NTNU – Trondheim
Norwegian University of
Science and Technology

MCTF and JPEG 2000 Based Wavelet Video Coding Compared to the Future HEVC Standard

Frøy Brede Tureson Erlid

Master of Science in Communication Technology
Submission date: June 2012
Supervisor: Andrew Perkis, IET

Norwegian University of Science and Technology
Department of Electronics and Telecommunications

Problem Description

High Efficiency Video Coding (HEVC) is an emerging video coding standard under joint development by VCEG and MPEG. This new standard is meant to be the successor of H.264, by providing significantly better compression capability and functionalities [1]. While HEVC is based on the traditional hybrid video coding structure, other ways of compressing video exists. One such way is wavelet coding with Motion Compensated Temporal Filtering (MCTF) [2]. The task is to develop a video coder that uses MCTF for reducing temporal redundancy, and JPEG 2000 for spatial coding. This video coder shall be compared to HEVC by both objective and subjective assessments.

[1] G. J. Sullivan and J.-R. Ohm, “Recent developments in standardization of high efficiency video coding (HEVC)," in SPIE Applications of Digital Image Processing XXXIII, ser. Proceedings of SPIE, A. G. Tescher, Ed., vol. 7798, no. 7798-30, August 2010.

[2] C.-C. Lin, Y.-T. Hwang, K.-H. Tseng, and S.-W. Chen, “Wavelet Based Lossless Video Compression Using Motion Compensated Temporal Filtering," in IEEE Workshop on Signal Processing Systems 2007, October 2007, pp. 686-691.

Preface

This thesis was written in the spring of 2012 in my final semester of the Master of Science program Communication Technology, at the Norwegian University of Science and Technology.

I would like to thank my supervisor, Professor Andrew Perkis, for valuable guidance and advice during the work on this thesis. I would also like to thank Magnus Jeffs Tovslid for his proofreading, and for the collaboration when we were setting up the lab. Lastly, I would like to thank everyone who participated in the subjective assessment.

Frøy Tureson Erlid
Trondheim, June 11, 2012

Abstract

Video and multimedia content has over the years become an important part of our everyday life. At the same time, the technology available to consumers has become more and more advanced. These technologies, such as streaming services and advanced displays, has enabled us to watch video content on a large variety of devices, from small, battery powered mobile phones to large TV-sets.

Streaming of video over the Internet is a technology that is getting increasingly popular. As bandwidth is a limited resource, efficient compression techniques are clearly needed. The wide variety of devices capable of streaming and displaying video suggest a need for scalable video coders, as different devices might support different sets of resolutions and frame rates.

As a response to the demands for efficient coding standards, VCEG and MPEG are jointly developing an emerging video compression standard called High Efficiency Video Coding (HEVC). The goal for this standard is to improve the coding efficiency as compared to H.264, without affecting image quality. A scalable video coding extension to HEVC is also planned to be developed.

HEVC is based on the classic hybrid coding approach. This however, is not the only way to compress video, and attention is given to wavelet coders in the literature. JPEG 2000 is a wavelet image coder that offers spatial and quality scalability. Combining JPEG 2000 with Motion Compensated Temporal Filtering (MCTF) gives a wavelet video coder which offers both temporal, spatial and quality scalability, without the need for complex extensions.

In this thesis, a wavelet video coder based on the combination of MCTF and JPEG 2000 was implemented. This coder was compared to HEVC by performing objective and subjective assessments, with the use case being streaming of video with a typical consumer broadband connection. The objective assessment showed that HEVC was the superior system in terms of both PSNR and SSIM. The subjective assessment revealed that observers preferred the distortion produced by HEVC over the proposed system. However, the results also indicated that improvements to the proposed system can be made that could possibly enhance the objective and subjective quality. In addition, indications were also found that suggest that a use case operating with higher bit rates is more suitable for the proposed system.

Sammendrag

Video og multimedia har i løpet av årene blitt en viktig del av vår hverdag. Samtidig har teknologien som er tilgjengelig for forbrukerne blitt stadig mer avansert. Denne teknologien, som for eksempel videostrømmingstjenester og avanserte skjermer, har gjort det mulig for oss å se videomateriale på et stort og variert utvalg enheter, fra små, batteridrevne mobiltelefoner til store fjernsynsapparater.

Strømming av video over Internett er en teknologi som blir mer og mer populær. Ettersom båndbredde er en begrenset ressurs, er det et klart behov for effektive kompresjonsteknikker. Den store variasjonen av enheter som kan strømme og vise video antyder et behov for skalerbare videokodere, ettersom ulike enheter kan støtte ulike sett oppløsninger og bildefrekvenser.

Som en respons til det økende behovet for effektive videokodingstandarder har VCEG og MPEG sammen startet utviklingen av en kommende videokodingstandard kalt High Efficiency Video Coding (HEVC). Målet for denne standarden er å forbedre kodingseffektiviteten sammenliknet med H.264, uten å tape bildekvalitet. En skalerbar utvidelse av HEVC er også planlagt å bli utviklet.

HEVC er basert på det klassiske hybridkoder-designet. Dette er dog ikke den eneste måten å komprimere video på, og wavelet-kodere blir viet oppmerksomhet i litteraturen. JPEG 2000 er en wavelet-bildekoder som tilbyr skalerbarhet i oppløsning og kvalitet. Ved å kombinere JPEG 2000 med bevegelseskompensert temporal filtrering, får man en wavelet-videokoder som tilbyr skalerbarhet i forhold til både bildefrekvens, oppløsning og kvalitet, uten å måtte lage komplekse utvidelser.

I denne masteroppgaven ble en wavelet-videokoder basert på kombinasjonen av JPEG 2000 og bevegelseskompensert temporal filtrering implementert. Denne koderen ble sammenliknet med HEVC ved å gjennomføre objektive og subjektive tester, hvor bruksscenariet var strømming av video over en typisk bredbåndstilkobling. Den objektive testen viste at HEVC var det overlegne systemet i forhold til både PSNR og SSIM. Den subjektive testen avslørte at testdeltakerne foretrakk distorsjonen produsert av HEVC over distorsjonen produsert av det egenutviklede systemet. Resultatene indikerte dog også at det egenproduserte systemet kan forbedres slik at objektiv og subjektiv kvalitet muligens blir bedre. I tillegg var det også indikasjoner på at det foreslåtte systemet vil fungere bedre i et bruksscenario som opererer med høyere bitrater.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Ever since television became commercially available, video content has become an increasing part of our daily life. Today, in addition to watching TV on High Definition (HD) displays in our living room, we stream content from the Internet to our computers, laptops, tablets and mobile phones. The technological development since the early days has been immense. The large flatscreens and home theatre setups of today stand in sharp contrast to the flickering, black-and-white TV sets of the past that were more a furniture and status symbol than a source of entertainment. Better capturing and viewing equipment has given us better video quality. This increase in content quality dictates the necessity of video compression.

According to Cisco, mobile video traffic was 52 percent of the total mobile data traffic at the end of 2011, and by 2016, they predict that two-thirds of the world's mobile data traffic will be video [1]. In addition, they predict that by the end of 2015, 62 percent of the consumer Internet traffic will be video [2]. It is evident that streaming video over the Internet is increasing in popularity, and that a variety of devices, both mobile and stationary, are used for streaming.

It seems that to an increasing degree, we want high resolution video on our TV screens, and we want to stream video on our home computers, laptops, battery powered and small screen devices. The demands for quality is high, no matter what type of device we use. These demands calls for new and more efficient video coding standards, and the variety of devices capable of streaming suggests the need for scalable coders.

High Efficiency Video Coding (HEVC) is a future video coding standard under development, aiming to be the successor of H.264/AVC. This emerging standard is based on the traditional hybrid coding structure. At the same time, there is a lot of research and focus on wavelet coders. An important topic is how these coders remove temporal redundancy, and Motion Compensated Temporal Filtering (MCTF) is a technique for doing this. By combining MCTF with JPEG 2000, a wavelet image coder, a wavelet video coder is obtained. The advantage of this

approach is that it in a natural way offers scalability in terms of both quality and spatial and temporal resolutions. To make a standard such as HEVC support scalability requires complex extensions to be developed. Indeed, the development of such an extension to HEVC is planned [3]. The motivation for this thesis is therefore to investigate how a video coder based on MCTF and JPEG 2000 performs compared to HEVC. If the former system has a performance comparable to HEVC in terms of quality, it could suggest that efforts for creating the next-generation scalable video coding system should focus on wavelet based alternatives rather than extending HEVC.

In the work on this thesis, a wavelet video coder based on the combination of JPEG 2000 and MCTF was developed. By utilizing both objective and subjective assessment methods, the developed codec was compared to version 5.2 of the HEVC Test Model.

The thesis is organized as follows. Chapter 2 presents the theory relevant for the work that was performed. Information about hybrid video coding in general and the emerging HEVC standard is given, in addition to theory on motion estimation and wavelet coding with MCTF. An introduction to the JPEG 2000 standard is also made. Chapter 3 describes a proposed wavelet codec that was made as part of the thesis. A use case is presented, and the methods used to test the proposed codec in comparison to HEVC is described, along with a description of the test material. In chapter 4, the results from the objective and subjective test that was performed is presented and discussed. An evaluation of the proposed wavelet codec is also made. Finally, in Chapter 5, the thesis is concluded.

Chapter 2

Theory

This chapter presents the theory that was used in this thesis. Section 2.1 gives a general description of the hybrid video coding structure, while section 2.2 gives theory on motion estimation. An overview of HEVC is presented in section 2.3, and wavelet video coding, MCTF and JPEG 2000 is described in section 2.4, 2.5 and 2.6.

2.1 Hybrid Video Coding

Hybrid video coding is perhaps the most known and used principle of video coding. The term *hybrid video coding* is mainly used to express the combination of motion compensated DPCM with block transform coding [4]. Most of the well-known video standards today, like H.264, is based on this principle. The emerging High Efficiency Video Coding (HEVC) standard is not an exception [5]. A block diagram of the general design of a hybrid encoder is shown in Figure 2.1.

The hybrid coder works by estimating the motion that has occurred between frames. This gives a set of motion vectors that describes the motion, and a predicted frame. By subtracting the predicted frame from the original frame, one ends up with an estimation error signal. The idea is that most of the time, the bits needed to encode the estimation error and the motion vectors are less than what is needed to encode the frame itself.

2.1.1 Types and Grouping of Frames

There are three main type of frames associated with a hybrid coder. If we bypass the inter-frame prediction process altogether, we get an Intra frame (I-frame). The I-frame will thus be encoded as a regular frame, and can be decoded without having

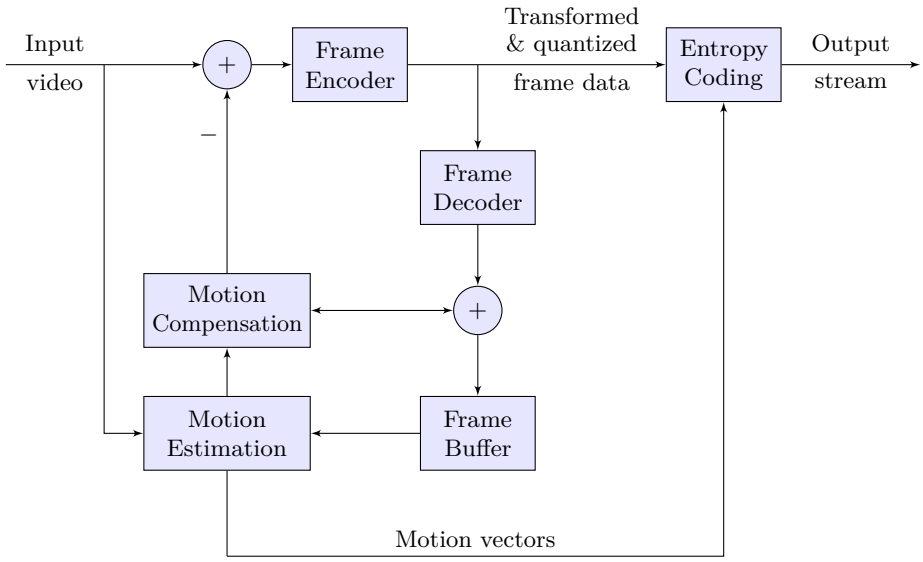


Figure 2.1: Basic hybrid video encoder (Adapted from [6])

to decode other frames first. If we allow a frame to be predicted from previous frames, we get a Predicted frame (P-frame). To decode a P-frame, the associated motion vectors and the previous frames that the P-frame are predicted from must first be decoded. By allowing predictions from both previous and future frames, we get a Bidirectional predicted frame (B-frame). Decoding a B-frame thus requires both previous and future frames to be decoded first, together with the associated motion vectors. Usually, only I- and P-frames are used as anchor frames, meaning that only these frame types are used as reference to predict other frames [7]. In H.264, this restriction was removed [8], which means that both referenced and non-referenced B-frames can be used.

The Group of Pictures (GOP) structure specifies how the different frame types are ordered, i.e. what frame type the successive frames in the video will have. A GOP usually starts with an I-frame, followed by a number of B-frames separated by P-frames. Often there is a fixed number of B-frames between anchor frames. Therefore, a GOP-structure is often characterized by two numbers, M and N, where M is the distance between anchor frames, and N is the distance between I-frames, and thus the GOP-length [7]. For example, an $M = 3$, $N = 9$ GOP structure would be $IBBPBBPBB$. In other cases, when a GOP is not required to start with an I-frame, Intra-Period and GOP-Size parameters might be used instead.

2.1.2 Spatial Transform

As mentioned, the hybrid coding structure is mainly associated with block transforms. One such transform that is widely used in image and video compression is the two-dimensional Discrete Cosine Transform (DCT). A frame is split into rectangular blocks of size $M \times N$, and the DCT is taken over each block. The 2D DCT is used in standards such as JPEG and H.261, 2 and 3 [4].

The fact that the DCT uses trigonometric functions, means that it must be implemented with floating point precision. When the available arithmetic precision is limited, rounding must be performed, which introduce errors. An alternative to sinusoidal transforms like the DCT is integer transforms. These transforms avoids rounding errors, as floating point precision is not needed. An example is the 4×4 transform used in H.264/AVC, with the transform matrix given in Equation 2.1 [4].

$$\mathbf{T^{int}}(\mathbf{4}) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}. \tag{2.1}$$

2.2 Motion Estimation and Compensation

Video is in essence a temporal sequence of frames that captures scenes of motion. Thus, frames of a scene will usually have a high degree of correlation between them. We want to remove this redundant information between frames, in order to compress well. By performing Motion Estimation (ME), a set of motion vectors can be found which estimates the motion between frames. These vectors can be used to describe a current frame as a transformation of previous or future frames. This is called motion compensation (MC), and is a common technique for reducing temporal redundancy.

There are several ways of estimating motion. In this section, estimation by matching is discussed.

2.2.1 Motion Estimation by Matching

Motion estimation by matching is done by partitioning a frame into several matching areas [4]. These matching areas can in general have arbitrary shapes. The areas are then compared to equal shapes in reference frames, in order to find the match that best describes the motion of the area. When a match is found, the displacement of the position of the block gives a motion vector $\overrightarrow{v} = [k, l]$ for the area.

When the matching areas are rectangular blocks, this method is referred to as block matching [4]. The rest of this discussion assumes that the matching areas are rectangular. The position of a block is described by the upper-left pixel.

Often, a search range r is defined, which limits the size of the vector components such that $|k| \leq r$ and $|l| \leq r$. In this case, only a limited area of the reference frame will be used to look for a matching block. This area is called the search window. A step size can also be defined [4]. For example, with a step size of one, vectors pointing to any integer pixel position within the search window is legal. In this case, the vectors have full-pixel accuracy. With a step size less than one, sub-pixel accuracy can be obtained. This creates the need for interpolation between pixels. Figure 2.2 shows an example of block motion estimation.

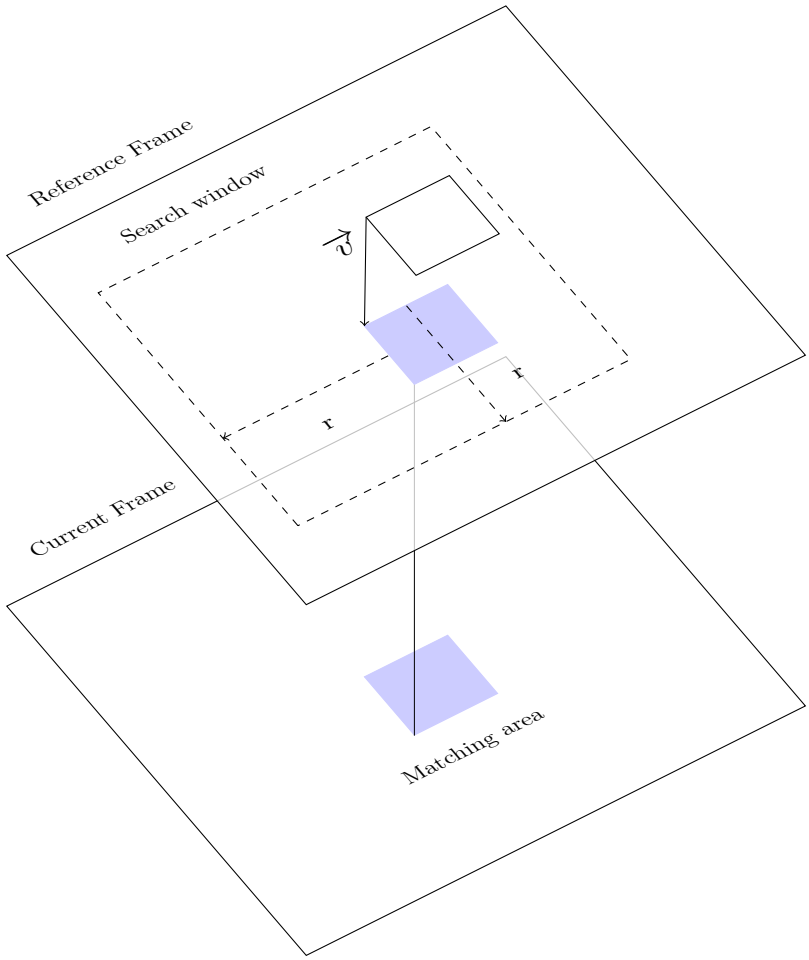


Figure 2.2: Example of Block matching motion estimation. The distance r between the upper-left corner of the matching area (shaded) and the edges of the search window (dashed) is the search range. The black bordered block is the match found.

In order to find an optimal match of a block within a reference frame, it is necessary to define a cost function that can be minimized. This function could for example

be the Sum of Absolute Differences (SAD) or the Mean Squared Error (MSE). There is generally a tradeoff between efficiency and complexity associated with the choice of cost function. In the case of SAD and MSE, MSE can be considered the most efficient choice. SAD performs an absolute value calculation of a difference, whereas MSE raises the difference by a power of two. This is a much more expensive operation than taking the absolute value, so MSE is also the most computationally complex choice.

In the case of MSE, the optimal motion vector can be defined as

$$\overrightarrow{v}_{opt} = \min_{|k|,|l| \leq r} \left\{ \frac{1}{M \cdot N} \sum_{m} \sum_{n} \left(x(m,n) - y(m+k, n+l) \right)^2 \right\} \tag{2.2}$$

where $x(m,n)$ is the current frame, $y(m,n)$ is the reference frame and M and N gives the block dimensions. It is worth noting that the optimality of a block match is in relation to the specific cost function that is used.

In order to be guaranteed to find the optimal vector, a full search needs to be performed, where all candidate positions are checked. This is however not a very efficient solution as the computational complexity is very high. There is $4r^2$ blocks to check, and each of these checks needs to calculate the value of the cost function. Fast search algorithms, like the Diamond Search [9], provides fast and efficient ways to find good matches, although they generally do not guarantee to find the globally optimal vector [4]. By using search patterns, the number of block comparisons are minimized.

2.2.2 Overlapping Block Motion Compensation

In regular block motion compensation, the current frame is partitioned into non-overlapping blocks. Motion estimation is performed on each block, which gives a set of motion vectors. The frame can then be described as a transformation of the reference blocks according to these vectors. Each pixel is thus associated with one motion vector.

Blocks can also overlap. By doing this, pixels can belong to several blocks, and thus be associated with several motion vectors. A pixel is then "transformed" by applying a weighting function to the different vectors. This is called Overlapping Block Motion Compensation (OBMC) [4].

With non-overlapping blocks, there can be discontinuities at the block edges that can cause so called blocking artifacts when a frame is decoded. By letting the blocks overlap, this problem can be avoided or reduced, as the edges of one block also is contained within other blocks.

2.3 High Efficiency Video Coding

The H.264, or "Advanced Video Coding" (AVC) standard has for many applications been the latest big improvement in video coding efficiency. The 2004 development of the Fidelity Range Extensions for AVC included the specification of a feature set called the High Profile, which has become the "flagship" technology for many digital video applications. Further extensions has been added to AVC since then, but these extensions has primarily focused on capabilities rather than coding efficiency [5].

The demands for video with higher quality and resolution is however always increasing. As an example, HDTV is well on its way to replace standard definition TV [5]. This demand implies a need for new standards with higher coding efficiency. In addition, video content is to a greater extent than before being streamed over the Internet, to a variety of platforms. Since bandwidth is a limited resource, this gives another incentive for developing more efficient standards. As a consequence, two of the major video coding standardization organizations, the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG), established a Joint Collaborative Team on Video coding (JCT-VC) in January 2010, and issued a Call for Proposals (CfP). The new project was given the name "High Efficiency Video Coding" (HEVC) [5].

The HEVC standard is intended to provide significantly better compression capability than the existing H.264/AVC standard. Some of the best-performing proposals resulting from the CfP actually showed similar quality compared to H.264/AVC at roughly half the bit rate [5].

In February 2012, HEVC reached ISO/IEC Committee Draft status. It is expected that the first version of the standard will be completed in the beginning of 2013 [10]. It will then be a part of the MPEG-H standard suite [11].

2.3.1 Picture Partitioning

One of the elements that is beneficial for higher compression performance in video with high resolution that has been introduced in HEVC, is the larger block structures with flexible mechanisms for sub-partitioning [5]. These structures are described here.

The top-level partitioning structure in HEVC is the *Largest Coding Unit* (LCU). The frames are divided into a sequence of LCUs, which is an NxN block of luma samples, together with the two corresponding blocks of chroma samples. The LCU concept is the equivalent of the macroblock concept in previous standards like H.264. The maximum allowed size of the LCU is 64×64 luma samples [12]. An example of an LCU partitioned frame is shown in Figure 2.3.

The *Coding Unit* (CU) is the basic unit of region splitting for inter and intra prediction. It is always square, and it can have a size from 8×8 luma samples to the size of the LCU. The CU block can be recursively split into four equally

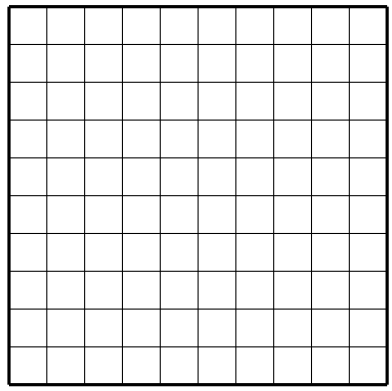


Figure 2.3: Example of a picture divided into LCUs (Reproduced from [12])

sized blocks, starting from the LCU. This process gives a content adaptive coding tree structure consisting of CU blocks. Both *skipped* and *non-skipped* CU types are allowed. The skipped CU has an inter-prediction mode without coding of motion vector differences and residual information. The non-skipped CU is assigned to either an intra or inter-prediction mode [12]. Figure 2.4 shows an example of a CU structure.

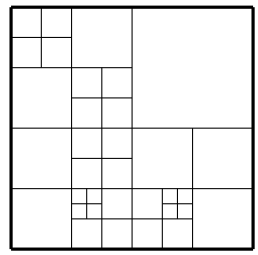


Figure 2.4: Example of Coding Unit Structure (Reproduced from [12])

The *Prediction Unit* (PU) is the basic unit for information related to the prediction processes. It is generally not restricted to being square in shape, so that a partitioning of the frame that matches the boundaries of real objects can be possible. Each CU can contain one or more PUs. $N \times N$ sized PUs are only allowed when the corresponding CU size is greater than the smallest allowed CU size [12]. Four types of PU structures is shown in Figure 2.5.

The *Transform Unit* (TU) is the basic unit for transforms and quantization. The shape of the TU depends on the PU partitioning mode. Each CU can contain one or more TUs, where multiple TUs can be arranged in a quadtree structure [12]. An example of a TU structure is shown in Figure 2.6.

A *slice* is a packetization unit of coded video data for transmission. A slice consists of a header and a series of successive CUs in coding order. The boundaries of a slice

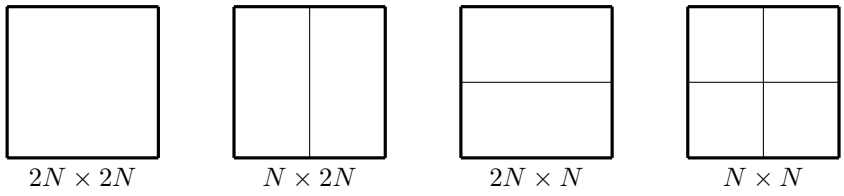


Figure 2.5: Four types of Prediction Unit structures (Reproduced from [12])

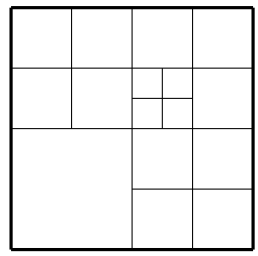


Figure 2.6: Example of a Transform Unit structure (Reproduced from [12])

can be located inside an LCU so that packetization efficiency can be maximized. Slices are designed to be independently decodable. Therefore, no prediction is performed across slice borders, and entropy coding is restarted between slices [12].

2.3.2 Transform and Quantization

HEVC supports transforms of sizes from 4×4 to 32×32. These transform are DCT-like integer transforms. The transform coefficients are derived from the transform matrix $\mathbf{c}$ and the residual samples r by a series of steps involving arithmetic and binary shift operations. The transform matrix for the 4×4 transform is shown in Eq. 2.3.

$$\mathbf{c}(\mathbf{4}) = \begin{bmatrix} 64 & 64 & 64 & 64 \\ 83 & 36 & -36 & -83 \\ 64 & -64 & -64 & 64 \\ 36 & -83 & 83 & -36 \end{bmatrix} \tag{2.3}$$

After the transform coefficients have been derived, they are quantized. The quantization is done similar to the quantization in H.264 [5], which uses scalar quantizing [8].

2.3.3 Entropy Coding

For entropy coding, HEVC uses a Context Adaptive Binary Arithmetic Coding (CABAC) coder [12]. Earlier versions of the HEVC Test Model also had a Con-

text Adaptive Variable Length Coding (CAVLC) coder for the Low Complexity configuration, but this has been removed in later versions.

2.3.4 Encoder Configurations

The HEVC test model currently has two defined configurations: “High Efficiency” (HE) and “ Low Complexity” (LC) [12]. The difference between these configurations is mainly that some tools are disallowed in the LC configuration. LC is thus a subset of HE in terms of coding tools. Some differences between the two configurations are listed below.

- For a CU of size $2N \times 2N$, both HE and LC allows the PUs shown in Figure 2.5. In addition, HE allows PUs of size $2N \times \left(\frac{N}{2} + \frac{3N}{2}\right)$ and $\left(\frac{N}{2} + \frac{3N}{2}\right) \times 2N$, provided that $N > 4$.
- In LC, the transform block sizes are always square, while HE allows non-square blocks.
- For in-loop filtering, both configurations has a deblocking filter. In HE, two additional filter stages called Sample-Adaptive Offset (SAO) filter and Adaptive Loop Filter (ALF) are also supported.

2.3.5 Temporal Prediction Structures

The encoder has three different temporal prediction structures: Intra-only, low-delay and random-access [12].

The intra-only structure is shown in Figure 2.7. With this configuration, each frame is encoded as an Instant Decoder Refresh (IDR) frame. No temporal reference frames are used.

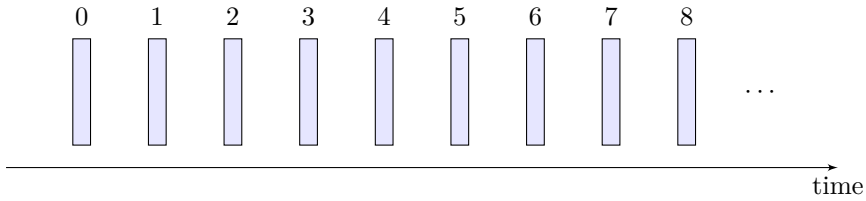


Figure 2.7: Intra-only temporal prediction structure. All frames are coded as IDR frames. (Reproduced from [12])

Figure 2.8 shows the low-delay structure. In this configuration, only the first frame of a sequence is encoded as an IDR frame. The rest of the frames are encoded as Generalized P/B (GPB) frames. A GPB frame is a frame where the two reference picture lists are identical [13]. In the low-delay structure, These lists can only contain frames with a Picture Order Count (POC) smaller than the current frame.

In other words, the GPB frames acts as P-frames, and can only reference temporally previous frames.

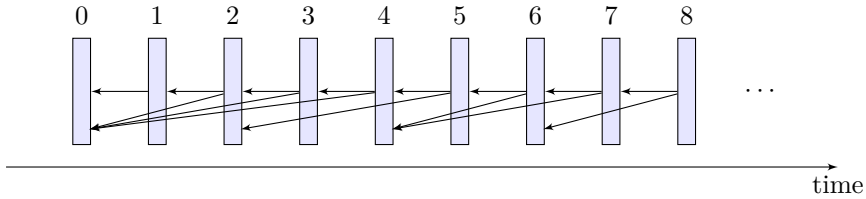


Figure 2.8: Low-delay temporal prediction structure. A frame can only reference previous frames. (Reproduced from [12])

The random-access structure is shown in Figure 2.9. In this configuration, a hierarchical B structure is used for coding. I-frames are inserted cyclically, with a period of about one second. The first I-frame of the sequence is encoded as an IDR frame, and the rest are encoded as non-IDR I-frames. The frames between I-frames (in display order) are encoded as B-frames.

The lowest temporal layer consists of I- and GPB-frames. The GPB frames can refer to I- or other GPB-frames. The second and third temporal layers consists of referenced B-frames, and the highest temporal layer consists of non-referenced B-frames.

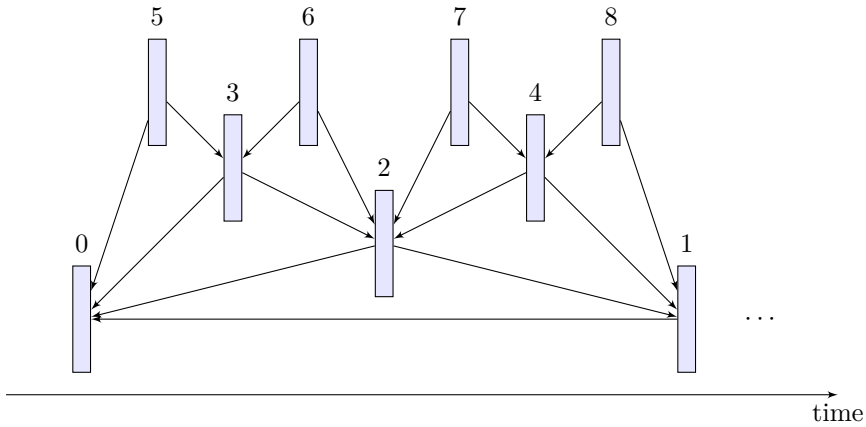


Figure 2.9: Random-access temporal prediction structure. Frames can reference both previous and future frames. Numbers refer to encoding order. (Reproduced from [12])

2.4 Wavelet Based Video Coding

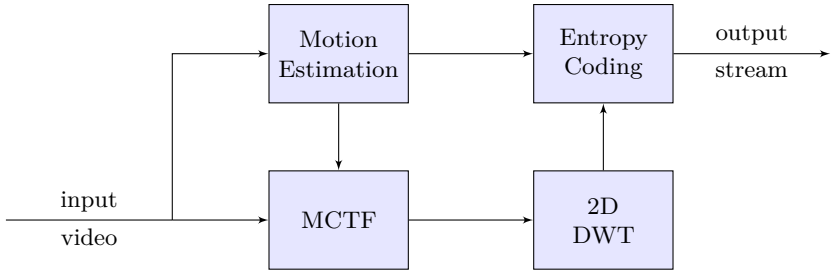


Figure 2.10: Basic T+2D encoder with MCTF

In a wavelet based video coding scheme, the video frames get decomposed into multiple subbands by temporal and spatial wavelet transforms. After this decomposition, quantization and entropy coding are applied to the subbands. These coding schemes can be placed in two categories: In *T+2D*, the temporal filtering is applied first, and then spatial filtering is applied on the resulting temporal subbands. In *2D+T* (also referred to as *in-band*), the spatial filtering is performed first, and then temporal filtering is performed on the resulting spatial subbands. Thus, if MC is to be performed, it must be done in the wavelet domain before filtering temporally. Either way, the temporal and spatial filtering can be described independently [14]. In Figure 2.10, a block diagram of a general T+2D encoder with MCTF is shown. MCTF is further explained in the following section.

A wavelet video coder that is often referred to in literature is the Motion Compensated Embedded Zeroblock Coder (MC-EZBC). This coder uses the T+2D structure with MCTF followed by EZBC image coding [14]. In [15], an improved version of this coder called the Enhanced MC-EZBC is described. Among the presented improvements is an adaptive MCTF framework.

2.5 Motion Compensated Temporal Filtering

In wavelet video coding, temporal redundancy can be removed by applying motion compensation before the temporal decomposition. This combination of MC and temporal filtering is called *Motion Compensated Temporal Filtering* (MCTF). In wavelet-based MCTF, a wavelet transform is performed in the temporal direction along previously determined motion trajectories [14]. The wavelet transform in the temporal direction decomposes the original frame sequence into several frequency subbands, which represents the “velocity of temporal change” [16]. The motion compensation helps concentrating the energy into the lowpass subbands.

2.5.1 Temporal Decomposition

A temporal decomposition of three levels with a Haar filter is shown in Figure 2.11. First, MC is performed on the 8 input frames in consecutive pairs (A and B denotes even and odd numbered frames). The first temporal decomposition level is obtained by filtering the pairs of frames with the Haar filter along the motion trajectories determined by the MC. This filtering results in 4 temporal lowpass and 4 temporal highpass frames. The resulting lowpass frames are in the same fashion pairwise motion compensated and used for further decomposition. This process is repeated until the lowest temporal level, with one lowpass and one highpass frame (LLL and LLH in the figure), is reached [14]. Note that the original frames are treated as "Level 0" lowpass frames.

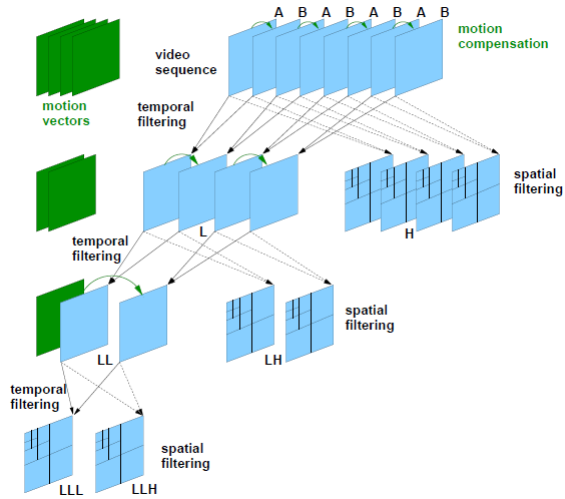
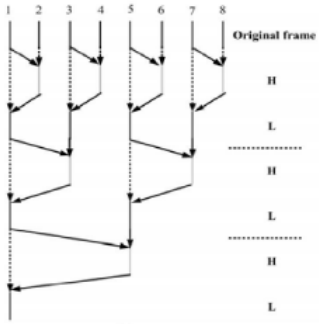


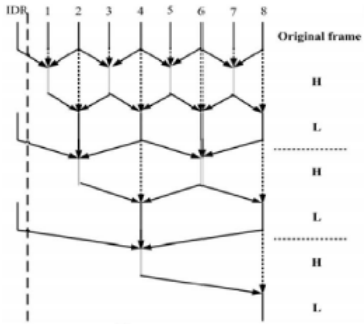
Figure 2.11: T+2D Temporal decomposition using Haar filter (From [14])

As can be seen from figure 2.11, the temporal filtering in this case results in 8 temporal subband frames. One of the subband frames is a lowpass subband, while the rest contains higher frequencies. Lowpass (L) frames can be regarded as averaged frames, and highpass (H) frames can be regarded as frames containing error residuals. L frames are generally easier to compress than H frames [17], so we want to concentrate the energy in the L frames. This is done by applying MC in the decomposition. Blurring in the L frames caused by fast motion is avoided, and the amount of energy or high frequency components in the H frames is minimized [17].

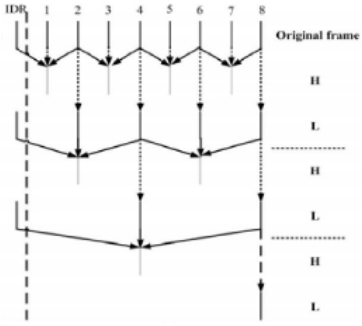
Different types of wavelet filters can be used for the temporal decomposition. In the example of Figure 2.11, a Haar filter is used. The Haar filter is used for uni-directional motion prediction. Other filter types, like the LGT 5/3 filter or the 1/3 filter could be used for bi-directional prediction [18]. The decomposition structures for the Haar, 5/3 and 1/3 filters is shown in Figure 2.12.



(a) Haar filter



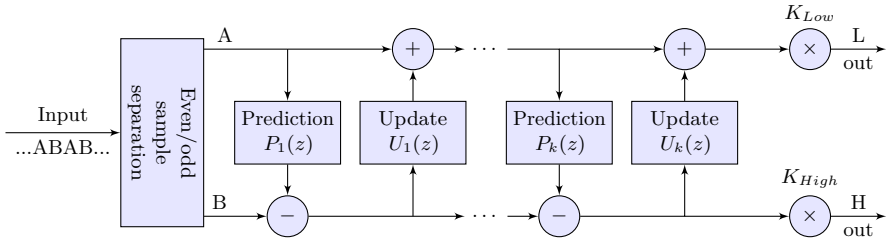
(b) 5/3 filter



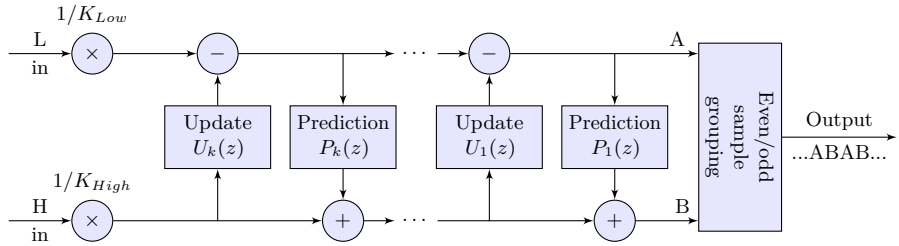
(c) 1/3 filter

Figure 2.12: Temporal decomposition structures (From [18])

2.5.2 Lifting Structure



(a) Analysis filter



(b) Synthesis filter

Figure 2.13: Lifting structure of a biorthogonal filter. (a) shows the analysis filter and (b) shows the synthesis filter. (Reproduced from [4])

Any pair of biorthogonal filters can be implemented in a lifting structure as shown in Figure 2.13. The first step of the lifting filter is a decomposition of the signal into its even- and odd-indexed polyphase components. Then, the two basic operations are *prediction steps* P and *update steps* U. Finally, normalization factors K_{Low} and K_{High} is applied to achieve orthonormal decomposition [4]. The output of the lifting filter are two subbands of the original signal. Thus we can use a lifting filter to split a signal into low- and highpass bands. Another nice property is that the lifting structure makes perfect reconstruction possible when the synthesis filter uses the same prediction and update steps as the analysis filter [4].

By using pairs of biorthogonal wavelet filters, we can split a video signal into temporal subbands. If we regard even- and odd-indexed frames as even and odd polyphase components of the video signal, we see that we can transform these frames along the temporal axis into lowpass frames L and highpass frames H.

As explained in [4, sec. 13.4.2], MC can be integrated into the lifting steps [14]. Figure 2.14 shows the block diagram of a lifting based MCTF module. The filter admits two inputs subject to motion trajectories and generates one high and one low frequency subband output h_t and l_t [18].

The lifting steps in Figure 2.14 can be described as

$$h_t = x_{2t+1} - x_{2t} \tag{2.4}$$

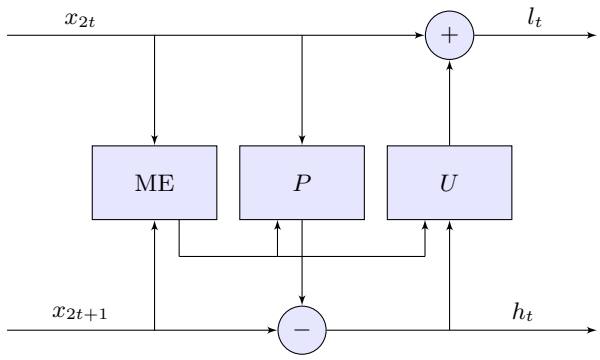


Figure 2.14: Lifting based invertible MCTF (Reproduced from [18])

$$l_t = x_{2t} + \frac{h_t}{2} \tag{2.5}$$

for the Haar filter, and

$$h_t = x_{2t+1} - \frac{x_{2t} + x_{2t+2}}{2} \tag{2.6}$$

$$l_t = x_{2t} + \frac{h_{t-1} + h_t}{4} + \frac{1}{2} \tag{2.7}$$

for the 5/3 filter. For the 1/3 filter, which is the same as the 5/3 filter, but with the update step omitted, the equations becomes

$$h_t = x_{2t+1} - \frac{x_{2t} + x_{2t+2}}{2} \tag{2.8}$$

$$l_t = x_{2t} \tag{2.9}$$

[18]. As can be seen from Equation 2.9 and Figure 2.12 (c), the lowpass subbands of a temporal layer is simply copied down from the layer above in the 1/3 filter. In the Haar and 5/3 filter, highpass subbands are also used to construct the lowpass bands, and thus inverse motion compensation must be used.

2.5.3 In-band MCTF

In a T+2D scheme, the temporal subband frames are spatially filtered with a 2D wavelet transform after the temporal decomposition. This filtering decomposes the subband frames into spatial subbands. In a 2D+T scheme however, the video frames are spatially decomposed *before* the temporal decomposition. After the spatial decomposition, the prediction and update steps (assuming a lifting implementation of the temporal filtering) are performed on each of the spatial subbands.

If ME is performed on each spatial subband, they all have their own set of motion vectors. It is fairly clear that there would be some correlation between these motion vector sets. In Figure 2.17 (b) for example, there is clearly some "structural"

correlations between the subbands that would translate to the motion vectors. A viable alternative would be to only perform ME on the lowpass subband, and use the resulting set of motion vectors for all subbands. However, if the motion vectors from the lowpass subband doesn't capture the underlying motion in the highpass subbands, mismatch artefacts will appear in the decoded video [19]. In [19], a solution to this is presented by a macroblock-level adaptive scheme that only transmits nescessary highpass subband motion vectors.

2.6 JPEG 2000

JPEG 2000 is an image compression standard developed by ISO and IEC in a collaboration called Joint Photographic Expert Group (JPEG). It is meant to be the successor of the highly successful original JPEG standard in many application areas [20].

JPEG 2000 has a lot of interesting features. In addition to basic compression functionality, some of the features are:

Both lossless and lossy compression: The standard has both an integer and real mode. In integer mode, reversible component and wavelet transforms are used, and quantizer step sizes are set to one. This makes lossless compression possible. In real mode, the irreversible wavelet transform is used.

Scalability: The standard supports both spatial and quality scalability. Several quality layers can be generated from the original image, providing a base layer for the basic quality, and enhancement layers to generate higher quality. Similarly, spatial scalability is achieved by generating several spatial layers. Depending on how the code stream is organized, an image can be transmitted progressively, so that the decoder can start to reconstruct the image before all of the information is available, either by pixel accuracy, spatial resolution or a combination of both.

Region of Interest (ROI) coding: In some applications there are parts of an image that are particularly important. ROI coding makes it possible to code and transmit part of an image with higher quality than the rest of the image.

Error resilience: The transmitted data are organized in independent code blocks. Entropy coding of coefficients is performed within these blocks. Errors in the bit stream of a code block will therefore be restricted within the block.

JPEG 2000 consists of several parts, where Part 1 is the core coding system. Among the other parts are Part 2 - Extensions, which adds functionality to the standard, and Part 3 - Motion JPEG 2000. This section discusses Part 1 of the standard.

2.6.1 General Overview

The general codec structure of JPEG 2000 is illustrated in Figure 2.15. At the encoder, the image is first preprocessed. Then, the forward intracomponent transform is applied. The transform coefficients are then quantized and entropy coded before forming the output code stream [21]. The decoder is essentially a mirror of the encoder, where each functional block either exactly or approximately inverts the effect of its corresponding block in the encoder [22].

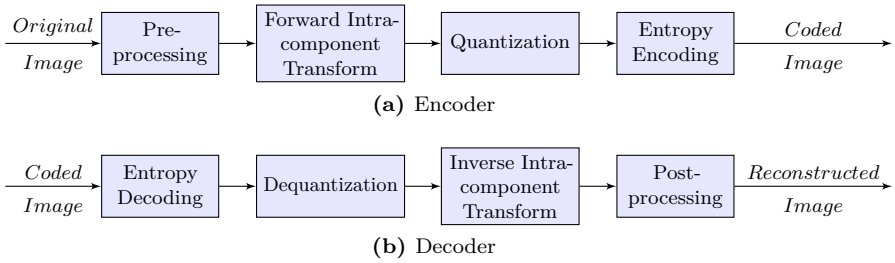


Figure 2.15: Structure of the JPEG 2000 (a) encoder and (b) decoder. (Adapted and reproduced from [22])

2.6.2 Preprocessing

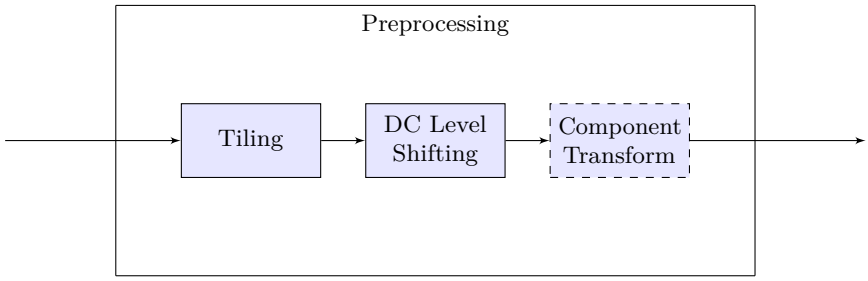


Figure 2.16: Preprocessing stage of JPEG 2000. The image is first partitioned into tiles. Then, the samples of the tiles are DC level shifted. For multicomponent images, a component transform can also be applied.

Before any transformations of the image take place, the image is partitioned into rectangular, nonoverlapping tiles. These tiles are later compressed individually, as if they were independent images. The tile dimensions can be of arbitrary size, up to and including the entire image. It is thus possible to regard the entire image as one tile. This is useful as partitions with larger tiles performs visually better than with smaller tiles [21].

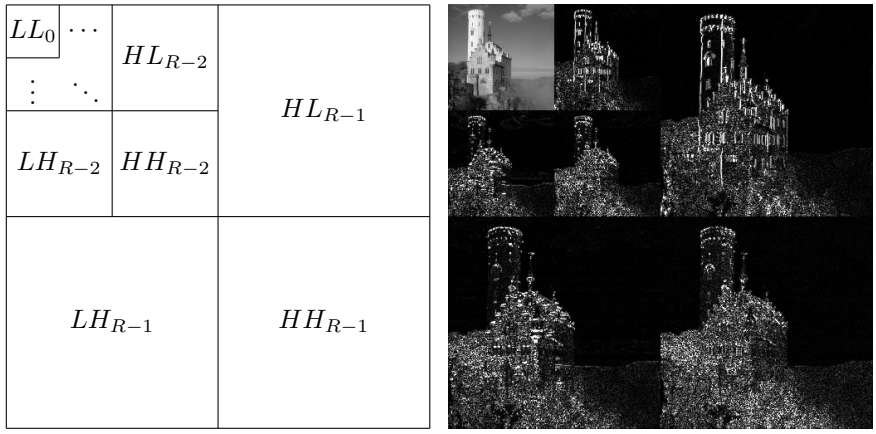
After partitioning the image into tiles, all samples of a tile are DC level shifted, provided that the samples are unsigned (non-negative). This is done to ensure that

the sample data has a nominal dynamic range approximately centered about zero. If a component has a precision of $P^{bits}/_{sample}$, shifting is done by subtracting 2^{P-1} from all samples [22].

To reduce the correlation between components, an inter-component transformation can be applied to the tile-component data. Two inter-component transforms are defined; the Irreversible Color Transform (ICT) and the Reversible Color Transform (RCT). Both of these essentially map image data from the RGB to YCbCr color space [22]. The ICT may be used for lossy coding, and the RCT may be used for lossless or lossy coding [21]. For the inter-component transform to be applied, the image must have at least three components [22].

The preprocessing stage in the encoder is depicted in Figure 2.16.

2.6.3 Discrete Wavelet Transform



(a) Subband structure of a $(R-1)$-level transform (Reproduced from [22])

(b) Lichtenstein (From `wikipedia.org`)

Figure 2.17: Subbands of a picture after DWT

After preprocessing the input image, the forward intracomponent transform is applied. This transform is a Discrete Wavelet Transform (DWT). The DWT splits the tiles into several decomposition levels. Each decomposition level contains four subbands that describe horizontal and vertical spatial frequency characteristics [21]. The structure of an $(R-1)$-level transform is depicted in Figure 2.17 (a). An $(R-1)$-level decomposition has R resolution levels, where level 0 and $R-1$ corresponds to the lowest and highest resolutions, respectively [22]. An example of the subbands of a picture after the DWT is shown in Figure 2.17 (b).

To perform the DWT, the standard uses a one-dimensional subband decomposition. The image is decomposed into low- and highpass samples. Lowpass samples

represents a down-sampled version, while highpass samples represents a down-sampled residual version needed for perfect reconstruction [21]. As an image is two-dimensional, the filtering is performed in both the horizontal and vertical directions. This process is applied recursively to the lowpass subbands at each level in the decomposition, until the desired number of levels are obtained.

The DWT can be reversible or irreversible. The reversible transform is the Le Gall 5/3 filter, while the irreversible filter is the Daubechies 9/7 filter [21]. The irreversible filter gives the highest compression efficiency, but cannot be used for lossless compression. For lossless compression, the reversible filter must be used, as it allows for perfect reconstruction.

2.6.4 Quantization and Entropy Coding

After the DWT has been applied, all coefficients are quantized. Quantization is a lossy operation, except in the trivial case were the quantization step size is 1 and the coefficients are integers, as is the case when the reversible DWT is used. The quantization is done with a uniform scalar quantizer with a dead-zone around the origin. A different quantizer is used for each subband, and each quantizer has its step size as its only parameter [22].

After quantizing, the quantizer indices for each subband are partitioned into code blocks. These code blocks are then individually coded with a bit-plane coder as described in [22]. Entropy coding is done with an arithmetic coding system [21].

Chapter 3

Method

In this chapter, the methodology that was used for this thesis is presented. In section 3.1, a use case for the codecs and assessments is given. Section 3.2 describes the test material that was used. Section 3.3 and 3.4 describes the baseline and proposed codecs, respectively. Finally, the objective and subjective assessments that were conducted is described in section 3.5 and 3.6.

3.1 Use Case

Streaming video content over the Internet is becoming increasingly popular. Service providers such as YouTube allow users to stream and watch video content at any time. At the same time, TV's, computer monitors and other devices capable of displaying High Definition content are becoming more affordable and available for general consumers. The increased availability of such devices has created a demand for higher quality.

With equipment capable of displaying high quality video, the constraint on the obtainable quality for streamed video becomes the capacity of the users broadband connection. 720p (1280×720) is a widely used HD resolution. With a frame-rate of 25 fps, and assuming a depth of 8 bit per color component, uncompressed video would need a bandwidth of approximately 553 Mbps. So, compression is obviously needed, as an average Norwegian broadband connection has a capacity of about 11 Mbps [23].

The use case is a consumer streaming 720p25 video over the Internet with a typical broadband connection. The user watches the video on a flat screen display.

3.2 Test Material

This section describes the test material that was used for the objective and subjective tests. When choosing test material, it was attempted to find sequences that was somewhat typical for the use case. The SVT High Definition Multi Format Test Set [24] seemed to be a good fit. In addition, this test set is well documented, is specifically made for testing purposes and provides sequences with different coding difficulties. Using this test set therefore provides a variety of testing conditions and "stress" to test video coders with. The test set is further described in the following subsection.

3.2.1 The SVT High Definition Multi Format Test Set

The SVT High Definition Multi Format Test Set is a set of demanding video clips taken from the TV-program "Fairytale" by Sveriges Television (SVT). All the sequences were filmed in 50 fps with professional 65mm film equipment by SVT in October 2004. The sequences were digitized and mastered in $3840 \times 2160p50$ [24].

The test set consists of five sequences. Four of these sequences were used for the assessments. These sequences are listed in Table 3.1. They were downloaded in 720p50 format with 16 bits per colour plane. They were then converted from 16 bit RGB format to planar 8 bit YUV 4:2:0 format with the command-line tool "sgi2yuv" [25]. The conversion from 50 to 25 fps was done by dropping every other frame.

Name	Coding Difficulty	No. Frames
CrowdRun	Difficult	250
DucsTakeOff	Difficult	250
IntoTree	Easy	250
OldTownCross	Easy	250

Table 3.1: List of the test sequences that were used. All sequences have a resolution of 720p25, and a duration of 10 seconds.

CrowdRun is classified as a difficult sequence, and shows a group of runners moving towards the camera. There is some degree of camera movement, and the runners cause a lot of in-picture motion. The amount of objects causes a high amount of edges.

In the DucksTakeOff sequence, there is little or no camera movement. There are rapid movements when the ducks fly away, and the ripples in the water represents slower movements. This sequence is also classified as difficult.

The IntoTree sequence has a low degree of object movement, but there is a lot of camera movement. The camera moves towards the mansion and zooms in on the

Figure 3.1: First frame of CrowdRun sequence.

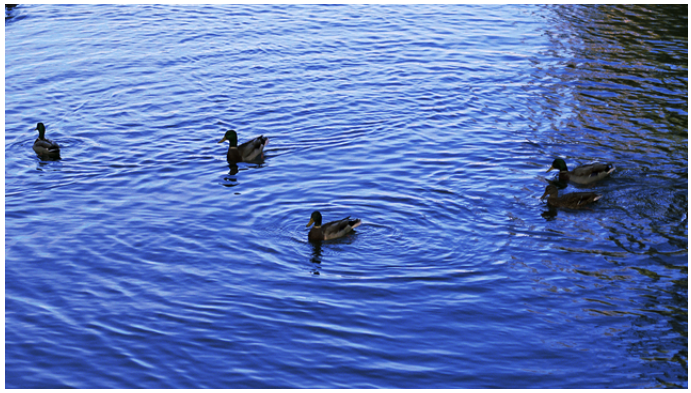


Figure 3.2: First frame of DucksTakeOff sequence.

Figure 3.3: First frame of IntoTree sequence.

tree on its right. There appears to be a slight amount of grain noise in the video, mostly noticeable in the sky. It is classified as having easy coding difficulty.

In the OldTownCross sequence, there is little object movement except for a few cars. The camera moves across the scenery. In addition to a slight amount of grain noise, there appears to be some instances of salt-and-pepper noise present. The coding difficulty for this sequence is described as easy.

Figure 3.4: First frame of OldTownCross sequence.

3.3 Baseline Codec - HEVC HM-5.2

Version 5.2 of the HEVC Test Model (HM-5.2) [26] was used as the baseline codec for comparison with the proposed codec. The encoder was set to use the Low Complexity (LC) configuration and the Random Access (RA) temporal prediction structure, as this seemed to be the best fit for the use case. RA is the appropriate choice for bandwidth-sensitive use cases. The LC configuration was chosen since a typical platform for the user would be streaming video in a web browser, perhaps on a battery-powered laptop. In these cases, LC seems appropriate.

The GOP-size was 8, with an Intra-period of 32. The search range for motion estimation was 64. The Configuration file [27] that was used for HM-5.2 is given in Appendix A.

3.4 Proposed Wavelet Codec - MCTF-JP2

With the use case presented in Section 3.1 in mind, a proposed wavelet codec was developed. A block diagram of the codec, hereby referred to as MCTF-JP2, is presented in Figure 3.5. It uses a T+2D structure, with JPEG 2000 as the spatial codec. For MCTF, a bi-directional 1/3 filter was implemented. Motion vectors were

compressed losslessly by an arithmetic coder. The implementation was written in Matlab, version 7.11.0.584 (R2010b).

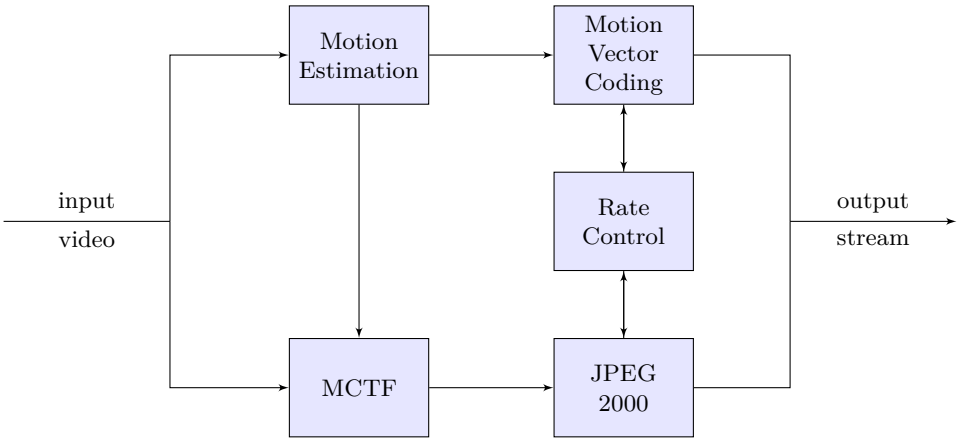


Figure 3.5: Block diagram of the proposed MCTF-JP2 encoder.

The following subsections describe the implementation of MCTF-JP2. The Matlab code for the codec can be found in the accompanying zip-file, the content of which is listed in Appendix H.

3.4.1 Motion Estimation

Motion Estimation was done by block matching with minimum Mean Squared Error (MSE) as the matching criteria. Although less complex alternatives than MSE exists, the focus was more on achieving quality when implementing the codec. The block matching was done with full-pixel accuracy. For luma components, the block size was 40×40, and the maximum search range was set to 64. For chroma components, the block dimensions and maximum search range were set to be half of the corresponding luma values. This was done since the chroma components of the test material was subsampled with the YUV 4:2:0 scheme.

The maximum search range corresponds to the range used to produce the lowest temporal layer. The search range for producing the second lowest layer is half of the maximum range, and so on. The search range used to produce motion vectors for layer i can thus be described as $r_{max} \cdot 2^{i-L}, 1 \leq i \leq L$, where r_{max} is the maximum search range and L is the total number of temporal layers. Here, the temporal layers are indexed from highest to lowest layer. Adapting the search range to the layers was done since the distance between the current and reference frames gets lower further up in the temporal hierarchy. This made it possible to save bytes spent on coding motion vectors without any significant loss of quality.

Diamond Search (DS) was the block matching algorithm that was implemented. The DS implementation was based on the algorithm described in [9], and can be found in Appendix B.1. Full Search (FS) was also implemented, but as this algorithm turned out to have an extreme running time, this was not used. As DS is not guaranteed to find the optimal vector, it is interesting to see how this algorithm affects the performance of MCTF-JP2 compared to FS. In Figure 3.6, the Y-PSNR values for the first 25 frames of the CrowdRun sequence, coded at $10Mbps$, is shown. It is clear that the loss of using the suboptimal DS algorithm is very small. The average difference between FS and DS is in this case only $0.054dB$, with a maximum difference of $0.45dB$. At the two peaks, DS actually makes the codec perform better than with FS. This is probably due to lucky factors when compressing the vector fields. As a description of the extreme running time of FS, it can be noted that the encoding time with DS was about three minutes and 20 seconds for the three GOPs in Figure 3.6. Encoding with FS on the other hand, took over three hours.

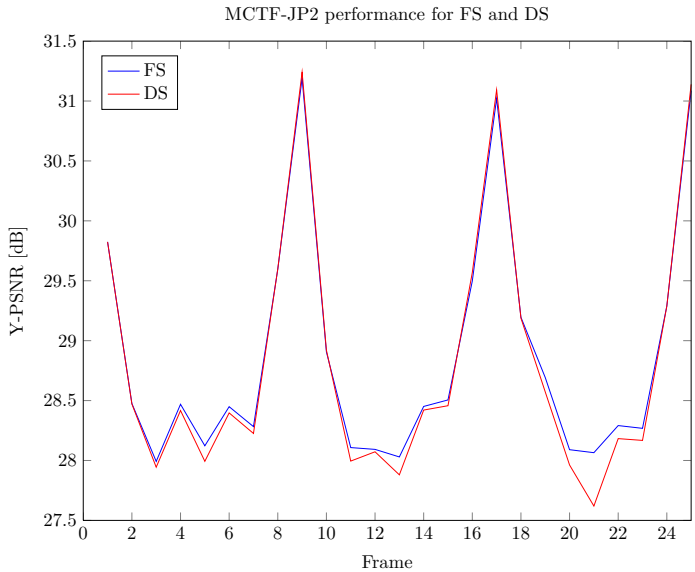


Figure 3.6: Performance comparison of MCTF-JP2 with FS and DS for block matching. The plot shows the PSNR values for the first 25 frames of the CrowdRun sequence. The settings for MCTF-JP2 are as described throughout this section.

A simple form of Overlapped Block Motion Compensation (OBMC) was utilized for luma components. The block overlap was half of the block size. Each pixel belongs to four blocks (except at frame edges). The pixel prediction is the average of all "block predictions". For chroma components, regular block compensation was used as the improvement for these components did not match the overhead in bit-rate, which caused an overall loss of quality.

3.4.2 Motion Vector Compression

For the compression of motion vectors, a combination of run-length coding and arithmetic coding was used. At first, a Canonical Huffman coder was implemented, but an open-source arithmetic coder [28] proved to be much more effective. Adding run-length coding [29] before arithmetic coding further increased the compression efficiency. The run-length coding of the vector fields was done in a raster-scan order.

3.4.3 Motion Compensated Temporal Filtering

The MCTF step was implemented as a bi-directional 1/3 lifting filter as described in 2.5.2. Three temporal layers were used, which gives a GOP size of 8 frames. A bi-directional filter was chosen for the same reasons as the choice of the RA structure for HM-5.2, namely that the use case is more sensitive to bandwidth than delay. For a discussion of why the 1/3 filter was chosen over the 5/3 filter, see Section 4.3.4.

The highpass subbands produced by an MCTF filter has an increased precision compared to the input frame. For an 8-bit frame with a dynamic range of $[0, 255]$, the resulting highpass subband has a dynamic range of $[-255, 255]$. This precision increase is undesirable. With efficient motion compensation, a very small amount of the coefficients will be in the extremities of the range. A simple way of reducing the range to 8 bit is to cut off the edges of the range and shift the values to get a range of $[0, 255]$. A slightly more sophisticated and efficient method, as described in [6], is scaling of the coefficients. This approach was implemented in MCTF-JP2. In this method, all values of a highpass subband are scaled by a factor x, and then 127 is added. The resulting values are then rounded to the nearest integer in the $[0, 255]$ range. The scaling factor x is computed as

$$x = \begin{cases} y & \text{if } y < 1 \\ 1 & \text{if } y > 1 \end{cases} \tag{3.1}$$

where y is defined as

$$y = \frac{\left\lceil \dfrac{127}{\max(\text{abs}(HP(x,y)))} \cdot 10 \right\rceil}{10}. \tag{3.2}$$

This gives a scaling factor x between 0.5 and 1, with 0.1 incrementations [6]. x is sent together with the subband. As it can take six distinct values, it can be coded with three bits.

The implementation of the 1/3 lifting filter can be found in Appendix B.2, and examples of subband frames generated by the filter can be seen in Appendix C.

3.4.4 JPEG 2000 Codec

JasPer version 1.900.1 [30] was used for encoding subbands as JPEG 2000 images. JasPer is the JPEG 2000 reference software implementation.

Redundant and unnecessary information from the main and tile header of the JPEG 2000 files was removed. For example, JasPer puts a comment in the main header, which is not needed, and information such as image size is unnecessary as this information was stored separately in a video stream header. In total, for each produced code stream, 112 header bytes were removed.

All subbands were coded with the irreversible wavelet transform. Only one quality layer was used, and the subbands were coded as one tile. The default code block size of 64×64 was used. For lowpass subbands, 7 spatial levels for luma components and 6 for chroma components were used. For highpass subbands, 5 and 4 spatial levels were used for luma and chroma, respectively. The number of levels to use were chosen by experimenting with different values.

3.4.5 Rate Control

The rate control mecanism for MCTF-JP2 was implemented as follows. The byte budget for each Group of Pictures (GOP) was calculated from the target bit rate given as input to the encoder. The amount of bytes used for coding the motion vectors was subtracted from the budget, leaving the amount of bytes available to code the subband frames of a GOP. If a GOP did not use its entire budget, the left-over bytes were given to the next GOP.

The remaining byte budget of the GOP was divided between lowpass (LP) and highpass (HP) frames by a ratio given to the encoder as input. The LP and HP budgets was then divided on the different frame components, by ratios also given as input. The values of the ratios used for the different test sequences are given in Table 3.2. These values were chosen by means of experimentation. Note that as there are only one LP subband in a GOP, the entire LP budget is used for this subband. The HP budget is on the other hand divided equally on all the HP subbands in the GOP.

Sequence	LP (%)	HP (%)	Luma (%)	Cb (%)	Cr (%)
CrowdRun	25	75	90	5	5
DucksTakeOff	25	75	85	12	3
IntoTree	25	75	86.5	9	4.5
OldTownCross	25	75	90	5	5

Table 3.2: Percentage of available bits given to LP/HP frames and color components in a GOP. The Luma, Cb and Cr values goes for both LP and HP frames.

The compression ratio C for each subband, given as input to JasPer, is calculated as

$$C = \frac{B + 112 - 0.375}{M \cdot N}, \tag{3.3}$$

where B is the subband byte budget. 112 bytes are added to the budget to account for the removed header bytes, and 0.375 bytes are subtracted because of the three bits needed to code the scaling factor. The product of the subband dimensions M and N gives the uncompressed subband size in bytes.

3.5 Objective Assessment

When designing video codecs, numerical measures of the performance is valuable. They give a cost-effective way to monitor the effect of changes in algorithms and methods of compression. They also give an indication of how the coded video will be percieved by humans. And, as opposed to subjective evaluation methods, they give (assuming a deterministic coding system) deterministic results which are easy to reproduce. Also, an objective test can be automatized, which certainly is an advantage.

A comparison of MCTF-JP2 and HM-5.2 by conducting an objective assessment was performed. Two different metrics was computed from video coded by the codecs with bit-rates in the range of 7 to 12 Mbps.

3.5.1 Objective Quality Metrics

To evaluate the objective performance of the baseline and proposed codec, the objective quality metrics Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM) were used. PSNR gives a measure of the signal-to-noise ratio, while SSIM gives a measure of the structural similarity of the distorted picture and the reference. Unlike PSNR, SSIM is designed to take into consideration how errors are perceived by the visual system of humans [31]. Using both metrics gives better and complimentary insight into how the systems are performing.

PSNR is defined as

$$PSNR = 10 \cdot \log_{10}\left(\frac{I_{max}^2}{MSE}\right) \tag{3.4}$$

where I_{max} is the maximum intensity level of the image (255 for 8 bit images), and MSE is the mean squared error between the original and reconstructed image.

SSIM is defined as

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{3.5}$$

where μ_x and μ_y are the averages of image x and y, σ_{xy} is the covariance between x and y, and σ_x^2 and σ_y^2 are the variances of x and y. C_1 and C_2 are constants to

avoid instability when $\mu_x^2 + \mu_y^2$ or $\sigma_x^2 + \sigma_y^2$ are close to zero. They are defined as $C_1 = (0.01 \cdot I_{max})^2$ and $C_2 = (0.03 \cdot I_{max})^2$ [31, 32]. The value of the SSIM index is between 0 and 1, where greater values indicate greater structural similarity between the distorted and reference picture. If the two pictures are identical, a value of 1 is obtained.

The PSNR and SSIM values were calculated by using the MeTriX MuX [33] Matlab software.

3.5.2 Methodology

The objective assessment was conducted using the following steps:

1. The sequences were encoded and decoded with HM-5.2. The quantization parameters were chosen to get close to the target bit rates as shown in Table 3.3.
2. The sequences were then encoded and decoded with MCTF-JP2. The encoded sequences had target bit rates equal to those obtained after encoding with HM-5.2.
3. Finally, average PSNR and SSIM values were calculated for each reconstructed sequence. In addition, the differences between the results obtained by HM-5.2 and MCTF-JP2 were also computed.

	Sequence			
	CrowdRun	DucksTakeOff	IntoTree	OldTownCross
Target bit rate (Mbps)	QP value			
7	28.8	30.3	21.7	18.75
8	28.0	29.5	21.0	18.25
9	27.5	29.0	20.5	17.8
10	26.5	28.5	20.0	17.5
11	26.0	27.8	19.5	17.0
12	25.3	27.25	19.0	16.75

Table 3.3: Quantisation Parameter values used for the baseline codec. The parameters were chosen to get close to the respective target bit rates.

3.6 Subjective Assessment

Video content is created for humans to watch. Without viewers, the need for video and video codecs is not there. Therefore, the evaluation of how a video codec affect the quality of the content needs to involve subjective assessments at some point.

The process of subjective assessment involves getting a variety of people to give their opinion of the questions at test. The group of participants should have a good distribution of age, gender and background. They should be non-experts in the field of video coding, and a high number of participants is desirable. As a subjective assessment involves gathering opinions of humans, many factors can have an influence of the obtained results. For this reason, guidelines and standardized procedures have been developed, such as the recommendations in [34].

In order to establish how MCTF-JP2 visually performs relative to HM-5.2, a subjective assessment was conducted. The participants were asked to evaluate the annoyance of the distortion produced by the two codecs. The hypothesis for this assessment is given in Hypothesis 1. Before conducting the assessment, a research protocol that explained the background, purpose and methodology of the test was delivered to and approved by the supervisor. This research protocol can be found in App. D.

Hypothesis 1 *The distortion produced by MCTF-JP2 is less annoying than the distortion produced by HM-5.2.*

3.6.1 Environment

The assessment was conducted at the Café Media lab at NTNU. The environment was set up to provide viewing condition as close to those recommended in [34, 35] as possible.

The test environment was set up at a limited area of the lab, surrounded by dark blue curtains. The curtains allowed distractions of the surroundings to be minimized, as well as enabling control of the light intensity. The wall behind the monitor had a neutral grey color, and was illuminated with a table lamp. Illumination from other sources than the monitor and lamp was kept to a minimum.

The observers were placed $1.8m$ from the screen. The viewing distance was calculated as three times the picture height. All observers were seated within an angle of $\pm 30^\circ$ horizontally from the center of the display. The seating arrangement are depicted in Figure 3.7. Pitures of the test environment can be seen in Appendix E.

3.6.2 Methodology

The Stimulus Comparison Adjectival Categorical Judgement (SCACJ) method defined in [34] was the basis for this assessment. The observers were shown pairs of impaired sequences, which were considered to have approximately equal objective quality. The pairs consisted of one sequence A coded by HM-5.2, and one sequence B coded by MCTF-JP2. Both combinations AB and BA were used. Both sequences lasted for 10 seconds. Three seconds of mid-grey were displayed between the sequences, and five seconds of mid-grey were displayed after the sequence pair to

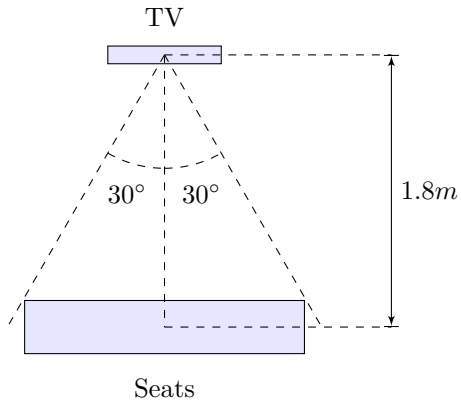


Figure 3.7: Seating arrangement of observers

allow for voting time. The last mid-grey period also included a centered, white-coloured number indicating the next presentation. The structure of the presentations is shown in Figure 3.8.

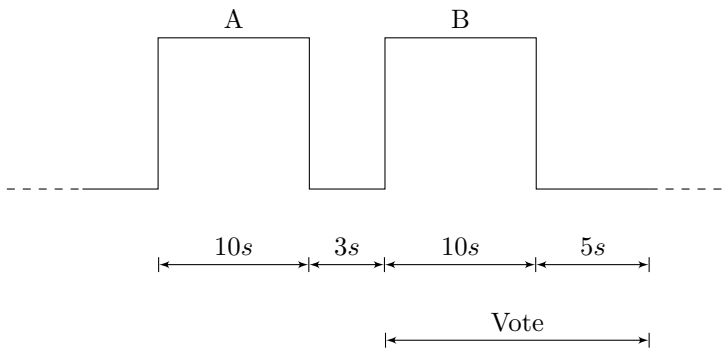


Figure 3.8: Presentation structure of the subjective assessment. Each presentation consisted of a pair of sequences A and B. The total length of a single presentation was 28 seconds. (Adapted from [34])

The observers were asked to vote on a 7-point scale how annoying they considered the distortion of the second sequence to be in comparison to the first. The comparison scale is shown in Table 3.4.

The test conditions for each sequence were subjectively chosen to provide sets ranging from visually bad to good quality. At each test point, both HM-5.2 and MCTF-JP2 had, as close as possible, equal objective quality in terms of the metrics used. The approximate PSNR values for each of the test conditions are given in Table 3.5.

Annoyance comparison scale	
-3	Much more annoying
-2	More annoying
-1	Slightly more annoying
0	Same
1	Slightly less annoying
2	Less annoying
3	Much less annoying

Table 3.4: Annoyance comparison scale.

Sequence	Test condition						
	1	2	3	4	5	6	7
CrowdRun	27.5	28	29	30	31	32	34
DucksTakeOff	27	28	29	30	31	32	34
IntoTree	33	34	35	36	37	38	39
OldTownCross	35	35.5	36	37	38	39	40

Table 3.5: Approximate PSNR values for the two codecs at the different test conditions.

3.6.3 Session

A session consisted of 56 presentations. As each presentation lasted for 28 seconds as shown in Figure 3.8, the total length of a session was just over 26 minutes. The display order of the presentations was randomized. This display order were used for all test sessions. Preferably, a random order should have been made for each session, but this was unfortunately not possible due to time constraints.

Up to three observers at a time participated in a session. A written handout with instructions was given to the assessors. In addition, a verbal explanation of the timing and rating process was given, and a training sequence was shown to familiarize the assessors with the procedure. The goal of the test was explained, and the participants were allowed to ask questions before the session istelf started. The handout given to the participants can be seen in Appendix F.

3.6.4 Participants

A total of 18 observers participated in the assessment. Their age ranged from 20 to 25 years, and 33.3% of the observers were female. 44.4% reported to use some form of visual correction such as glasses or lenses. All of the participants were students, and could be regarded as non-experts of video assessment.

3.6.5 Equipment

To display the assessment material, a Samsung Plasma TV (model PS50 C687 G5S) was used. It had a screen size of 50 inches in the 16:9 format. Any filtering or signal processing of the TV was turned off. The brightness and contrast settings was set using a PLUGE signal for HDTV displays [36]. A screen luminance of $70^{cd}/_{m^2}$ was measured when displaying peak white. This measurement was done with a Konica Minolta LS-100 Luminance meter. The luminance and chromaticity of the lamp that illuminated the background wall could not be accurately measured, due to lack of equipment.

The video signals were fed to the TV via an HDMI cable by a computer running on an Intel Core i7 960 3.20GHz processor with 6 GB of RAM and an ATI Radeon HD 4800 Series video card. The computer ran on a Windows 7 OS. VLC Media Player version 2.0.1 was the software used for video playback. As the assessment material was raw YUV 4:2:0, VLC was used with the command line

```
vlc -demux rawvideo -rawvid-fps 25 -rawvid-width 1280 -rawvid-height 720
-rawvid-chroma=I420 -rawvid-aspect-ratio 16:9 -fullscreen filename.yuv.
```

3.6.6 Analysis

After conducting the subjective assessment, the gathered data was analyzed by applying statistical methods. For each test condition, a mean and a 95% confidence interval were calculated as described in [34].

Mean scores $\bar{u}_{jk}$ were calculated for each of the presentations. The mean scores were calculated as

$$\bar{u}_{jk} = \frac{1}{N}\sum_{i=1}^{N/2}(u_{ijkAB} - u_{ijkBA}) \tag{3.6}$$

where u_{ijkAB} is the score given by observer i for test condition j, sequence k for the AB combination, and vice versa for u_{ijkBA}. N is the total number of scores for the test condition. N is thus twice the number of assessors, since both AB and BA presentation combinations were used.

Note that in 3.6, BA scores are negated to make all scores reflect that MCTF-JP2 is evaluated against HM-5.2. Thus, the scores denotes that MCTF-JP2 is rated as either worse, same or better than HM-5.2 when they are negative, zero or positive, respectively.

The mean scores were then used to calculate 95% confidence intervals. The confidence intervals were calculated as

$$[\bar{u}_{jk} - \delta_{jk}, \bar{u}_{jk} + \delta_{jk}] \tag{3.7}$$

where

$$\delta_{jk} = 1.96\frac{S_{jk}}{\sqrt{N}} \tag{3.8}$$

The standard deviation for each presentation, S_{jk}, is given by

$$S_{jk} = \sqrt{\sum_{i=1}^{N/2} \frac{(\bar{u}_{jk} - u_{ijkAB})^2 + (\bar{u}_{jk} + u_{ijkBA})^2}{(N-1)}} \tag{3.9}$$

Some of the results were studied more closely. In these cases, the results for the specific sequence were checked for outliers by screening the observers, as described in [34]. This was done by calculating kurtosis coefficients to check for normality. The scores of each presentation and observer were then compared to the associated mean value plus and minus the standard value times two if normal and times $\sqrt{20}$ if non-normal. Counters P_{ik} and Q_{ik} were incremented each time a value above or below these thresholds were encountered. The kurtosis coefficient is given by

$$\beta_{2jk} = \frac{\frac{1}{N}\sum_{i=1}^{N}(u_{ijk} - \bar{u}_{jk})^4}{\left(\frac{1}{N}\sum_{i=1}^{N}(u_{ijk} - \bar{u}_{jk})^2\right)^2} \tag{3.10}$$

for test condition j, sequence k. An assessor is taken to be an outlier if the ratio $\frac{P_{ik} + Q_{ik}}{J}$, where J is the number of test conditions, is greater than 5% and the ratio $\frac{P_{ik} - Q_{ik}}{P_{ik} + Q_{ik}}$ is less than 30%.

The procedure used for finding outliers of a specific sequence is described programmatically in Algorithm 1.

Algorithm 1 Procedure for finding outliers of a specific test sequence k.

for all observers i **do**
  $P_{ik} = 0$
  $Q_{ik} = 0$
  for all test conditions j **do**
    if $2 \leq \beta_{2j} \leq 4$ **then** ▷ Normal distribution
      if $u_{ijk} \geq \bar{u}_{jk} + 2 \cdot S_{jk}$ **then**
        $P_{ik} = P_{ik} + 1$
      end if
      if $u_{ijk} \leq \bar{u}_{jk} - 2 \cdot S_{jk}$ **then**
        $Q_{ik} = Q_{ik} + 1$
      end if
    else ▷ Non-normal distribution
      if $u_{ijk} \geq \bar{u}_{jk} + \sqrt{20} \cdot S_{jk}$ **then**
        $P_{ik} = P_{ik} + 1$
      end if
      if $u_{ijk} \leq \bar{u}_{jk} - \sqrt{20} \cdot S_{jk}$ **then**
        $Q_{ik} = Q_{ik} + 1$
      end if
    end if
  end for
  if $\frac{P_{ik} + Q_{ik}}{J} > 0.05$ and $\frac{P_{ik} - Q_{ik}}{P_{ik} + Q_{ik}} < 0.3$ **then**
    Reject assessor i
  end if
end for

Chapter 4

Results and Discussion

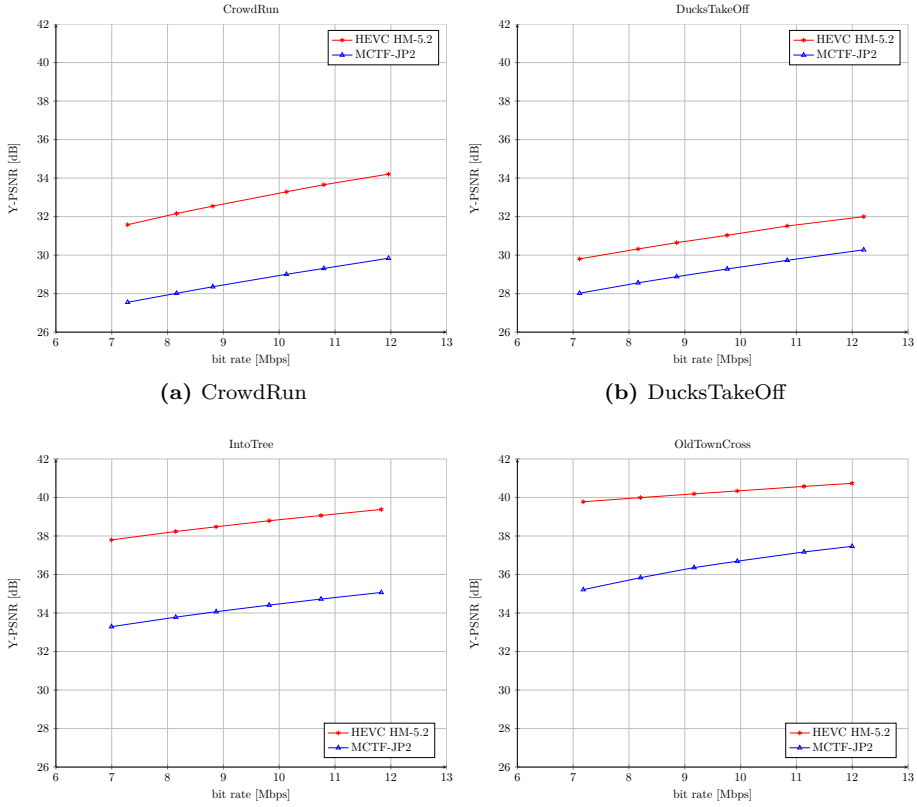
This chapter contains the results obtained from the assessments that was performed. In section 4.1, the results from the objective assessment is presented and discussed. The results from the subjective assessment is presented and discussed in section 4.2. This section also gives a discussion of possible parameters that might have influenced the results. Finally, an evaluation of the proposed codec is given in section 4.3.

4.1 Objective Results

The results obtained after the objective assessment is shown graphically in Figure 4.1 and 4.2 for PSNR and SSIM values, respectively. The values presented are averages over all reconstructed frames. Only luma (Y) values are considered. Detailed results along with full sized plots, can be found in Appendix G.1.

As can be seen in Figure 4.1 and 4.2, MCTF-JP2 is outperformed by HM-5.2 for all of the sequences in the test set, both in terms of PSNR and SSIM. HM-5.2 is for the most part better by a rather large margin, and the curves suggest that MCTF-JP2 needs considerably higher bit-rate to achieve the same objective quality. It is clear that in terms of objective quality, HEVC is the better choice for the use case under consideration.

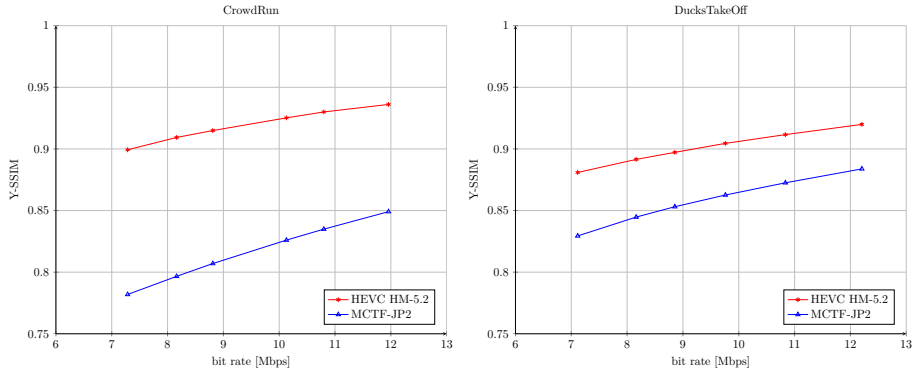
The differences between the two codecs are more clear in Figure 4.3. For PSNR, HM-5.2 scores for the most part 3.5 to $4.5dB$ better than MCTF-JP2. The exception is the results for the DucksTakeOff sequence, where HEVC is better by a margin of about $1.75dB$ across the bit rate range. Compared to CrowdRun, the MCTF-JP2 performance in DucksTakeOff is slightly better. HM-5.2 however drops around $2dB$ across the bit rate range in DucksTakeOff compared to CrowdRun. The same can be seen in the SSIM results; HM-5.2 loses performance, and MCTF-JP2 gains performance when comparing DucksTakeOff to CrowdRun. Both of these sequences



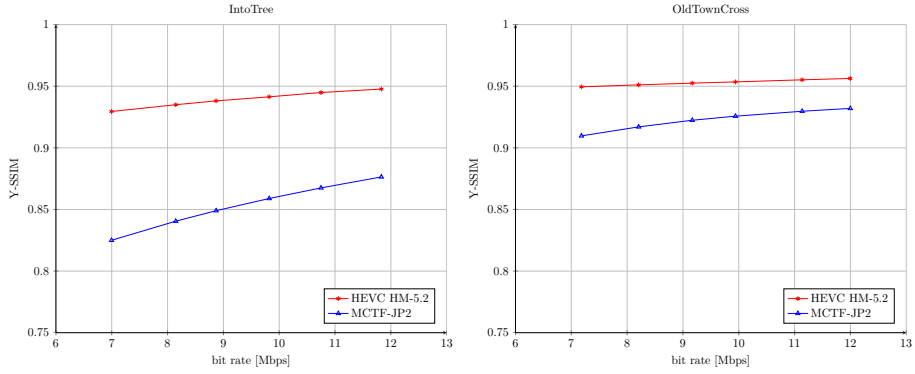
(a) CrowdRun **(b)** DucksTakeOff

(c) IntoTree **(d)** OldTownCross

Figure 4.1: PSNR results

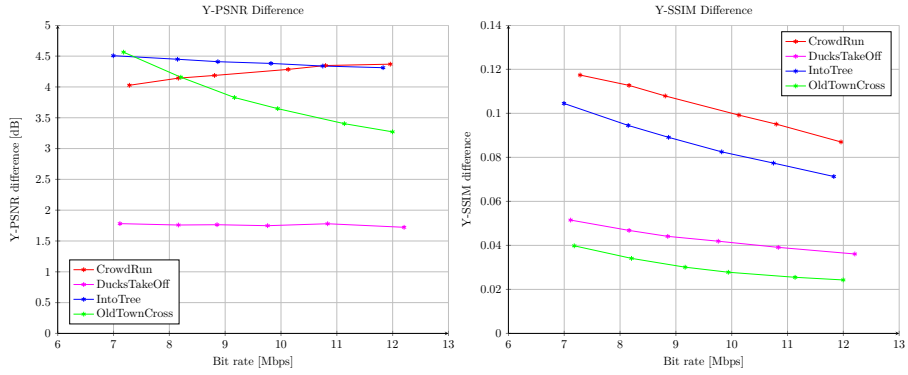


(a) CrowdRun **(b)** DucksTakeOff



(c) IntoTree **(d)** OldTownCross

Figure 4.2: SSIM results



(a) Y-PSNR differences **(b)** Y-SSIM differences

Figure 4.3: Differences between HM-5.2 and MCTF-JP2 results.

are difficult in terms of coding difficulty. This could suggest that different types of content suits the different codecs to a certain degree. HEVC seem to handle content with a lot of objects with hard edges in motion, like in CrowdRun, better than content with soft edges moving around, like the water in DucksTakeOff.

By looking at the differences between the two codecs, shown graphically in Figure 4.3, MCTF-JP2 is clearly getting closer to the SSIM performance of HM-5.2 as the bit rates gets higher. The same behaviour can however not bee seen for PSNR, with the exception of the OldTownCross sequence, which is also the easiest sequence in terms of coding difficulty. So, the difference in noise produced by the two codecs stays fairly constant, while the difference in structural similarity decreases. This might suggest that with increasing bit rates, MCTF-JP2 gets more efficient at distributing the distortions to the least relevant areas relative to HM-5.2. This again suggest that the proposed codec could be better suited for a use case in higher bit rate ranges.

4.2 Subjective Results

The mean values and 95% confidence intervals that were calculated from the data gathered in the subjective assessment is shown graphically in Figure 4.4. These values are not screened for outliers. Increasing test conditions denotes increasing objective quality as shown in Table 3.5.

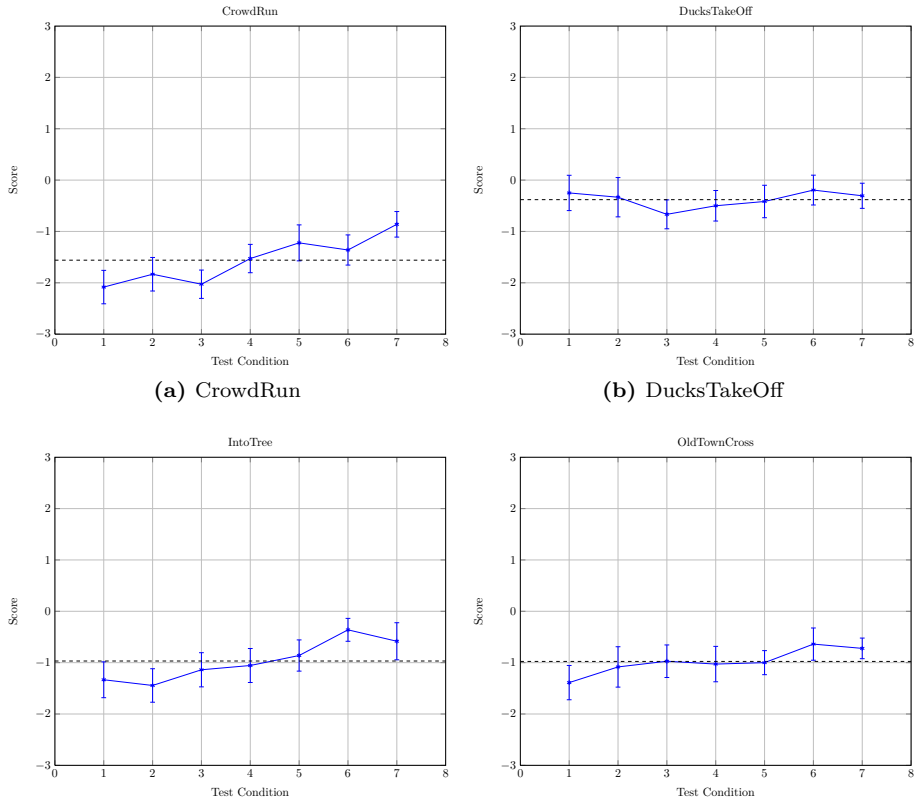
More detailed results and larger plots can be found in Appendix G.2.

As can be seen in Figure 4.4 (b), (c) and (d), the results for these three sequences fits somewhat nicely with the grand means of the respective sequences across the test points. The grand means are tabulated in Table 4.1. For IntoTree and OldTownCross, the grand means are approximately -1, which clearly contradicts the hypothesis that the distortion produced by MCTF-JP2 is less annoying than the distortion produced by HM-5.2.

Sequence	Grand mean
CrowdRun	-1.56
DucksTakeOff	-0.38
IntoTree	-0.97
OldTownCross	-0.98

Table 4.1: Grand means of the sequences under assessment

The story of CrowdRun and DucksTakeOff stands somewhat out from the big picture. Although the CrowdRun results are well below zero, they show a tendency for the MCTF-JP2 codec to be evaluated more positively relative to HM-5.2 with increasing objective quality. DucksTakeOff does not show the same tendency, but is at the other hand pretty close to the zero line.



(a) CrowdRun **(b)** DucksTakeOff

(c) IntoTree **(d)** OldTownCross

Figure 4.4: 95% Confidence intervals. Grand means are showed with black dashed lines.

When examining the CrowdRun results further, it is clear that there are some differences between the results of AB and BA presentation pairs. For the AB pairs, MCTF-JP2 was evaluated in comparison to HM-5.2, and vice versa for BA pairs. As mentioned in Section 3.6.6, the BA results are adjusted to make all results reflect that MCTF-JP2 is being compared to HM-5.2. The AB and BA means, accompanied by the overall results is shown in Figure 4.5.

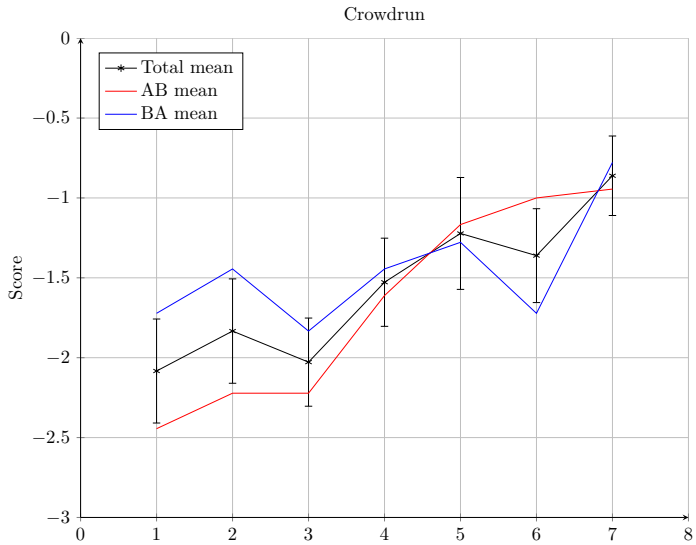


Figure 4.5: Mean values for AB and BA presentations for the CrowdRun sequence. The black graph shows overall confidence intervals. Note that in this graph, the score axis range from −3 to 0.

The AB results clearly show an improvement after the third test point of CrowdRun. The BA means fluctuates more, and does not show the same clear upwards tendency. Assessors seems to be more forgiving to MCTF-JP2 when evaluating BA pairs at lower test points than when the corresponding AB pairs are evaluated.

When screening the assessors, no outliers were found for CrowdRun.

The CrowdRun sequence was in terms of coding difficulty the most difficult sequence in the test set. The results for this sequence shows a clear tendency for MCTF-JP2 to be rated more equal to HM-5.2 at higher test points. The reason for this could simply be that as the objective quality gets higher, it gets more difficult to see the differences of the distortions produced by the two codecs. It could however also suggest that the proposed codec is more efficient when the objective quality gets higher, or that HEVC “saturates” faster. As this behaviour was only clearly observed for the CrowdRun sequence, it is hard to conclude on the cause of this tendency.

For DucksTakeOff, the results across the test points are fairly constant and fits well with its grand mean. However, the curve lies fairly close to 0. This means that for

this sequence, many of the assessors rated the distortion of MCTF-JP2 as equally annoying to the distortion produced by HM-5.2. This is interesting, especially when considering that DucksTakeOff is rated as a "Difficult" sequence to code. It is therefore not obvious that the hypothesis can be rejected for this sequence. Looking at AB and BA means in Figure 4.6, they are on opposite sites of the confidence interval extremities in test point 4 to 6. At the sixth test point, the AB mean is actually above 0. Looking closer at this test condition, a rather large portion of the scores are zero, as can be seen in Figure 4.7. Interestingly, only one of the observers gave the AB presentation a lower score than the BA presentation. This was also the only negative AB-score in this test condition.

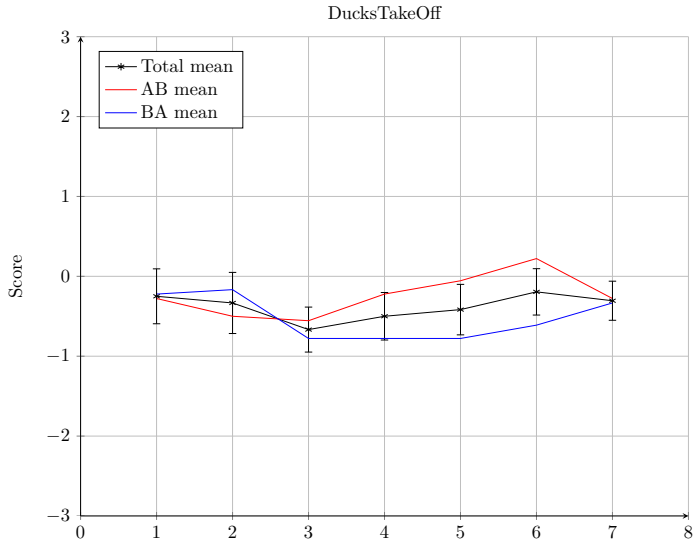


Figure 4.6: Mean values for AB and BA presentations for the DucksTakeOff sequence. The black graph shows overall confidence intervals.

Two outliers were found for DucksTakeOff. The screened versus unscreened results can be seen in Figure 4.8. The removal of outliers further flattened the graph of the results. The intervals after screening are also tabulated in Table 4.2. As can be seen, two of the test conditions has positive upper limits of their confidence intervals, and the upper limits for the other conditions are not far from zero.

For three out of four sequences, the numbers were clearly in favour of HM-5.2. For these sequences, the hypothesis can be rejected. For the last sequence, the numbers were slightly in favour of HEVC, but not enough to be able to readily reject the hypothesis. All in all though, HM-5.2 proved to have the least annoying distortions of the two codecs under test. Still, the tendencies in the results of DucksTakeOff and CrowdRun that were discussed above, shows that there might be possible to make improvements to MCTF-JP2 that could increase the overall subjective quality of the codec.

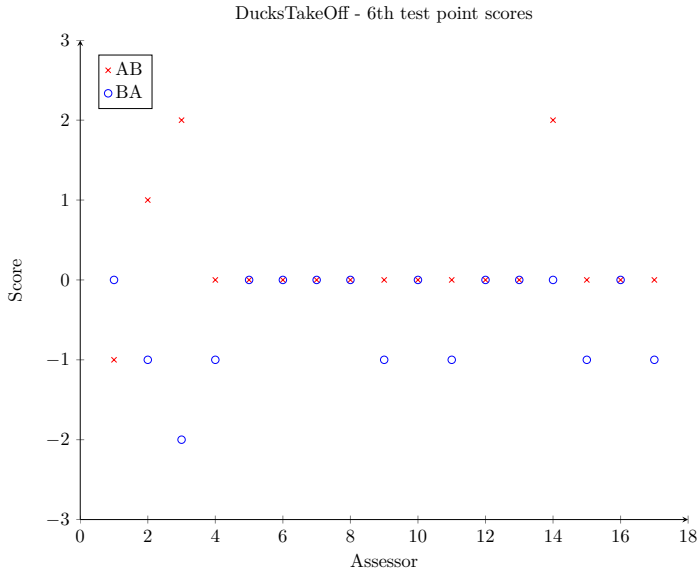


Figure 4.7: Scatter plot for the sixth test condition of DucsTakeOff.

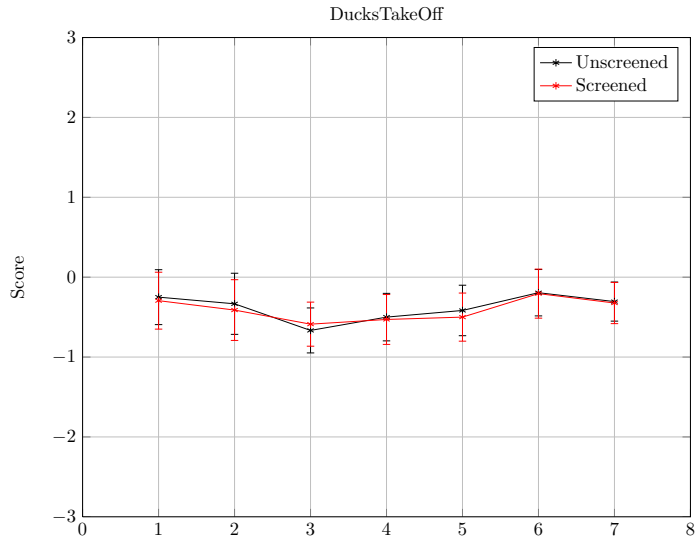


Figure 4.8: Results for DucksTakeOff. The black graph shows the unscreened data, and the red graph shows the screened data, where two outliers has been removed.

Test Condition	Mean	95% Confidence Interval
1	-0.29	$[-0.65, 0.06]$
2	-0.41	$[-0.79, -0.03]$
3	-0.59	$[-0.86, -0.31]$
4	-0.53	$[-0.84, -0.22]$
5	-0.50	$[-0.80, -0.20]$
6	-0.21	$[-0.51, 0.10]$
7	-0.32	$[-0.58, -0.07]$

Table 4.2: 95% confidence intervals for the DucksTakeOff sequence after removing two outliers.

4.2.1 Influences on Results

The results obtained from a subjective assessment can be influenced by many parameters, both wanted and unwanted. Removing unwanted elements that can disturb or influence the formation of opinions by the assessors can be a challenge. Some of the parameters that may have influenced the results of the subjective assessment that was performed is discussed here.

The most important influence of the results is the assessment methodology, i.e how the assessment is conducted. In this assessment, the observers were shown pairs of sequences, both of which where impaired with distortions, and both had approximately equal objective quality. One of the sequences, A, in a pair was coded with HM-5.2, while the other, B, was coded with MCTF-JP2. Both AB and BA pairs were shown to the assessors. Surprisingly, AB and BA pairs were clearly rated differently for some of the test points. This shows that which codec that is first in a pair has an effect of the result. Had only one of the pair orderings been used, the results obtained would have looked different.

The environment in which the assessment took place seemed to be the biggest source of disturbance. Computers and equipment in the lab produced noise. This noise was kept to a minimum, but some of the equipment could not be turned off. There were also some degree of noise from the hallway outside of the lab, which was impossible to control. The results would have been more reliable if the test had been conducted in a lab with better control of the noise level, but all in all, this did not seem to be a big issue.

The test was conducted in a limited part of the lab, enclosed with thick blue curtains. This provided good control of the light intensities in the testing area. However, to illuminate the wall behind the screen, an ordinary table lamp was used. This was clearly not optimal, as the lamp screen hindered the light to spread out, causing only a part of the wall to be illuminated. Also, there was no way of knowing the chromaticity of the lamp. The preferred chromaticity of the background is specified in the recommendations that was followed for the assessment.

A 50-inch plasma screen was used to display the test material. Plasma screens have a reputation for being prone to picture burn-in, where previously viewed content can be faintly seen on the screen. There did not seem to be any permanent burn-in on the screen, but temporary burn-in could be experienced when a text document with white background and black test was displayed for a period of time and then removed. This phenomenon did not seem to be an issue when displaying the test material, but it could be a possible influence on the observers, and so it affects the reliability of the results.

Selection of the test material to use is also important. For this assessment, four sequences from the well-known and documented SVT High Definition Multi Format Test Set was used. Two of the sequences were difficult, and two were easy in terms of coding difficulty. A higher number of test sequences with more varied content could have benefited the reliability of the assessment.

All the assessors that participated were students with an age in the range of 20 to 25 years. A higher versatility in terms of both background and age would be preferable. Also, only about a third of the assessors were female.

4.3 Drawbacks and Possible Improvements of the Proposed Codec

There is little doubt that improvements can be made to MCTF-JP2. By making these improvements, better results in terms of both objective and subjective assessments could be achieved. Considering the amount of work and manpower associated with the development of HEVC, it is inconceivable that results very close to HEVC could be achieved. However, getting rid of the obvious obstacles could help reveal where the true limitations of the scheme lies.

Some of the drawbacks and possible improvements to MCTF-JP2 is presented in this section.

4.3.1 Structural Improvements

Due to the limited time that was available for developing MCTF-JP2, there is some degree of duplicate code, and some of the code is badly structured. By removing duplicate code and structuring the code better, perhaps by making it strictly object-oriented, would improve readability, modifiability and could possibly improve perfomance with respect to running-time. A better architecture would definitely ease any further development of the codec.

A practical video codec needs to take complexity in terms of running-time into account. This was not considered in this work, but if this issue is to be addressed, the first step would be to decide on a suitable programming language. The implementation was done in Matlab. Although Matlab has its advantages in the early

stages of development, it has drawbacks, especially when considering running-time. A language such as C++ or Java would be a better choice.

4.3.2 Motion Estimation

In the way OBMC is performed in MCTF-JP2, each predicted pixel is a simple average of all block predictions (typically four, as most pixels belongs to four blocks, except at edges). A more sophisticated weighting of the predictions could improve performance in terms of quality. Also, adding support for sub-pixel accuracy could give an improvement.

The Diamond Search implementation does not keep track of which nodes that has already been visited and evaluated. As a result, some calculations are unnecessarily done multiple times. Avoiding this would reduce the average asymptotic running time, at the expense of memory usage.

4.3.3 Motion Vector Coding

The biggest drawback of the compression of the motion vectors is that it is lossless. As a consequence, the vector data from a certain video segment or sequence always takes the same amount of the byte budget, regardless of what the target bit-rate is set to be. A scalable vector compression scheme would be more appropriate, and could perhaps increase the overall coding performance at lower bit-rates.

The vectors are coded by a third-party arithmetic coding implementation, which is of a general nature, not specifically made for this use case. A more sophisticated, specialized arithmetic coder would certainly increase performance. In addition, the motion vectors are run-length coded before arithmetic coding. This is done by raster scanning the vector coefficients. The run-length coding might be more effective with for example zig-zag scanning.

4.3.4 MCTF

The MCTF filter that was implemented was a 1/3 lifting filter. This filter does not utilize the update step. Implementing a filter with a proper update step such as the 5/3-filter could give the coding efficiency a boost. There are however challenges to this, as one need to address the motion continuity problem [18] to avoid unnecessary distortions. An attempt to implement the 5/3 filter was made, but there were difficulties getting it to function properly. Due to time constraints, further work on the 5/3 filter was abandoned.

The HP subbands resulting from the MCTF procedure contains error residuals. For input with 8-bit precision, these HP frames can then contain values in the range of $[-255, 255]$. There is thus an increase of precision associated with the HP frames.

This is of course an undesired effect, both in terms of efficiency and the fact that JasPer does not support coding n-bit images. The scaling technique described in Section 3.4.3 is adopted to adress this issue. This method seems to be slightly more efficient, or at least not worse than a simple cutoff, but it could be worth while investigating if there are better solutions to this problem.

4.3.5 Rate Control

The rate control mechanism that was implemented was very simple. The division of available bytes on the different subbands was based on experimentation. An adaptive solution that in some way optimizes the way the available bytes are used would probably giver better results. There might for example be beneficial to spend more bytes in the lower temporal layers than the higher ones. In any case, the way the byte budget is used should not rely on parameters given as input to the encoder.

Chapter 5

Conclusions

A wavelet MCTF codec based on JPEG 2000 and the T+2D structure, MCTF-JP2, has been developed and presented. It has been tested by means of both objective and subjective assessments and compared to the HEVC test model HM-5.2. The use case for the assessments was streaming of 720p25 video with a typical consumer broadband connection.

The objective assessment showed that HM-5.2 by far was the superior coding system. Both PSNR and SSIM was used as metrics, and HEVC came out with the best results for both in all cases. For PSNR, the great majority of values showed that HM-5.2 was 3.5 to $4.5dB$ above MCTF-JP2 across the bit rate range. SSIM results revealed that the proposed codec got closer to the performance of HEVC with increasing bit rates.

In the subjective assessment, the annoyance of the distortions produced by the two codecs was evaluated relative to each other. The hypothesis was that MCTF-JP2 would produce less visually annoying distortions than HM-5.2. This hypothesis was clearly rejected for three of the sequences, as the results indicated the opposite. The results for the last sequence was also in favour of HEVC, but not enough to make a clear rejection of the hypothesis.

All in all, when considering the results, HEVC is the better choice for the presented use case. The results, together with the evaluation of the proposed codec in Section 4.3 also suggest that some improvements can be made to MCTF-JP2 to reduce the gap between the two systems.

5.1 Future Work

In order to find the true limitations of the wavelet MCTF structure that was used for the proposed codec, improvements to MCTF-JP2 such as those presented in

Section 4.3 should be made. It could also be worthwhile to investigate whether or not a 2D+T structure is a better solution than the T+2D structure.

The obtained results suggests that MCTF-JP2 might perform better for higher bit rates than those used for testing the use case presented in Section 3.1. The proposed codec should therefore be tested against an alternative use case. This use case should operate in a higher bit rate range, and it could be beneficial to test different kinds of materials, both in terms of content and temporal and spatial resolutions.

HEVC is still under development. During the work on this thesis, two JCT-VC meetings has been held. Version 6 of the test model has been released, with version 7 just around the corner at the moment of writing. For another and perhaps more fruitful direction of future work than already discussed, investigations on improving HEVC could be made. Work on scalability extensions for HEVC would be especially interesting.

Appendices

Appendix A

HEVC HM-5.2 Configuration

Listing A.1 gives the configuration file used for HM-5.2. This is the Random Access, Low Complexity configuration. The original file can be found in [26]. Note that comments and some whitespace has been removed in order to fit the text to the page.

```
#======== File I/O =====================
BitstreamFile                     : str.bin
ReconFile                         : rec.yuv

#======== Unit definition ================
MaxCUWidth                        : 64
MaxCUHeight                       : 64
MaxPartitionDepth                 : 4
QuadtreeTULog2MaxSize             : 5
QuadtreeTULog2MinSize             : 2
QuadtreeTUMaxDepthInter           : 3
QuadtreeTUMaxDepthIntra           : 3

#======== Coding Structure =============
IntraPeriod                       : 32
DecodingRefreshType               : 1
GOPSize                           : 8

Frame1: B 8 1 0.442    0 4 1 4  -8 -10 -12 -16  0
Frame2: B 4 2 0.3536   0 2 1 3  -4 -6  4        1  0   4  5  1 1 0 0 1
Frame3: B 2 3 0.3536   0 2 1 4  -2 -4  2 6      1  0   2  4  1 1 1 1
Frame4: B 1 4 0.68     0 2 0 4  -1  1  3 7      1  0   1  5  1 0 1 1 1
Frame5: B 3 4 0.68     0 2 0 4  -1 -3  1 5      1  0  -2  5  1 1 1 1 0
Frame6: B 6 3 0.3536   0 2 1 4  -2 -4 -6 2      1  0  -3  5  1 1 1 1 0
Frame7: B 5 4 0.68     0 2 0 4  -1 -5  1 3      1  0   1  5  1 0 1 1 1
Frame8: B 7 4 0.68     0 2 0 4  -1 -3 -7 1      1  0  -2  5  1 1 1 1 0
ListCombination                   : 1
DisableInter4x4                   : 1
```

```
#========== Motion Search ==========
FastSearch                    : 1
SearchRange                   : 64
BipredSearchRange             : 4
HadamardME                    : 1
FEN                           : 1

#========== Quantization ==========
QP                            : 32
MaxDeltaQP                    : 0
MaxCuDQPDepth                 : 0
DeltaQpRD                     : 0
RDOQ                          : 0
ChromaQpOffset                : 0
ChromaQpOffset2nd             : 0

#========== Entropy Coding ==========

#========== Deblock Filter ==========
LoopFilterDisable             : 0
LoopFilterAlphaC0Offset       : 0
LoopFilterBetaOffset          : 0

#========== Misc. ==========
InternalBitDepth              : 8

#========== Coding Tools ==========
MRG                           : 1
SAO                           : 0
ALF                           : 0
ALFEncodePassReduction        : 0
LMChroma                      : 0
NSQT                          : 0
AMP                           : 0
ALFMaxNumFilter               : 16

#========== Slices ==========
SliceGranularity          : 0
SliceMode                 : 0
SliceArgument             : 1500
LFCrossSliceBoundaryFlag  : 1
EntropySliceMode          : 0
EntropySliceArgument      : 180000

#========== PCM ==========
PCMEnabledFlag                : 0
PCMLog2MaxSize                : 5
PCMLog2MinSize                : 3
PCMInputBitDepthFlag          : 1
PCMFilterDisableFlag          : 0

#========== Tiles ==========
TileInfoPresentFlag           : 1
UniformSpacingIdc             : 0
TileBoundaryIndependenceIdc   : 1
NumTileColumnsMinus1          : 0
```

```
ColumnWidthArray                        : 2 3
NumTileRowsMinus1                       : 0
RowHeightArray                          : 2
TileLocationInSliceHeaderFlag           : 0
TileMarkerFlag                        : 1
MaxTileMarkerEntryPoints              : 4
TileControlPresentFlag                : 1
LFCrossTileBoundaryFlag               : 1

#============ WaveFront ================
WaveFrontSynchro                        : 0
WaveFrontFlush                          : 0
WaveFrontSubstreams                     : 1

#============ Quantization Matrix ================
ScalingList                           : 0
ScalingListFile                       : scaling_list.txt

### DO NOT ADD ANYTHING BELOW THIS LINE ###
### DO NOT DELETE THE EMPTY LINE BELOW ###
```

Listing A.1: Content of configuration file `encoder_randomaccess_loco.cfg` used for HM-5.2

Appendix B

Matlab Code

This appendix provides exerpts from the Matlab source code of MCTF-JP2. The source code can in its entirety be found in the accompanying zip-file.

B.1 Motion Estimation Implementation

Listing B.1 shows the implementation of motion estimation for OBMC in MCTF-JP2. The main function `obmc()` returns motion vectors for the current frame. The function takes the current and reference frame, along with the block size and search range as parameters. The subfunctions `ds()` and `fullSearch()` are implementations of Diamond Search and Full Search, respectively. These functions finds the motion vector for a specific block in the current frame.

```
1  function [mvx, mvy] = obmc(cur_frame, ref_frame, blockSize,
       searchRange)
2
3  dim = size(cur_frame);
4
5  if ( dim(1) < blockSize*3/2 || dim(2) < blockSize*3/2 || mod( dim(1),
       blockSize/2 ) || mod( dim(2), blockSize/2 ) )
6      error('Bad block size');
7  end
8
9  b = blockSize/2;
10 mvx = zeros( (dim / b) - 1 );
11 mvy = zeros( (dim / b) - 1 );
12
13 for i = 1:b:dim(1)-b
14     for j = 1:b:dim(2)-b
15         [dx, dy] = ds(cur_frame, ref_frame, blockSize, searchRange, i,
               j);
16         %[dx, dy] = fullSearch(cur_frame, ref_frame, blockSize,
               searchRange, i, j);
```

```
17          mvy( (i-1)/b + 1, (j-1)/b + 1 ) = dy;
18          mvx( (i-1)/b + 1, (j-1)/b + 1 ) = dx;
19      end
20  end
21
22  end
23
24
25
26  function [dx, dy] = ds(cur_frame, ref_frame, blockSize, searchRange, i
        , j)
27  % Diamond Search
28
29  ldsp_ind = [0 2; -1 1; 1 1; -2 0; 0 0; 2 0; -1 -1; 1 -1; 0 -2]; %Large
        Diamond Search Pattern indices
30  sdsp_ind = [0 1; -1 0; 0 0; 1 0; 0 -1]; %Small Diamond Search Pattern
        indices
31
32  dim = size(cur_frame);
33
34  MSE_min = 256^2;
35  dy = 0;      %Current y vector comp
36  dx = 0;      %Current x vector comp
37  %ldsp
38  while ( abs(dy) < searchRange - 3 && abs(dx) < searchRange - 3) %Less
        than search range - 3 because ldsp might add 2, and sdsp might add
         1
39      best_y = 0; %Best y relative ind
40      best_x = 0; %Best x relative ind
41      for k = 1:length(ldsp_ind)
42          ref_y = i + dy + ldsp_ind(k,2);
43          ref_x = j + dx + ldsp_ind(k,1);
44          if ( ref_y > 0 && (ref_y + blockSize - 1) <= dim(1) ...
45                  && ref_x > 0 && (ref_x + blockSize - 1) <= dim(2) ) %
                    Check if block is inside frame
46              % Inline mse
47              MSE = sum( sum(  ( double(cur_frame(i:i+blockSize-1, j:j+
                    blockSize-1)) - double(ref_frame(ref_y:ref_y+blockSize
                    -1, ref_x:ref_x+blockSize-1)) ).^2 ./blockSize ) ./
                    blockSize );
48              if MSE < MSE_min
49                  MSE_min = MSE;
50                  best_x = ldsp_ind(k,1);
51                  best_y = ldsp_ind(k,2);
52              end
53          end
54      end
55      dx = dx + best_x;
56      dy = dy + best_y;
57      if ( best_x == 0 && best_y ==0 )
58          break;
59      end
60  end
61  %sdsp
62  sdsUpdate = 0;
63  for k = 1:length(sdsp_ind)
64      ref_y = i + dy + sdsp_ind(k,2);
```

```
65      ref_x = j + dx + sdsp_ind(k,1);
66      if ( ref_y > 0 && (ref_y + blockSize - 1) <= dim(1) ...
67              && ref_x > 0 && (ref_x + blockSize - 1) <= dim(2) ) %Check
                    if block is inside frame
68          % Inline mse
69          MSE = sum( sum(   ( double(cur_frame(i:i+blockSize-1, j:j+
                blockSize-1)) - double(ref_frame(ref_y:ref_y+blockSize-1,
                ref_x:ref_x+blockSize-1)) ).^2 ./blockSize ) ./blockSize )
                ;
70          if MSE < MSE_min
71              MSE_min = MSE;
72              best_x = sdsp_ind(k,1);
73              best_y = sdsp_ind(k,2);
74              sdsUpdate = 1;
75          end
76      end
77  end
78  if (sdsUpdate)
79      dx = dx + best_x;
80      dy = dy + best_y;
81  end
82
83  end
84
85
86
87
88
89
90
91
92
93  function [dx, dy] = fullSearch(cur_frame, ref_frame, blockSize,
        searchRange, i, j)
94  % Full search
95
96  dim = size(cur_frame);
97
98  MSE_min = 256^2;
99  dy = 0;
100 dx = 0;
101 for k = -(searchRange-1):1:(searchRange-1)
102     for l = -(searchRange-1):1:(searchRange-1)
103         if( i+k > 0 && i+k+blockSize-1 <= dim(1) && j+l > 0 && j+l+
                blockSize-1 <= dim(2) )
104             % Inline mse
105             MSE = sum( sum(   ( double(cur_frame(i:i+blockSize-1, j:j+
                    blockSize-1)) - double(ref_frame(i+k:i+k+blockSize-1,
                    j+l:j+l+blockSize-1)) ).^2 ./blockSize ) ./blockSize )
                    ;
106             if MSE < MSE_min
107                 MSE_min = MSE;
108                 dy = k;
109                 dx = l;
110             end
111         end
112     end
```

```
113 end
114
115 end
```

Listing B.1: Content of Matlab file `../matlab_prototype/codec/obmc.m`. This function file provides motion vectors used for OBMC. The choice between DS and FS is hardcoded in the function `obmc()`.

B.2 1/3 MCTF Filter Implementation

Listing B.2 shows the implementation of the 1/3 MCTF filter. The `analysis()` and `synthesis()` functions takes frames or subbands as input together with their motion vector fields. An optional parameter specifies whether or not the vector fields are overlapping. If this parameter is not given as input, it is assumed that OBMC is to be utilized.

```
1  classdef OneThreeMCTF
2
3      methods (Static)
4
5          function h_band = analysis(A_1, B, A_2, mvx_f, mvy_f, mvx_b,
               mvy_b, blockSize, varargin)
6              obmc = 1;
7              if (nargin == 9)
8                  obmc = varargin{1};
9              end
10
11             h_band = double(B) - OneThreeMCTF.prediction(A_1, A_2,
                   mvx_f, mvy_f, mvx_b, mvy_b, blockSize, obmc);
12
13         end
14
15         function L = synthesis(l_band_1, h_band, l_band_2, mvx_f,
               mvy_f, mvx_b, mvy_b, blockSize, varargin)
16             obmc = 1;
17             if (nargin == 9)
18                 obmc = varargin{1};
19             end
20
21             L = double(h_band) + OneThreeMCTF.prediction(l_band_1,
                   l_band_2, mvx_f, mvy_f, mvx_b, mvy_b, blockSize, obmc)
                   ;
22
23         end
24
25     end
26
27     methods (Static, Access = private)
28
29         function a = prediction(prev_frame, fut_frame, mvx_f, mvy_f,
               mvx_b, mvy_b, blockSize, obmc)
30             dim = size(prev_frame);
31             a = zeros(dim);
```

```
32
33              if (obmc)
34                  b = blockSize / 2;
35
36                  weights = ones( dim / b ) * 2;
37                  weights(1,1) = 1;
38                  weights(1,end) = 1;
39                  weights(end,1) = 1;
40                  weights(end,end) = 1;
41                  weights(2:end-1,2:end-1) = 4;
42
43                  mvDim = size(mvx_f);
44                  for i = 1:mvDim(1)
45                      for j = 1:mvDim(2)
46                          dx = mvx_f(i,j);
47                          dy = mvy_f(i,j);
48                          for y = (i-1)*b+1 : (i-1)*b+blockSize
49                              for x = (j-1)*b+1 : (j-1)*b+blockSize
50                                  a(y,x) = a(y,x) + ( prev_frame(y+dy, x
                                        +dx) / weights(ceil(y/b),ceil(x/b)
                                        ) );
51                              end
52                          end
53                      end
54                  end
55
56                  mvDim = size(mvx_b);
57                  bb = zeros(dim);
58                  for i = 1:mvDim(1)
59                      for j = 1:mvDim(2)
60                          dx = mvx_b(i,j);
61                          dy = mvy_b(i,j);
62                          for y = (i-1)*b+1 : (i-1)*b+blockSize
63                              for x = (j-1)*b+1 : (j-1)*b+blockSize
64                                  bb(y,x) = bb(y,x) + (fut_frame(y+dy, x
                                        +dx) / weights(ceil(y/b),ceil(x/b)
                                        ));
65                              end
66                          end
67                      end
68                  end
69
70                  a = floor( (a + bb) /2 );
71
72              else %not obmc
73                  for i = 1:dim(1)
74                      for j = 1:dim(2)
75                          a(i,j) = floor( ...
76                              (prev_frame( i + mvy_f( ceil(i/blockSize),
                                    ceil(j/blockSize) ), j + mvx_f( ceil(
                                    i/blockSize), ceil(j/blockSize) ) )
                                    +...
77                              fut_frame( i + mvy_b( ceil(i/blockSize),
                                    ceil(j/blockSize) ), j + mvx_b( ceil(i
                                    /blockSize), ceil(j/blockSize) ) ) ...
78                              )/ 2 );
79                      end
```

```
80                      end
81                  end
82              end
83
84          end
85
86  end
```

Listing B.2: Content of Matlab file `../matlab_prototype/codec/OneThreeMCTF.m`. This class implements the lifted 1/3 MCTF filter.

Appendix C

Examples of Subband Frames

This appendix provides examples of subband frames after filtering with the 1/3 MCTF filter.

Figure C.1: Example of a Luma component lowpass frame from the OldTownCross sequence after 1/3 MCTF filtering. For the 1/3 filter, lowpass frames acts as IDR frames.

Figure C.2: Example of a Luma component highpass frame from the OldTownCross sequence after 1/3 MCTF filtering. The picture has been contrast adjusted for visibility.

Appendix D

Research Protocol for Subjective Assessment

D.1 Synopsis

Video is becoming an increasing part of our daily life. As the technology available to consumers has become more and more advanced, regular users are now able to watch video content on a multitude of devices. Streaming of video over the Internet has become increasingly popular, and since bandwidth is a limited resource, efficient compression techniques are clearly needed. The wide variety of devices capable of displaying video includes everything from small, battery powered devices to large TV-sets. This wide range of devices suggests the need for scalable video coders, as different devices may support different sets of resolutions and frame rates, and transcoding is an impractical and inefficient solution.

To meet the demands for efficient coding standards, the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG) are jointly developing HEVC, an emerging video compression standard. Its goal is to improve the coding efficiency compared to H.264/AVC, without affecting image quality. A scalable video coding extension to HEVC is also planned.

While HEVC is based on the hybrid coding approach, attention is given to 3D wavelet coders in the literature. JPEG 2000 is a wavelet image coder which offer spatial and quality scalability. A combination of JPEG 2000 and wavelet based Motion Compensated Temporal Filtering (MCTF) gives a 3D wavelet video coder with temporal, spatial and quality scalability, without the need for complex extensions.

D.2 Background

Currently, JPEG 2000 is mostly used for video where high quality is important, and available bandwidth is less of an issue. In these cases, JPEG 2000 is used for intra-coding, i.e. without exploiting temporal redundancy.

This subjective assessment will examine the potential of JPEG 2000 in combination with MCTF for use in bandwidth-constricted cases. The aim is to find out how annoying the distortion of MCTF-JP2, the proposed JPEG 2000-based codec, is perceived in comparison with HEVC when the objective quality is comparable. This will be achieved by performing a subjective test where HEVC and MCTF-JP2 is tested at points with equal objective quality.

If MCTF-JP2 proves to have less annoying distortion than HEVC at equal objective quality points, it shows that there is a potential for wavelet codecs in bandwidth-constricted use cases. This means that codecs with both temporal, spatial and quality scalability can be achieved without the need for implementing complex extensions or transcoding streams.

D.3 Hypothesis

The distortion produced by MCTF-JP2 is less annoying than the distortion produced by HEVC.

D.4 Methodology and Design

D.4.1 Measures Used to Test the Hypothesis

- Two codecs will be tested; HEVC test model HM-5.2 and MCTF-JP2.
- The test will measure the annoyance of the distortion produced by HEVC and MCTF-JP2
- The data set will be tested with PSNR/SSIM values ranging from what can be considered bad to good quality.
- The test will use the Stimulus Comparison Adjectival Categorical Judgement (SCACJ) method in accordance with [34]. Each presentation will consist of two impaired sequences, which are considered to have equal objective quality. The observer is asked to vote on a 7 point scale how (s)he evaluates the distortion annoyance of the second sequence in comparison to the first. The scale used for this test will be defined as in Table D.1.

- The test will consist of 56 presentations, where each presentation consists of two sequences of 10s each, 3s mid-grey between the sequences and 5s mid-grey at the end to allow voting time.

- Each presentation lasts 28s. With 56 presentations, the test will last for about 26 min. Allowing 4 min of training and introduction, a test session will last 30 min.

- Each sequence in the data set will have 14 associated presentations. A presentation consists of two impaired sequences A and B. Both possible pairs AB and BA will be used. This gives 7 test points for each sequence in the data set.

- The viewing conditions for the test will be in accordance with [35]

Annoyance comparison scale	
-3	Much more annoying
-2	More annoying
-1	Slightly more annoying
0	Same
1	Slightly less annoying
2	Less annoying
3	Much less annoying

Table D.1: Annoyance comparison scale.

D.4.2 Resources Required

- At least 20 observers will participate in the test.

- The room Café Media will be required for one work week of testing.

- A computer capable of mounting a secondary monitor and handle raw YUV 4:2:0 720p25 video material will be required

- A monitor capable of displaying 720p25 video material will be required

- The data set consists of four sequences from the SVT High Definition Multi Format Test Set. All sequences have a resolution of 720p25 and a duration of 10s.

- The participants will be compensated with a cinema ticket.

D.5 Results Analysis

In order to test the hypothesis, the obtained results will be used to calculate 95% confidence intervals.

D.6 Priority and Timetable

The test sessions will be held in week 16.

After finishing the test sessions, two weeks will be used analysing the results of the test.

The remaining time before thesis deadline (June 11) will be spent writing the thesis. Approximately two weeks before deadline, a finished draft will be delivered to the supervisor.

Week	Task
13	Preparing for subjective tests
14	Easter
15	Preparing for subjective tests
16	Subjective testing sessions
17 - 18	Analysing results
19 - 21	Writing report
22	Writing report, deliver finished draft to supervisor
22 - 23	Writing report
24	Thesis deadline

Table D.2: Timetable for subjective testing

Appendix E

Pictures of Test Environment

The pictures below shows the test environment used for the subjective assessments. All pictures are taken by fellow student Magnus Jeffs Tovslid, and is used with permission.

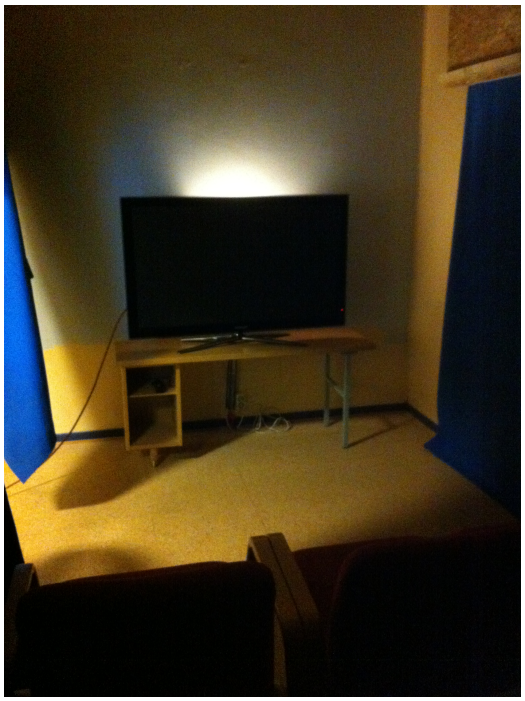


Figure E.1: Picture of the viewing area, taken from behind the seats. The curtains on the left were closed during testing. (M. J. Tovslid)

Figure E.2: Picture of the test environment taken from the outside. (M. J. Tovslid)

Figure E.3: Picture of the three seats in the viewing area. (M. J. Tovslid)

Appendix F

Subjective Assessment Handout

About you

- **Your Name:** _____
- **Your Age:** _____
- **Your Gender:** ☐ Male ☐ Female
- **Do you normally wear glasses, lenses or other forms of vision correction?** ☐ Yes ☐ No

The information above will not be used for any other purposes than this study.

Instructions

In this test you will be presented with different pairs of video sequences. Both sequences in a pair may have visual degradations. You are asked to judge how annoying the visual degradations of the second sequence are compared to the first.

Both sequences in a pair lasts for 10 seconds. Between them are 3 seconds of mid-grey color to allow you to separate them from each other.

Voting is done according to the scale defined in the table below. After viewing a sequence pair, you have 5 seconds to vote. To vote, check the box of the corresponding score in the provided score sheet.

Thank you for participating!

Annoyance comparison scale	
-3	Much more annoying
-2	More annoying
-1	Slightly more annoying
0	Same
1	Slightly less annoying
2	Less annoying
3	Much less annoying

Table F.1: Scale to compare the visual annoyance of the second sequence in a pair relative to the first

#	-3	-2	-1	0	1	2	3
1	☐	☐	☐	☐	☐	☐	☐
2	☐	☐	☐	☐	☐	☐	☐
3	☐	☐	☐	☐	☐	☐	☐
4	☐	☐	☐	☐	☐	☐	☐
5	☐	☐	☐	☐	☐	☐	☐
6	☐	☐	☐	☐	☐	☐	☐
7	☐	☐	☐	☐	☐	☐	☐
8	☐	☐	☐	☐	☐	☐	☐
9	☐	☐	☐	☐	☐	☐	☐
10	☐	☐	☐	☐	☐	☐	☐
11	☐	☐	☐	☐	☐	☐	☐
12	☐	☐	☐	☐	☐	☐	☐
13	☐	☐	☐	☐	☐	☐	☐
14	☐	☐	☐	☐	☐	☐	☐
15	☐	☐	☐	☐	☐	☐	☐
16	☐	☐	☐	☐	☐	☐	☐
17	☐	☐	☐	☐	☐	☐	☐
18	☐	☐	☐	☐	☐	☐	☐
19	☐	☐	☐	☐	☐	☐	☐
20	☐	☐	☐	☐	☐	☐	☐
21	☐	☐	☐	☐	☐	☐	☐
22	☐	☐	☐	☐	☐	☐	☐
23	☐	☐	☐	☐	☐	☐	☐
24	☐	☐	☐	☐	☐	☐	☐
25	☐	☐	☐	☐	☐	☐	☐
26	☐	☐	☐	☐	☐	☐	☐
27	☐	☐	☐	☐	☐	☐	☐
28	☐	☐	☐	☐	☐	☐	☐

#	-3	-2	-1	0	1	2	3
29	☐	☐	☐	☐	☐	☐	☐
30	☐	☐	☐	☐	☐	☐	☐
31	☐	☐	☐	☐	☐	☐	☐
32	☐	☐	☐	☐	☐	☐	☐
33	☐	☐	☐	☐	☐	☐	☐
34	☐	☐	☐	☐	☐	☐	☐
35	☐	☐	☐	☐	☐	☐	☐
36	☐	☐	☐	☐	☐	☐	☐
37	☐	☐	☐	☐	☐	☐	☐
38	☐	☐	☐	☐	☐	☐	☐
39	☐	☐	☐	☐	☐	☐	☐
40	☐	☐	☐	☐	☐	☐	☐
41	☐	☐	☐	☐	☐	☐	☐
42	☐	☐	☐	☐	☐	☐	☐
43	☐	☐	☐	☐	☐	☐	☐
44	☐	☐	☐	☐	☐	☐	☐
45	☐	☐	☐	☐	☐	☐	☐
46	☐	☐	☐	☐	☐	☐	☐
47	☐	☐	☐	☐	☐	☐	☐
48	☐	☐	☐	☐	☐	☐	☐
49	☐	☐	☐	☐	☐	☐	☐
50	☐	☐	☐	☐	☐	☐	☐
51	☐	☐	☐	☐	☐	☐	☐
52	☐	☐	☐	☐	☐	☐	☐
53	☐	☐	☐	☐	☐	☐	☐
54	☐	☐	☐	☐	☐	☐	☐
55	☐	☐	☐	☐	☐	☐	☐
56	☐	☐	☐	☐	☐	☐	☐

Appendix G

Results

This appendix provides detailed results obtained from the objective and subjective assessments, along with large plots. Objective and subjective reults are given in Section G.1 and G.2, respectively.

G.1 Objective Results

G.1.1 PSNR Results

Sequence		Bit rate (Mbps)					
		7	8	9	10	11	12
CrowdRun	HM-5.2	31.58	32.16	32.54	33.29	33.66	34.21
	MCTF-JP2	27.55	28.02	28.36	29.00	29.31	29.84
	Difference	04.03	04.14	04.18	04.29	04.35	04.37
DucksTakeOff	HM-5.2	29.80	30.32	30.65	31.03	31.51	32.00
	MCTF-JP2	28.02	28.56	28.88	29.28	29.73	30.28
	Difference	01.78	01.76	01.77	01.75	01.78	01.72
IntoTree	HM-5.2	37.80	38.23	38.48	38.79	39.06	39.38
	MCTF-JP2	33.29	33.78	34.07	34.41	34.72	35.07
	Difference	04.51	04.45	04.41	04.38	04.34	04.31
OldTownCross	HM-5.2	39.78	39.99	40.19	40.34	40.57	40.73
	MCTF-JP2	35.21	35.83	36.36	36.69	37.17	37.46
	Difference	04.57	04.16	03.83	03.65	03.40	03.27

Table G.1: Y-PSNR results

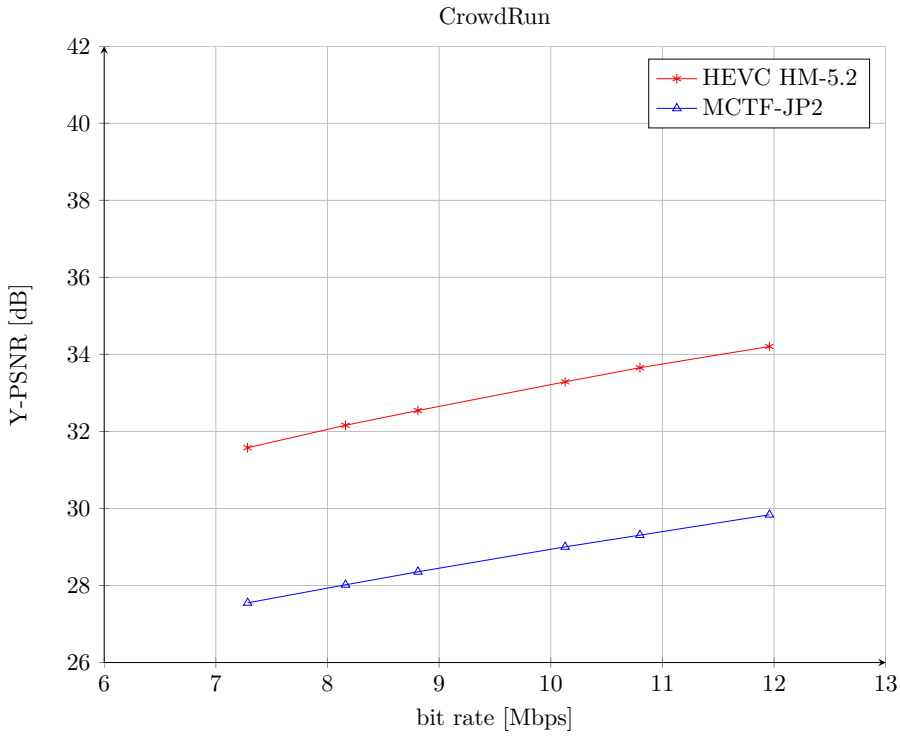


Figure G.1: Y-PSNR results for the CrowdRun sequence

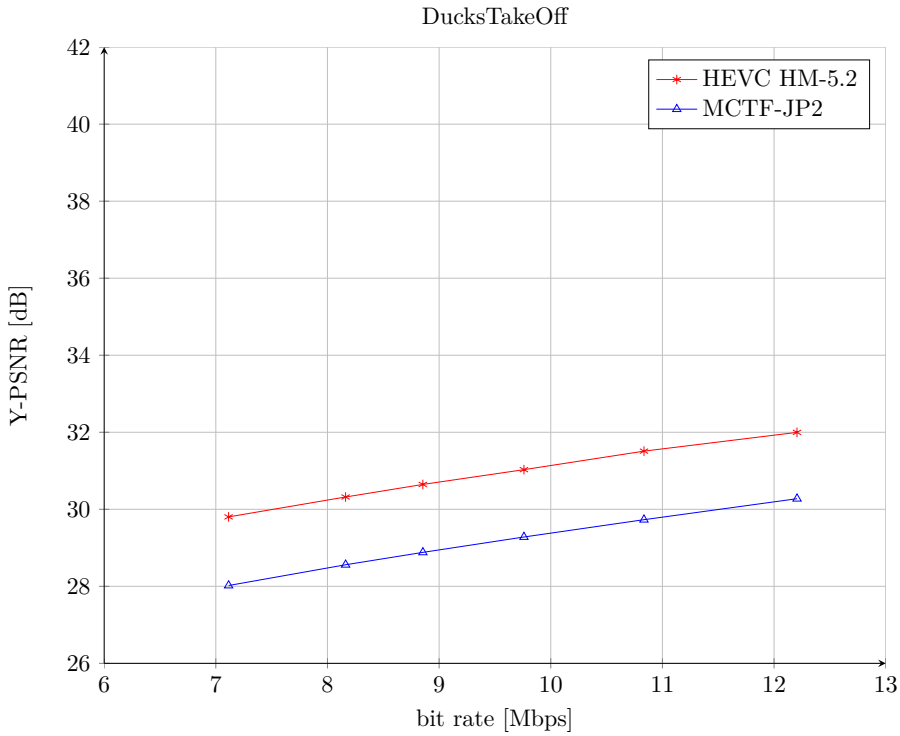


Figure G.2: Y-PSNR results for the DucksTakeOff sequence

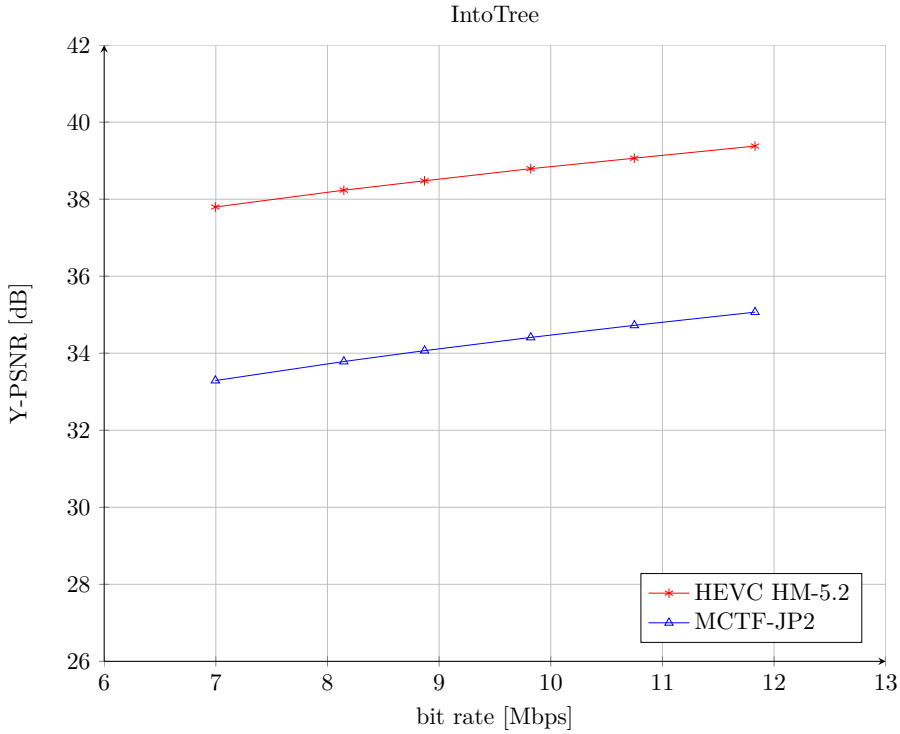


Figure G.3: Y-PSNR results for the IntoTree sequence

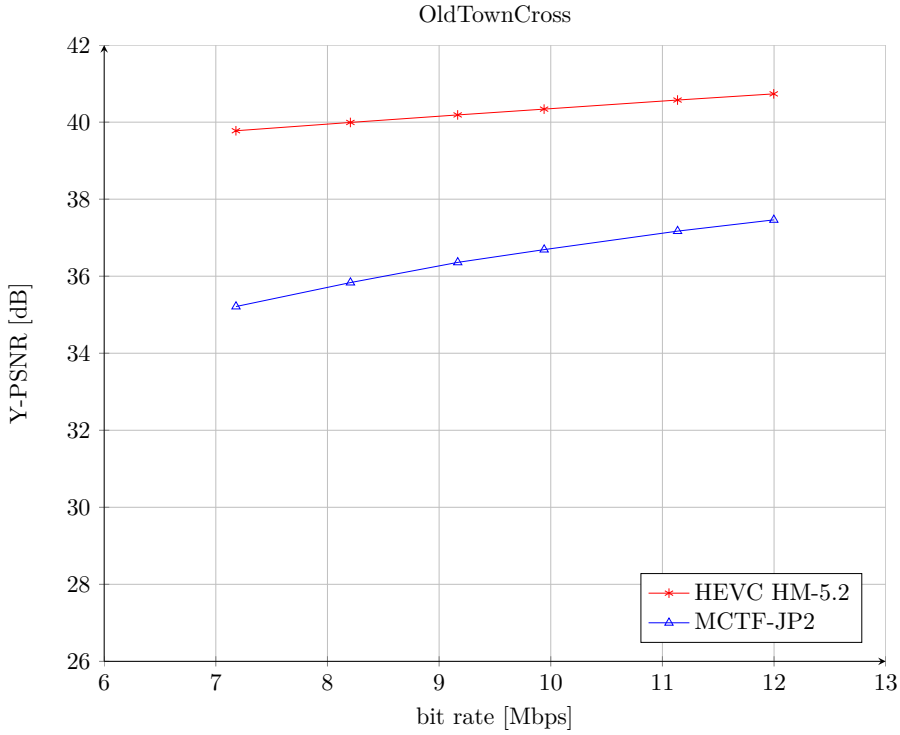


Figure G.4: Y-PSNR results for the OldTownCross sequence

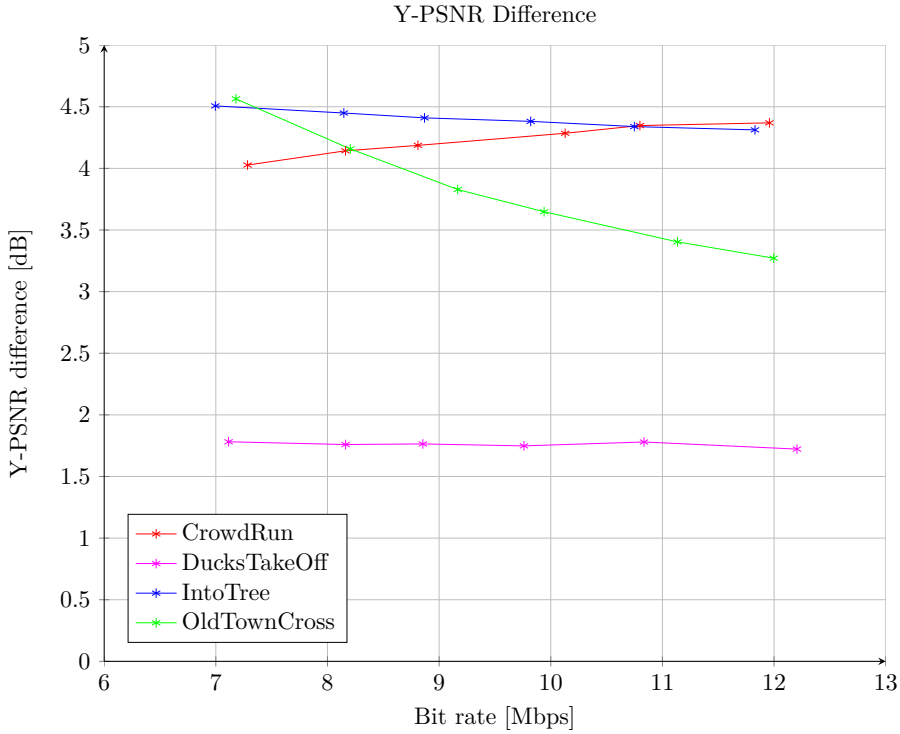


Figure G.5: Differences between the Y-PSNR results for HM-5.2 and MCTF-JP2

G.1.2 SSIM Results

Sequence		Bit rate (Mbps)					
		7	8	9	10	11	12
CrowdRun	HM-5.2	0.8993	0.9093	0.9149	0.9252	0.9300	0.9361
	MCTF-JP2	0.7819	0.7966	0.8070	0.8260	0.8349	0.8491
	Difference	0.1174	0.1127	0.1079	0.0992	0.0951	0.0870
DucksTakeOff	HM-5.2	0.8809	0.8915	0.8972	0.9045	0.9116	0.9199
	MCTF-JP2	0.8294	0.8447	0.8531	0.8626	0.8725	0.8838
	Difference	0.0515	0.0468	0.0441	0.0419	0.0391	0.0361
IntoTree	HM-5.2	0.9295	0.9350	0.9381	0.9414	0.9449	0.9477
	MCTF-JP2	0.8250	0.8405	0.8490	0.8589	0.8675	0.8764
	Difference	0.1045	0.0945	0.0891	0.0825	0.0774	0.0713
OldTownCross	HM-5.2	0.9495	0.9511	0.9525	0.9535	0.9552	0.9563
	MCTF-JP2	0.9097	0.9170	0.9224	0.9257	0.9297	0.9320
	Difference	0.0398	0.0341	0.0301	0.0278	0.0255	0.0243

Table G.2: Y-SSIM results

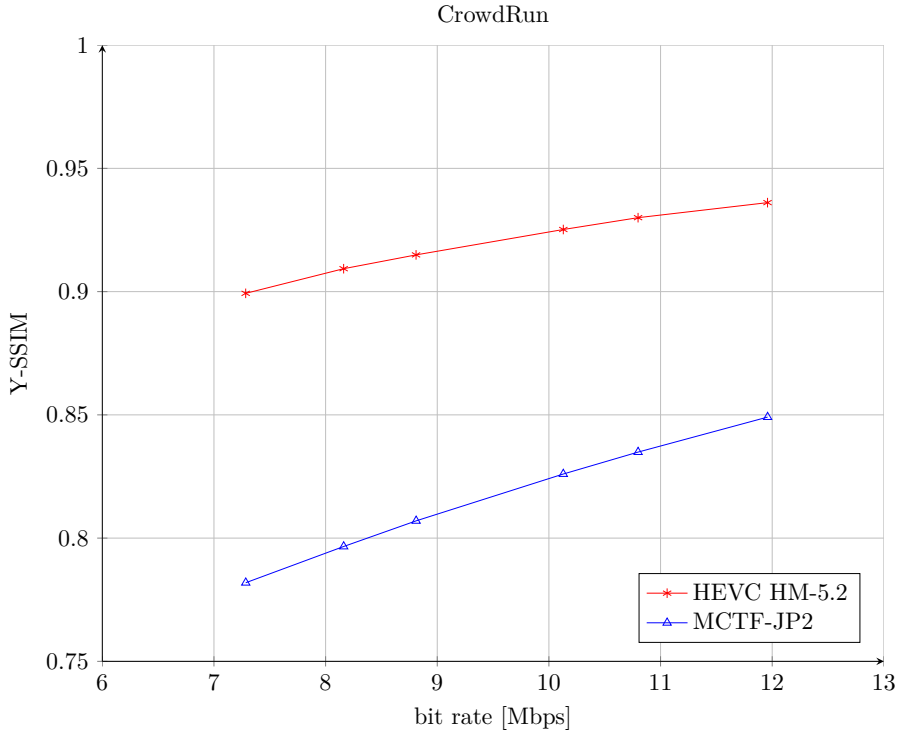


Figure G.6: Y-SSIM results for the CrowdRun sequence

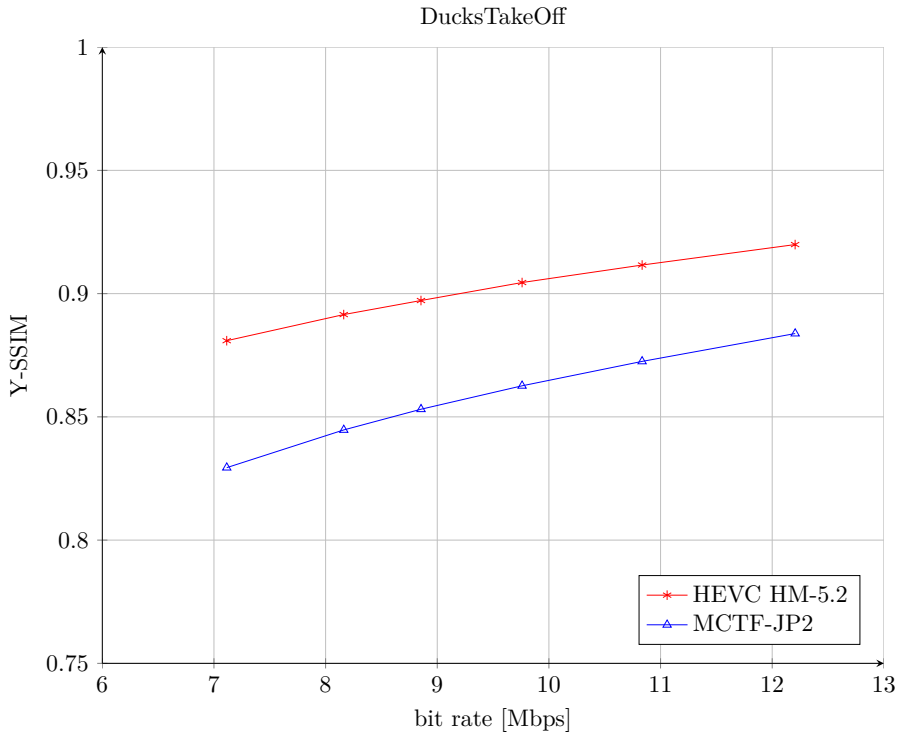


Figure G.7: Y-SSIM results for the DucksTakeOff sequence

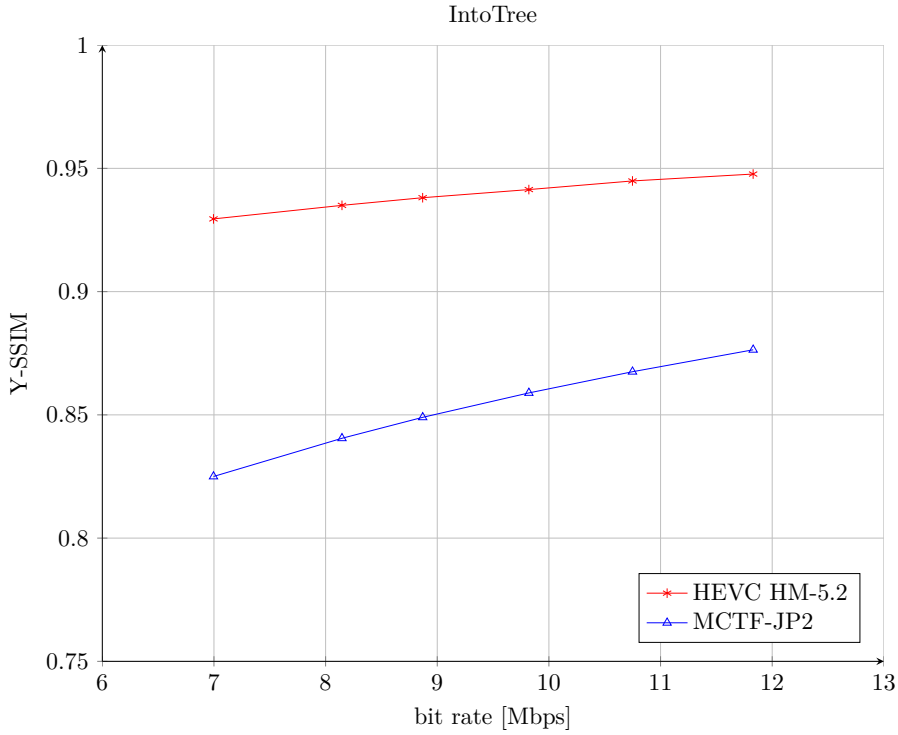


Figure G.8: Y-SSIM results for the IntoTree sequence

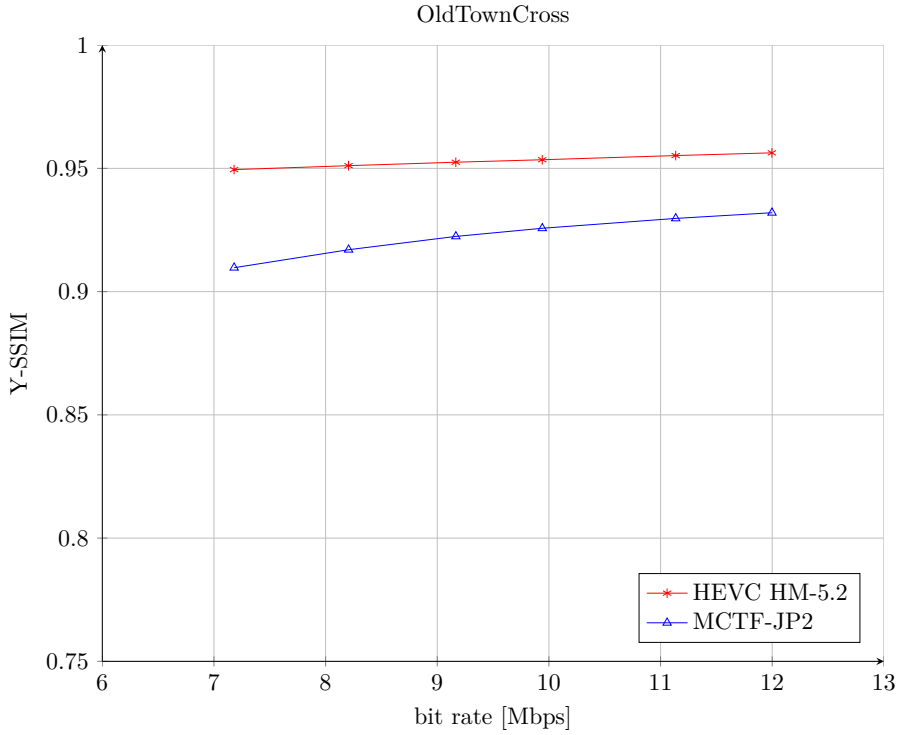


Figure G.9: Y-SSIM results for the OldTownCross sequence

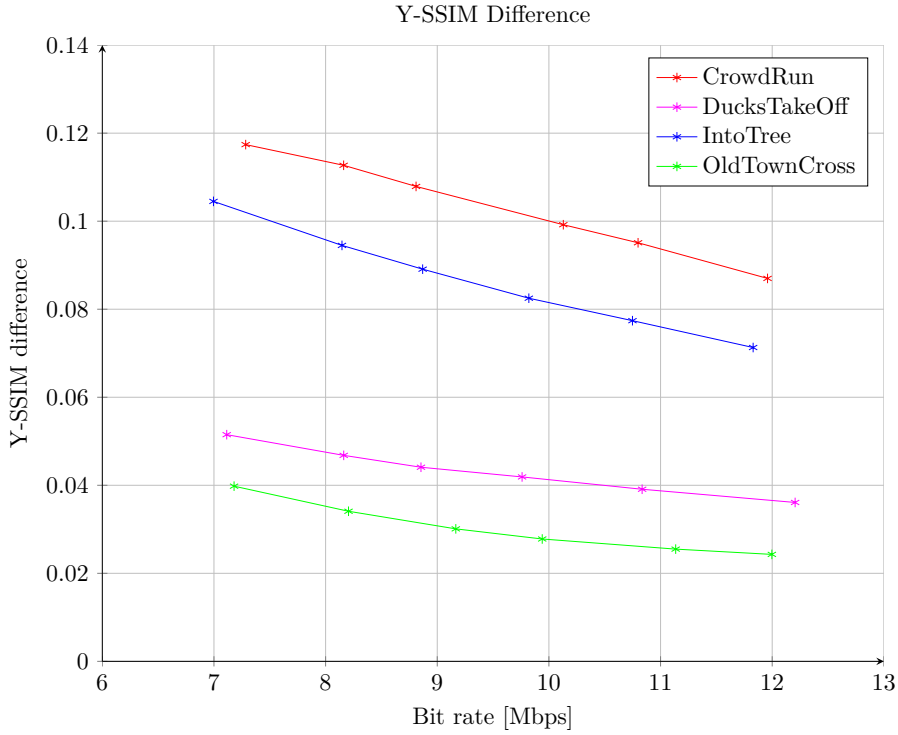


Figure G.10: Differences between the Y-SSIM results for HM-5.2 and MCTF-JP2

G.2 Subjective Results

Test Condition	AB mean	BA mean	Total Mean	95% Confidence interval
1	-2.44	-1.72	-2.08	$[-2.41, -1.76]$
2	-2.22	-1.44	-1.83	$[-2.16, -1.51]$
3	-2.22	-1.83	-2.03	$[-2.30, -1.75]$
4	-1.61	-1.44	-1.53	$[-1.80, -1.25]$
5	-1.17	-1.28	-1.22	$[-1.57, -0.87]$
6	-1.00	-1.72	-1.36	$[-1.65, -1.07]$
7	-0.94	-0.78	-0.86	$[-1.11, -0.61]$

Table G.3: Subjective results for the CrowdRun sequence

Test Condition	AB mean	BA mean	Total Mean	95% Confidence interval
1	-0.28	-0.22	-0.25	$[-0.59, 0.09]$
2	-0.50	-0.17	-0.33	$[-0.72, 0.05]$
3	-0.56	-0.78	-0.67	$[-0.95, -0.39]$
4	-0.22	-0.78	-0.50	$[-0.80, -0.20]$
5	-0.06	-0.78	-0.42	$[-0.73, -0.10]$
6	0.22	-0.61	-0.19	$[-0.48, 0.10]$
7	-0.28	-0.33	-0.31	$[-0.55, -0.06]$

Table G.4: Subjective results for the DucksTakeOff sequence

Test Condition	AB mean	BA mean	Total Mean	95% Confidence interval
1	-1.50	-1.17	-1.33	$[-1.68, -0.98]$
2	-1.89	-1.00	-1.44	$[-1.77, -1.12]$
3	-1.00	-1.28	-1.14	$[-1.47, -0.81]$
4	-1.39	-0.72	-1.06	$[-1.39, -0.72]$
5	-0.83	-0.89	-0.86	$[-1.17, -0.56]$
6	-0.50	-0.22	-0.36	$[-0.58, -0.14]$
7	-0.44	-0.72	-0.58	$[-0.94, -0.22]$

Table G.5: Subjective results for the IntoTree sequence

Test Condition	AB mean	BA mean	Total Mean	95% Confidence interval
1	-1.56	-1.22	-1.39	$[-1.72, -1.06]$
2	-0.94	-1.22	-1.08	$[-1.48, -0.69]$
3	-0.78	-1.17	-0.97	$[-1.29, -0.66]$
4	-0.89	-1.17	-1.03	$[-1.37, -0.68]$
5	-1.00	-1.00	-1.00	$[-1.23, -0.77]$
6	-0.83	-0.44	-0.64	$[-0.95, -0.33]$
7	-0.94	-0.50	-0.72	$[-0.92, -0.52]$

Table G.6: Subjective results for the OldTownCross sequence

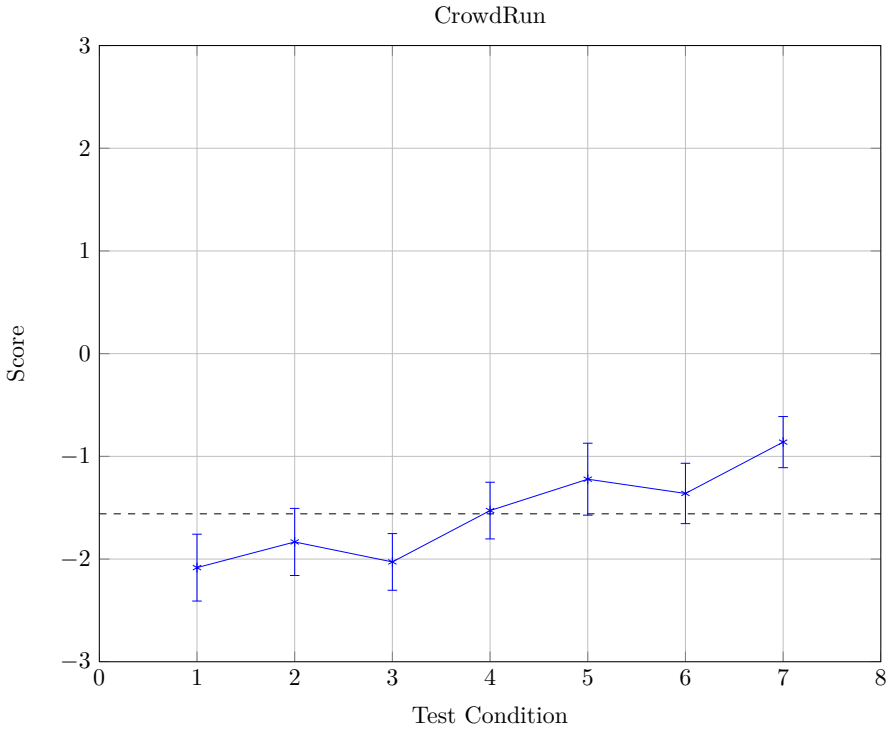


Figure G.11: 95% confidence intervals for the CrowdRun sequence. Black dashed lines shows the grand mean.

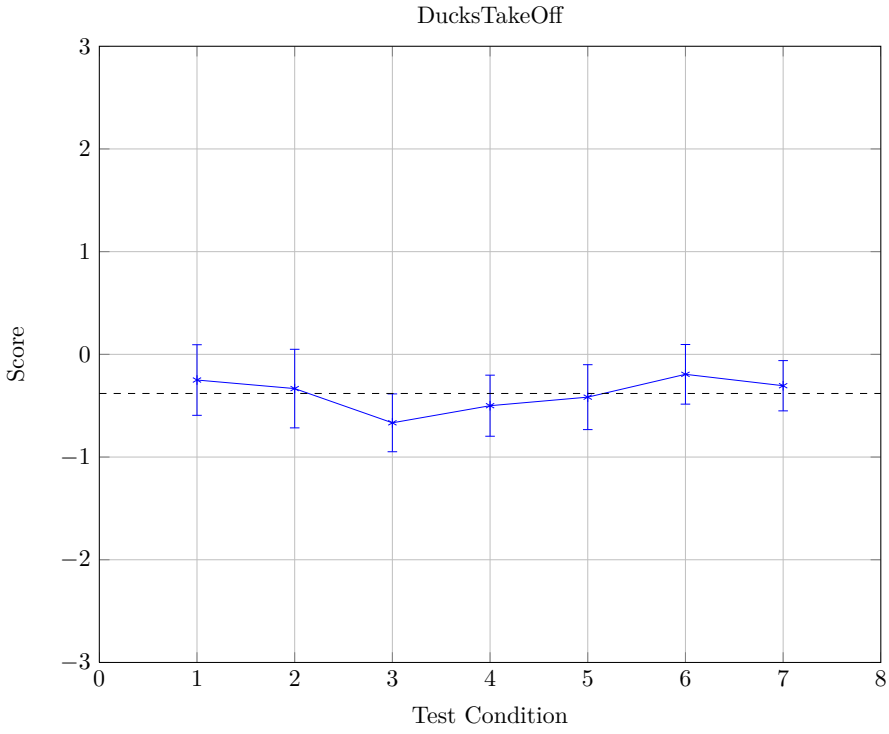


Figure G.12: 95% confidence intervals for the DucksTakeOff sequence. Black dashed lines shows the grand mean.

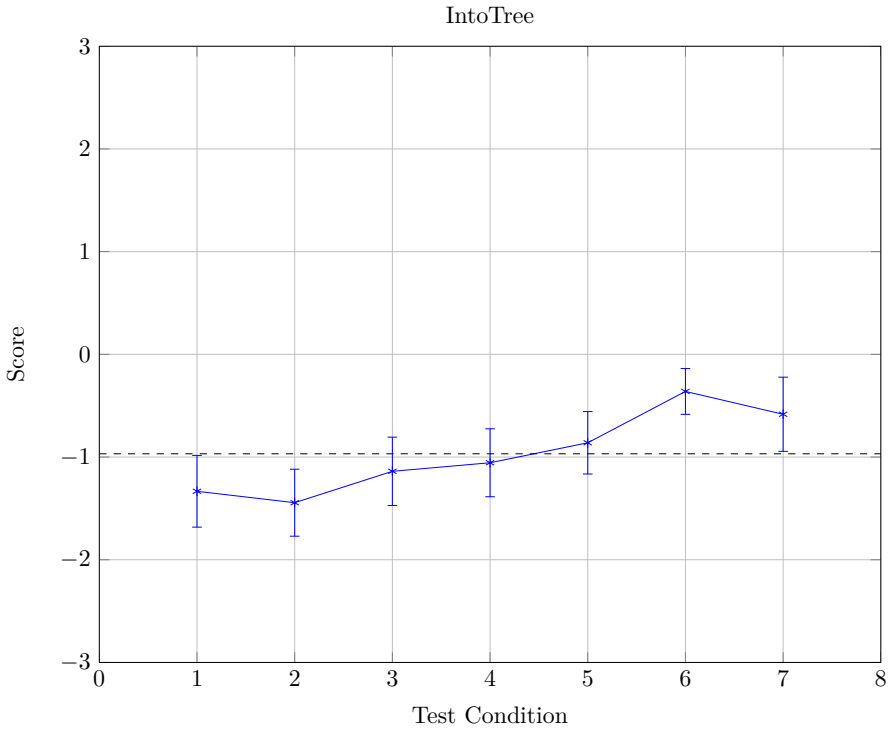


Figure G.13: 95% confidence intervals for the IntoTree sequence. Black dashed lines shows the grand mean.

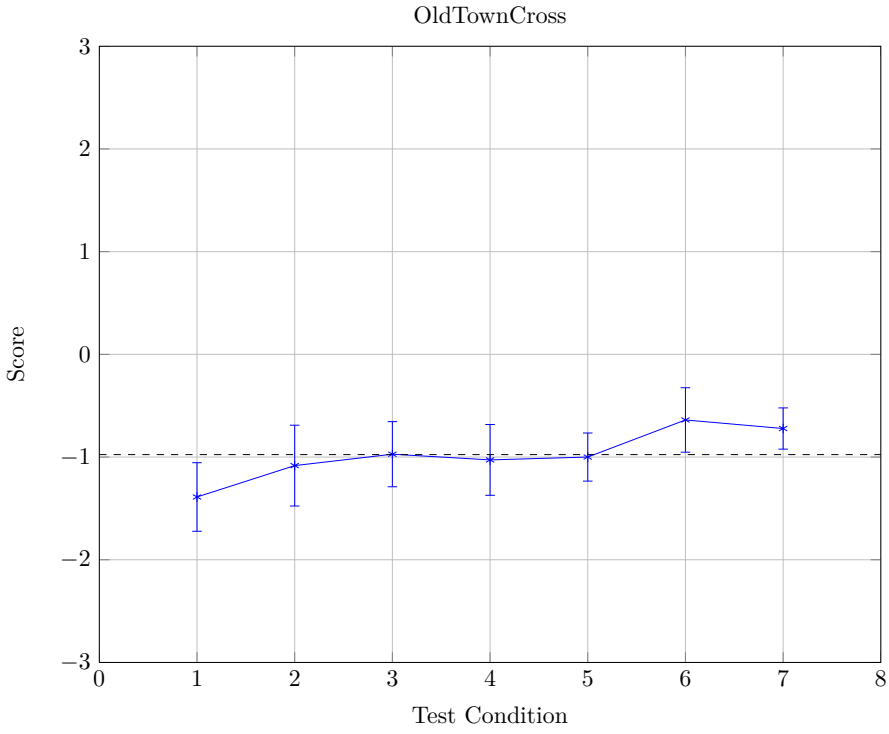


Figure G.14: 95% confidence intervals for the OldTownCross sequence. Black dashed lines shows the grand mean.

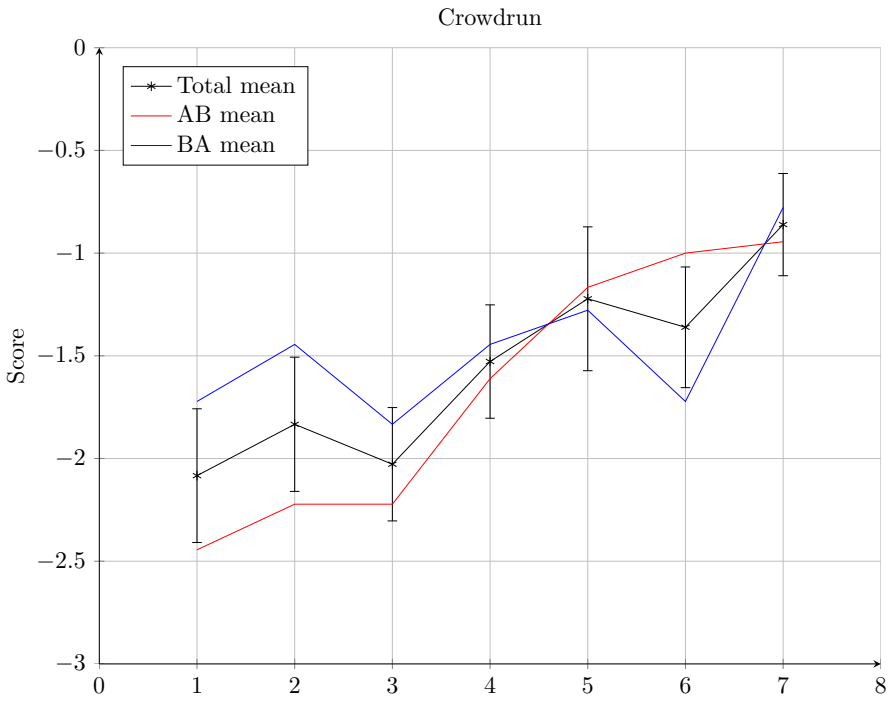


Figure G.15: Mean values for AB and BA presentations for the CrowdRun sequence. The black graph shows overall confidence intervals. Note that in this graph, the score axis range from −3 to 0.

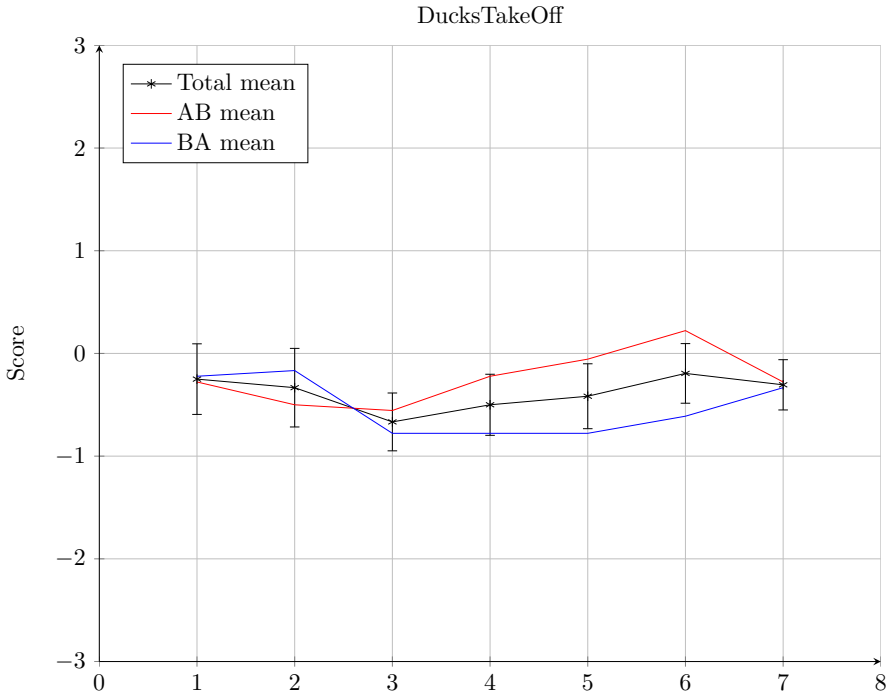


Figure G.16: Mean values for AB and BA presentations for the DucksTakeOff sequence. The black graph shows overall confidence intervals.

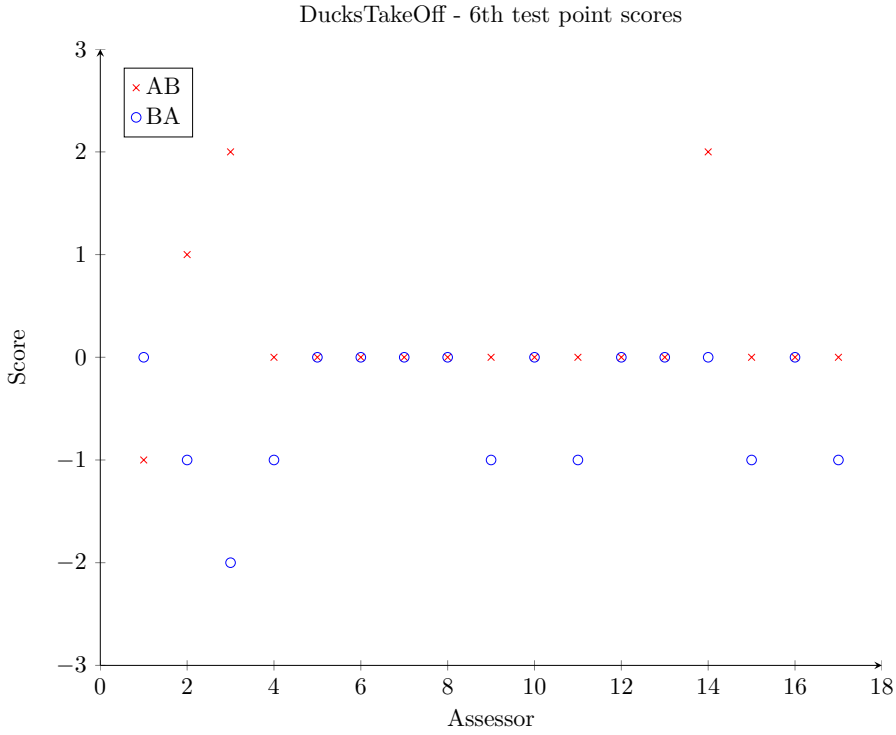


Figure G.17: Scatter plot for the sixth test condition of DucsTakeOff.

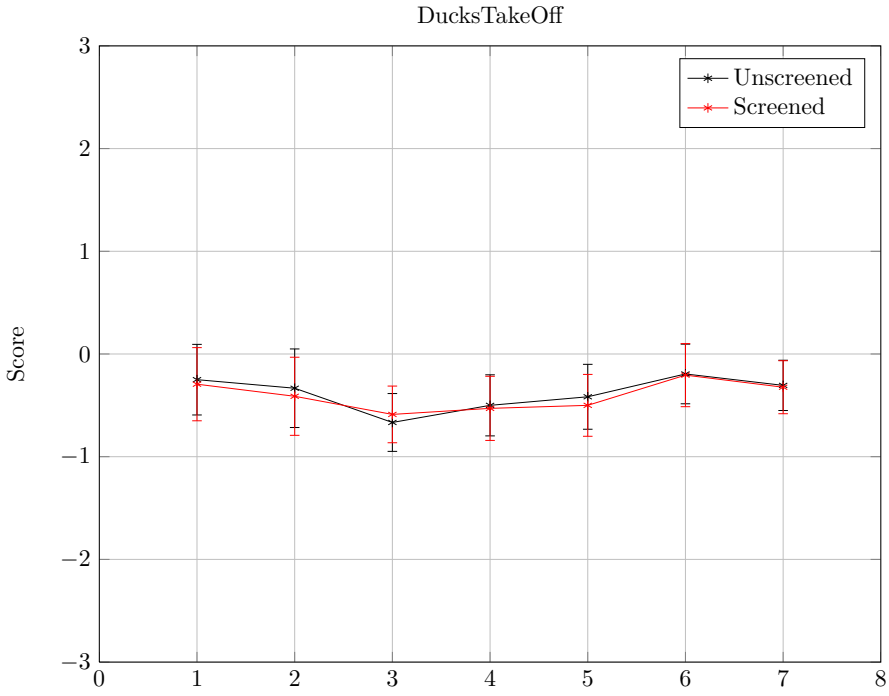


Figure G.18: Results for DucksTakeOff. The black graph shows the unscreened data, and the red graph shows the screened data, where 2 outliers has been removed.

Appendix H

Attached Zip File Content

Content of the attached zip-file:

- Matlab code for the developed MCTF-JP2 codec
- Usage description for MCTF-JP2 (pdf file)

References

[1] Cisco VNI Mobile, "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2011–2016," White Paper, February 2012, Accessed on May 9, 2012. [Online]. Available: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.html

[2] Cisco VNI, "Cisco Visual Networking Index: Forecast and Methodology, 2010-2015," White Paper. [Online]. Available: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360_ns827_Networking_Solutions_White_Paper.html

[3] "Requirements of the scalable enhancement of HEVC," Document N12783, ISO/IEC JTC1/SC29/WG11 Coding of Moving Pictures and Audio, May 2012, Accessed on May 25, 2012. [Online]. Available: http://mpeg.chiariglione.org/working_documents/mpeg-h/hevc/HEVC_ext_Reqs.zip

[4] J.-R. Ohm, *Multimedia Communication Technology: Representation, Transmission and Identification of Multimedia Signals*, 1st ed. Springer-Verlag, 2004.

[5] G. J. Sullivan and J.-R. Ohm, "Recent developments in standardization of high efficiency video coding (HEVC)," in *SPIE Applications of Digital Image Processing XXXIII*, ser. Proceedings of SPIE, A. G. Tescher, Ed., vol. 7798, no. 7798-30, August 2010.

[6] Øystein Ødegaard Auli, "Wavelet based Video coding with optical flow for motion compensation," Master's thesis, Norwegian University of Science and Technology, June 2011. [Online]. Available: http://daim.idi.ntnu.no/masteroppgave?id=6211

[7] J. D. Gibson, T. Berger, T. Lookabaugh, D. Lindbergh, and R. L. Baker, *Digital Compression for Multimedia: Principles and Standards*, 1st ed. Morgan Kaufmann Publishers, 1998.

[8] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC Video Coding Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560 – 576, July 2003.

[9] S. Zhu and K.-K. Ma, "A New Diamond Search Algorithm for Fast Block-Matching Motion Estimation," *IEEE Transactions on Image Processing*, vol. 9, no. 2, pp. 287 – 290, February 2000.

[10] ITU-T WP 3/16 and ISO/IEC JTC 1/SC 29/WG 11, "HEVC reaches ISO/IEC committee draft status," Press Release, February 2012, Accessed on February 23, 2012. [Online]. Available: http://www.itu.int/en/ITU-T/studygroups/com16/video/Pages/pr201202.aspx

[11] "MPEG-H Context and Objectives," Document w12312, ISO/IEC JTC1/SC29/WG11 Coding of Moving Pictures and Audio, December 2011, Accessed on February 23, 2012. [Online]. Available: http://wg11.sc29.org/

[12] K. McCann, B. Bross, I.-K. Kim, S. ichi Sekiguchi, and W.-J. Han, *HM5: High Efficiency Video Coding (HEVC) Test Model 5 Encoder Description*, Document JCTVC-G1102, Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, November 2011, Accessed on February 23, 2012. [Online]. Available: http://phenix.it-sudparis.eu/jct/index.php

[13] G. Sullivan and J.-R. Ohm, "Meeting report of the seventh meeting of the Joint Collaborative Team on Video Coding (JCT-VC), Geneva, CH, 21–30 Nov. 2011," Document JCTVC-G1100, Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, November 2011, Accessed on April 7, 2012. [Online]. Available: http://phenix.it-sudparis.eu/jct/index.php

[14] P. Eder, D. Engel, and A. Uhl, "JPEG2000-based Scalable Video Coding with MCTF," Department of Computer Sciences, University of Salzburg, Austria, Tech. Rep. 2007-04, October 2007, Accessed on February 19, 2012. [Online]. Available: http://www.cosy.sbg.ac.at/research/tr/2007-04_Eder_Engel_Uhl.pdf

[15] Y. Wu, K. Hanke, T. Rusert, and J. W. Woods, "Enhanced MC-EZBC Scalable Video Coder," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 10, pp. 1432–1436, October 2008.

[16] J.-R. Ohm, "Three-Dimensional Subband Coding with Motion Compensation," *IEEE Transactions on Image Processing*, vol. 3, no. 5, pp. 559–571, September 1994.

[17] G. Baruffa, P. Micanti, and F. Frescura, "Performance Assessment of JPEG 2000 Based MCTF and H.264 FRExt for Digital Cinema Compression," in *16th International Conference on Digital Signal Processing 2009*, July 2009, pp. 1–5.

[18] C.-C. Lin, Y.-T. Hwang, K.-H. Tseng, and S.-W. Chen, "Wavelet Based Lossless Video Compression Using Motion Compensated Temporal Filtering," in *IEEE Workshop on Signal Processing Systems 2007*, October 2007, pp. 686–691.

[19] A. Gao, N. Canagarajah, and D. Bull, "Macroblock-Level Mode Based Adaptive in-Band Motion Compensated Temporal Filtering," in *IEEE International Conference on Image Processing 2006*, October 2006, pp. 3165–3168.

[20] D. S. Taubman and M. W. Marcellin, *JPEG2000: Image Compression Fundamentals, Standards and Practice.* Springer Science+Business Media, 2002.

[21] A. Skodras, C. Christopoulos, and T. Ebrahimi, "The JPEG 2000 Still Image Compression Standard," *IEEE Signal Processing Magazine*, vol. 18, no. 5, pp. 36–58, September 2001.

[22] M. D. Adams, "The JPEG-2000 Still Image Compression Standard," Bundled with JasPer JPEG 2000 reference software, December 2005, Accessed on March 9, 2012. [Online]. Available: http://www.ece.uvic.ca/~frodo/jasper/

[23] Statistics norway, "The internet survey," December 2011, Accessed on March 6, 2012. [Online]. Available: http://www.ssb.no/english/subjects/10/03/inet_en/

[24] L. Haglund, "The SVT High Definition Multi Format Test Set," February 2006, Accessed on March 6, 2012. [Online]. Available: ftp://vqeg.its.bldrdoc.gov/HDTV/SVT_MultiFormat/

[25] T. Oelbaum, "Videotools," Accessed on March 6, 2012. [Online]. Available: http://www2.ldv.ei.tum.de/Members/tobias/Members/tobias/videotools

[26] "High Efficiency Video Coding test model HM-5.2." [Online]. Available: http://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-5.2

[27] F. Bossen, *Common test conditions and software reference configurations*, Document JCTVC-G1200, Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, November 2011, Accessed on February 23, 2012. [Online]. Available: http://phenix.it-sudparis.eu/jct/index.php

[28] K. Skretting, "Arithmetic Coding and Huffman Coding in MATLAB," Arithmetic Coding Software, Accessed on March 8, 2012. [Online]. Available: http://www.ux.uis.no/~karlsk/proj99/index.html

[29] S. Eireiner, "RLE de/encoding," Run-length Coding Software, Accessed on March 15, 2012. [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/4955-rle-deencoding

[30] M. D. Adams, "JasPer 1.900.1," JPEG 2000 Reference Software, Accessed on March 9, 2012. [Online]. Available: http://www.ece.uvic.ca/~frodo/jasper/

[31] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, April 2004.

[32] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multi-scale structural similarity for image quality assessment," in *Conference Record of the Thirty-Seventh Asilomar*

Conference on Signals, Systems and Computers, 2003, vol. 2, November 2003, pp. 1398 – 1402.

[33] M. Gaubatz, "MeTriX MuX Visual Quality Assessment Package," Visual quality metrics software, Accessed on March 8, 2012. [Online]. Available: http://foulard.ece.cornell.edu/gaubatz/metrix_mux/

[34] ITU-R, "Methodology for the subjective assessment of the quality of television pictures," International Telecommunication Union, Reccommendation ITU-R BT.500-12, 2009.

[35] ——, "Subjective assessment methods for image quality in high-definition television," International Telecommunication Union, Reccommendation ITU-R BT.710-4, 1998.

[36] ——, "Specifications and alignment procedures for setting of brightness and contrast of displays," International Telecommunication Union, Reccommendation ITU-R BT.814-2, 2007.