MacInnis *et al.* [52] proposes a similar technique for interlaced sequences.

3.8 Discussion

This section showed an overview of approaches that are commonly found in the literature and industry. Though researchers all over the world invented solutions that provide good results, the techniques are not always suitable for a real time implementation due to their complexity. Especially iterative techniques are too expensive for consumer electronics products and, therefore, not found in the industry. Some complex techniques are available for off-line processing as software plug-ins. Table 3.1 shows a comparison of the discussed methods.

method	domain	iterative	complexity
adaptive	spatial	no	low
adaptive	DCT	no	low/intermediate
adaptive	wavelet	no	high
adaptive	temporal	no	low
POCS	all	yes	high
CLS	all	yes	high
MAP	all	yes	high
anisotropic diffusion	all	yes	high

Table 3.1: Comparison of the different blocking artefact repair approaches.

We explicitly do not give a comparison in terms of picture quality improvement. The results of each algorithm strongly depend on its implementation and a real comparison would require an exhaustive assessment of all methods. Instead, we will make some observations that lead us to the proposal of a new approach to deal with the blocking artefact.

All methods for de-blocking report that non-adaptive smoothing causes excess blurring of the decoded image. Simply constraining the signal bandwidth to remove high-frequency block edges also removes the natural high frequencies that are present in an image, as an ideal low-pass filter cannot be realized. To avoid smoothing of natural edges, previous work suggested to exploit edge information. The detection of edges from highly compressed images is, however, a very difficult task. Edges are sensitive to quantization and, therefore, distorted [80]. To this respect, filtering in the sub-band domain gives results that are more satisfying.

The success of set-theoretic methods like POCS depends on the constraint sets. Bandwidth constraints on the entire image yield over-smoothing, whereas methods that only affect the edge pixels are not sufficient for very low bit rates (for example, the constraint-based algorithm in proposed by Yang *et al.* [99]). In the extreme case that a sequence does not have the bandwidth to code for any AC coefficients, the blocking effect concerns all pixels in the block. An algorithm has, therefore, to take into account the spatial masking properties of the human visual system. Though there exists a multitude of constraints that can be combined, the solution space still leaves too many feasible images, making POCS-based methods not perform well [19].

Mayor drawback of all iterative methods (POCS, CLS) and statistical methods (MAP, anisotropic diffusion) is that they require several iterations to converge to a solution, making the techniques less suitable for real-time implementations. This observation is supported by the number of solutions found in the industry. Most registered solutions are found in spatial adaptive approaches that have low computational complexity and do not require mayor adaptation of existing architectures. We focus on a solution that can be implemented in a real-time system.

Some algorithms use quantization information from the bit stream as an input for post-processing algorithms. These include motion vectors and quantization scale parameters. Often, these parameters are not available for the post-processor because the decoder is an embedded component that has a bit stream as input and a sequence of images as output. Parameter extraction requires modification of the decoder, so we aim at "pure" post-processing techniques for which bit stream parameters are not available.

A drawback of most of the algorithms discussed before is that they are initially adopted from still image de-blocking techniques. The algorithms cannot be simply extended to video because

the blocking artefact behaves more complex in digital video. A model that takes into account the process of motion compensated prediction is desirable.

Considering the aspects discussed above, we seek a solution that meets the following requirements:

- The filtering scheme has to be adaptive, preferably not based on an edge map.
- The algorithm must have a limited computational complexity to make it suitable for real-time implementations.
- The technique must model the process of motion compensation that takes place in block-DCT-based video coding schemes.
- To avoid modification of existing decoders, the post-processor does not use coding parameters, except for the size and location of the coding grid.

We will keep these requirements in mind in the development and evaluation of a de-blocking filter in the next chapters.

Chapter 4

Motion compensated blocking artefact repair

Almost all algorithms used for blocking artefact repair are based on adaptive filtering of a single image. The filters can be driven by the coding parameters from the bit stream or by parameters derived from local characteristics in the pixel domain, DCT domain, sub-band domain, etcetera. Most de-blocking algorithms for video reviewed in the previous chapter were adopted from still image (JPEG) de-blocking methods that are essentially two-dimensional. Few methods exploit the fact that video has a third, temporal dimension [19].

The risk of spatial methods is that blocking artefacts coinciding with natural edges are filtered too. Moreover, for video, the blocking artefact is not limited to the fixed 8×8 coding grid, but instead it propagates to the current picture by motion compensated (MC) prediction, especially if the residual coding mechanism does not provide sufficient data to compensate for the artefact. The propagation of the block edges in an MPEG-2 sequence is illustrated in Figure 4.1 and Figure 4.2. Where traditional algorithms assume an encoding scheme that solely performs 8×8 block DCT followed by quantization, we extend the encoder model with the motion estimation and compensation process. As shown in Chapter 2, the process of motion estimation yields a residual image that is calculated by

$$F_\varepsilon(\vec{x}, n) = F_o(\vec{x}, n) - F_p(\vec{x}, n), \ \forall \vec{x} \in \mathcal{I}, \tag{4.1}$$

where F_ε is the residual image, F_o is the original sequence, and F_p is the MC prediction. The residual image is then block DCT transformed and quantized. The decoder de-quantizes and inverse transforms the residual image, then *adds* this to the MC prediction. For bi-directionally predicted pictures, a de-blocking algorithm should have access to a future reference image that is only available in the decoder or at the end of a group of pictures (GOP). For optimal blocking artefact reduction, a post-processor should de-block these reference frames before predicting derived pictures. Such a filter scheme is proposed in the AVC standard [78] and is applied in several implementations ([17], [31]). However, we do not have access to the decoder parameters, so we lack information about the picture types, reference frames, quantization scale parameters, and the motion vectors. Instead, we try to estimate the motion from neighbouring frames to make a MC prediction of the image to be filtered.

Our starting point for MC artefact repair is a robust motion estimator which provides accurate motion vectors. Motion vectors from the decoder relate a frame with one or more reference frames, which are generally not the neighbouring frames and which are not available behind the decoder. We need a dense and accurate motion estimation based on the neighbouring (previous or next in sequence) frames [19], that yields the *true motion* in a sequence instead of the best matching block. Philips Electronics developed several robust and accurate motion estimation algorithms for different MC video processing tasks such as noise-filtering, picture rate conversion, and de-interlacing. A commercial available IC is described in [22].

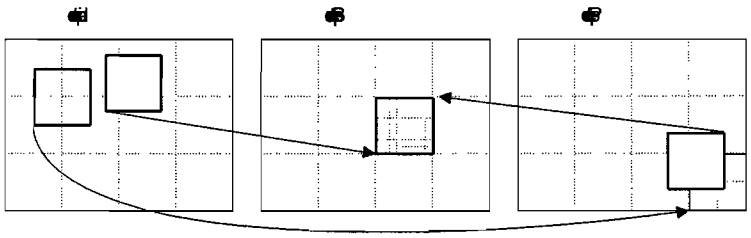


Figure 4.1: Propagation of the blocking artefact.

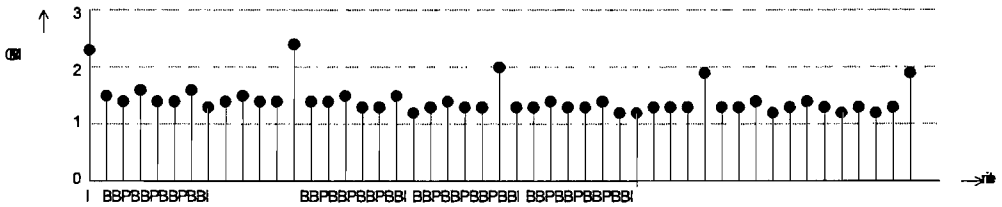


Figure 4.2: Propagation of the blocking artefact from I pictures to P and B pictures. The graph shows the GBIM of frames 505 to 553 from the 500*kbps* MPEG-2 encoded *tennis* sequence.

For the experiments used in this project, a state-of-the-art Philips propriety motion estimator was used, requiring three consecutive frames. The estimator is based on a sub-pixel accurate three-dimensional recursive search (3-D RS) block matcher, assisted by a global motion model to account for panning, zooming, rotation, or travelling of the camera [23, pages 175–226]. 3-D RS block matchers give a high spatial and temporal consistent (smooth) motion vector fields with relatively low computational effort [21]. The three-frame approach has the advantage over two-frame methods that it is able to solve the occlusion[1] problem and it is more robust in the sense that spurious vectors occur less frequently [50]. Drawback of the estimator is that it needs access to three frames and that it needs on average twice as much computational effort than a classical two-frame 3-D RS block matcher. To prevent the estimator from biasing, the original pictures are fed to the estimator prior to post-processing.

From an initial approach, based on an observation in the literature, we will propose an algorithm and describe some experiments in Section 4.1. From the observations, we come to an improved model in Chapter 4.2, working towards a solution that can be implemented next to a three-frame based motion estimator such that the existing architecture does not change considerably in Section 4.3.

4.1 Step discontinuity compensation

Refer to a two-frame motion estimation method as shown in Figure 4.3. Suppose the block emphasized in the Figure contains the blocking artefact, located at the coding grid in frame n. Generally, its MC prediction from the previous frame $n-1$ will not be located at the coding grid. As a result, the predicted image may not contain the blocking artefact at the same spatial position as the original decoded block. We can exploit this fact to use the MC prediction to compensate for the data near the coding block edges.

A similar idea has been proposed earlier by Park *et al.* [69], who exploited the self-similarity in a natural image to remove discontinuities. In Figure 4.4, a *range block* $b_{\vec{x}_s,n}(x,y)$, shifted horizontally by half a block size with respect to the coding grid, overlaps two neighbouring coding grid blocks, showing a horizontal discontinuity (vertical edge) in the centre. The algorithm searches its best matching block located at the coding grid ($\vec{x}_b \in \mathcal{B}$) within the same image n, called the

[1] Occlusion is the covering and uncovering of an object (or the background) by another object.

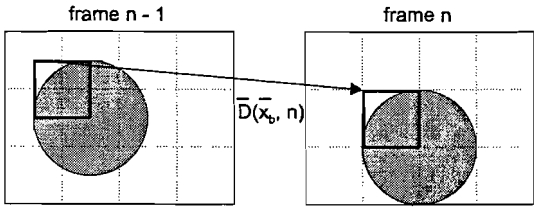


Figure 4.3: Two-frame motion estimation.

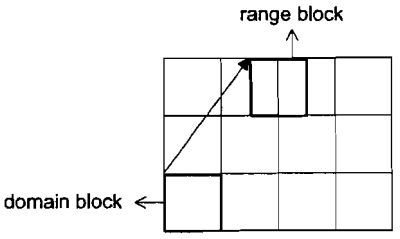


Figure 4.4: Domain and range blocks used by Park *et al.*. The solid grid is the coding grid. The arrow represents the best match.

domain block $b_{\vec{x}_b,n}(i,j)$. Because the match is less reliable at the block boundary, the MSE metric is changed into a weighted MSE ($WMSE$):

$$WMSE = \sum_{i=0}^{7}\sum_{j=0}^{7} w_i w_j \left(b_{\vec{x}_s,n}(i,j) - b_{\vec{x}_b,n}(i,j)\right)^2 . \tag{4.2}$$

The weights w_i depend on the type of the two overlapped coding blocks: both blocks can be low-frequency blocks (case 1), one block can be a low-frequency block and the other a high-frequency block (case 2), and both blocks can be high-frequency blocks (case 3):

$$\vec{w} = \begin{cases} (0.2, & 0.15, & 0.1, & 0.05, & 0.05, & 0.1, & 0.15, & 0.2)^T, \text{ in case 1,} \\ (0.15, & 0.15, & 0.1, & 0.1, & 0.1, & 0.1, & 0.15, & 0.15)^T, \text{ in case 2,} \\ (0.135, & 0.3, & 0.125, & 0.11, & 0.11, & 0.125, & 0.3, & 0.35)^T, \text{ in case 3.} \end{cases} \tag{4.3}$$

The classification is made in the DCT domain, where a block is a *low-frequency block* if all DCT coefficients other than $B(0,0), B(1,0)$ $B(0,1)$ are zero, and a block is a *high-frequency block* if one or more coefficients with $u > 2$ and $v > 2$ are nonzero. Using the fact that the range block contains a discontinuity in the centre and that domain block is continuous in the centre, the range block can be recovered by a linear combination of the original decoded range block and its corresponding domain block, which is further left undocumented. The same procedure follows for the vertical blocking artefact.

Motion estimation on a shifted grid

We extend the algorithm above to the temporal domain, where the range block is in the current frame and the domain block is more likely to be found in a temporal neighbouring frame by the process of motion estimation. For this reason, let us define a motion estimation grid that is shifted by half the block size, both horizontally and vertically:

$$\forall\, \vec{x}_s = \begin{pmatrix} x_s \\ y_s \end{pmatrix} \in \mathcal{I},\ \exists\, \mathcal{B}_{\vec{x}_s} = \{\vec{x} \in \mathcal{I} \mid x_s \leq x < x_s + 8 \wedge y_s \leq y < y_s + 8\}, \tag{4.4}$$

with the upper left corner $\vec{x}_s \in \mathcal{S}$ given by:

$$\mathcal{S} = \{\vec{x} \in \mathcal{I} \mid x \bmod 8 = 4 \wedge y \bmod 8 = 4\ \wedge x \neq N - 4 \wedge y \neq M - 4\}, \tag{4.5}$$

as illustrated in Figure 4.5.

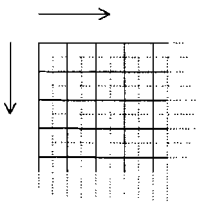


Figure 4.5: Coding block grid (solid) versus motion estimation grid (dotted).

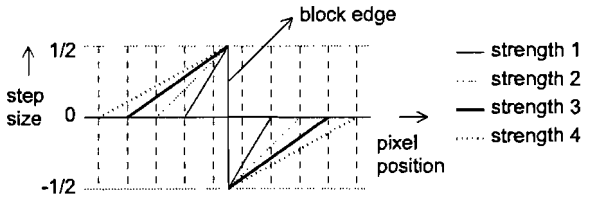


Figure 4.6: One-dimensional compensation signals with different strengths to remove the (positive) step discontinuity at the block boundary.

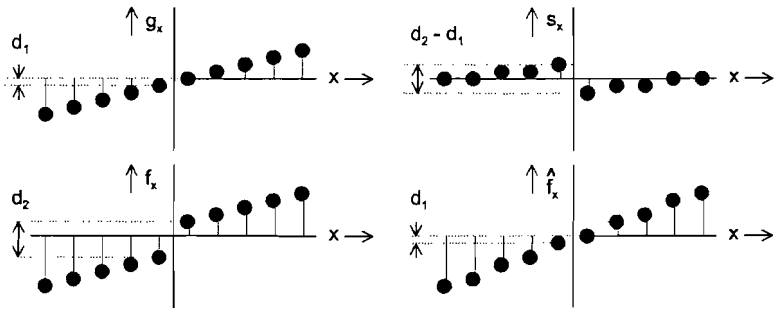


Figure 4.7: Discontinuity compensation, the vertical axis representing the block boundary.

Discontinuity compensation

Wang *et al.* [90] model the blocking artefact as a two-dimensional step discontinuity due to impaired DC coefficients. To compensate for this step, a two-dimensional compensation signal is added to the blocky signal. The length of the compensation signal, equal to its strength, depends on the visibility of the artefact, derived from the DCT coefficients of the decoded image. For the one-dimensional case, these linear compensation signals are depicted in Figure 4.6, for discontinuities that are positive for increasing pixel position.

A first approach is to compensate for the step located at the block boundary, given the predicted step size from the MC predicted image. For an initial solution, observe Figure 4.7 for the one-dimensional approach. Let signal $\vec{f}$ be a signal with a discontinuity d_2 across the block boundary (the vertical coordinate axis). Let $\vec{g}$ be its MC prediction, not showing the discontinuity at the location of the block border. At the block boundary, $\vec{g}$ has a natural discontinuity d_1. If we want to compensate for the false edge, we can add a for example a linear compensation signal $\vec{s}$ with discontinuity $d_2 - d_1$ at the position of the border. Signal $\hat{\vec{f}}$ gives the de-blocked result with the original discontinuity d_1.

Compensation signal

From earlier observations, we have seen that for very low bit rates and high velocity motion sequences, the temporal redundancy in the sequence is reduced so that more coding blocks require updating. Because the encoder has to divide the available bits over all coding blocks, only a few (or even no) AC coefficients are encoded. With only low frequency components present, it is not sufficient to filter the border pixels alone. Therefore, we use the compensation signal with strength 3 in Figure 4.6, defined by

$$w_i = \frac{i}{N-1}, \; 0 < i < \frac{N}{2}, \tag{4.6}$$

where N is the horizontal (or vertical) block size. For $\frac{N}{2} \leq i < N$ the signal has a similar form (cf. Figure 4.8). Such a compensation signal was also chosen by Kryukov *et al.* [39].

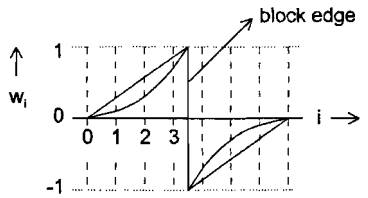


Figure 4.8: Linear and sinusoidal compensation signal to remove the (positive) step discontinuity at the block boundary.

The use of linear compensation functions might cause a new artefact in the centre of the coding block, because the continuity of the derivative is not guaranteed. We solve this with a smooth, sinusoidal compensation signal, which has a flat slope at in the centre of the coding block:

$$w_i = -\cos\left(\frac{\pi * i}{N-1}\right) + 1,\ 0 < i < \frac{N}{2}. \tag{4.7}$$

4.1.1 Experiments

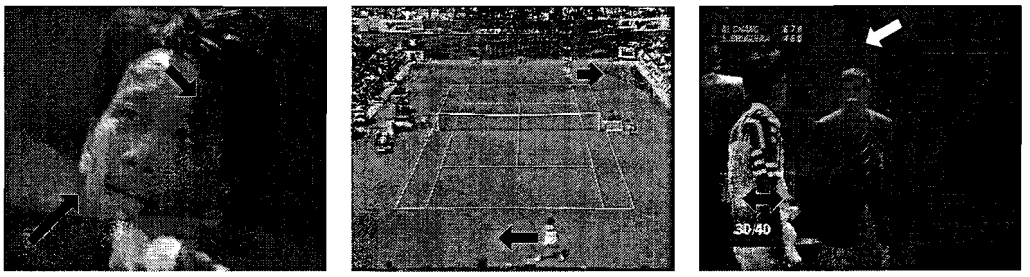
Though the concept described above has not yet been optimized, we did perform some experiments to verify the concept. For testing of the algorithms, we initially used a *tennis* game sequence which has lots of complex motion (pan and zoom), different scales of detail (faces, court lines, logos), and difficult objects to track with a motion estimator (occlusion, transparent scoreboard overlay). The sequence consists of five shots. The first shot shows a close-up of a tennis player serving a ball (frames 1 to 83). The motion magnitude clips to the maximum magnitude allowed by MPEG-2. Next, a long shot of the tennis court is shown, with the sharp court lines and brand logos (frames 84 to 373). The next two shots show a medium shot (frames 374 to 442) and a close-up (frames 443 to 493) of a walking tennis player in front of a panning background. The sequence ends with a pan showing the audience (frames 494 to 600). Some screen shots are shown in Figure 4.9.

The video material is progressive and originally MPEG-2 compressed with a bit rate of 1,984 *kbps*. The codec that has been used has been developed by the MPEG Software Simulation Group (*MSSG*)[60]. The format is known as Common Intermediate Format (CIF) (352 × 288 pixels), at a frame rate of 25 frames per second. In Table 4.1, the effect of the bit rate on the distortion is measured in PSNR and GBIM. Figure 4.10 shows the effect quantitatively.

bit rate [*kbps*]	MSE	$PSNR$ [dB]	M_{hGBIM}	M_{vGBIM}	M_{GBIM}
1,984	0	∞	1.12562	1.29680	1.21121
1,500	53.995	30.807	1.19107	1.39440	1.29274
1,000	86.399	28.766	1.27510	1.50659	1.39085
750	114.317	27.550	1.38079	1.63678	1.50879
500	167.202	25.898	1.66603	1.99692	1.83148
300	287.932	23.538	2.29145	2.91899	2.60522
250	306.671	23.264	2.40951	3.19175	2.80063

Table 4.1: Bit rate versus distortion, for the luminance signal only.

The algorithm proposed above has been tested on the 500*kbps* MPEG-2 encoded and decoded tennis sequence, for the linear compensation signal (`comp. linear`), the sinusoidal signal (`comp. sine`), and a recursive alternative (`comp. rec.`) which uses the filtered output to make the next prediction, so preventing the propagation of the artefact. The motion estimation of the latter method was based on the original decoded sequence. The algorithms have been compared with the MPEG-4 de-blocking algorithm proposed in [36] and explained in Subsection 3.7.1. The



(a) frame 76 (b) frame 280 (c) frame 440

(d) frame 472 (e) frame 524

Figure 4.9: Frames from each shot of the original *tennis* sequence. White overlayed arrows indicate motion of the background, black arrows that of individual "objects". The arrow tail lengths give an indication of the motion magnitude.

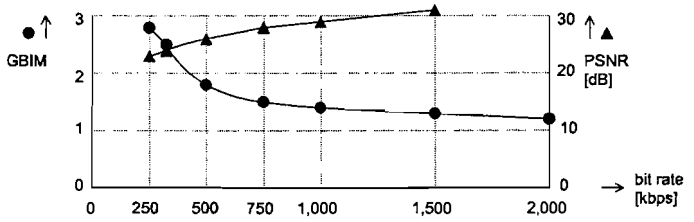
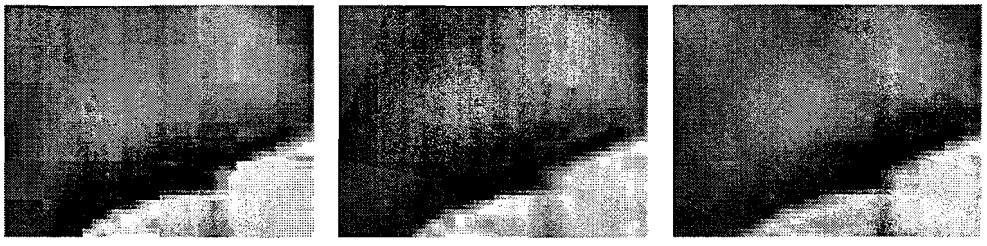


Figure 4.10: Bit rate versus distortion in GBIM (bullets) and PSNR (triangles).



(a) original (b) MPEG-4 (c) proposed

Figure 4.11: Results of different post-processing algorithms on an image detail of frame 481 from the *tennis* sequence. (a) shows the decoded image, (b) the MPEG-4 de-blocked image, and (c) is the recursively de-blocked image with the proposed sinusoidal compensation signal.

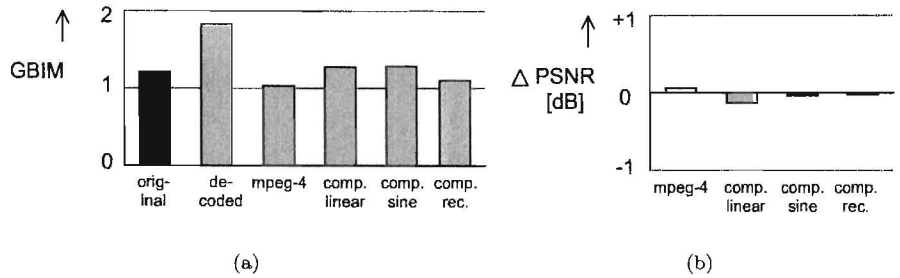


Figure 4.12: Objective test results for the step edge compensation algorithm in block impairment (GBIM) (a) and image fidelity (PSNR) (b).

results in terms of image fidelity (PSNR) and block impairment (GBIM) are given in Table 4.2 and in Figure 4.12. Because the motion compensated filter needs the first and the last frame for three-frame motion estimation, the non-motion compensated algorithm has a slight advantage. Therefore, we calculated the metrics only for the intermediate frames. Figure 4.11 shows the post-processing results for an image detail of frame 471, where a part of the tennis player's shoulder (light) is shown, moving against a panning background (dark).

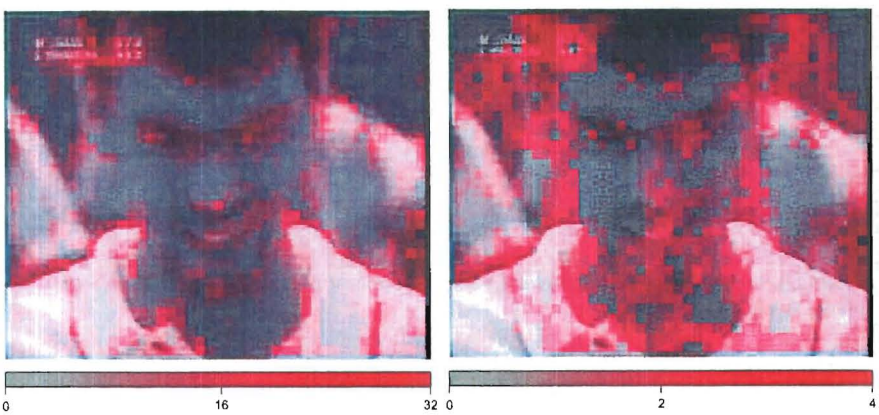
The PSNR metric measures the image fidelity. To compare the improvement with respect to the original decoded sequence, the table includes the *PSNR improvement*, $\Delta PSNR$. The improvements or deteriorations are within tenths of *dB*s. A better indicator of the blocking artefact is given by the GBIM. However, PSNG and GBIM are averaged over each image and over the entire sequence, so severe local artefacts will not appear in the metrics. This indicates the need for a subjective evaluation of the sequences.

algorithm	MSE	$PSNR$ $[dB]$	$\Delta PSNR$ $[dB]$	M_{hGBIM}	M_{vGBIM}	M_{GBIM}
original	0	∞	∞	1.12552	1.29699	1.21126
decoded	166.901	25.906	0	1.66517	1.99610	1.83064
mpeg-4 de-blocked	164.643	25.965	+0.059	0.97651	1.09317	1.03484
compensated (linear)	172.028	25.774	-0.132	1.18346	1.35707	1.27027
compensated (sinusoidal)	167.609	25.887	-0.019	1.19348	1.37107	1.28228
compensated (recursive)	167.800	25.883	-0.023	1.02726	1.18095	1.10411

Table 4.2: Objective test results for the step edge compensation algorithm.

4.1.2 Evaluation

The results show that, when not imposing restrictions on the motion vector reliability, the proposed algorithm performs reasonably well in terms of blocking impairment, even though slightly deteriorating the image fidelity with respect to its initial decoded reference. Subjective observations show that the algorithm correctly smoothes the artefact where the motion vectors are reliable, yet introduces many false edges where the predictions are not reliable. This can be seen in high-motion sequences, areas with sharp edges and high texture, and in occlusion areas, as in Figure 4.11(c). The reason is that even a sub-pixel shift of the motion vectors cause the predicted step edge to become unreliable. The robustness of any motion compensated filter, therefore, depends on the reliability of the motion vectors, for which we introduce two metrics.



(a) (b)

Figure 4.13: Frame 481 of the decoded tennis game sequence, showing a moving tennis player at a panning background. The overlay (a) shows the nonzero clipped MAD and the rounded non-zero LSI (b) values for each coding block.

Motion vector reliability

The usual reliability metric for motion vectors is the Sum of Absolute Differences or SAD, summing the errors between the original block and its MC prediction:

$$SAD(\vec{x}_b, n) = \sum_{\vec{x} \in \mathcal{B}_{\vec{x}_b}} |\varepsilon(\vec{x}, n)|. \tag{4.8}$$

Another similar metric commonly found in the literature is the Mean Absolute Difference (MAD), which is in fact the SAD averaged over the number of pixels in a block:

$$MAD(\vec{x}_b, n) = \frac{1}{|\mathcal{B}_{\vec{x}_b}|} \sum_{\vec{x} \in \mathcal{B}_{\vec{x}_b}} |\varepsilon(\vec{x}, n)| = \frac{1}{|\mathcal{B}_{\vec{x}_b}|} SAD(\vec{x}_b, n). \tag{4.9}$$

As a measure for the inconsistency of a motion vector, we introduce the Local Spatial Inconsistency (LSI) metric, based on the smoothness metric in [23, pages 175–226], defined for a block $\mathcal{B}_{\vec{x}_b}$ as:

$$\begin{aligned} LSI(\vec{x}_b, n) = \frac{1}{8} \sum_{i=-1}^{1} \sum_{j=-1}^{1} & \left| D_x(\vec{x}_b, n) - D_x\left(\vec{x}_b - \begin{pmatrix} 8i \\ 8j \end{pmatrix}, n\right) \right| + \\ & \left| D_y(\vec{x}_b, n) - D_y\left(\vec{x}_b - \begin{pmatrix} 8i \\ 8j \end{pmatrix}, n\right) \right|. \end{aligned} \tag{4.10}$$

The LSI measure calculates the first order approximation of the vector differences between a block and its eight spatial neighbouring blocks.

For frame 481 of the tennis sequence, the reliability metrics are overlayed in Figure 4.13. Where the reliability is low, the compensation algorithm is more likely to introduce false edges, as the detail in 4.11(c) shows.

4.2 Three-frame approach

The use of a three-frame motion estimator was introduced to overcome the occlusion problems with the two-frame approach. The three-frame motion estimator gives, apart from the previous

field, also access to the next frame. The problem that rises is to make a prediction out of two frames instead of one.

The use of order statistical filters, such as the median filter, has been proven successful in interpolation operations such as picture rate conversion. Picture rate conversion typically deals with interpolating data from neighbouring frames. At locations degraded by quantization, we can possibly use interpolated data from neighbouring frames to reduce the blocking artefact as in the previous section. To make a MC prediction from two neighbouring frames, the *dynamic median* yields good results, as assessed by De Haan in [23, pages 136–143]. The dynamic median F_{dyn} is defined as:

$$F_{dyn}(\vec{x},n) = \text{median}\left\{\begin{array}{c}\frac{F(\vec{x},n-1)+F(\vec{x},n+1)}{2},\\ F(\vec{x}-\vec{D}(\vec{x},n),n-1),\\ F(\vec{x}+\vec{D}(\vec{x},n),n+1)\end{array}\right\}. \tag{4.11}$$

Another simple prediction obtaining similar objective and subjective results is the motion compensated average, defined as

$$F_{mca}(\vec{x},n) = \frac{F(\vec{x}-\vec{D}(\vec{x},n),n-1)+F(\vec{x}+\vec{D}(\vec{x},n),n+1)}{2}. \tag{4.12}$$

Because the current field is also available, we can also use this information for making a prediction. If we take into account that, for occlusion areas, the covering areas are better represented in the previous frame and the uncovering areas are better represented in the next frame, we propose an interpolation that is weighted by the sum of absolute differences between the two:

$$F_{mcwa}(\vec{x},n) = \frac{SAD_{fwd}F(\vec{x}-\vec{D}(\vec{x},n),n-1)+SAD_{backwd}F(\vec{x}+\vec{D}(\vec{x},n),n+1)}{SAD_{fwd}+SAD_{backwd}}, \tag{4.13}$$

where the match errors are defined as

$$SAD_{fwd} = \sum_{\vec{x}\in\mathfrak{B}_{\vec{x}_b}} |F(\vec{x}-\vec{D}(\vec{x},n),n-1)-F(\vec{x},n)| \tag{4.14}$$

for the forward prediction and

$$SAD_{backwd} = \sum_{\vec{x}\in\mathfrak{B}_{\vec{x}_b}} |F(\vec{x}+\vec{D}(\vec{x},n),n+1)-F(\vec{x},n)| \tag{4.15}$$

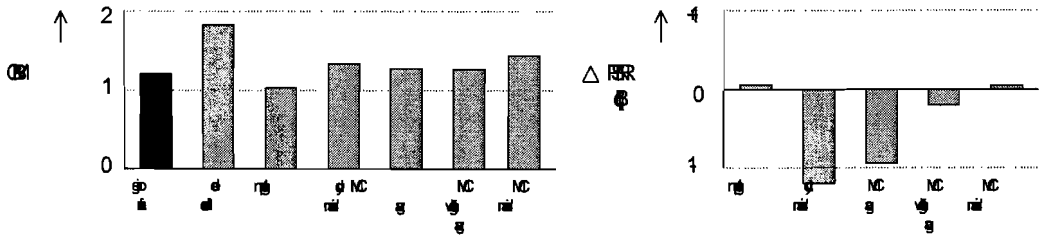
for the backward prediction.

Lastly, we will evaluate a MC median, defined as:

$$F_{mcmed} = \text{median}\left\{F(\vec{x}-\vec{D}(\vec{x},n),n-1),F(\vec{x},n),F(\vec{x}+\vec{D}(\vec{x},n),n+1)\right\}. \tag{4.16}$$

The four interpolation methods above are performed on the shifted coding grid introduced in the previous section and then compared to the original sequence, with the results listed in Table 4.3 and Figure 4.14.

As shown in Table 4.3, the motion compensated median gives results closest to the original sequence. Though the median filter yields the most reliable interpolation, the figures in terms of block impairment reveal that algorithm just performs moderate. This can also be seen in the pictures in Figure 4.15. We can explain this by the fact that the median filter often chooses the original decoded value, which is the blocky source. The MC weighted average does not use the original pixel directly, gives the second best results in image reliability, and performs average on block impairment. It should be noted that the MC median *does* give good subjective results with respect of other noise such as ringing and mosquito noise. Another drawback of the MC median is that moving objects change in size and that very small moving objects are erased entirely.



(a) (b)

Figure 4.14: Objective test results for the step edge compensation algorithm in block impairment (GBIM) (a) and image fidelity (PSNR) (b).

algorithm	MSE	$PSNR$ $[dB]$	$\Delta PSNR$ $[dB]$	M_{hGBIM}	M_{vGBIM}	M_{GBIM}
decoded	166.901	25.906	0	1.66517	1.99610	1.83064
dynamic median	220.239	24.701	-1.204	1.24643	1.44100	1.34372
MC average	206.341	24.984	-0.921	1.19235	1.36294	1.27765
MC weighted average	174.442	25.714	-0.192	1.18930	1.34995	1.26963
MC median	164.880	25.959	+0.053	1.34664	1.55070	1.44867

Table 4.3: Objective test results for interpolation.

To investigate the combination of the interpolation algorithms with the step edge compensation algorithm from the previous section, we will apply the sinusoidal step edge compensation algorithm again, described in the previous chapter. The results can be found in Table 4.4 and in Figure 4.16.

The results also show that reusing the filtered frame as a reference for the next interpolation increases the performance of the filter in terms of blocking impairment. Assuming the previous frame to be already de-blocked, the image can be used for predicting the erroneous information in the current field. This proves again that recursive filtering yields better results, because the artefact will propagate less to the consecutive frames. The motion vectors used for interpolation were extracted from a shifted grid, on the original, non-filtered sequence.

Soft thresholding

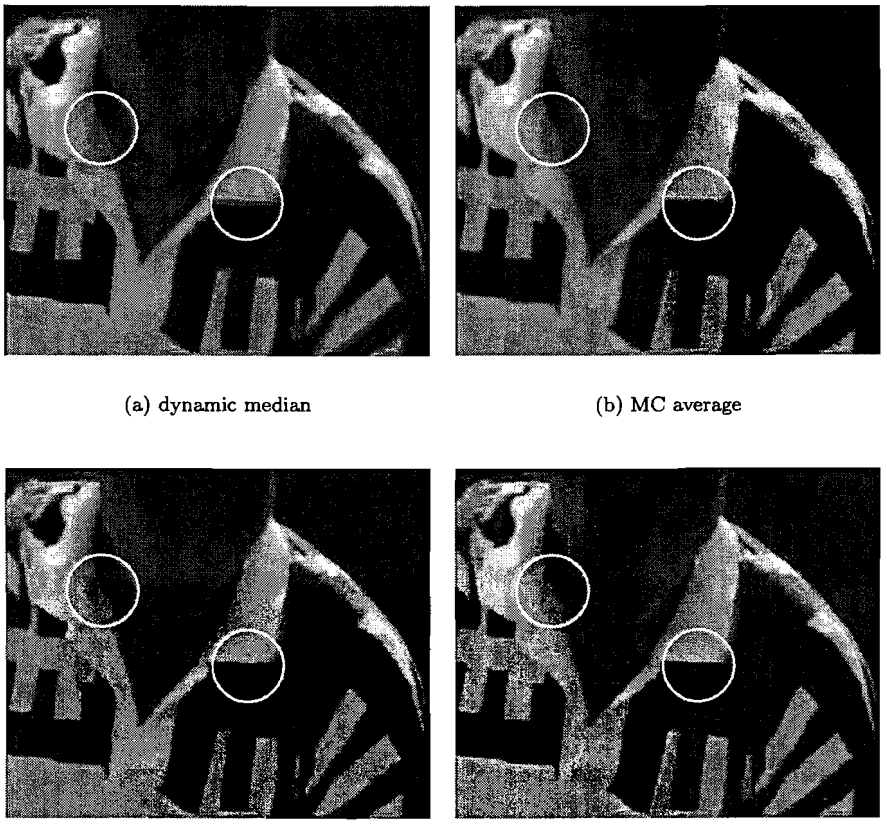
In the previous section, we introduced two reliability metrics for the motion vectors. Here, we will assess the influence of thresholding on the mean absolute difference (MAD) and the local spatial inconsistency (LSI). Two-mode, or more general, multi-mode filtering techniques in the literature, often set a threshold to choose between one type of processing and another type of processing. In ambiguous areas, this leads to a new artefact where the algorithm switches between the different kinds of processing. This effect can be seen in Figure 4.17.

To avoid this kind of artefact, we will use a soft thresholding scheme that gradually switches from full post-processing to no post-processing as the reliability decreases, as depicted in Figure 4.18. The output image F_{soft} is then calculated from:

$$F_{soft}(\vec{x}, n) = kF(\vec{x}, n) + (1 - k)F_{filtered}(\vec{x}, n), \tag{4.17}$$

where $F(\vec{x}, n)$ is the decoded sample, $F_{filtered}(\vec{x}, n)$ is the filtered sample, and k is the weighting function from Figure 4.18:

$$k = \begin{cases} \frac{MAD_{3f}}{MAD_{max}}, & \text{if } MAD_{3f} < MAD_{max}, \\ 1 & \text{, if } MAD_{3f} \geq MAD_{max}. \end{cases} \tag{4.18}$$



(a) dynamic median

(b) MC average

(c) MC weighted average

(d) MC median

Figure 4.15: Frame no. 25 (I-frame) interpolated with different interpolation algorithms. The overlayed circles on the left show a piece of the collar of the tennis player's shirt where the median filter leaves the most blockiness. On the other hand, the median filter does not blur the natural edge of the collar, as the other algorithms do, indicated by the overlayed circles on the right.

MAD_{3f} is the *three-frame match error*, which we will choose as the minimum of the forward and the backward prediction error, like in [50]:

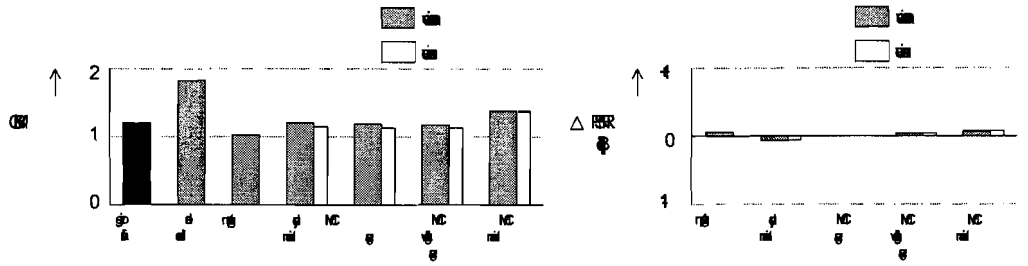
$$MAD_{3f} = \min\{MAD_{fwd}, MAD_{backwd}\}. \tag{4.19}$$

Similarly, we soft threshold the LSI metric (LSI_{max}). Table 4.5 lists the results for different MADs and LSIs for the recursive MC weighted average algorithm. Infinite values mean that the processing is always turned on, i.e. $k = 0$.

The optimal three-frame match error lies around 32 (see Figure 4.19). The GBIM is then close to the original value. Any threshold for the local spatial inconsistency does not contribute considerably to a better performance. We use soft thresholding on the MAD_{3f} instead to prevent the introduction of false edges.

Fall-back processing

The MAD value still can locally jump from low values to high values, as illustrated in Figure 4.13. The soft threshold introduced earlier causes some blocks to be processed heavily and others to be not processed at all. For the non-reliable estimates, a non-motion compensated fall-back function



(a) (b)

Figure 4.16: Objective test results for the step edge compensation algorithm on a three-frame interpolation in block impairment GBIM (a) and image fidelity (PSNR) (b).

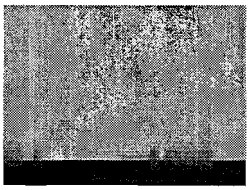


Figure 4.17: Detail of the *tennis* sequence, deblocked with the MPEG-4 post-processing algorithm, showing the effect of a hard threshold.

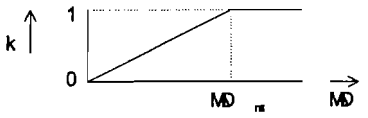
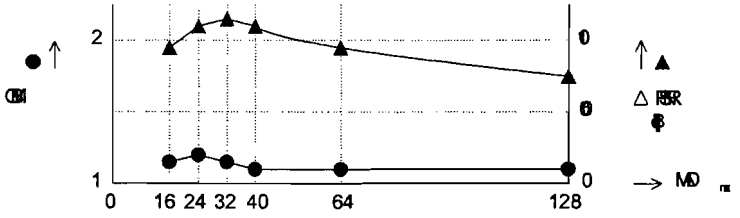
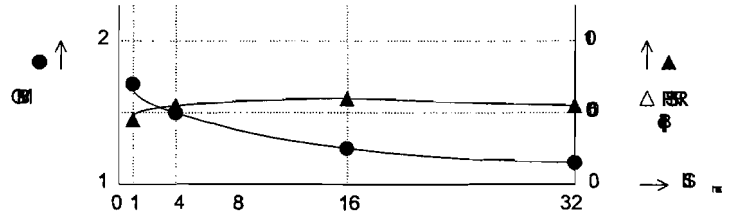


Figure 4.18: Mean absolute differences versus the weight k.



(a)



(b)

Figure 4.19: MAD (a) and LSI (b) thresholds versus the distortion.

algorithm	recursive	MSE	$PSNR$ $[dB]$	$\Delta PSNR$ $[dB]$	M_{hGBIM}	M_{vGBIM}	M_{GBIM}
original		0	∞	∞	1.12552	1.29699	1.21126
decoded		166.901	25.906	0	1.66517	1.99610	1.83064
dynamic median		169.716	25.834	-0.072	1.13445	1.29315	1.21380
dynamic median	✓	169.084	25.850	-0.056	1.06631	1.22802	1.14717
MC average		166.584	25.914	+0.008	1.10936	1.26448	1.18692
MC average	✓	166.031	25.929	+0.023	1.05710	1.20692	1.13201
MC weighted avg.		165.351	25.947	+0.041	1.11022	1.25857	1.18440
MC weighted avg.	✓	165.149	25.952	+0.046	1.06593	1.21113	1.13853
MC median		164.089	29.980	+0.074	1.28497	1.47962	1.38230
MC median	✓	164.016	25.982	+0.076	1.26992	1.46368	1.36680

Table 4.4: Performance results for the step edge compensation signals.

MAD_{max} /64	LSI_{max}	MSE	$PSNR$ $[dB]$	$\Delta PSNR$ $[dB]$	M_{hGBIM}	M_{vGBIM}	M_{GBIM}
∞	∞	165.149	25.952	+0.046	1.06593	1.21113	1.13853
128	∞	164.067	25.981	+0.075	1.08052	1.22777	1.15415
64	∞	163.290	26.001	+0.095	1.10242	1.25209	1.17726
40	∞	162.730	26.016	+0.110	1.13717	1.29081	1.21399
32	∞	162.562	26.021	+0.115	1.16331	1.32070	1.24201
24	∞	162.591	26.020	+0.114	1.21004	1.37578	1.29291
16	∞	163.309	26.001	+0.095	1.30048	1.48730	1.24389
∞	32.0	164.820	25.961	+0.055	1.10161	1.25083	1.17622
∞	16.0	164.713	25.964	+0.058	1.16525	1.32120	1.24323
∞	4.0	164.873	25.959	+0.053	1.43150	1.62244	1.52697
∞	1.0	165.021	25.955	+0.049	1.57125	1.80862	1.68994
∞	0.5	165.852	25.934	+0.027	1.60378	1.86035	1.73207

Table 4.5: Performance results of soft thresholding on the recursive compensation algorithm based on the MC weighted average.

can be used to reduce this artefact, with the risk of over-smoothing edges that coincide with block edges (see Figure 4.20). Equation 4.17 then changes to

$$F_{soft}(\vec{x},n) = kF_{non-mc}(\vec{x},n) + (1-k)F_{non-mc}(\vec{x},n), \tag{4.20}$$

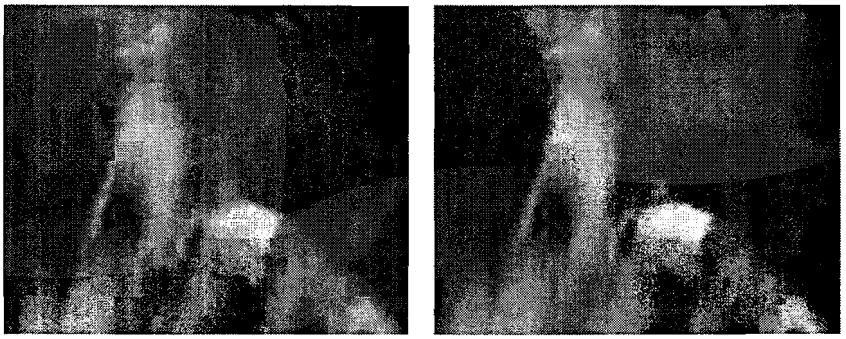
where F_{non-mc} is the output of a conventional non-motion compensated de-blocking algorithm. As an example, we will use a blind version of the MPEG-4 de-blocking algorithm, i.e. it assumes an infinitely high *qscale*. Table 4.6 contains the results for fall-back processing for $MAD_{max} = 32$.

Motion compensated median filtering

As we already remarked before in this section, the use of the median filter gives good results for noise like ringing and mosquito noise, but preserves the blocking artefact because it is likely to choose the original decoded sequence. A way to remove the blockiness as well is to use a non-motion compensated filtered frame as a prediction of the current frame instead of the original decoded sequence. The filtered output then becomes:

$$F_{median-fb} = \text{median}\left\{F(\vec{x}-\vec{D}(\vec{x},n),n-1), F_{blind}(\vec{x},n), F(\vec{x}+\vec{D}(\vec{x},n),n+1)\right\}, \tag{4.21}$$

where F_{blind} the non-motion compensated filtered image with the blind MPEG-4 de-blocking algorithm. There is no reason to use the shifted grid motion compensation scheme that we used in previous approaches, so we use the motion vectors from a non-shifted motion estimator. Like



(a) soft thresholding (b) fall-back processing

Figure 4.20: Figure (a) shows some remaining artefacts were the motion vectors are not reliable. Figure (b) is the same image with fall-back processing.

algorithm	MSE	$PSNR$ $[dB]$	$\Delta PSNR$ $[dB]$	M_{hGBIM}	M_{vGBIM}	M_{GBIM}
original	0	∞	∞	1.12552	1.29699	1.21126
decoded	166.901	25.906	0	1.66517	1.99610	1.83064
step edge comp.	165.149	25.952	+0.046	1.06593	1.21113	1.13853
soft thresholding	162.562	26.021	+0.115	1.16331	1.32070	1.24201
blind MPEG-4	165.584	25.941	+0.035	1.35693	0.95856	1.15775
fall-back	164.373	25.972	+0.066	1.00682	1.18062	1.09372

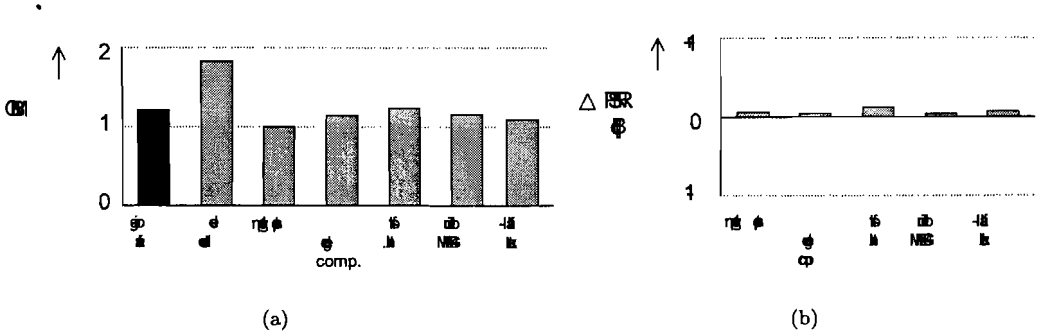
Table 4.6: Performance results for fall-back processing for $MAD_{max} = 32$.

for the previous algorithms, we can apply the MC median recursively and with soft thresholding with $MAD_{max} = 32$. Because we do not use shifted vectors, we can also assess the performance of the median approach with a commercial available 3-D RS block matcher.

algorithm	soft thr.	recur-sive	3-D RS	$PSNR$ $[dB]$	$\Delta PSNR$ $[dB]$	M_{hGBIM}	M_{vGBIM}	M_{GBIM}
original				∞	∞	1.12552	1.29699	1.21126
decoded				25.906	0	1.66517	1.99610	1.83064
median-fb		✓		25.941	+0.035	1.35693	0.95856	1.15775
median-fb	✓	✓		25.963	+0.057	0.91904	1.09324	1.00614
median-fb	✓			26.087	+0.181	1.07647	1.21274	1.14461
median	✓	✓	✓	25.941	+0.035	0.91307	1.09073	1.00190
median-fb	✓		✓	26.069	+0.163	1.07391	1.21873	1.14632

Table 4.7: Performance results for fall-back median filtering approach processing without shifted motion estimation and compensation.

Table 4.7 shows the results of the fall-back median filter with different modifiers. The results show that the recursive scheme performs better in block impairment (GBIM), but gives a slightly worse signal-to-noise ratio (PSNR). Since we are after blocking artefact reduction, we will include the latter filter in our assessment. We will refer to this algorithm, the recursive median filter with fallback processing, as the `median-fb` filter. It can also be seen that the fall-back median filter gives good results in combination with a commercial available 3-D RD block matcher, which was one of the goals to be met.



(a) (b)

Figure 4.21: Objective test results for the step edge compensation algorithm and fall-back processing in block impairment (GBIM) (a) and image fidelity (PSNR) (b).

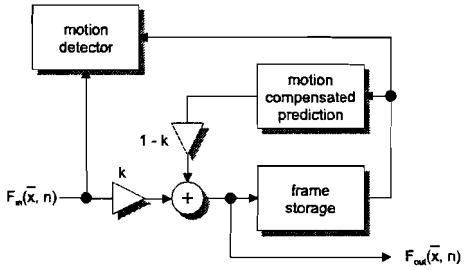


Figure 4.22: General loop filter design.

4.2.1 Evaluation

In the previous section, some improvements have been proposed for the initial algorithm. The best results were obtained when applying a sinusoidal compensation signal to the block edges of the original frame, based on the step size of the same edge of an interpolated frame. The interpolation was performed with a motion compensated weighted average based on the previous and the next frame, as defined in Equation 4.13. The filtered result is then used for the calculation of the next interpolation. It was not possible to do the motion estimation on the filtered result, so both recursive and non-recursive algorithms use the same motion vectors. Figure 4.21 shows the result of the algorithm (`step edge comp.`) and of the proposed further processing:

- Soft thresholding (`soft thr.`) prevents new edges from being created but also leaves more edges unaffected, increasing the blockiness.
- Fall-back processing (`fallback`) with a blind MPEG-4 (`blind MPEG-4`) processor improves the results again for the non-processed image parts. The latter introduces the risk of real sharp edges coinciding with block edges being blurred. However, we consider this a small risk since natural sharp edges are already blurred for low bit rates.

The blocking artefact is, for highly compressed image sequences, not limited to only the boundary discontinuity. In the next section, we propose a noise-filtering approach.

4.3 Noise-filtering

Another word for the blocking artefact is grid noise (see Chapter 3). The word noise suggests a noise-like approach for filtering. Temporal noise filters are often implemented as recursive filters (Infinite Impulse Response filters) to avoid expensive field or frame memories. Moreover, Infinite

Impulse Response (IIR) filters have stronger noise suppression than Finite Impulse Response filters [23, page 56]. This was also experienced in the previous sections. The general design for a noise reduction filter is given in Figure 4.22. The output $\hat{F}(\vec{x},n)$ of the filter is given as:

$$\hat{F}(\vec{x},n) = kF(\vec{x},n) + (1-k)\hat{F}(\vec{x} - \vec{D}(\vec{x},n), n-1). \tag{4.22}$$

Here, k is a control parameter that lies between 0 and 1 and weighs the amount of recursion. A motion detector calculates for each pixel the probability of motion. Generally, this motion detector calculates the difference between two frames, low-pass filters the result to reduce the noise in the difference image, then smoothes the result again to obtain a consistent motion detection. Where in the past accurate motion estimation was not feasible, high performance accurate motion estimators are now available in consumer electronics. We will use the motion estimator introduced before to filter the pixels along the motion trajectory. In this way, we can access temporal neighbouring pixels that have a strong correlation with the pixel under observation.

Error distribution

In the previous sections, we modelled the blocking artefact as a step discontinuity that can be smoothed with a smooth compensation function. However, the blocking artefact is not limited to this step continuity. To illustrate that, we will have a closer look at the error distribution within each coding block. For each 8×8 coding block in an image n, we calculate the SAD between the original picture and its corresponding MPEG-2 encoded and decoded picture for each relative pixel position in the coding block:

$$SAD(i,j) = \sum_{\vec{x}_b \in \mathcal{B}} \left| F_o\left(\begin{pmatrix} x_b + i \\ y_b + j \end{pmatrix}, n\right) - F\left(\begin{pmatrix} x_b + i \\ y_b + j \end{pmatrix}, n\right) \right|, \forall\ 0 \leq i,j < 8, \tag{4.23}$$

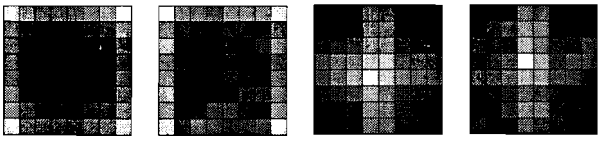
where F_o is the original source sample and F is the encoded and decoded sample. Figure 4.23(a) illustrates the distribution of the SAD for all blocks in an I-frame. The figure shows that pixels near the block boundary deviate most from the original sequence, and that pixels in the centre of the block are more reliable. With a numerical minimum of $6,079$ and maximum of $11,274$, the amount of error is about twice as much at the border as in the centre. As an illustration, Figure 4.23(b) shows the same distribution for an bi-directionally predicted image (B-picture) which contains high velocity motion (a moving tennis player against a panning background, cf. Figure 4.9(d)). The distribution is somewhat noisier and with a minimum of $9,379$ and a maximum of $11,819$, the sample values in the centre are less reliable as in I-pictures, but the distribution still shows the characteristic pattern. Now observe the distributions of the shifted blocks. From Figure 4.23(c) and 4.23(d) it can be found that near the centre, the predicted data is more reliable than near the block borders. We can exploit the characteristics of this distribution of interpolated data to reconstruct the original image. Because of the absence of correct data in the neighbourhood of block edges, the de-blocking algorithm again becomes an interpolation problem.

Using the observations above, we will propose a new filter scheme that uses the distribution of the errors with respect to the pixel position in the block. This position is characterized by the first order approximation of the pixel distance (the *Manhattan distance*) to the block border, defined for an 8×8 block as

$$d(\vec{x}) = d\left(\begin{pmatrix} x \\ y \end{pmatrix}\right) = d(x) + d(y), \text{ with } d(x) = \begin{cases} x \bmod 8 & , \text{if } x \bmod 8 < 4, \\ 7 - (x \bmod 8), & \text{if } x \bmod 8 \geq 4, \end{cases} \tag{4.24}$$

see also Figure 4.24.

Instead of a motion detector, we can use a motion estimator to control the value of k. Therefore, we adapt the control parameter k such that it becomes dependent on the distance in the shifted coding grid, just like we constrained the recursion with a soft threshold for the SAD. Such a function is proposed in Figure 4.25. Here, d_{max} is the maximum pixel distance to the block boundary, i.e. 6 for an 8×8 coding block. Note that for zero distance, the output is determined



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

Figure 4.23: Sum of absolute differences (SAD) between the original tennis sequence and its MPEG-2 encoded ($500kbps$) reconstruction, summed for each pixel position in an 8×8 block. Full white represents the greatest SAD (unreliable), black the smallest (reliable). (a) is the distribution of I-frame 481, (b) of B-frame 474. (c) and (d) are the respective distributions of the shifted grids.

Figure 4.24: Manhattan distance to the block border.

only by the interpolation. Because the interpolation is also based on a future frame, errors will slowly decay in time. To make errors decay faster, a more robust weighing scheme can be proposed like shown in Figure 4.26, giving up some reduction in block impairment. The outcome of both algorithms is found in Table 4.8.

For the interpolation, the MC weighted average proposed in Equation 4.13 is used. The motion compensation is performed on a shifted grid to profit from the continuity of the MC interpolation at the location of the coding block boundaries. As in the previous section, we can apply soft thresholding on the vector reliability and do fall-back processing with a non-motion compensated algorithm. For the noise-filtering approach, the results can be found in Table 4.8.

algorithm	MSE	$PSNR$ $[dB]$	$\Delta PSNR$ $[dB]$	M_{hGBIM}	M_{vGBIM}	M_{GBIM}
original	0	∞	∞	1.12552	1.29699	1.21126
decoded	166.901	25.906	0	1.66517	1.99610	1.83064
noise filtered	161.786	26.041	+0.135	1.20042	1.37814	1.28928
noise filtered, soft threshold	158.903	26.119	+0.213	1.28976	1.48710	1.38843
noise filtered, robust	160.040	26.089	+0.183	1.38491	1.62182	1.50337
noise filtered, fall-back	159.856	26.094	+0.188	1.12062	1.32772	1.22417

Table 4.8: Performance results for noise-filtering approach processing.

Algorithmic cost

Motion estimation on a shifted grid requires a mayor adaptation of the motion estimation to the shifted grid. Furthermore, an advanced motion estimator that was used for the experiments would require at least twice as much computational effort than an existing 3-D RS block matcher [50]. One of our goals was to implement a MC algorithm next to an existing architecture. Here we will evaluate a similar noise reduction algorithm as above, using a less complex motion estimator that is available in consumer electronics products as a non-shifted 3-D RS block matcher. The block diagram of the post-processor is depicted in Figure 4.27. The motion estimator uses the previously filtered frame for motion estimation, so saving an extra frame memory for the filtered result.

Figure 4.25: Pixel distance d to the block edge versus the weight of recursion k.

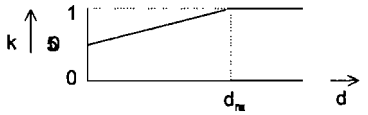


Figure 4.26: Pixel distance d to the block edge versus the weight of recursion k.

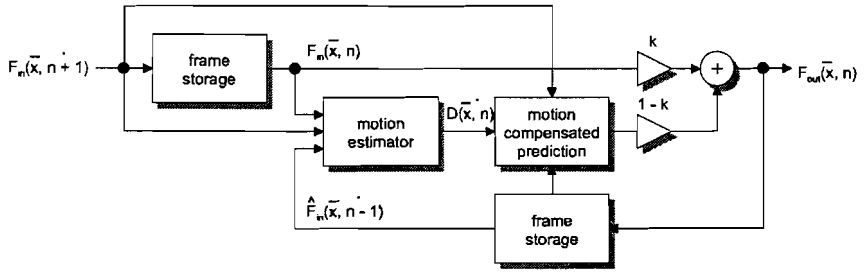


Figure 4.27: Motion compensated de-blocking.

From Table 4.9, it can be seen that the non-shifted interpolation is not able to profit from the continuity at the coding block grid and, therefore, is not able to reduce the blocking artefact as effective as the shifted version does. Moreover, the motion estimation is less accurate, introducing higher values for the MAD so that the fall-back algorithm is called more often, introducing more blurring. A screen shot of the shifted and the non-shifted noise-filtering algorithm with fall-back processing is shown in Figure 4.28.

4.3.1 Evaluation

In this chapter, we proposed several MC methods to deal with the blocking artefact. From an initial algorithm that uses a smooth compensation signal to remove the discontinuities at block boundaries, we arrived at a recursive noise-filtering method that both give good results in terms of PSNR and GBIM when compared to a common non-motion compensated method. For the step edge compensated algorithm (`comp`) and the noise-filtering algorithm (`noise`), the numerical results are depicted in Figure 4.29 for the unrestricted, the soft thresholded (`st`) and the fall-back processed (`fb`) *tennis* sequence. Furthermore, the `median-fb` filter dealt with in the previous subsection is evaluated.

MC methods can reduce the blocking artefact, but with an effort that is much higher than that of existing methods. When restricting the motion vector reliability, the blocking artefact reduction will be less, but the image fidelity rises. Fall-back processing is a good alternative to improve the image quality where the motion vectors are unreliable.

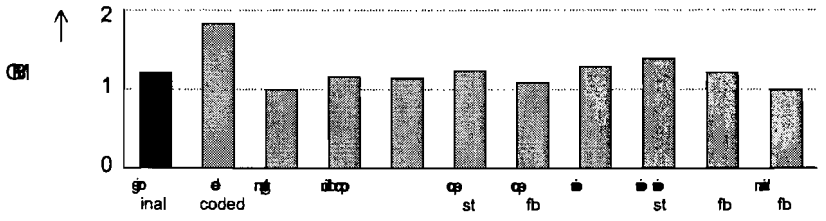
The non-shifted algorithm was found inferior to the shifted grid algorithm, except for the fall-back median filter. In the next chapter, we will assess the three shifted MC methods, the step edge compensation, the noise-filtering approach, with and without soft thresholding and fall-back processing, and the fall-back median filter, on several regular and demanding sequences.

algorithm	MSE	$PSNR$ $[dB]$	$\Delta PSNR$ $[dB]$	M_{hGBIM}	M_{vGBIM}	M_{GBIM}
original	0	∞	∞	1.12552	1.29699	1.21126
decoded	166.901	25.906	0	1.66517	1.99610	1.83064
non-shifted	170.152	25.822	-0.084	1.31943	1.49902	1.40923
non-shifted, soft threshold	160.682	26.071	+0.165	1.38689	1.60721	1.49705
non-shifted, fall-back	174.494	25.713	-0.193	0.90252	1.19453	1.04853

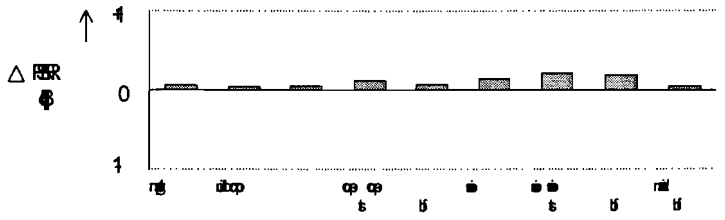
Table 4.9: Performance results for noise-filtering approach processing without shifted motion estimation and compensation. The difference is especially visible at the edge indicated by a white circle.

(a) (b)

Figure 4.28: Image post-processed with the noise-filtering algorithm with fall-back processing, with prediction on a shifted grid (a) and a non-shifted grid (b).



(a)



(b)

Figure 4.29: Objective test results for both step edge compensation and noise reduction algorithms in block impairment (GBIM) (a) and image fidelity (PSNR) (b).

Chapter 5

Evaluation

In the previous chapter, we presented several algorithms for the removal of the blocking artefact. For the evaluation of the algorithms, we will compare them to some existing non-motion compensated algorithms in the literature and industry since a reference implementation of a motion compensated de-blocking algorithm was absent. A de-blocking algorithm from the Moscow State University, available as software plug-in and for which the implementation is not documented, is also assessed. Both methods that use bit stream parameters as well as "pure" post-processing algorithms that lack bit stream information are evaluated.

5.1 Sequences

To assess the algorithms, some more and less critical sequences are chosen with respect to motion estimation. Though the objective of the test is not to test the quality of the motion estimator, the filter must be robust to erroneous estimates. The following sequences were evaluated, with the corresponding criteria:

- `birds`. Panning sky with flying birds. Their wings are smaller than the block size, making it hard to find the motion vectors. The air has a smooth gradient that makes it very susceptible to quantization. The sequence consists of 50 frames.
- `golf`. Panning background with high frequency details (grass, tree). Occlusion of the background by the golf player. The scene has 48 frames.
- `k3`. Video clip, 60 frames, containing high velocity vertical motion.
- `ngc`. Sequence of 77 frames captured from National Geographic Channel. Contains motion in all directions with high velocities.
- `ngc2`. Another 60 frames from National Geographic Channel, with transparency, high velocities, and combined video formats. Very hard to estimate motion.
- `susie`. Susie on the phone, 50 frames. Almost no motion, so motion estimation should give near-zero motion vectors here. An artificial alignment stripe pattern is present at the top and at the bottom of the image.

The sources are uncompressed, or at least compressed at such a bit rate that there is hardly noticeable degradation of the image quality. The resolution of the sequences is 720×576 pixels at 25 frames per second (PAL), except for the `birds` sequence, which has a resolution of 720×480 at 30 frames per second (NTSC). Each sequence has a duration close to two seconds. A screen shot for each sequence is pictured in Figure 5.1.

(a) birds (b) golf (c) k3

(d) ngc (e) ngc2 (f) susie

Figure 5.1: Test sequences used for evaluation.

5.2 Algorithms

The MPEG-2 codec is used for the experiments. The reference encoder and decoder are provided by the MPEG Software Simulation Group (*MSSG*)[60]. Because we focus on low bit rate Internet Video, we also want to measure the performance of de-blocking algorithms for different low bit rates. Therefore, each sequence is evaluated for an encoding bit rate of $500kbps$ and $1,000kbps$. The following algorithms were selected for evaluation:

- `mpeg-4`. MPEG-4 de-blocking algorithm proposed in [36] and explained in Subsection 3.7.1. The post-processing was done inside the reference MPEG-2 decoder, since the algorithm uses the quantization scale parameters that are encoded in the bit stream. The technique requires a modification of the decoder and is as such not "pure".
- `msu`. Version 2.0 of the de-blocking algorithm developed at the Graphics and Media Laboratory of the Moscow State University (MSU) [89]. The filter is meant as a VideoCD (MPEG-1) restoration filter and is used as a software plug-in for video processing software (`VirtualDub`).
- `shift`. Shifted grid filter proposed by Nosratinia *et al.* [64] in Subsection 3.2.2. The method uses an estimated quantization scale parameter, obtained by the DATES block grid estimator developed at Philips Research Laboratories.
- `3-mode`. Three-mode filter, implemented by Philips Research Laboratories, that uses three different compensation signals with three different strengths (cf. Figure 4.6). The algorithm does not use stream information, but measures the local activity like the MPEG-4 de-blocking algorithm.
- `blind`. The same MPEG-4 de-blocking algorithm observed before, but with no knowledge of the quantization parameter. The quantization parameter is assumed infinite.

- `comp`. The MC recursive step edge compensation algorithm proposed in Chapter 4, with the sinusoidal compensation signal. The implementation using soft thresholding with $MAD_{max} = 32$ is referred to as the `comp-st` algorithm, whereas the algorithm that incorporates blind MPEG-4 fall-back processing is called `comp-fb`.

- `noise`. The MC recursive noise-filtering method proposed in Chapter 4. Again, `noise-st` and `noise-fb` refer to the method using soft thresholding and fall-back processing respectively.

- `median-fb`. The recursive MC median filter with fall-back processing, proposed in Chapter 4.

5.3 Objective results

The motion estimator requires the previous image and the next image in the sequence and as a result, the first and the last frames are not post-processed. Therefore, the first and the last frame of each sequence were not included in the calculation of the metrics.

The results in terms of image fidelity (MSE, PSNR) and blocking artefact (GBIM) are listed in Appendix B. Figures 5.4 and 5.5 give the PSNR improvement (positive axis) with respect to the decoded sequence. Figures 5.2 and 5.3 show the GBIM metric for each algorithm, with the original sequence as a reference. A GBIM close to the original GBIM indicates a good estimate, a GBIM above the original GBIM means that the algorithm *on average* preserves some blocking artefact, while a GBIM lower than the original indicates over-smoothing. PSNR is a measure for image fidelity. A positive value represents an improvement with respect to the initial decoded sequence. The filtered result is then closer to the original sequence. Note that GBIM and PSNR are global metrics: severe local errors are levelled out over the entire image and over the entire sequence, and are not represented by the metrics.

The figures reveal that for different bit rates, all algorithms behave quite similarly. Both $500kbps$ and $1,000kbps$ encoded sequences show almost the same amount of blockiness for moderate and high velocity motion. Differences in blockiness for the two bit rates become visible for sequences with low velocity motion (e.g. the `golf` and `susie`). Still the post-processed results arrive at around the same quality in terms of blockiness and PSNR improvement, except for the methods using soft thresholding. In that case, the blockiness will depend more on the blockiness in the original sequence.

Objective observations show that MC methods are able to reduce the blocking effect. However, the differences with existing methods, whether or not using the quantization parameter from the bit stream, are hardly noticeable in areas where the motion estimation is accurate (see Figures 5.6, 5.7, and 5.8). As an example, observe the motion vector reliability in Figure 5.9 and compare an existing non-motion compensated method such as MPEG-4 de-blocking (Figure 5.6(c)) with a MC method (Figure 5.7(a) or 5.7(b)). Where the motion estimation is not reliable, unconstrained processing leads to the introduction of new edges in case of the step edge compensation signal (Figure 5.7(a)). Constraining the motion vector reliability prevents this new artefact, but instead leaves the blocky original unaffected (Figure 5.7(c)). A non-motion compensated fall-back algorithm can remove those unfiltered areas (Figure 5.7(e)). MC de-blocking with fall-back processing does not give a substantial improvement over fall-back processing alone. In fact, it was observed that the MC filter leaves some block edges untouched where both the interpolation is reliable and the motion vectors are zero, or more in general, where the motion vector components are zero modulo the block size (0 mod 8).

For the noise-filtering approach, unconstrained filtering yields the best possible results in terms of blocking impairment (GBIM), but can cause severe artefacts in fast changing sequences (see screen shot in Figure 5.7(b)), which explains the strong deviation in the PSNR improvement figure for the `ngc2` sequence. Here, the motion cannot be tracked and the scenes change to quickly for the

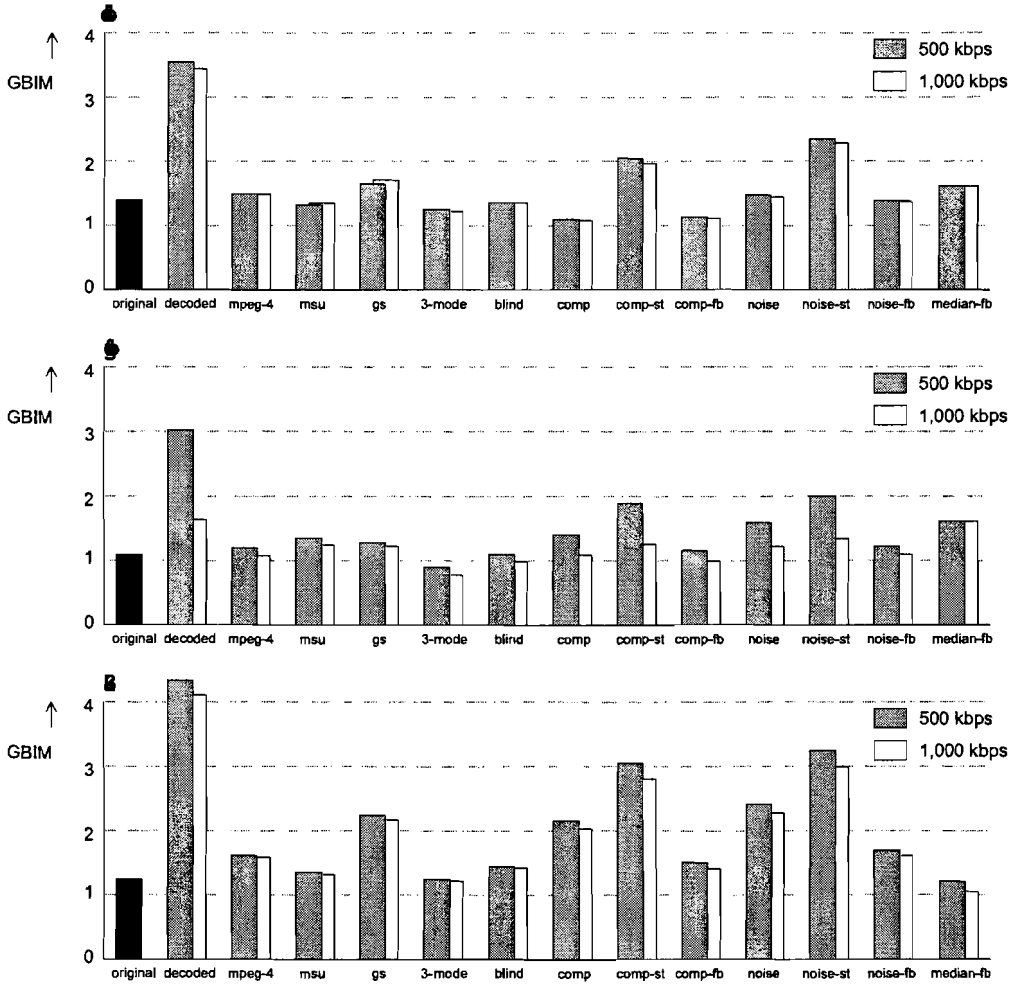


Figure 5.2: Objective test results for all test sequences in block impairment (GBIM).

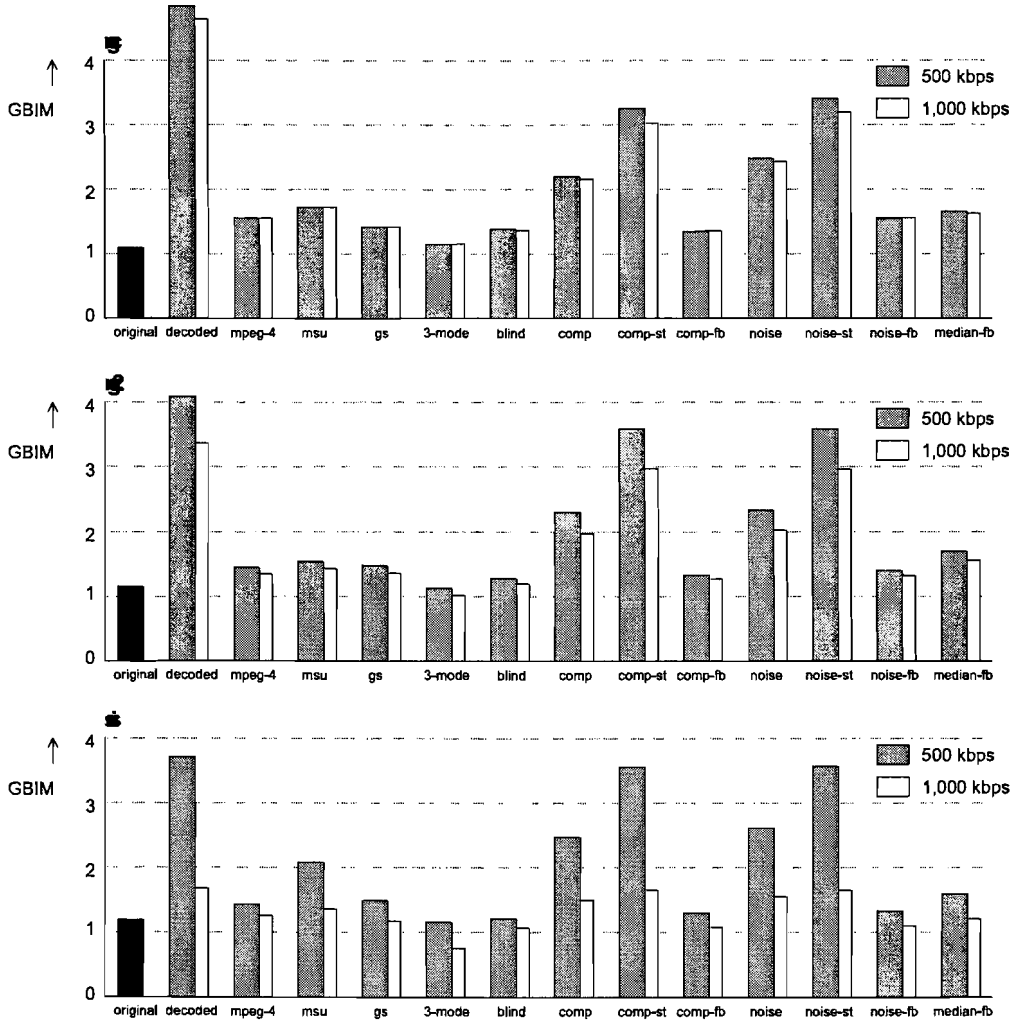


Figure 5.3: Objective test results for all test sequences in block impairment (GBIM).

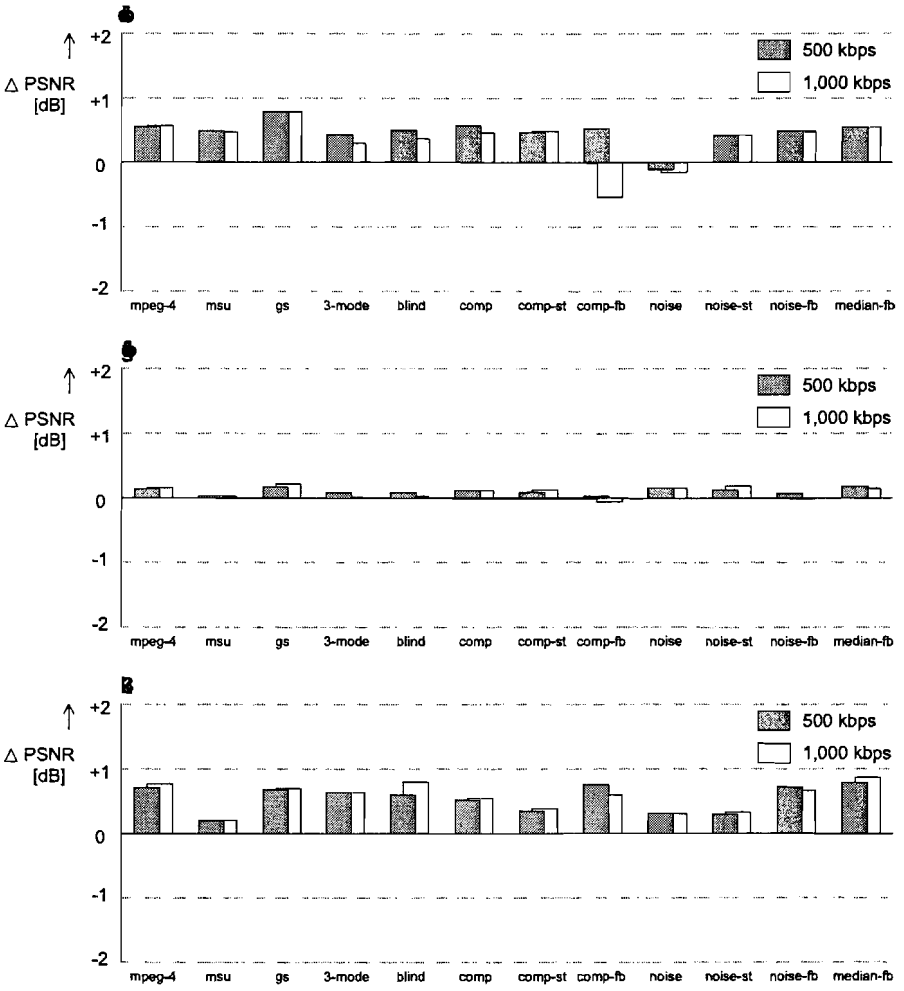


Figure 5.4: Objective test results for all test sequences in image fidelity (PSNR).

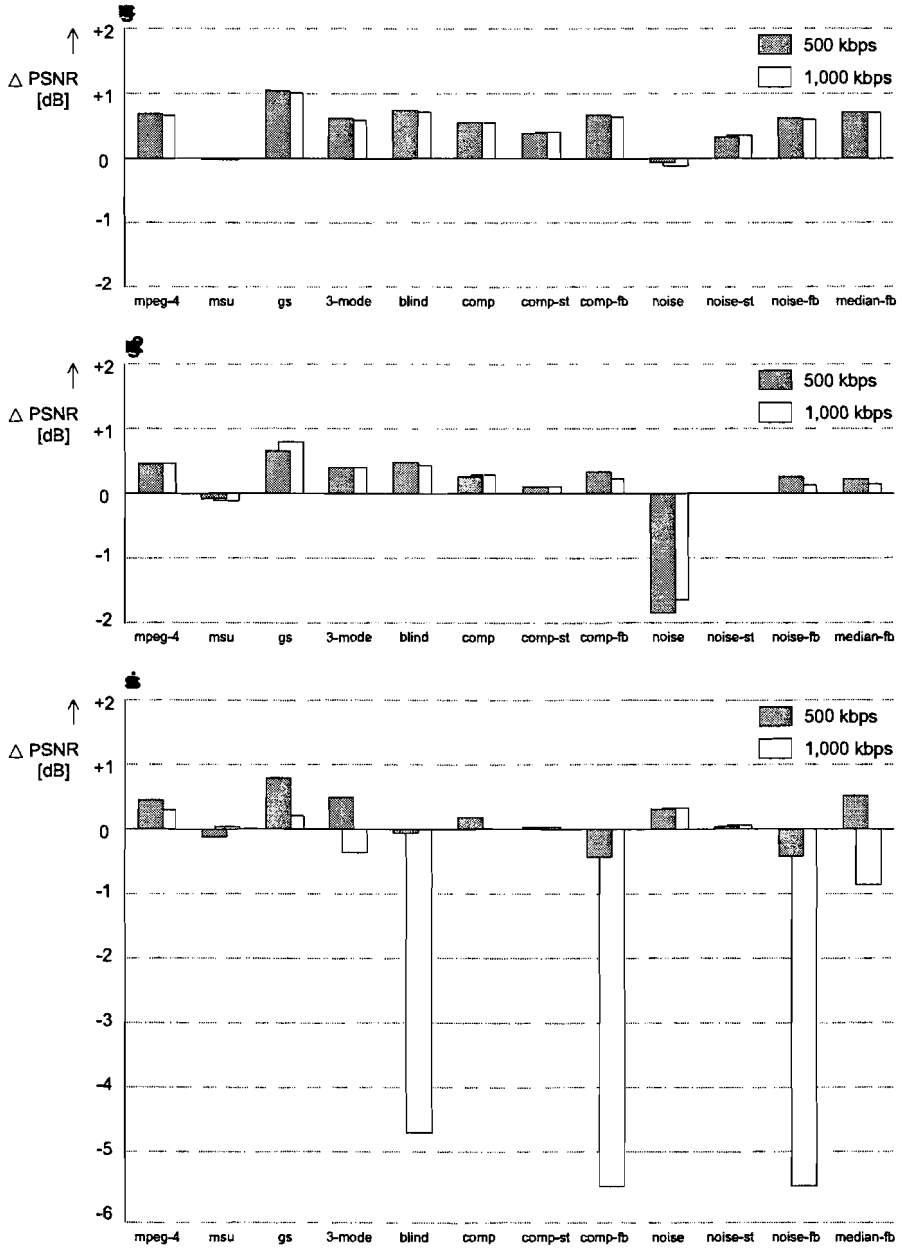


Figure 5.5: Objective test results for all test sequences in image fidelity (PSNR).

(a) original

(b) decoded

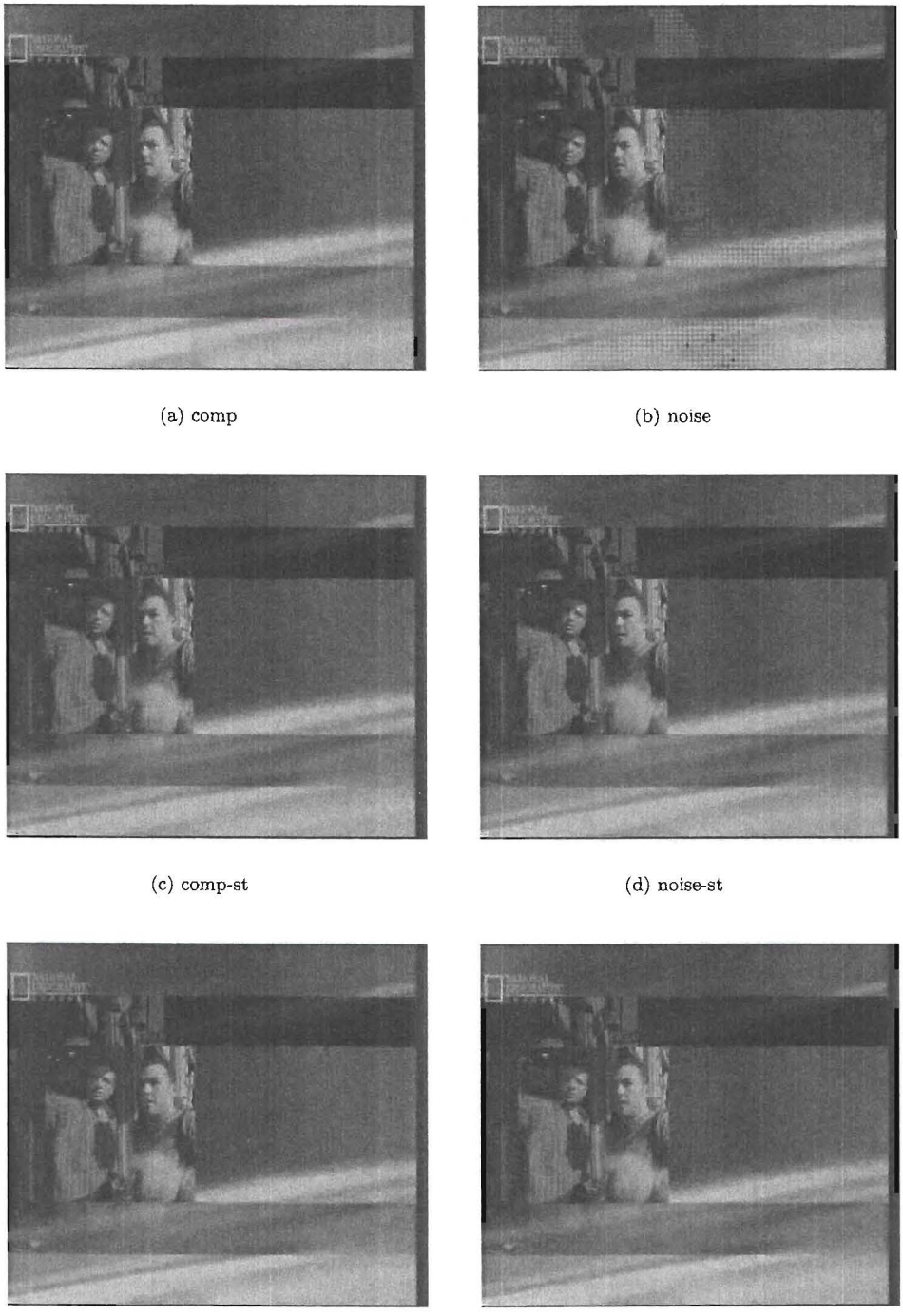
(c) mpeg-4

(d) msu

(e) gs

(f) blind

Figure 5.6: Frame 39 from the *NGC2* sequence at $1,000kbps$ (non-motion compensated methods).



(a) comp

(b) noise

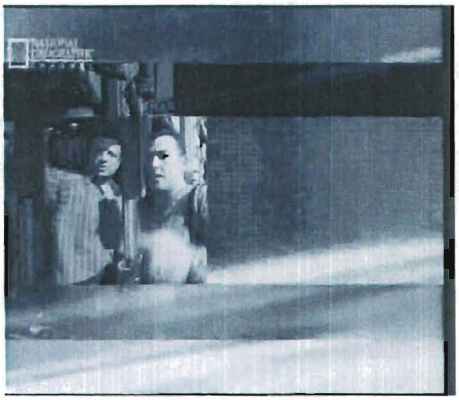
(c) comp-st

(d) noise-st

(e) comp-fb

(f) noise-fb

Figure 5.7: Frame 39 from the *NGC2* sequence at $1,000kbps$ (motion compensated methods).



(a) median-fb

Figure 5.8: Frame 39 from the *NGC2* sequence at $1,000kbps$.

Figure 5.9: MAD of frame 39 from the *NGC2* sequence.

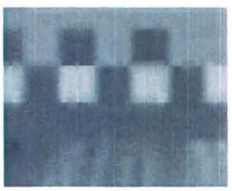


Figure 5.10: Smearing of artificial edges due to blind post-processing in the *susie* sequence.

errors to fade. The artefact remains visible even with soft thresholding and fall-back processing (cf. Figures 5.7(d) and 5.7(f)).

In general, step edge compensation yields slightly better results for de-blocking than the noise-filtering approach.

The large deviation in PSNR improvement for the `susie` sequence can be explained by a failing fall-back algorithm. The artificial alignment stripe pattern contains ambiguous areas where the algorithm flips between strong and weak filtering, showing smeared edges near the original sharp edges (Figure 5.10). This suggests the use of a more reliable fall-back algorithm.

The difference in PSNR improvement for all non-motion compensated and MC methods in image fidelity is lies roughly in the order of tenths of dBs.

5.4 Subjective evaluation

GBIM gives an indication of the block impairment for still images, but does not give a measure for the blocking artefact that are not aligned with the coding grid. Furthermore, the GBIM metric does not give the warranty that a picture is not over-smoothed. The PSNR must be observed next to the GBIM to guard the image fidelity. Both GBIM and PSNR are global metrics, averaged over the entire image and over the entire sequence, not taking into account local errors. For example, the `ngc2` sequence shows a post-processing artefact (cf. Figure 5.7(f)) which is not reflected by the PSNR improvement ($+0.118dB$). A more reliable comparison follows from a subjective assessment.

A simple method to compare different algorithms to each other is by *paired comparison.* In a paired comparison, a representative number of observers is presented several pairs of sequences in split screen (left and right) on a single monitor, each for which they have to decide which one of the two they prefer. For the analysis afterwards, it is obligatory that the viewer makes a decision. Since the difference in block impairment and image fidelity is low for the best-performing algorithms assessed before, we expect the observers to need more time (one minute maximum) to make a decision. Therefore, we aim at a reduced number of stimuli (pairs) per observer, limiting the number of algorithms to be evaluated and the number of sequences. An average duration of 15 minutes was acceptable for most people.

We decided to compare a low-cost, well-performing non-motion compensated algorithm and three motion compensated methods that gave the best objective results in terms of block impairment (GBIM), four algorithms in total:

- `mpeg-4`, a common, low-cost de-blocking technique that is popular in the industry and often used as a reference in the literature [36].
- `comp-fb`, the MC step edge compensation algorithm with blind MPEG-4 fall-back processing proposed in Chapter 4.
- `noise-fb`, the MC noise-filtering approach proposed in Chapter 4, including fall-back processing.
- `median-fb`, the fall-back median proposed in Chapter 4.

Without the decoded sequence, there are $\binom{4}{2} = 6$ possible combinations of algorithms. The decoded sequence is left out on purpose. The reason for this is that the difference between the decoded sequence and any post-processed sequence is that obvious, that a comparison would yield a statistically significant difference between decoded sequence and the post-processed sequences, leaving the post-processed sequences clustered with no statistical significant mutual difference. The six combinations appear in random order, and at random screen positions (left or right), to prevent biasing of the test results by the display order or the screen position.

The chosen sequences include both moderate and high velocity motion in different directions, different scale of detail, and other difficulties like gradients and logos. Three sequences meeting those requirements are selected, `birds`, `golf`, and `ngc`, making a total of $3 \times 6 = 18$ stimuli per

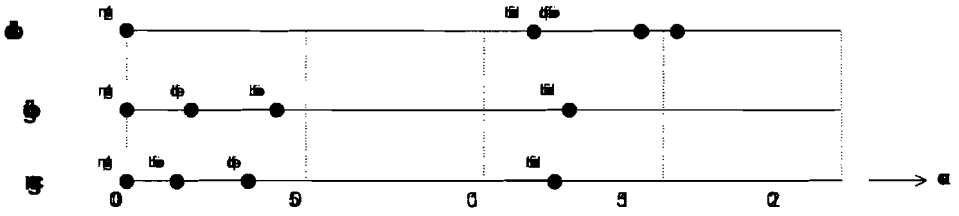


Figure 5.11: Perception test results.

observer (about 15 minutes). A repetition measurement is not performed for the same reason mentioned before. The two-second sequences are concatenated in palindromical order to make a longer, continuous sequence. As for the objective measurements, the first and last frames of each sequence are left out. These are typical I-frames, showing heavy blockiness when displayed unprocessed. To avoid the influence of artefacts in the chrominance pictures, the observers are only presented with luminance information.

Prior to the actual test, the observers were given a set of two training pairs. The first pair showed the decoded sequence and a randomly chosen post-processed sequence (again at random screen position) to demonstrate the maximum possible difference. The second pair showed two different randomly chosen post-processed sequences to demonstrate the minimum difference they might expect. The blockiness in all sequences was almost identical, so the observers were asked to rate the overall picture quality.

The test setup was made in a visual perception room on a 37" flat panel LCD monitor against a grey wall. The room is visually isolated from its environment, providing a moderate amount of background illumination to prevent eyestrain. The original SD resolution was kept in order to prevent the introduction of scaling artefacts. Since the performance difference of the algorithms was almost imperceivable at the standardized viewing distance of 6 times the active screen height, the viewing distance was reduced to 3 times the active screen height, a normal viewing distance for a customer in a shop.

5.5 Subjective results

Among the viewers, 2 women and 13 men, were several people with experience with picture quality (about 10), along with some less experienced viewers. Less experienced viewers on average took more time to make a judgement (around 60 seconds) than experienced viewers (around 30 seconds). Most of the observers reported that it was difficult to differentiate between the algorithms.

The results of the test can be found in Table C.1 in Appendix C. At first glance, each particular pair of algorithms shows a different score for different sequences. A numerical analysis indeed supports the observation that the results depend on the sequence, assuming that a maximum of 5% of the decisions in the experiment were false [73]. For more details, see Appendix C. This means that the results have to be evaluated per sequence, and not as a total.

Table C.1 gives a multi-dimensional comparison of the algorithms. The easiest way to compare different algorithms is to evaluate them one-dimensionally. The *Thurstone model* is the most common statistical model for paired comparisons. This model yields so-called *z-scores* for each algorithm. The z-score is a measure that indicates how much the ratio between "better" and "worse" deviates from the fifty-fifty situation, normalized by the worst performing algorithm (in our case the `mpeg-4` algorithm). The calculated values that result from the Thurstone model, described in [73], can be found in table C.2 in Appendix C. The one-dimensional visual result is shown in Figure 5.11

The relations have to be thought of in terms of *statistical significant difference*. For example, for the `birds` sequence, the distance along the z-axis between `median-fb` and `comp-fb` is relatively small, indicating *no* significant difference, as well as the distance between `comp-fb` and `noise-fb`,

and between median-fb and noise-fb. We group these mutual close algorithms by underlining them like

$$\texttt{mpeg} - 4 < \underline{\texttt{median} - \texttt{fb} < \texttt{comp} - \texttt{fb} < \texttt{noise} - \texttt{fb}}, \tag{5.1}$$

where the "less than" operator in $a < b$ indicates that algorithm b performs better than algorithm a. The meaning of the underline is that the grouped algorithms lie in each other's *confidence intervals*, which means that their mutual order could change in another experiment, assuming that the observers made a wrong decision in a maximum of 5% of the cases. Similarly, we can write such a relationship for the golf and the ngc sequence:

$$\underline{\texttt{mpeg} - 4 < \texttt{comp} - \texttt{fb} < \texttt{noise} - \texttt{fb}} < \texttt{median} - \texttt{fb}, \text{ for the golf sequence,} \tag{5.2}$$

$$\underline{\texttt{mpeg} - 4 < \texttt{noise} - \texttt{fb} < \texttt{comp} - \texttt{fb}} < \texttt{median} - \texttt{fb}, \text{ for the ngc sequence.} \tag{5.3}$$

When investigating equations 5.1, 5.2, and 5.3, the only conclusion that can be drawn is that the both comp-fb and noise-fb perform equally. As also remarked during the perception test, the birds and the ngc sequences will not occur often in an average television broadcast, and if they occur, their appearance will be short. To this respect, the golf sequence is a more natural sequence that is more likely to occur in normal television broadcast. If we assume this sequence to be more representative for broadcast content, the beneficial properties of the fall-back median filter with respect to reduction of noise like ringing and mosquito noise becomes clear from the outlying z-value, as previously remarked in Section 4.2. The noise-filtering characteristics of the fall-back median filter was also noticed by some observers, who reported the mosquito noise in the detailed parts (e.g. grass) of the golf sequence as most annoying.

Chapter 6

Conclusions and Recommendations

6.1 Conclusions

In the last decades, numerous approaches have been developed to repair the most objectionable degradation of block transformed images and video sequences, the blocking artefact. Next to adaptive filtering methods that target on *image enhancement* and are very popular in the industry, *image restoration* techniques can be found in the literature. These approaches can be subdivided into set theoretic methods such as Projection Onto Convex Sets (POCS) and Constrained Least Squares (CLS), and statistical estimation methods like Maximum A Posteriori (MAP) probability based restoration and anisotropic diffusion.

We concluded in Chapter 3 that non-adaptive filtering offers no improvement and that edge-based filtering is not appropriate since the edges of highly compressed images are heavily distorted.

Set theoretic methods rely on the definition of the constraint sets. Often, these constraints leave too many images in the solution set, including many distorted images, making POCS-based methods not perform very well, especially considering their high computational load. Generally, iterative restoration techniques are too expensive for real-time applications and will be only found in off-line processing applications.

There are algorithms that use bit stream information, often the quantization parameter, to repair the distorted sequence. We aimed at a method that does not use this information, because a decoder is often implemented as a standard component, which has an input bit stream and presents the decoded pictures at the output. Parameter extraction would require a modification of the decoder.

Many existing algorithms model the process of block DCT coding and quantization for still images, but do not take into account the process of motion compensation (MC) when extending traditional algorithms to digital video. In this report, we presented a motion compensated approach for the removal of the blocking artefact. In Chapter 3, we set up the following requirements for a post-processor. We repeat these requirements here:

- The filtering scheme has to be adaptive, preferably not based on an edge map.
- The algorithm must have a limited computational complexity to make it suitable for real-time implementations.
- The technique must model the process of motion compensation that takes place in block-DCT-based video coding schemes.
- To avoid modification of existing decoders, the post-processor does not use coding parameters, except for the size and location of the coding grid.

All but one requirement from above list has been met. The real-time implementation introduced in the Chapter 4 was not able to give satisfying results, except for the MC median with fall-back processing. The use of a state-of-the-art motion estimator on a shifted coding grid makes the observations in this report more of academical value instead of practical. We will list further conclusions below.

- Motion compensated methods are able to reduce the blocking artefact. Though compared to existing methods, whether or not using the quantization scale parameter, the methods that use unrestricted step edge compensation or noise-filtering developed in this study on average yield worse results than existing methods and are far more complex.
- The result of a motion compensated filter strongly depends on the reliability of the motion vectors. With reliable vectors, an improvement in image fidelity as well as subjective results can be achieved comparable to non-motion compensated methods. Without reliable vectors, some artefacts remain. In those cases, *reliable* fall-back processing improves the result. Non-motion compensated fall-back processing introduces the risk of real sharp edges coinciding with block edges being blurred too. Because natural sharp edges are already blurred for low bit rates we consider this a small risk. Even if the vectors are reliable, the MC interpolation does not always give useful information. If both the interpolation is reliable and the motion vectors are zero (or more in general, when the motion vector components are zero modulo the block size (0 mod 8)), the blocking artefact is still present in the interpolation and the artefact remains after filtering.
- Comparing the MC techniques presented in this report with the fall-back processing alone, there is no considerable gain in blocking artefact reduction, nor in image fidelity. The calculation of a MC interpolation requires accurate and dense motion vectors. The state-of-the-art motion estimator that was used for the experiments would require almost all the resources of a current commercial multimedia processor. Considering the complexity, less complex non-motion compensated algorithms still provide the best alternative for de-blocking. However, the MC median with fall-back processing does give better subjective results in terms of ringing artefact and mosquito noise.
- Lastly, we found that the GBIM metric does not give a measure for those blocking artefact that are not aligned with the coding grid and does not give any warranty that a picture is not over-smoothed. The Peak Signal-to-Noise Ratio (PSNR) metric guards the post-processed image fidelity and has to be observed next to the GBIM metric. Subjective evaluation remains necessary for the assessment of algorithms on picture quality improvement.

6.2 Recommendations

In this project, a few motion compensated algorithms were proposed and evaluated for the removal of the blocking artefact. The process of motion compensation was added to the model of block-based transformation and quantization. Since a well-documented existing MC method could not be found, we compared the proposed algorithms to existing adaptive methods, optionally using parameters from the encoder.

- As stated in Chapter 4, an optimal de-blocking algorithm has access to the decoder, were the picture types, motion vectors, reference images, and quantization scale parameters are available. Future research might include the extraction of these parameters, and how they can optimally be used to improve picture quality.
- It was assumed that the pictures were directly available from the decoder, such that the block sizes are 8×8 samples and the blocks are aligned to the image origin. Block edges were not blurred by spatial down and up-sampling elsewhere in the video chain. Post-processors should also consider non-aligned coding grids. For the detection of the coding grid and

its offset from a re-sampled sequence, accurate algorithms are already available at Philips Electronics.

- The motion estimator that was used is a advanced propriety 8×8 block matcher from Philips Research Laboratories. Less complex motion estimators are available for consumer electronics, but the performance of such an algorithm was not evaluated.
- The motion vector reliability threshold used in this project, expressed in Mean Absolute Differences (MAD), was experimentally determined. Song *et al.* [83] proposes that this threshold depends on a noise figure. This can be implemented with a noise estimator.
- A simple model for occlusion problems was proposed. Possibly, better results can be gained with a more advanced occlusion model. For example, instead of block-based interpolation, also pixel-based interpolation can be evaluated. The calculation of motion vectors per pixel is often implemented with median filters as a process called *block erosion*, which is computationally expensive. The block-based approach was also preferred over the pixel-based approach by Tang *et al.* [85].
- The influence of post-processing filters like sharpness and contrast improvement was not evaluated. These image enhancement methods are often part of the video chain, and can have an effect opposite to that of artefact reduction techniques.

6.3 Final remarks

Although post-processing is a means to reduce artefacts caused by lost information, none of the techniques is fully able to restore the lost data. In the development of next-generation video codecs, more effort is dedicated to the reduction of coding artefacts at the encoder side. The *lapped orthogonal transform*, mentioned in the Chapter 1, is such a technique. To conclude this report, we will give a short overview of the recent developments, with the Advanced Video Coding (AVC) scheme as an example.

The ITU Video coding Experts Group *VCEG* and the ISO/IEC MPEG, joint in the Joint Video Team (JVT), established an Advanced Video Coding standard called H.264, H.26L, ISO/IEC MPEG-4 Advanced Video Coding, or MPEG-4 Part 10 [24]. The codec claims to achieve an average gain of $5.8dB$ over MPEG-2 at a transmission rate of $1024kbit/s$ and a frame rate of $30frames/s$. We will list some improvements with respect to MPEG-2.

- In AVC, an alternative 4×4 DCT using integer arithmetic is applied, achieving a transformation to the frequency domain with less computational effort, but with the advantages of the frequency domain transform [79]. The smaller block size reduces the ringing artefact, and the integer transform reduces the mismatch between encoder and decoder that was previously caused by different implementations of the DCT and IDCT.
- AVC uses a block-based coding scheme similar to that of MPEG-2 and MPEG-4. It reduces the blocking artefact by an adaptive in-loop *de-blocking filter* that is present in both the encoder and decoder system [78]. The filter smoothes block edges based on prediction type (intra or inter macroblock coding), motion vector values, prediction error energy (the number of coded transform coefficients), difference in reference pictures, and the gradient along the block edge.
- Further compression is achieved by spatial prediction, the use of context adaptive entropy coding, and the use of multiple reference pictures with motion compensation at different block sizes (see Chapter 2).

As techniques like AVC gradually come into use, future systems will make post-processing for e.g. ringing and blocking artefacts more and more obsolete.

Abbreviations and Acronyms

3-D RS	Three-dimensional Recursive Search
AC	Alternating Current
ADRC	Adaptive Dynamic Range Coding
ASP	Advanced Simple Profile
AVC	Advanced Video Coding
BAM	Blocking Artefact Meter
BD	Blu-ray Disk
BDCT	Block Discrete Cosine Transform
BGD	Block Grid Detector
CABAC	Context-based Adaptive Binary Arithmetic Coding
CD	Compact Disk
CIF	Common Intermediate Format
CLS	Constrained Least Squares
DADAR	Digital Artifact Detection And Repair
DATES	Digital Artifact Estimator
DC	Direct Current
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DVB	Digital Video Broadcast
DVD	Digital Versatile Disk
DWT	Discrete Wavelet Transform
FIR	Finite Impulse Response
GB	Gigabyte, 1,024 kilobytes
GBIM	General Block Impairment Metric
GOP	Group of Pictures
HD	High Definition
HVS	Human Visual System
IDCT	Inverse Discrete Cosine Transform
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IIR	Infinite Impulse Response
IP	Internet Protocol
ISO	International Organization for Standardization
ITU	International Telecommunication Union
JND	Just Noticeable Difference
JPEG	Joint Photographic Experts Group
JVT	Joint Video Team

kbit	kilobit, one thousand bits (1,000, *not* 1,024)
kbps	kbit per second
LCD	Liquid Crystal Display
LMMSE	Linear Minimum Mean Squared Error
LOT	Lapped Orthogonal Transform
LSI	Local Spatial Inconsistency
MAD	Mean Absolute Difference
MAP	Maximum A Posteriori
Mbit	Megabit, one thousand kilobits (1,000, *not* 1,024)
MC	Motion Compensated
MLP	Multi-Layer Perceptron
MPEG	Moving Picture Expert Group
MP@ML	Main Profile at Mail Level
MSDS	Mean Squared Difference of Slope
MSE	Mean Squared Error
MSSG	MPEG Software Simulation Group
MSU	Moscow State University
NTSC	National Television Systems Committee
PAL	Phase Alternation Line
PCA	Principal Component Analysis
POCS	Projection Onto Convex Sets
PSNR	Peak Signal-to-Noise Ratio
RGB	Red, Green, and Blue colour space
SAD	Sum of Absolute Differences
SAF	Spatial Adaptive Filtering
SD	Standard Definition
SP	Simple Profile
SECAM	Système Électronique Couleur Avec Mémoire
SVCD	Super Video CD
TMAP	Temporal Maximum A Posteriori
TV	Television
VCEG	Video Coding Experts Group
VHS	Video Home System
VLC	Variable Length Code
VOP	Video Object Plane
YCbCr	Y (luminance), Cb and Cr (chrominance signal for blue and red colour differences respectively) colour space

References

[1] S.A. Basith and S.R. Done, "Digital video, MPEG and associated artifacts", in *Surveys and Presentations in Information Systems Engineering (SURPRISE) '96 Journal*, Imperial College London, Vol. 4, June 1996. Available from Internet: <http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/sab/report.html>.

[2] L. Chiariglione, *MPEG: achievements and current work*, International Organization for Standardization, Nov. 2001. Technical Report No. ISO/IEC JCT 1/SC 29/WG 11 N.

[3] B. Cahill and C. Heneghan, "Locally adaptive deblocking filter for low bit rate video", in *2000 International Conference on Image Processing*, vol. 2, pp. 664–667, Sept. 2000.

[4] T. Chen, H.R. Wu, and B. Qiu, "Adaptive postfiltering of transform coefficients for the reduction of blocking artifacts," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 5, pp. 594–602, May 2001.

[5] N.I. Cho, "A convolutional model and a cepstral filtering algorithm for the reduction of blocking artifacts", in *Proceedings of the 43rd IEEE Midwest Symposium on Circuits and Systems*, Vol. 2, pp. 608–611, Aug. 2000.

[6] H. Choi and T. Kim, "Blocking-artifact reduction in block-coded images using wavelet-based subband decomposition", in *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 10, No. 5, pp. 801–805, Aug. 2000.

[7] K.S. Choi and S.J. Ko, "POCS-based enhancement of de-interlaced video in the DCT domain", in *Proceedings of the 2003 International Technical Conference on Circuit Systems, Computers and Communications*, Vol. 2 pp. 1290–1293, July 2003. Available in Portable Document Format from Internet: <http://dali.korea.ac.kr/publication/int_pro/paper/IntPro082.pdf>.

[8] J. Chou, M. Crouse, and K. Ramchandran, "A simple algorithm for removing blocking artifacts in block-transform coded images", in *IEEE Signal Processing Letters*, Vol. 5, No. 2, pp. 33–35, Feb. 1998.

[9] M. Crouse and K. Ramchandran, "Nonlinear constrained least squares estimation to reduce artifacts in transform-coded images", in *Proceedings of the International Conference on Image Processing*, Vol. 1, pp. 462–465, Oct. 1995.

[10] S. Delcorso and C. Miró, *Correction of temporal artefacts introduced by the MPEG-2 compression*, Suresnes (France): Philips Research France, Dec. 2001. Technical Report No. PRF C 2001-797.

[11] C. Derviaux, F.X. Coudoux, M.G. Gazalet, and P. Corlay, "Blocking artifact reduction of DCT coded image sequences using a visually adaptive postprocessing", in *Proceedings of the International Conference on Image Processing*, Vol. 1, pp. 5–8, Sept. 1996.

[12] D.L. Donoho, "De-noising by soft-thresholding", in *IEEE Transactions on Information Theory*, Vol. 41, No. 3, pp. 613–627, May 1995.

[13] A. Drouot, *A blocking artifact detection algorithm for decoded analog video*, Limeil-Brévannes (France): Laboratoires d'Électronique Philips, Sept. 1999. Technical Report No. LEP C 99-719.

[14] G. Fan and W.K. Cham, "Postprocessing of low bit-rate wavelet-based image coding using multiscale edge characterization", in *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 11, No. 12, pp.1263–1272, Dec. 2001.

[15] N.P. Galatsanos and A.K. Katsaggelos, "Methods for choosing the regularization parameter and estimating the noise variance in image restoration and their relation", in *IEEE Transactions on Image Processing*, Vol. 1, No. 3, pp. 322–336, July 1992.

[16] A. Gesnot, C. Miró, and J. Caviedes, *DCT frequency deblocking*, Limeil-Brévannes (France): Laboratoires d'Électronique Philips, July 2000. Technical Report No. LEP C 2000-742.

[17] C. Gomila, *Decoder apparatus and method for smoothing artifacts created during error concealment*, July 2004. Patent No. WO2004064396.

[18] R.A. Gopinath, M. Lang, H. Guo and J.E. Odegard, "Wavelet-based post-processing of low bit rate transform coded images", in *Proceedings of the IEEE International Conference on Image Processing*, Vol. 2, pp. 913–917, Nov. 1994.

[19] B.K. Gunturk, Y. Altunbasak, and R.M. Mersereau, "Multiframe blocking-artifact reduction for transform-coded video", in *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 12, No. 4, pp. 276–282, April 2002.

[20] S. Gupta, Y.T. Tse, *Coding parameter adaptive transform artifact reduction process*, July 1999. US Patent No. US5920356.

[21] G. de Haan, P.W.A.C. Biezen, H. Huijgen, and O.A. Ojo, "True-motion estimation with 3-D recursive search block matching", in *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 3, No. 7, pp. 368–379, Oct. 1993.

[22] G. de Haan, "IC for motion compensated de-interlacing, noise reduction, and picture rate conversion", in *IEEE Transactions on Consumer Electronics*, Vol. 45, No. 3, pp. 617–624, Aug. 1999.

[23] G. de Haan, *Video Processing for multimedia systems*, University Press Eindhoven, 3rd ed., 2003. ISBN 90-9014015-8.

[24] T. Hallbach and M. Wien, "Concepts and performance of next-generation video compression standardization", in *5th Nordic Signal Processing Symposium (NORSIG)*, Norway, Oct. 2002. Available in Portable Document Format from Internet: `<http://www.ncesd.org/vc/docs/H264_explained.pdf>`

[25] F. van der Heijden, *Image based measurement systems: object recognition and parameter estimation*, Chichester (UK): John Wiley and Sons, 1994. ISBN 0-471-950629.

[26] International Organisation for Standardization, *Information technology - generic coding of moving pictures and associated audio*, ISO, Feb. 2000. ITU-T Recommendation H.262 | ISO/IEC International Standard 13818-2.

[27] International Organisation for Standardization, *Information technology – coding of audio-visual objects – part 2: visual*, ISO, Dec. 2001. ISO/IEC International Standard 14496-2.

[28] International Telecommunication Union, *Advanced video coding for generic audiovisual services*, ITU-T, May 2003. Recommendation H.264.

[29] International Telecommunication Union, *Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios*, ITU-R, Oct. 1995. Recommendation BT.601-5, 1995.

[30] B. Jeon and J. Jeong, "Blocking artifacts reduction in image compression with block boundary discontinuity criterion", in *IEEE Transaction on Circuits and Systems for Video Technology*, Vol. 8, No. 3, pp. 345–357, June 1998.

[31] A. Joch, J. Au, B. Yu-sheng, *Low complexity deblocking filter*, May. 2004. US Patent No. US20040101059.

[32] J. Jung, *FaDA: a fast deblocking algorithm for MPEG-4*, Suresnes (France): Philips Research France, Aug. 2002. Technical Report No. PRF C 2002-922.

[33] P. Karandikar, U. Satyanarayana, *Deblocking block-based video data*, Nov. 2002. Patent No. WO2002096117.

[34] A. Kaup, "Image restoration for frame- and object-based video coding using an adaptive constrained least-squares approach", *Signal Processing*, Vol. 80, No. 11, pp. 2337–2345, 2000. Available in Portable Document Format from Internet: <http://www.lnt.de/lms/publications/web/lnt2000_39.pdf>.

[35] C. Kim, *Hybrid technique for reducing blocking and ringing artifacts in low-bit-rate coding*, Dec. 2003. US Patent No. US20030235248.

[36] S.D. Kim, J. Yi, and J.B. Ra, "A deblocking filter with two separate modes in block-based video coding", in *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 9, No. 1, pp. 156–160, Feb. 1999.

[37] Y. Kim and T. Yi, "Efficient post-processing for block-based compression video", in *Fourth EURASIP Conference focused on Video/Image Processing and Multimedia Communications*, Vol. 1, pp. 101–105, July 2003.

[38] Y. Kim, C.S. Park, and S.J. Ko, "Fast POCS based post-processing technique for HDTV", in *IEEE Transactions on Consumer Electronics*, Vol. 49, No. 4, pp. 1438–1447, Nov. 2003.

[39] S.N. Kryukov, T.O. Semenkova, and Z.A. Zaklika, *Removal of block encoding artifacts*, March 2003. US Patent No. US20030053708.

[40] Y.K. Lai, J. Li, and C.C.J. Kuo, "Image enhancement for low bit-rate JPEG and MPEG coding via postprocessing", in *Proceedings SPIE*, Vol. 2727, pp. 1484–1494, 1996. Available in Portable Document Format from Internet: <http://research.microsoft.com/users/jinl/paper_1996/vcip96.PDF>.

[41] S.H. Lee and M.G. Kang, "Spatio-temporal video filtering algorithm based on 3-D anisotropic diffusion equation", in *Proceedings of the 1998 International Conference on Image Processing*, Vol. 2, pp. 447–450, Oct. 1998.

[42] Y.L. Lee, H.C. Kim, and H.W. Park, "Blocking effect reduction of JPEG images by signal adaptive filtering", in *IEEE Transactions on Image Processing*, Vol. 7, No. 2, pp. 229–234, Feb. 1998.

[43] Y.L. Lee, H.W. Park, *Loop-filtering method for image data and apparatus therefor*, Dec. 2003. US Patent No. US6665346.

[44] E. Lesellier, *A blocking artifact detection-correction system for most frequent MPEG-2 encoding formats*, Limeil-Brévannes (France): Laboratoires d'Électronique Philips, May 2001. Technical Report No. LEP C 2001-772.

[45] E. Lesellier and A. Gesnot, *An overview of postprocessing methods for low-bitrate block.DCT-based compression.* Suresnet (France): Philips Research France, July 2001. Technical Report No. PRF C 2001-779.

[46] J. Li and C.C.J. Kuo, "Coding artifact removal with multiscale postprocessing", in *Proceedings of the International Conference on Image Processing*, Vol. 1, pp. 45–48, Oct. 1997.

[47] Z. Li and E.J. Delp, "MAP-based post processing of video sequences using 3-D Huber-Markov random field model", in *Proceedings of the 2002 IEEE International Conference on Multimedia and Expo*, Vol. 1, pp. 153–156, Aug. 2002.

[48] A.W.C. Liew and H. Yan, "Blocking artifacts suppression in block-coded images using overcomplete wavelet representation", in *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 14, No. 5, pp. 450–461, April 2004.

[49] T.-S. Liu and N. Jayant, "Adaptive postprocessing algorithms for low bit rate video signals", in *IEEE Transactions on Image Processing*, Vol. 4, No. 7, pp. 1032–1035, July 1995.

[50] G.A. Lunter, *Block-based motion estimation: some methods to deal with occlusion and convergence*, Eindhoven: Philips Research Laboratories, Feb. 2002. Technical Note No. 2002/076.

[51] J. Luo, D.W. Chen, K.J. Parker, and T.S. Huang, "Artifact reduction in low bit rate DCT-based image compression", in *IEEE Transactions on Image Processing*, Vol. 6, No. 9, pp. 1363–1368, Sept. 1996.

[52] A.G. MacInnos, S. Zhong, and J.R. Alvarez, *Method and apparatus for performing deblocking filtering*, March 2003. European Patent No. EP1296522 A2.

[53] S. Mallat and S. Zhong, "Characterization of signals from multiscale edges", in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 14, No. 7, pp. 710–732, July 1992.

[54] H.S. Malvar and D.H. Staelin, "The LOT: transform coding without blocking effects", in *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 37, No. 4, pp. 553–559, April 1989.

[55] C.E. Manning, *Block matching algorithms for motion compensated video compression*, Cork (Ireland): University College Cork, 1996. Master's thesis. Available from Internet `<http://www.newmediarepublic.com/dvideo/>`.

[56] T. Meier, K.N. Ngan, G. Crebbin, "A region-based algorithm for enhancement of images degraded by blocking effects", in *1996 IEEE TENCON – Digital Signal Processing Applications*, Vol. 1, pp. 405–408, Nov. 1996.

[57] S. Minami and A. Zakhor, "An optimization approach for removing blocking effects in transform coding", in *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 5, No. 2, pp. 74–82, April 1995.

[58] C. Miró, A. Gesnot, and J.E. Caviedes, *Device jointly implementing a post-processing and a decoding of data*, Dec. 2002. US Patent No. US20020196856.

[59] G. Morrison, "Video coding standards for multimedia: JPEG, H.261, MPEG", *IEE Colloquium on Technology Support of Multimedia*, pp. 2/1–2/4, Apr. 1992.

[60] MPEG Software Simulation Group, "*MPEG-2 Encoder / Decoder, Version 1.2*", Jul. 1996. C source code. Available from Internet: `<http://www.mpeg.org/MSSG>`.

[61] M. Najafo, A. Krylov, and D. Kortchagine, "Image deblocking with 2-D Hermite transform", in *Proceedings of the International Conference on Computer Graphics and Vision (GraphiCon 2002)*, Sept. 2003. Available in Portable Document Format from Internet: <http://www.graphicon.ru/2003/Proceedings/Technical/Krylov1.pdf>.

[62] M. Nikolova, "Regularization functions and estimators", in *International Conference on Image Processing*, Vol. 1, pp. 457–460, Sept. 1996.

[63] A. Nosratinia, "Postprocessing of JPEG-2000 images to remove compression artifacts", in *IEEE Signal Processing Letters*, Vol. 10, No. 10, pp. 296–299, Oct. 2003.

[64] A. Nosratinia, "Embedded post-processing for enhancement of compressed images", in *Proceedings of the Data Compession Conference*, pp. 62–71, March 1999.

[65] T. Özcelik, J.C. Brailean, and A.K. Katsaggelos, "Image and video compression algorithms based on recovery techniques using mean field annealing", in *Proceedings of the IEEE*, Vol. 83, No. 2, pp. 304–316, Feb. 1995.

[66] T.P. O'Rourke and R.L. Stevenson, "Improved image decompression for reduced transform coding artifacts", in *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 5, No. 6, pp. 490–499, Dec. 1995.

[67] H. Paek, R.C. Kim, and S.U. Lee, "On the POCS-based postprocessing technique to reduce the blocking artifacts in transform coded images", in *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 8, No. 3, June 1998.

[68] H. Paek, R.C. Kim, and S.U. Lee, "A DCT-based spatially adaptive post-processing technique to reduce the blocking artifacts in transform coded images", in *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 10, No. 1, pp. 36–41, Feb. 2000.

[69] K.-N. Park, K.-K. Kwon, S.-W. Ban, and K.-I. Lee, "Blocking artifacts reduction in block-coded images using self-similarity", in *Proceedings of the IEEE International Symposium on Industrial Electronics*, Vol. 3, pp. 1667–1670, June 2001.

[70] K.-N. Park, G.-W. Lee, K.K. Kwon, B.-S. Kom, and K.-I. Lee, *Blocking artifact reduction in block-coded image using interpolation and SAF based on edge map*, Daegu (Rep. of Korea): Kyungpook National University, School of Electrical Engineering and Computer Science, June 2001. Available in Portable Document Format from Internet: <http://www.kmutt.ac.th/itc2002/CD/pdf/18_07_45/TA2_PI/8.pdf>.

[71] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion", in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 12, No. 7, pp. 629–639, July 1990.

[72] G. Qiu, "MLP for adaptive postprocessing block-coded images", in *IEEE Transactions on circuits and systems for Video Technology*, Vol. 10, No. 8, pp. 1450– 1454, Dec. 2000.

[73] R.J.E. Rajae-Joordens and J. Engel, *Statistical analysis for paired comparison and ranking studies*, Eindhoven: Philips Research Laboratories, Feb. 2003. Technical Note No. 2002/527.

[74] B. Ramamurthi and A. Gersho, "Nonlinear space-variant postprocessing of block coded images", in *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 34, No. 5, pp. 1258–1268, Oct. 1986.

[75] H. Reeve and J. Lim, "Reduction of blocking effect in image coding", in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '83)*, Vol. 8, pp. 1212–1215, April 1983.

[76] I.E.G. Richardson, *H.264 / MPEG-4 Part 10: Context-based adaptive arithmetic coding CABAC*, Video and image compression resources and research (vcodex), March 2003. White paper. Available in Portable Document Format from Internet: <http://www.vcodex.com/h264_cabac.pdf>.

[77] I.E.G. Richardson, *H.264 / MPEG-4 Part 10: Prediction of intra macroblocks*, Video and image compression resources and research (vcodex), March 2003. White paper. Available in Portable Document Format from Internet: <http://www.vcodex.com/h264_intrapred.pdf>.

[78] I.E.G. Richardson, *H.264 / MPEG-4 Part 10: Reconstruction filter*, Video and image compression resources and research (vcodex), March 2003. White paper. Available in Portable Document Format from Internet: <http://www.vcodex.com/h264_loopfilter.pdf>.

[79] I.E.G. Richardson, *H.264 / MPEG-4 Part 10: Transform and quantization*, Video and image compression resources and research (vcodex), March 2003. White paper. Available in Portable Document Format from Internet: <http://www.vcodex.com/h264_transform.pdf>.

[80] M.Y. Shen and C.C. Jay Kuo, "Review of postprocessing techniques for compression artifact removal", in *Journal of Visual Communcation and Image Representation*, Vol. 1, No. 1, pp. 2–14, March 1998. Article No. VC970378. Available in Portable Document Format from Internet: <http://viola.usc.edu/newextra/Publication/PDF/selected/1998_JVCIR_Shen.pdf>.

[81] T. Shin, K. Cho and B.-H. Ahn, "Blocking effect reduction with content-based AC prediction in an MPEG-2 compressed video", in *IEEE Transactions on Consumer Electronics*, Vol. 45, No. 3, pp. 625–631, Aug. 1999.

[82] L.I. Smith, *A tutorial on principal components analysis*, Orthago (New Zealand), University of Othago, Computer Science Department, Feb. 2002. Student tutorial. Available in Portable Document Format from Internet: <http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf>.

[83] B.C. Song and K.W. Chun, "Motion-compensated temporal filtering for denoising in video encoder", in *Electronic Letters*, Vol. 4, No. 13, June 2004, pp. 802–804.

[84] S. Suthaharan, "A perceptually significant block-edge impairment metric for digital video coding", *2003 International Conference on Multimedia and Expo (ICME)*, Vol. 2, pp. 585–588, July 2003.

[85] C.Q. Tang and O.C. Au, "Comparison between block-based and pixel-based temporal interpolation for video coding", in *Proceedings of the 1998 IEEE International Symposium on Circuits and Systems*, Vol. 4, June 1998, pp. 122–125.

[86] G.A. Triantafyllidis, D. Tzovaras, D. Sampson, and M.G. Strintzis, "A hybrid algorithm for the removal of blocking artifacts", in *Proceedings of the 2002 IEEE International Conference on Multimedia and Expo*, Vol. 1, pp. 161–164, Aug. 2002.

[87] G.A. Triantafyllidis, D. Tzovaras, and M.G. Strintzis, "Blocking artifact detection and reduction in compressed data", in *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 12, No. 10, pp. 877–890, Oct. 2002.

[88] P.N. Tudor, "Mpeg-2 video compression", in *Electronics & Communication Engineering Journal*, Vol. 7, No. 6, pp. 257–264, Dec. 1995.

[89] D. Vatolin and S. Grishin, *VirtualDub MSU deblocking Filter v2.0*, Moscow: Moscow State University, Department of Computer Science, Graphics and Media Laboratory, Feb. 2005. Available from Internet: <http://www.compression.ru/video/deblocking/index_en.html>.

[90] C. Wang, W.-J. Zhang, and X.-Z. Fang, "Adaptive reduction of blocking artifacts in DCT domain for highly compressed images", in *IEEE Transactions on consumer electronics*, Vol. 50, No. 2, pp. 647–754, May 2004.

[91] J. Widén, *Principles of compression techniques*, Luleå University of Technology, Centre for Distance-spanning Technology (CDT), 1999. Course notes. Available from Internet: <http://www.cdt.luth.se/~johnny/courses/smd074_1999_2/CodingCompression/kap28/slide0.html>.

[92] H.R. Wu, M. Yuen, and B. Qiu, "Video coding distortion classification and quantitative impairment metrics", in *1996 3rd International Conference on Signal Processing*, Beijing: Institute of Electrical and Electronics Engineers, Vol. 2, pp. 962–965, Beijing, Oct. 1996.

[93] H.R. Wu and M. Yuen, "A generalized block-edge impairment metric for video coding", in *IEEE Signal Processing Letters*, Vol. 4, No. 11, pp. 317–320, Nov. 1997.

[94] Z. Xiaoqing, *Motion-compensated noise reduction in black-and-white motion picture films*, Stanford (CA): Stanford University, Electrical Engineering Department, March 2002. Student's report. Available in Portable Document Format from Internet: <http://www.stanford.edu/class/ee392j/Winter2002/projects/xiaoqing_report.pdf>.

[95] Z. Xiong, M.T, Orchard and Y.Q. Zhang, "A simple deblocking algorithm for JPEG compressed images using overcomplete wavelet representations", in *Proceedings of the 1997 International Symposium on Circuits and Systems*, Vol. 2, pp. 1077–1080, June 1997.

[96] L. Yan, "A nonlinear algorithm for enhancing low bit-rate coded motion video sequence", in *IEEE International Conference on Image Processing*, Vol. 2, pp. 923–927, Nov. 1994.

[97] S. Yang and Y.-H. Hu, "Coding artifacts removal using biased anisotropic diffusion", in *International Conference on Image Processing*, Vol. 2, pp. 346–349, Oct. 1997.

[98] S. Yang, S. Kittitornkun, Y.-H. Hu, T.Q. Nguyen, D.L. Tull, "Blocking artifact free inverse discrete cosine transform", in *Proceedings of the 2000 International Conference on Image Processing*, Vol. 4, pp. 869–872, Sept. 2000.

[99] Y. Yang, N.P. Galatsanos, and A.K. Katsaggelos, "Regularized reconstruction to reduce blocking artifacts of block discrete cosine transform compressed images", in *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 3, No. 6, pp. 421–432, Dec. 1993.

[100] Y. Yang, N.P. Galatsanos, and A.K. Katsaggelos, "Projection-based spatially adaptive reconstruction of block-transform compressed images", in *IEEE Transactions on Image Processing*, Vol. 4, No. 7, pp. 896–908, July 1995.

[101] Y. Yang and N.P. Galatsanos, "Removal of compression artifacts using projections onto convex sets and line process modeling", in *IEEE Transactions on Image Processing*, Vol. 6, No. 10, pp. 1345–1357, Oct. 1997.

[102] Y. Yang, *Video processing of compressed (MPEG-2) digital video for LCoS displays*, Briarcliff Manor, New York: Philips Research USA, Feb. 2003. Technical Note No. TN-2003-003.

[103] S. Yao, G. Feng, X. Lin, K.P. Lim, and W. Lin, "A coding artifacts removal algorithm based on spatial and temporal regularization", in *Proceedings of the 2003 International Conference on Image Processing*, Vol. 2, pp. 215–218, Sept. 2003.

[104] D.C. Youla, "Generalized image restoration by the method of alternating orthogonal projections", in *IEEE Transactions on Circuits an Systems*, Vol. 25, No. 9, pp. 694–702, Sept. 1978.

[105] D.C. Youla and H. Webb, "Image restoration by the method of convex projections: part 1 - theory", in *IEEE Transactions on Medical Imaging*, Vol. 1, No. 2, pp. 81–94, Oct. 1982.

[106] A. Zakhor, "Iterative procedures for reduction of blocking effects in transform image coding", in *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 2, No. 1, pp. 91–95, March 1992.

[107] M. Zhao, R.E.J. Kneepkens, P.M. Hofman, and G. de Haan, "Content adaptive de-blocking", in *2004 IEEE International Symposium on Consumer Electronics*, pp. 299-203, Sept. 2004.

[108] Q. Zhou, *Image processing circuit and method for reducing a difference between pixel values across an image boundary*, May 2001. US Patent No. US6236764.

[109] V. Zota, "Kompressionisten: Aktuelle Video-Codecs im Vergleich", in *C'T - Magazin für Computertechnik*, Heft 10, pp. 146–159, 2003. Available from Internet: <http://www.heise.de/ct/03/10/146/default.shtml>.

Appendix A

A block coding example

This Appendix illustrates the consequences of quantization with an example. Therefore, we use a codec developed by the MPEG Software Simulation Group (*MSSG*) as a reference [60]. Observe an intra coded 8×8 image detail in Figure A.1(a). The corresponding 8-bit sample values $b(x, y)$ are shown in Figure A.1(b), where full white represents the value $+127$, and full black the value -128. For convenience, we will drop the spatial ($\vec{x}_b$) and temporal (n) dependencies. The encoder performs the DCT transform of Equation 2.16, resulting in the DCT coefficient block $B(u, v)$ in A.1(c), where the coefficients are rounded to integer values according to the function

$$B'(v, u) = \lfloor B(v, u) + 0.499999 \rfloor. \tag{A.1}$$

Compare the coefficients with the contributions of the basis images in Figure 2.8(a) to the original image. Figure A.1(d) shows the DCT coefficients as an image, normalized by its maximum amplitude, where intermediate grey is the zero level. The more the coefficient deviates from zero, the more the corresponding basis image is present in the original image. Note that DCT coefficients can range from the value -1024 to $+1016$, which is representable using 12-bit numbers. Suppose the encoder uses the quantization matrix from Figure 2.8(b) with elements $quant(u, v)$. The quantized coefficients $Q(u, v)$ can be calculated with Equation A.2.

$$Q(u,v) = \begin{cases} \text{sign}(B'(u,v)) \cdot \left\lfloor \dfrac{\left\lfloor \frac{32 \cdot |B'(v,u)| + \left\lfloor \frac{quant(u,v)}{2} \right\rfloor}{quant(u,v)} \right\rfloor + \left\lfloor \frac{3 \cdot qscale + 2}{4} \right\rfloor}{2 \cdot qscale} \right\rfloor, & u \neq 0 \vee v \neq 0, \\ \text{sign}(B'(u,v)) \cdot \left\lfloor \dfrac{|B'(u,v)| + 4}{8} \right\rfloor & , u = 0 \wedge v = 0. \end{cases} \tag{A.2}$$

Suppose the encoder decides to use a quantization scale $qscale = 24$. Then the quantized coefficients become as in the matrix of Figure A.1(e). Traversing the quantized coefficients in zigzag order as indicated in Figure 2.9, yields the following sequence of zero run-length and value pairs (for the AC coefficients), where the EOB marker signals the End Of the Block:

$$(1,-1)(0,-5)(5,1)(0,5)(8,-1)(0,-2)(0,-3)(13,3)(0,1)(\text{EOB}).$$

Negative values are encoded as if they were positive. An additional bit added later encodes for the sign. Each pair is assigned a variable length code, as demonstrated in Table A.1. A total of 79 bits is required to encode the AC coefficients. The decoder reconstructs the quantized coefficients and de-quantizes them according to Equation A.3 (see Figure A.1(f)). The *quant* parameter is encoded in the stream.

$$\hat{F}(u,v) = \begin{cases} \text{sign}(Q(u,v)) \cdot \left\lfloor \dfrac{|Q(u,v)| \cdot quant(u,v) \cdot qscale}{16} \right\rfloor, & \text{for } u \neq 0 \vee v \neq 0, \\ 8 \cdot Q(u,v) & \text{, for } u = 0 \wedge v = 0. \end{cases} \tag{A.3}$$

The reconstructed samples can be found by rounding the IDCT samples (Figure A.1(g)). Figure A.1(h) shows the reconstructed result.

(run-length, absolute value)	variable length code	sign bit
(1, 1)	010	1
(0, 5)	11101	1
(5, 1)	000111	0
(0, 5)	11101	0
(8, 1)	0000101	1
(0, 2)	110	1
(0, 3)	0111	1
(8, 1)	0000101	1
(13, 3), no VLC defined, use escape code	000001	
6 bits to encode a zero-run-length of 13	001101	
12 bits to encode the value (2's complement)	000000000011	
(0, 1)	10	0
end-of-block marker	0110	

Table A.1: Encoded AC coefficients for $qscale = 24$.

Now suppose the encoder decided to use $qscale = 112$. Then the quantized coefficient matrix would look like Figure A.1(i), yielding the zero-run length and value pairs

$$(2, -1)(6, 1)(24, 1)(\text{EOB})$$

and their corresponding variable length codes in Table A.2. The encoder now outputs only 32 bits, gaining compression with a factor of more than two. After de-quantization (Figure A.1(j)), the reconstructed samples (Figure A.1(k)) look like in Figure A.1(l), clearly showing a ghost edge near the original edge (ringing artefact).

(run-length, absolute value)	variable length code	sign bit
(2, 1)	00101	1
(6, 1)	0000110	0
(24, 1)	0000000011101	0
end-of-block marker	0110	

Table A.2: Encoded AC coefficients for $qscale = 112$.

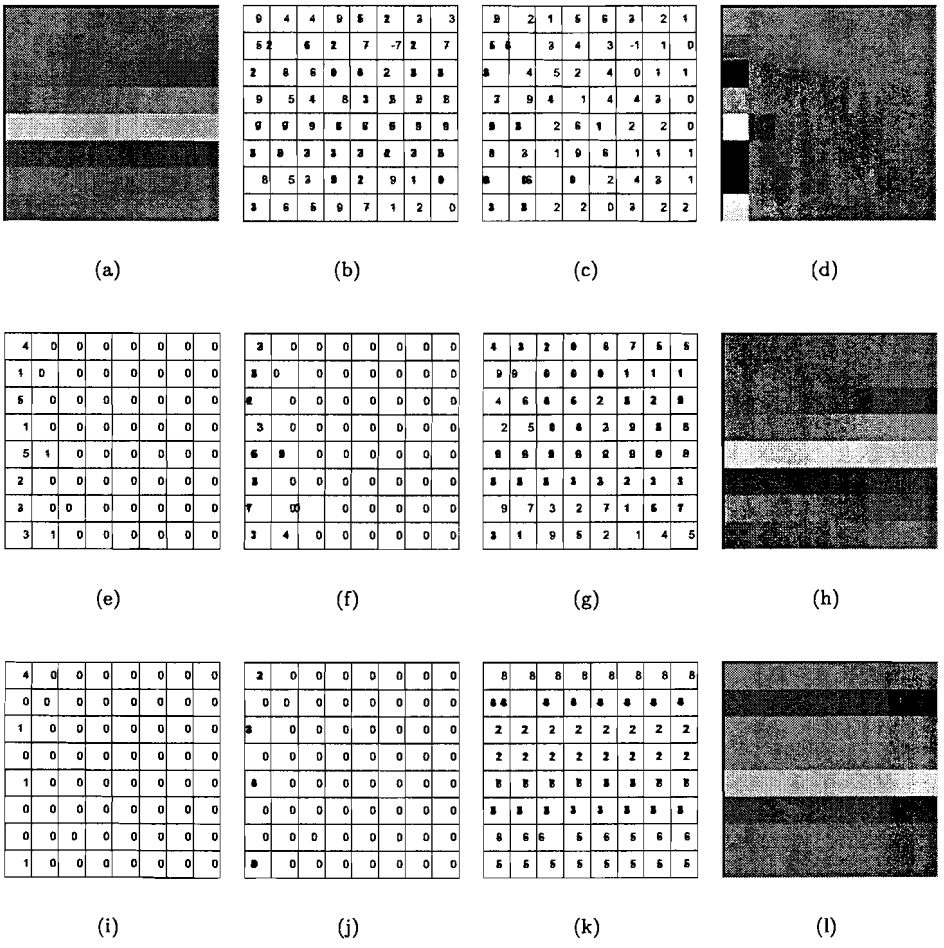


Figure A.1: MPEG-2 block coding example. Figure (a) is an original image detail. The corresponding sample values and its DCT coefficient matrix are shown in (b) and (c) respectively. (d) shows the DCT coefficient matrix as a scaled image. After quantization (e), the inverse quantization leads to an approximated coefficient block (f) from which a reconstruction can be made (g). (i) is the result of coarse quantization, yielding in coefficient matrix (j) and reconstruction (k). (h) and (l) show the reconstructed images after moderate quantization and after coarse quantization respectively.

Appendix B

Measurement results

The following tables show the numerical objective measurement results for the experiments in this report. The results of the MSU de-blocking algorithm could not be expressed in terms of image fidelity, because the reference decoder could not be used. Furthermore, the output format required conversion into another colour space, introducing more deviation from the original and biasing the error measures. The PSNR improvement with respect to the decoded sequence after colour space conversion, as well as the GBIM metric, could in fact be calculated.

For all sequences, the first and the last frames were not included in the measurement for reasons found in Chapter 4.

algorithm	MSE	$PSNR$ $[dB]$	$\Delta PSNR$ $[dB]$	M_{hGBIM}	M_{vGBIM}	M_{GBIM}
original	0	∞	∞	1.25771	1.54992	1.40382
decoded	84.907	28.841	0	3.03044	4.07051	3.55047
`mpeg-4`	74.623	29.402	+0.561	1.31613	1.68223	1.49923
`msu`			+0.490	1.19963	1.43915	1.31990
`shift`	70.853	29.627	+0.786	1.58516	1.85251	1.64577
`3-mode`	76.853	29.274	+0.433	1.22271	1.29609	1.25940
`blind`	73.300	29.421	+0.580	1.25295	1.49302	1.37299
`comp`	76.126	29.315	+0.474	1.06332	1.13404	1.09686
`comp-st`	76.054	29.320	+0.479	1.85294	2.23922	2.04608
`comp-fb`	75.220	29.367	+0.526	1.07413	1.19666	1.13540
`noise`	87.010	28.735	-0.106	1.36753	1.58733	1.47743
`noise-st`	77.020	29.265	+0.424	2.10398	2.60001	2.35200
`noise-fb`	75.893	29.329	+0.488	1.26647	1.50416	1.39032
`median-fb`	74.451	29.412	+0.571	1.50687	1.77963	1.64325

Table B.1: Measurement results for the *birds* sequence, encoded with $500kbps$.

algorithm	MSE	$PSNR$ $[dB]$	$\Delta PSNR$ $[dB]$	M_{hGBIM}	M_{vGBIM}	M_{GBIM}
original	0	∞	∞	1.25771	1.54992	1.40382
decoded	80.221	29.088	0	2.94276	3.92994	3.43635
`mpeg-4`	70.312	29.661	+0.573	1.30390	1.66612	1.48501
`msu`			+0.470	1.23129	1.46911	1.35020
`shift`	66.839	29.881	+0.793	1.57362	1.83665	1.70514
`3-mode`	72.071	29.553	+0.465	1.18420	1.28453	1.23437
`blind`	70.156	29.670	+0.583	1.24225	1.48296	1.36261
`comp`	71.955	29.560	+0.472	1.05171	1.10604	1.07888
`comp-st`	71.681	29.577	+0.489	1.80309	2.13673	1.96991
`comp-fb`	70.955	29.621	-0.533	1.06341	1.18046	1.12194
`noise`	83.069	28.936	-0.152	1.34768	1.55196	1.45282
`noise-st`	72.714	29.515	+0.428	2.05226	2.49789	2.27508
`noise-fb`	71.730	29.574	+0.483	1.26193	1.48536	1.37365
`median-fb`	70.229	29.666	+0.578	1.49103	1.76279	1.62691

Table B.2: Measurement results for the *birds* sequence, encoded with $1,000kbps$.

algorithm	MSE	$PSNR$ $[dB]$	$\Delta PSNR$ $[dB]$	M_{hGBIM}	M_{vGBIM}	M_{GBIM}
original	0	∞	∞	1.02611	1.16694	1.09653
decoded	58.239	30.479	0	2.66344	2.38934	3.02639
`mpeg-4`	56.387	30.619	+0.140	1.21650	1.19292	1.20471
`msu`			+0.040	1.36497	1.34408	1.35453
`shift`	55.631	30.678	+0.199	1.23465	1.31979	1.27722
`3-mode`	56.958	30.575	+0.096	0.89871	0.90227	0.90049
`blind`	57.023	30.570	+0.091	1.11914	1.07612	1.09763
`comp`	56.696	30.595	+0.116	1.26239	1.54099	1.40169
`comp-st`	57.065	30.567	+0.088	1.88326	1.90267	1.89297
`comp-fb`	57.827	30.509	+0.030	1.10487	1.20925	1.15706
`noise`	56.008	30.648	+0.169	1.46824	1.70658	1.58741
`noise-st`	56.535	30.608	+0.129	1.99942	2.00558	2.00250
`noise-fb`	57.372	30.544	+0.065	1.24040	1.31816	1.21928
`median-fb`	55.579	30.682	+0.203	1.22429	1.23058	1.22744

Table B.3: Measurement results for the *golf* sequence, encoded with $500kbps$.

algorithm	MSE	$PSNR$ $[dB]$	$\Delta PSNR$	M_{hGBIM}	M_{vGBIM}	M_{GBIM}
original	0	∞	∞	1.02611	1.16694	1.09653
decoded	30.237	33.325	0	1.65807	1.63111	1.64459
`mpeg-4`	29.148	33.485	+0.160	1.06532	1.08482	1.07507
`msu`			+0.034	1.21272	1.26603	1.23938
`shift`	28.732	33.547	+0.222	1.19275	1.25503	1.22389
`3-mode`	30.220	33.328	+0.003	0.76882	0.78717	0.77800
`blind`	30.033	33.355	+0.030	0.98838	0.98750	0.98794
`comp`	29.391	33.449	+0.124	1.01794	1.15929	1.08862
`comp-st`	29.361	33.453	+0.128	1.21748	1.28090	1.24919
`comp-fb`	30.630	33.269	-0.056	0.96209	1.03302	0.99756
`noise`	29.151	33.484	+0.159	1.14627	1.28503	1.21565
`moise-st`	28.959	33.513	+0.188	1.31302	1.37520	1.34411
`noise-fb`	30.288	33.318	-0.007	1.05842	1.12845	1.09344
`median-fb`	29.057	33.498	+0.173	1.05761	1.09005	1.07383

Table B.4: Measurement results for the *golf* sequence, encoded with $1,000kbps$.

algorithm	MSE	$PSNR$ $[dB]$	$\Delta PSNR$ $[dB]$	M_{hGBIM}	M_{vGBIM}	M_{GBIM}
original	0	∞	∞	1.35546	1.17273	1.26410
decoded	40.702	32.035	0	4.17340	4.50997	4.34169
`mpeg-4`	34.082	32.741	+0.706	1.53486	1.69861	1.61674
`msu`			+0.207	1.22451	1.48821	1.35636
`shift`	34.844	32.710	+0.675	1.97939	2.49571	2.23755
`3-mode`	35.157	32.671	+0.636	1.17659	1.32927	1.25293
`blind`	33.775	32.845	+0.810	1.37661	1.52132	1.44897
`comp`	36.083	32.558	+0.523	1.86521	2.45947	2.16234
`comp-st`	37.560	32.384	+0.349	2.81938	3.31506	3.06722
`comp-fb`	34.146	32.797	+0.762	1.34236	1.67461	1.50854
`noise`	37.820	32.354	+0.319	2.13241	2.71087	2.42164
`noise-st`	37.978	32.335	+0.300	2.99358	3.48212	3.23785
`noise-fb`	34.446	32.759	+0.724	1.53387	1.84417	1.68902
`median-fb`	33.798	32.842	+0.807	1.64048	1.88111	1.76080

Table B.5: Measurement results for the *k3* sequence, encoded with $500kbps$.

algorithm	MSE	$PSNR$ $[dB]$	$\Delta PSNR$ $[dB]$	M_{hGBIM}	M_{vGBIM}	M_{GBIM}
original	0	∞	∞	1.35546	1.17273	1.26410
decoded	35.889	32.580	0	3.95674	4.25623	4.10649
`mpeg-4`	30.398	33.302	+0.772	1.49885	1.68033	1.58959
`msu`			+0.206	1.22329	1.40942	1.31636
`shift`	30.531	33.283	+0.703	1.90344	2.43646	2.16995
`3-mode`	30.977	33.220	+0.640	1.14925	1.29892	1.22409
`blind`	29.861	33.380	+0.800	1.35507	1.50991	1.43249
`comp`	31.643	33.128	+0.548	1.76547	2.29509	2.03028
`comp-st`	32.799	32.972	+0.392	2.57873	3.04436	2.81155
`comp-fb`	30.450	33.295	+0.715	1.30195	1.54729	1.42462
`noise`	33.406	32.893	+0.313	2.01922	2.54477	2.28200
`noise-st`	33.262	32.911	+0.331	2.76065	3.22018	2.99042
`noise-fb`	30.798	33.246	+0.666	1.49742	1.72625	1.61184
`median-fb`	29.815	33.469	+0.889	1.60585	1.85456	1.73021

Table B.6: Measurement results for the *k3* sequence, encoded with $1,000kbps$.

algorithm	MSE	$PSNR$ $[dB]$	$\Delta PSNR$ $[dB]$	M_{hGBIM}	M_{vGBIM}	M_{GBIM}
original	0	∞	∞	1.18883	1.00864	1.09874
decoded	20.466	35.021	0	4.74410	4.93828	4.84119
`mpeg-4`	17.476	35.706	+0.685	1.55637	1.57295	1.56466
`msu`			-0.018	1.71648	1.74497	1.73073
`shift`	16.109	36.060	+1.039	1.50293	1.36281	1.43287
`3-mode`	17.762	35.636	+0.615	1.16588	1.15495	1.16042
`blind`	17.217	35.771	+0.750	1.40486	1.38532	1.39509
`comp`	18.001	35.578	+0.557	2.22937	2.19771	2.21354
`comp-st`	18.724	35.407	+0.386	3.26967	3.25373	3.26170
`comp-fb`	17.558	35.686	+0.665	1.36973	1.35849	1.36411
`noise`	20.776	34.955	-0.066	2.50591	2.47810	2.49201
`noise-st`	18.966	35.351	+0.330	3.41716	3.39747	3.40732
`noise-fb`	17.737	35.642	+0.621	1.55707	1.56123	1.55915
`median-fb`	17.272	35.757	+0.736	1.72208	1.63684	1.67946

Table B.7: Measurement results for the *ngc* sequence, encoded with $500kbps$.

algorithm	MSE	$PSNR$ $[dB]$	$\Delta PSNR$ $[dB]$	M_{hGBIM}	M_{vGBIM}	M_{GBIM}
original	0	∞	∞	1.18883	1.00864	1.09874
decoded	19.769	35.171	0	4.53701	4.74176	4.63939
`mpeg-4`	16.983	35.831	+0.660	1.54234	1.56791	1.55513
`msu`			-0.022	1.72389	1.74309	1.73349
`shift`	15.659	36.183	+1.012	1.49971	1.35914	1.42943
`3-mode`	17.243	35.765	+0.594	1.15864	1.15185	1.15525
`blind`	16.748	35.891	+0.720	1.39026	1.37665	1.38351
`comp`	17.423	35.719	+0.548	2.22869	2.10805	2.16837
`comp-st`	17.988	35.581	+0.410	3.07613	2.98492	3.03053
`comp-fb`	17.063	35.810	+0.639	1.38551	1.34916	1.36734
`noise`	20.308	35.054	-0.117	2.48143	2.39775	2.43955
`noise-st`	18.215	35.526	+0.355	3.23366	3.16508	3.19936
`noise-fb`	17.236	35.766	+0.595	1.57566	1.56731	1.57149
`median-fb`	16.766	35.886	+0.715	1.68193	1.61641	1.64917

Table B.8: Measurement results for the *ngc* sequence, encoded with $1,000kbps$.

algorithm	MSE	$PSNR$ $[dB]$	$\Delta PSNR$ $[dB]$	M_{hGBIM}	M_{vGBIM}	M_{GBIM}
original	0	∞	∞	1.28188	1.04333	1.16261
decoded	50.509	31.097	0	4.06136	4.11716	4.08926
mpeg-4	45.556	31.545	+0.448	1.35256	1.54353	1.44805
msu			+0.114	1.43767	1.64712	1.54240
shift	43.528	31.743	+0.646	1.47635	1.48215	1.47925
3-mode	46.060	31.498	+0.401	1.04796	1.20882	1.12839
blind	45.180	31.581	+0.484	1.19958	1.38743	1.28351
comp	47.706	31.345	+0.248	2.23166	2.36101	2.29634
comp-st	49.496	31.185	+0.088	3.51285	3.66047	3.58716
comp-fb	46.780	31.430	+0.333	1.12667	1.53798	1.33233
noise	77.363	29.245	-1.852	2.29691	2.36676	2.33184
noise-st	50.380	31.108	+0.011	3.52537	3.65671	3.59104
noise-fb	47.639	31.351	+0.254	1.21080	1.58537	1.39809
median-fb	47.786	31.338	+0.241	1.62473	1.80094	1.71784

Table B.9: Measurement results for the *ngc2* sequence, encoded with $500kbps$.

algorithm	MSE	$PSNR$ $[dB]$	$\Delta PSNR$ $[dB]$	M_{hGBIM}	M_{vGBIM}	M_{GBIM}
original	0	∞	∞	1.28188	1.04333	1.16261
decoded	37.542	32.386	0	3.27603	3.46846	3.37225
mpeg-4	33.766	32.846	+0.460	1.22710	1.46068	1.34389
msu			+0.090	1.29674	1.55920	1.42797
shift	32.071	33.070	+0.684	1.33006	1.38020	1.35513
3-mode	34.267	32.782	+0.396	0.92518	1.10985	1.01752
blind	33.983	32.818	+0.432	1.10003	1.30643	1.20323
comp	35.162	32.670	+0.284	1.81570	2.12447	1.97009
comp-st	36.727	32.481	+0.095	2.86936	3.07161	2.97049
comp-fb	35.625	32.613	+0.227	1.11778	1.42606	1.27192
noise	54.854	30.739	-1.647	1.90441	2.13386	2.01914
noise-st	37.643	32.374	-0.012	2.86066	3.06143	2.96105
noise-fb	36.529	32.504	+0.118	1.17074	1.46427	1.31751
median-fb	36.226	32.541	+0.155	1.47655	1.67570	1.57613

Table B.10: Measurement results for the *ngc2* sequence, encoded with $1,000kbps$.

algorithm	MSE	$PSNR$ $[dB]$	$\Delta PSNR$ $[dB]$	M_{hGBIM}	M_{vGBIM}	M_{GBIM}
original	0	∞	∞	1.15184	1.22066	1.18625
decoded	22.580	34.594	0	2.98830	4.42371	3.70601
`mpeg-4`	20.370	35.041	+0.447	1.18050	1.67918	1.42984
`msu`			-0.115	1.77001	2.38677	2.07839
`shift`	18.839	35.380	+0.786	1.12077	1.85169	1.48623
`3-mode`	20.652	34.981	+0.387	0.95976	1.36523	1.16251
`blind`	22.904	34.532	-0.062	0.98548	1.43508	1.21028
`comp`	21.596	34.787	+0.193	2.19665	2.75365	2.47515
`comp-st`	22.447	34.619	+0.025	2.91022	4.21101	3.56062
`comp-fb`	24.928	34.164	-0.430	1.00337	1.60090	1.30214
`noise`	21.027	34.903	+0.309	2.33057	2.92710	2.61645
`noise-st`	22.385	34.631	+0.037	2.91876	4.21320	3.56598
`noise-fb`	24.873	34.174	-0.420	1.02586	1.62904	1.32745
`median-fb`	19.940	35.134	+0.540	1.37736	1.85851	1.61794

Table B.11: Measurement results for the *susie* sequence, encoded with $500kbps$.

algorithm	MSE	$PSNR$ $[dB]$	$\Delta PSNR$ $[dB]$	M_{hGBIM}	M_{vGBIM}	M_{GBIM}
original	0	∞	∞	1.15184	1.22066	1.18625
decoded	3.946	42.169	0	1.49360	1.87070	1.68215
`mpeg-4`	3.680	42.472	+0.303	1.10970	1.41260	1.26115
`msu`			+0.041	1.21868	1.49614	1.35741
`gs`	3.760	42.379	+0.210	1.03886	1.30387	1.17137
`3-mode`	4.291	41.805	-0.364	0.64659	0.88250	0.76455
`blind`	11.684	37.455	-4.714	0.87556	1.26520	1.07038
`comp`	3.937	42.180	+0.011	1.32200	1.68687	1.50444
`comp-st`	3.919	42.199	+0.030	1.46407	1.83061	1.64734
`comp-fb`	14.096	36.640	-5.529	0.88101	1.28673	1.08387
`noise`	3.656	42.500	+0.331	1.36706	1.73774	1.55240
`noise-st`	3.896	42.225	+0.056	1.46942	1.83969	1.65456
`noise-fb`	14.088	36.642	-5.527	0.89326	1.29879	1.09603
`median-fb`	4.807	41.312	-0.857	1.03194	1.41999	1.22597

Table B.12: Measurement results for the *susie* sequence, encoded with $1,000kbps$.

Appendix C

Perception test results

This Appendix contains the results of the paired comparison perception test results dealt with in Chapter 5. A summary of the test parameters is listed below.

- Four algorithms were evaluated (`mpeg-4`, `comp-fb`, `noise_fb`, `median-fb`), making 6 combinations for each source.
- Three sources were shown (`birds`, `golf`, `ngc`), making a total of 18 pairs to be rated for overall picture quality by each test person.
- The sequences were shown in greyscale, displayed in palindromic order, excluding the first and last frame.
- Display: Sharp 37" large flat panel LCD monitor, type `LC-M3700`.
- The screen is situated in a moderately illuminated perception room against a grey background.
- Screen distance was 3 times the active display height. The SD content was not scaled to avoid scaling artefacts.
- 15 people, experienced and non-experienced viewers, were asked for their judgement on overall quality.

Table C.1 shows the result of the test, indicating the number of times that the sequence marked with "1" was preferred over the sequence marked with "2", for each combination and for each sequence. The table also indicates the percentage of the total number of comparisons. The percentage can also be thought of as *fractions*, which is simply the number in the last column in Table C.1 divided by the total number of times that the specific comparison was made (15 in this case).

A statistical model to evaluate the relationship between the algorithms one-dimensionally is the *Thurstone Model*, which is a special case of the *Generalized Linear Model* [73]. For a simple two-algorithm comparison, the model assumes a normal distribution of the probabilities:

$$P(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{(x-\mu)^2}{2\sigma^2}}. \tag{C.1}$$

If $\Phi(z)$ is the cumulative normal probability function that equals the fraction,

$$\Phi(z) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{z} e^{\frac{(x-\mu)^2}{2\sigma^2}} dx = \frac{1}{2}\left(1 + \mathrm{erf}\left(\frac{z-\mu}{\sigma\sqrt{2}}\right)\right), \tag{C.2}$$

i.e. the surface below the probability distribution function drawn in Figure C.1, the z-score is a measure that indicates how much the ratio between "better" and "worse" deviates from the fifty-fifty situation. In the fifty-fifty case, both algorithms perform equally and their z-score is zero.

sequence	mpeg-4	comp-fb	noise-fb	median-fb	# (sequence 1 preferred over sequence 2)
birds	1	2			0 (0.00 %)
golf	1	2			6 (40.0 %)
ngc	1	2			5 (33.3 %)
birds	1		2		1 (6.67 %)
golf	1		2		5 (33.3 %)
ngc	1		2		8 (53.3 %)
birds	1			2	2 (13.3 %)
golf	1			2	2 (13.3 %)
ngc	1			2	1 (6.67 %)
birds		1	2		9 (60.0 %)
golf		1	2		7 (46.7 %)
ngc		1	2		7 (46.7 %)
birds		1		2	7 (46.7 %)
golf		1		2	1 (6.67 %)
ngc		1		2	4 (26.7 %)
birds			1	2	12 (80.0 %)
golf			1	2	4 (26.7 %)
ngc			1	2	2 (13.3 %)

Table C.1: Visual perception test results for the listed sequences at $1,000kbps$.

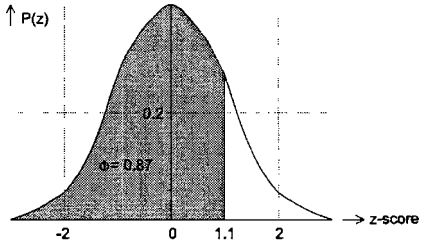


Figure C.1: The normal distribution probability function for $\mu = 0$ and $\sigma = 1$, showing the relation between the cumulative probability function and the z-score.

Note that we forced the observer to make a decision, so that the probabilities of "1 better than 2" and "2 better than 1" add to unity. For two or more algorithms, or in this case 4, the calculation can be generalized using a regression technique as described in [73].

Furthermore, the numbers in Table C.1 show sequence-dependency. A numerical analysis (by means of a χ^2 test) with statistical data-mining software *S-Plus 6.0 Professional Release* from the *Insightful* corporation indeed supports the observation that the results depend on the sequence, assuming that a maximum of 5% of the decisions in the perception test were false [73]. The results, therefore, have to be evaluated individually.

	mpeg-4	comp-fb	noise-fb	median-fb
birds	0	1.4326535422828564	0.18173456751485273	0.34130954418739745
golf	0	1.5357219276276166	0.4261479436595028	0.14923344530221638
ngc	0	1.1336589430669815	1.2335817955761954	1.1932447812559164

Table C.2: Z-scores for each algorithm, for each sequence.