NTNU
Norwegian University of
Science and Technology

Wavelet based Video coding with optical flow for motion compensation

Øystein Ødegaard Auli

Master of Science in Electronics

Submission date:	June 2011
Supervisor:	Andrew Perkis, IET
Co-supervisor:	Helge Stephansen, T-Vips AS

Norwegian University of Science and Technology
Department of Electronics and Telecommunications

Problem description

For high quality video compression in contribution environments, intra-only encoding with JPEG 2000 has been preferred by many. However, this solution does not exploit the temporal redundancy in a sequence of images, and therefore limits the possibility of further reduction of the bit rate.

There are several ways of estimating the motion in an image sequence. One of them is calculating the optical flow. By calculating the optical flow, each pixel will get its own vector that represents its motion, resulting in a vector field for the entire image.

The task is to explore the performance of a hybrid video codec that uses optical flow for motion estimation and JPEG 2000 to encode the estimation error. Its performance shall be assessed by considering the quality compared to a chosen baseline codec.

Assignment given: 13. January 2011
Supervisor: Andrew Perkis, IET
Co-Supervisor: Helge Stephansen, T-Vips AS

Preface

This Master's thesis was carried out at the Norwegian University of Science and Technology (NTNU), Faculty of Information, Mathematics and Electrical Engineering (IME), Department of Electronics and Telecommunication (IET) during the spring of 2011.

I would like to thank my supervisor Professor Andrew Perkis at the Centre for Quantifiable Quality of Service in Communication Systems (Q2S), my co-supervisor Helge Stephansen at T-Vips AS and Professor Touradj Ebrahimi for guidance and advice throughout this thesis. I would also like to extend my gratitude to Morten Flå and Ida Onshus for proofreading this thesis.

A thanks goes out to classmates, friends and Omega Loftet for the social aspect this spring.

Øystein Ødegaard Auli
Trondheim, June 13, 2011

Abstract

In broadcasting environments intra-only video coding with JPEG 2000 has shown to provide many desired features along with high picture quality. However, there is no exploitation of temporal redundancy, which can reduce the bit rate while maintaining the quality.

Optical flow algorithms are designed to find the apparent movement of brightness in an image, and can be used to estimate the motion of each pixel between consecutive images. This thesis explores the performance of a hybrid video codec that uses the 'Classic+NL' optical flow algorithm[1] for motion compensation and JPEG 2000 for encoding the estimation error.

The motion estimation proved to be inaccurate at the edges of objects. This can cause high frequency components in the residue image, which will decrease the efficiency of JPEG 2000. Occluded regions will also have poor estimation, as they are not present in the previous frame. Since noise is not considered when the optical flow is calculated, the energy of the noise may increase after motion compensation. Without addressing these issues, optical flow algorithms are not well suited for motion estimation in hybrid video codecs.

Even with the inaccurate motion compensation performed by the optical flow algorithm, there was an overall reduction in bit rate of 18.8%, compared to intra-only coding with JPEG 2000. The performance was highly content dependent, ranging from a reduction of 90% to an increase of 27%. The reduction comes at a cost of increased delay, higher complexity, vulerability to transmission errors, and a lack of a constant bit stream. The proposed hybrid codec is therefore not suited to replace intra-only coding with JPEG 2000 in contribution environments.

Contents

List of Figures

List of Tables

List of abbreviations

DCT	Discrete cosine transform
DWT	Discrete wavelet transform
EBCOT	Embedded Block Coding With Optimal Truncation
EBU	European Broadcasting Union
GOP	Group of pictures
H.264/AVC	H.264/MPEG-4 Advanced Video Coding
HD	High definition
HEVC	High Efficiency Video Coding
HS	Horn-Schunck's method
HVC	Hybrid Video Codec
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
JPEG	Image compression standard
JPEG	Joint Photographic Experts Group (organization)
JPEG 2000	Image compression standard
MCTF	Motion Compensated Temporal Filtering
MPEG	Moving Picture Experts Group
MSE	Mean square error
OBMC	Overlapped block motion compensation
OF	Optical flow
PRCD	Post Compression Rate Distortion

PSNR	Peak signal to noise ratio
SSIM	structural similarity
STD	Standard deviation
SVC	Scalable Video Coding
VBSMC	Variable block-size motion compensation

Chapter 1

Introduction

Since the television became commercially available in the late 1920s, there has been an enormous development in the broadcasting industry. From the first black and white television sets, to todays high definition flat screens, there has been a huge improvement in picture quality. Not only did the quality improve, but also the quantity; in 2008 the average US household received 118.6 TV channels [2]. The increase in quality and quantity had not been possible without the introduction of digital compression of video.

With digital compression of video and audio it is possible to reduce the bandwidth requirements without significantly reducing the perceived quality. In satellite distribution, a single transponder was used to carry one analog TV channel. However, it does not have the capacity to carry a single uncompressed digital channel. By employing state-of-the-art lossy video compression, a transponder can now carry 26 standard definition channels, or 6 high definition channels [3]. Even though lossy compression is used, the perceived quality is still higher than for analog channels. When analog signals are transmitted, their quality is reduced by the presence of noise. Analog noise reduction systems exist, but perfect reconstruction of the original signal is not possible. However, this is possible with digital transmission, and the quality experienced by TV viewers can be decided by the broadcasters.

While digital video not necessarily suffer from quality loss by transmission, compression errors will be introduced each time the video is lossy encoded. For every lossy compression, there will be a further reduction in quality, without necessarily a reduction of the bandwidth requirements. In most cases the bandwidth capacity is limited from the TV distributor to the customer, resulting in high compression. Therefore it is important that the quality of the video is as high as possible before the last encoding.

In the TV studio it is desired to maintain the highest possible quality of the signal at each processing step. If possible, the signal should be kept uncompressed. However, for high definition content, the bandwidth requirements for uncompressed video are extremely high and it may be necessary to use compression. The image compression standard JPEG 2000 offers both lossy and lossless compression, and has the possibility of greatly reducing the bit rate of a video. Even though JPEG 2000 uses advanced coding techniques, it is outperformed at lower bit rates by video coding standards that exploit the temporal correlation in a sequence of images.

By estimating the motion from one image to another, it is possible to reduce the bit rate of a video, without reducing the quality. In [4], Gary Demos proposed a video codec that uses wavelet transformations for encoding of the estimation error. The encoder estimates a motion vector for each pixel from one frame to the next, with the collection of these vectors resulting in a 'flowfield'. An investigation of the work by Gary Demos was wanted, to see how it could be applied in a contribution scenario. However, this proved to be a challenge since the article does not provide enough details to recreate the encoder.

Normally, motion vectors are not selected to predict a correctly appearing image. The vectors are selected to minimize the overall bit rate, without regard to the actual motion in the sequence. The term 'flowfield' is more accurate if the actual flow of the pixels is described by the vector field. To estimate the motion between two images, techniques that calculate the optical flow can be applied. *"Optical flow is the distribution of apparent velocities of movement of brightness patterns in an image." [5]* Thus, a vector field calculated with an optical flow algorithm, could represent the apparent motion between two images.

By using an optical flow algorithm for motion estimation and JPEG 2000 for encoding of the estimation error, it could be possible to further reduce the bit rate. This thesis will therefore explore the performance of a hybrid video codec that is built on this principle.

The thesis is organized as follows: Chapter 2 presents relevant information about optical flow, JPEG 2000 and hybrid video coding. It also includes a short overview of other types of wavelet based video coding. In Chapter 3 the methods used in this thesis are presented, which includes how the proposed hybrid codec was implemented and tested. It also gives an overview of how the results were analyzed. Chapter 4 presents the results of the hybrid codecs performance for different settings and approaches. A more general discussion of how optical flow performs for motion compensation in combination with JPEG 2000 can be found in Chapter 5. The thesis is concluded in Chapter 6, which also includes proposed future work.

Chapter 2

Theory

In the following sections theory on optical flow, JPEG 2000, hybrid video coding and wavelet video coding is presented.

2.1 Optical flow

Optical flow (OF) is defined as the apparent movement of brightness in a visual scene. Since the relative motion between objects and the viewer is a cause for the movement of brightness, it would be easy to say that optical flow represents the velocity of objects [5]. Although this may be true under certain circumstances, there are many exceptions. A moving object may for instance cause a constant brightness pattern, or the brightness pattern may change even though no objects move in the visual scene.

The first work on optical flow was done as early as in 1950 by Gibson [6]. In 1981 both Horn/Schunck and Lucas/Kanade published new approaches to finding the optical flow [5, 7]. After these significant contributions, many new algorithms where introduced, and a large amount was based on the work published in 1981. These algorithms are described as differential methods, since they are based on partial derivatives to find the flow.

2.1.1 Horn and Schunck's method

In the field of optical flow algorithms Horn and Schunck's differential method (HS) has been the basis for much work. The fundamentals of this approach are presented in the following section, as they will give a basic understanding of today's state of the art algorithms. For a more comprehensive explanation, refer to the original article by Horn and Schunck [5].

Some restrictions regarding the content of an image is made: The image is flat to avoid shading problems, the incident illumination is uniform and the reflectance of objects is constant and does not have spatial discontinuities. It is also assumed that no objects occlude one another. With these restrictions it can be said that the change of brightness between images arises directly from the motion of corresponding points.

The image brightness at point (x, y) at time t is denoted by $E(x, y, t)$. If assumed that the brightness of the point is constant, and has the ability to move, one has the following equality:

$$E(x, y, t) = E(x + \partial x, y + \partial y, t + \partial t) \tag{2.1}$$

Where ∂x and ∂y is the displacement along the x-axis and y-axis respectively, and ∂t is the displacement in in the temporal domain. The right side of the equation can then be expanded

$$E(x + \partial x, y + \partial y, t + \partial t) = E(x, y, t) + \partial x \frac{\partial E}{\partial x} + \partial y \frac{\partial E}{\partial y} + \partial t \frac{\partial E}{\partial t} + H.O.T. \tag{2.2}$$

where $H.O.T.$ represents the higher order terms in the expansion. By combining (2.1) and (2.2), subtracting $E(x, y, t)$ from both sides of the equation, and then dividing by ∂t, the following remains:

$$\frac{\partial x}{\partial t}\frac{\partial E}{\partial x} + \frac{\partial y}{\partial t}\frac{\partial E}{\partial y} + \frac{\partial E}{\partial t} + O(\partial t) = 0 \tag{2.3}$$

$O(\partial t)$ is a term of order ∂t. With the limit $\partial t \to 0$ the equation becomes:

$$\frac{\partial E}{\partial x}\frac{dx}{dt} + \frac{\partial E}{\partial y}\frac{dy}{dt} + \frac{\partial E}{\partial t} = 0 \tag{2.4}$$

Let

$$u = \frac{\partial x}{\partial t} \quad v = \frac{\partial y}{\partial t} \tag{2.5}$$

and

$$E_x = \frac{\partial E}{\partial x} \quad E_y = \frac{\partial E}{\partial y} \quad E_t = \frac{\partial E}{\partial t} \tag{2.6}$$

When replacing the terms in (2.4) one has:

$$E_x u + E_y v + E_t = 0 \tag{2.7}$$

or

$$(E_x, E_y) \cdot (u, v) = -E_t \tag{2.8}$$

Equation (2.8) expresses the constraint on the local flow velocity. Since this is an ill posed problem, it cannot be solved directly, and additional constraints are needed.

An important factor not considered in the previous equations is that there is a correlation in velocitys for adjacent pixels. For example, if an object is moving in the image, all pixels corresponding to the object will have the same or similar velocity. The velocity will in fact vary smoothly across the field, and discontinuities will reflect areas where one object occludes another. This smoothness can be added as an extra constraint in the algorithm, for example by minimizing the following expression:

$$\varepsilon_c^2 = \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 \tag{2.9}$$

As a result of this smoothness constraint, it is that likely the algorithm will have difficulties with occluding edges. Also the first constraint in equation (2.7) need to be expressed as a minimization problem:

$$\varepsilon_b = E_x u + E_y v + E_t \tag{2.10}$$

Combining these two minimization problems, the total error to be minimized is

$$\varepsilon^2 = \int\int (\alpha^2 \varepsilon_c^2 + \varepsilon_b^2) dxdy. \tag{2.11}$$

Because the brightness in the image is expected to have quantization noise, ε_b can not be expected to be zero. The weighing factor α^2 adjusts the strength of the smoothness. By minimizing this expression with an appropriate weighing factor it is possible to find a fairly accurate vector field representing the optical flow between the images. For more information about the weighting factor and a practical solution to minimize the expression, refer to the original paper by Horn and Schunck [5].

2.1.2 Principles and practices

The assumptions about the image content in Horn and Schunck's method limits the accuracy of the algorithm when used on natural images. In this section, some principles and practices that improve the calculation of the optical flow are presented.

Penalty function When minimizing error HS minimizes the square of ϵ_b and ϵ_c. This is referred to as a square penalty function, $p(x) = x^2$. Other penalty functions can be employed as well: For example the Charbonnier function $p(x) = \sqrt{x^2 + \epsilon^2}$ [8] or the Lorentzian function $p(x) = \log(1 + \frac{x^2}{2\sigma^2})$ [9].

Preprocessing: While the HS method assumes constant illumination of the images, changes in lighting can affect the calculation of the optical flow. Preprocessing of the images can therefore provide robustness against lighting changes [10].

Coarse-to-fine estimation: When there are large displacements, the calculation of the optical flow can be improved by using coarse-to-fine estimation techniques [1, 8]. The optimization involves downsampling of the image, often in several steps, and calculation of the optical flow for the lowest resolution. The estimation of the optical flow at the lowest resolution can be applied to correct the vector field at a higher resolution, a process referred to as warping. This process is then repeated at increasing resolutions.

Median filtering: To increase the robustness to sampling artifacts, median filtering of the vectors can be performed between warping steps. This process has shown to increase the accuracy of the optical flow significantly [1, 10].

2.1.3 Optical flow in video compression

Optical flow has already been proposed for use in video coding. For motion estimation, the algorithm has been proposed in several codecs, especially for low bit rates [11, 12, 13].

More recently, there has been a suggestion to employ optical flow algorithms to increase the performance of the new High Efficiency Video Coding (HEVC) standard [14]. Even though this method reduced the bit rate, it increased the decoding time substantially.

2.2 JPEG 2000

With the increasing use of the Internet and multimedia in the mid 90s is was clear that an image compression standard with new features and improved performance was desired. In March 1997 International Organization for Standardization (ISO)/International Electrotechnical Commission(IEC) issued the first call for proposal for their work on a new image compression standard that would follow the very successful JPEG standard. In 2000 a Discrete Wavelet Transform (DWT) based compression standard was complete (JPEG 2000), and it provided many features not available in previous standards. In the following sections only Part 1 of the JPEG 2000 standard will be discussed.

2.2.1 Features

Some of the more notable features in the JPEG 2000 standard are listed in this section. For more information about these features refer to [15] and [16].

Superior low bit-rate performance: Even though JPEG 2000 has a compression advantage over JPEG at high bit-rates, it is especially at low bit-rates (0.25 bit per pixel and below) where JPEG may suffer from severe blocking artifacts, that JPEG 2000 excels.

Lossless and lossy compression: When using the reversible (integer) wavelet transformation instead of the irreversible and skipping quantization of the coefficients, perfect reconstruction is possible.

Continuous-tone and bi-level compression: JPEG 2000 supports a pixel depth from 1 to 16 bit for each image component. This allows for both bi-level compression and higher color accuracy.

Tiling: An image can be split into rectangular tiles, which are encoded separately. The main advantage of this is the lower memory requirements at both the encoder and decoder.

Progressive transmission by pixel accuracy and resolution: The code stream can be organized so that the decoder can view the picture before all the information is available and thus improve the quality as more information becomes available. This can either be done by first displaying a lower resolution and later increasing it, or by improving the accuracy of the pixels.

Random code stream access and processing: JPEG 2000 has the ability to increase the quality of spatial regions in an image, and has mechanisms to allow spatial random access in the code stream.

Error resilience: The data in JPEG 2000 are stored in small independent code blocks, so that the loss of synchronization only will affect one block. This is especially helpful when transferring the image over an error prone channel.

2.2.2 Architecture overview

The basic building blocks of the JPEG 2000 encoder is shown in Figure 2.1. First the image samples are preprocessed, then a DWT is performed, followed by quantization and finally entropy coding. It is important to remember that this process is performed once for each tile. The samples can follow one of two paths; one that allows a perfect reconstruction of the samples and one that does not. These are referred to as the reversible and the irreversible path respectively.

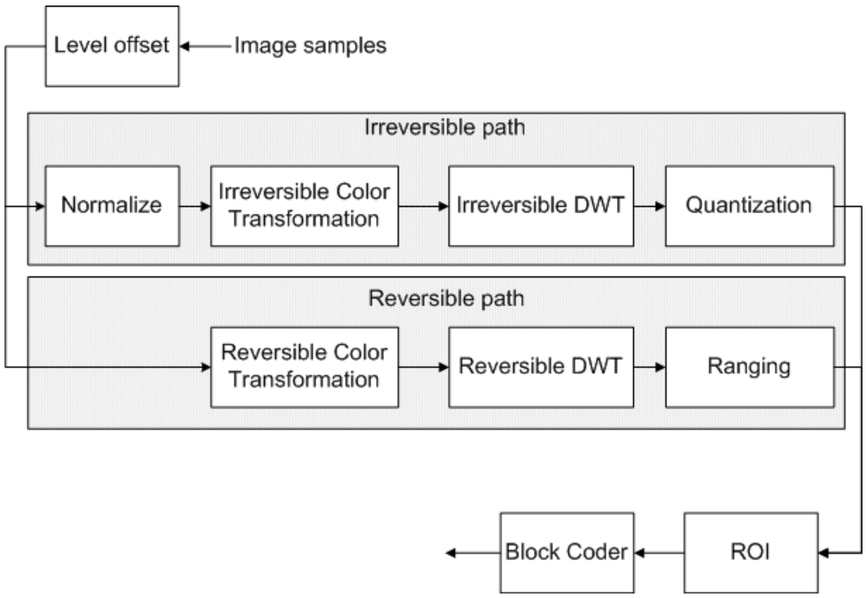


Figure 2.1: Overview of JPEG 2000 encoder.

2.2.3 Level offset and normalization

Before the wavelet transformation, the image samples are preprocessed. If the input bits are unsigned, a level offset is performed. Samples represented by B number of bits will then be in the range

$$-2^{B-1} \leq x[n] < 2^{B-1}. \tag{2.12}$$

This is done to avoid irregularities in the coefficients produced by the DWT. As shown in Figure 2.1 the level offset is performed for both the reversible and the irreversible path. After the level offset, samples following the irreversible path are normalized. This is done by dividing them by 2^B, such that the samples values are conformed to $[-\frac{1}{2}, \frac{1}{2}]$. Because of this normalization, the irreversible path is independent of the sample bit-depth. The decoder can choose the desired number of bits for the samples when decoding. Thus, a 12 bit image can be decoded as a 8 bit image, and vice versa.

2.2.4 Discrete Wavelet Transform

One essential difference between JPEG and JPEG 2000 is how the spatial decorrelation is performed. Whereas JPEG used a Discrete Cosine Transform (DCT) on an 8x8 pixel block, JPEG 2000 uses a DWT on the entire tile. While JPEG suffers from visual blocking artifacts due to its block size, JPEG 2000 does not have this problem. However, if the image is divided into tiles, similar artifacts can be seen at low bit-rates.

The DWT can be viewed as a pair of lowpass and highpass filters, followed by a decimator. As seen in Figure 2.2, the input samples are split into a highpass and a lowpass signal, and then downsampled by a factor of two. These samples are referred to as wavelet coefficients.

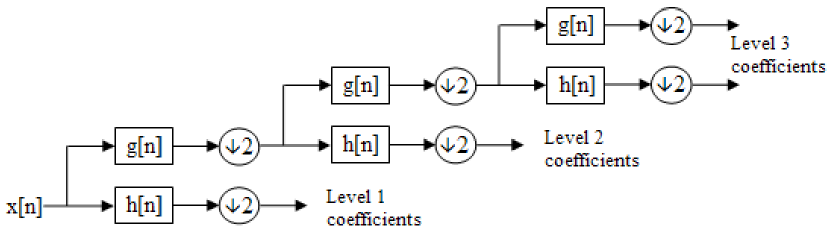


Figure 2.2: One dimensional three level wavelet transform, where h[n] is the highpass filter, and g[n] is lowpass the filter [17].

Because of the downsampling, the number of wavelet coefficients will be the same as the number of input samples. Since an image is two-dimensional it is necessary to extend the one-dimensional wavelet transformation. This is done by first performing the one-dimensional transformation on each row in the image, then performing the one-dimensional transformation vertically on the filtered, downsampled data. Since the filtering process is performed two times, the image will be divided into four set of frequency bands. These set of wavelet coefficients are referred to as subband images, or subbands.

In Figure 2.3 it can be seen how the different subbands are labeled. 1HH refers to the samples that have been highpass filtered in both the horizontal and vertical direction. These coefficients contain the high frequency parts of the image, for example sharp edges and noise. 1HL refers to the samples that have first been highpass filtered, then lowpassed filtered, while 1LH represents lowpass followed by highpass.

1LL refers to the samples that have been lowpass filtered two times, but for these coefficients there are still a lot of correleation. Therefore the process is performed again on the 1LL subband. Normally a JPEG 2000 image would use around five decomposition levels. It is important to remember that because of the downsampling, the components with high frequency at the first decomposition level (1HH , 1HL and 1LH) will contain 3/4 of the total number of wavelet coefficients, regardless of the number of decomposition levels.

In Figure 2.4 a two-level decomposition of an image is shown, with the different subbands corresponding to those in Figure 2.3. To emphasize the content in the

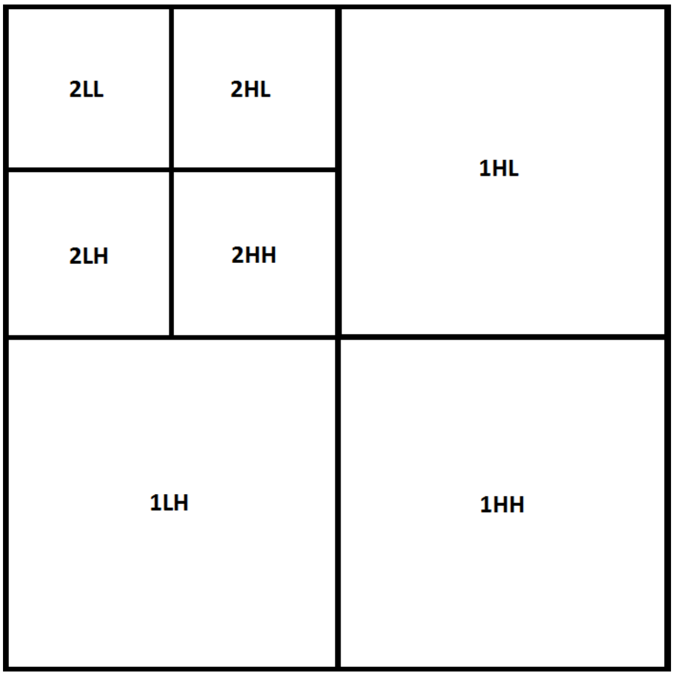


Figure 2.3: Labeling of subbands after a two-dimensional DWT.

image, the contrast has been adjusted for all the subbands (with the exception of the 2LL subband), and the samples have not been level shifted.

In JPEG 2000 Part 1, two different filter banks are available for wavelet transformation, the (9,7) irreversible floating-point filter bank and the (5,3) reversible integer filer bank. Filter taps for the analysis part of the filter banks can be found in Table 2.1 and 2.2. While the (9,7) filter bank has the advantage of giving the highest compression efficiency, the (5,3) filter bank allows for perfect reconstruction.

Table 2.1: Daubechies (9,7) analysis filter used by the irreversible DWT in JPEG 2000.

n	*Lowpass filter* $h_0(n)$	*Highpass filter* $h_1(n)$
0	0.6029490182363579	1.115087052456994
±1	0.2668641184428723	-0.5912717631142470
±2	-0.07822326652898785	-0.05754352622849957
±3	-0.01686411844287495	0.09127176311424948
±4	0.02674875741080976	

For more detailed information about these filter banks, refer to [15].

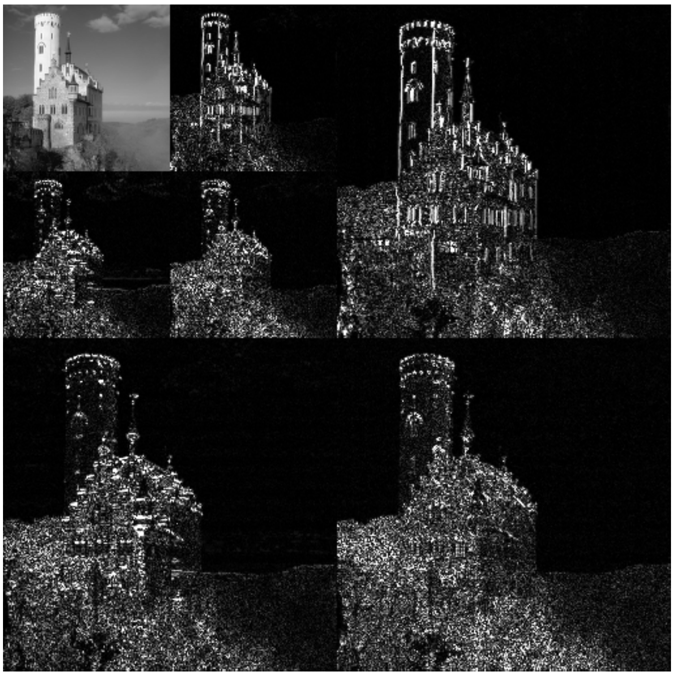


Figure 2.4: An image decomposed with a two-dimensional DWT [18]

Table 2.2: Integer (5,3) analysis filter used by the reversible DWT in JPEG 2000.

n	*Lowpass filter* $h_0(n)$	*Highpass filter* $h_1(n)$
0	6/8	1
± 1	2/6	-1/2
± 2	-1/8	

2.2.5 Quantization and entropy coding

Quantization is an irreversible process that reduces the entropy of the coefficients, and is therefore only used in the irreversible encoding path. To quantize the wavelet coefficients, JPEG 2000 employs a uniform quantizer with a central dead zone. The quantizer is shown in Figure 2.5, which displays how the quantizer maps wavelet coefficients to the appropriate quantization index. Around zero, the quantization step is $2\Delta_b$, instead of Δ_b, which is referred to as the central dead zone.

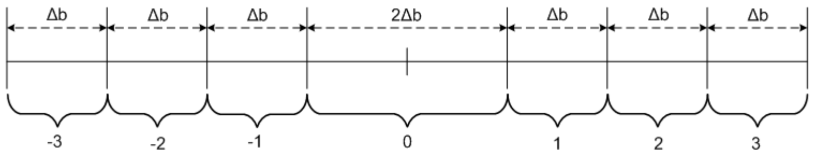


Figure 2.5: Uniform quantizer with a central dead zone.

The mathematical way to represent the quantization would be with the following equation:

$$q_b(u,v) = sign(y_b(u,v)) \left\lfloor \frac{|y_b(u,v)|}{\Delta_b} \right\rfloor \tag{2.13}$$

From both Figure 2.5 and equation (2.13) it is apparent that the parameter Δ_b decides the error introduced when quantizing. For each subband, Δb is selected and represented by two bytes, a 5-bit exponent ϵ_b and an 11-bit mantissa μ_b, according to

$$\Delta_b = 2^{R_b - \epsilon_b}(1 + \frac{\mu_b}{2^{11}}) \tag{2.14}$$

where R_b represents the number of bits in the nominal dynamic range of subband b. It is up to the encoder to select the quantization step, and the choice can be driven by a number of factors. For example the visual importance of each subband or rate control.

After the the coefficients have been quantized they are entropy coded to create the compressed bit-stream. This process is referred to as the Embedded Block Coding With Optimal Truncation (EBCOT). In JPEG 2000 each subband is divided into a rectangular group of coefficients referred to as code-blocks. These code-blocks are then bit-plane coded using arithmetic coding, starting with the most significant bit. The encoding is performed in three passes for each bit-plane, with the possibility of truncating the bit-stream after each pass. A rate-allocation process called Post Compression Rate Distortion (PRCD) is applied to achieve the desired bit-rate with the minimum amount of distortion, by truncating the appropriate code-blocks.

2.3 Hybrid video coding

Most video codecs are based on the hybrid video codec (HVC) principle. A HVC reduces the bit rate by exploiting the temporal correlation in a video, without reducing the quality. This is done by estimating the motion from image A to image B. Since the predicted image will differ from the original, the estimation error needs to be encoded as well. Figure 2.6 shows a block diagram of this process.

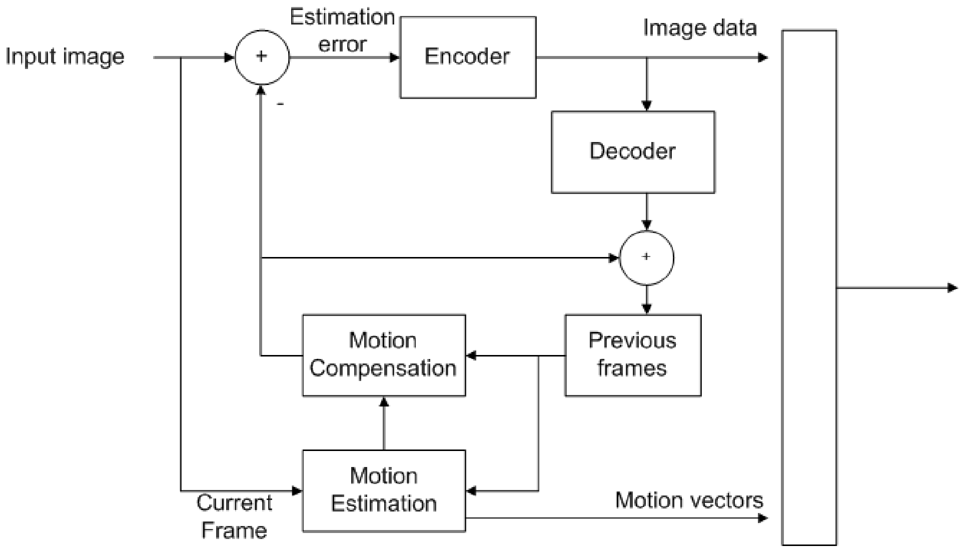


Figure 2.6: Block diagram of a generic hybrid video encoder.

If prediction of motion is satisfactory, the number of bits required to represent the estimation error is smaller than the image being predicted. This reduction in bits must outweigh the number of bits needed to represent the motion vectors, or there will be no overall reduction in bit rate.

In a HVC there are three types of frames: Intra coded frames (I-frames), predictive coded frames (P-frames), and bidirectionally predictive coded frames (B-frames). Normally, only I- and P-frames are anchor frames, which means that motion vectors point from these particular frames to others frames. The I-frame is encoded as a normal still image, completely independent of other frames, and can therefore be decoded without the presence of other frames. A P-frames contains the prediction error of a single frame, whose motion prediction is based on preceding I- and/or P-frames. Along with the motion vectors, preceding frames are therefore needed to decode a P-frame. B-frames can have vectors pointing from both preceding and following I- and/or P-frames. To limit complexity, and assure that errors do not propagate all the way through a video, motion vectors can only point between frames in a group of pictures (GOP). Figure 2.7 shows an example of how the frames within a GOP normally are organized.

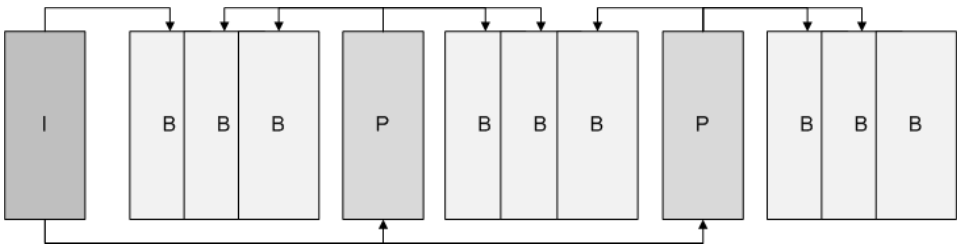


Figure 2.7: Example of normal GOP structure - arrow indicates motion vectors.

This definition of a GOP structure is rather strict, and it is important to notice that as the complexity and performance of a video codec increases, so does the flexibility of the GOP structure. More flexible GOP structures may allow B-frames to be anchor frames and increase the number of anchor frames each frame can have. As an example, H.264/AVC allows motion vectors to point from one GOP to another [19].

2.3.1 Motion estimation and compensation

The process of finding the motion vectors between two frames is referred to as motion estimation. A single vector can represent the motion of an arbitrary number of pixels, ranging from a single pixel to the entire image. When vectors represent the motion of the entire image, the estimation is referred to as global motion estimation, which will reflect the motion of the camera. This type of estimation works best if there are no moving objects in the scene.

For scenes with moving objects, it is normal to let the motion vector represent a group of pixels. To reduce the overhead a vector would normally represent a rectangular shaped block of pixels, instead of more complex group of pixels. The most common algorithm used in video codec is the block matching algorithm. The following equation is used for finding optimal vectors

$$Q(I(x, y, t), I(x + \Delta x, y + \Delta y, t + \Delta t)) \tag{2.15}$$

where $I(x, y, t)$ represents the group of pixels (macroblock) at horizontal position x, vertical position y, in frame t. Q denotes the evaluation matrix, and a motion vector $\vec{v}$ can be decomposed to Δx,Δy and Δt. There are several types of evaluation matrices that are commonly used, including mean square error (MSE), mean absolute difference (MAD), sum of absolute difference (SAD), sum of squared errors (SSE) and sum of absolute transformed differences (SATD). If equation (2.15) is minimized with respect to all valid vectors $\vec{v}$, the optimal vector is found. Which vectors that are valid depends on which frames that can be anchor frames, and other restrictions that may be applicable.

For standard block compensation, the image is divided into equally sized macroblocks. For each block an optimal vector is calculated so that all pixels are represented by a single vector. To minimize the bit rate of the vectors, encoders can choose non-optimal vectors, if they require fewer bits to be represented. More advanced designs allow for a variable size of the macroblock, and allows the encoder to select this at its own discretion. This type of motion compensation is referred to as variable block-size motion compensation (VBSMC).

Both standard block compensation and VBSMC suffers from discontinuities at the block borders (blocking artifacts). Depending on the transform coding used, these sharp edges may be visible in the decoded image. While some design employs deblocking filters to combat these artifacts, another solution is to use overlapped block motion compensation (OBMC). In OBMC the macroblocks are larger and overlap each other, so that each pixel belong to different blocks. The value assigned to each pixel will be weighted by a window function over the estimations. By using this method, there will be no block borders in the regular sense.

Interpolation can be used to increase the accuracy of the estimation. If the vectors Δx and Δy are allowed sub-pixel accuracy, then interpolation can be used to calculate sub-samples in the original image. An efficient optimization should be in place to determine if the extra accuracy obtained from interpolation outweigh the increased bit cost of the vectors. A precision of half a pixel can be found in older designs [20], while the newest employ quarter pixel precision [19].

2.3.2 Transform coding

To reduce the bit rate further, spatial correlation in the residual frame can be exploited. To do this, transform coding found in still image compression standards are used. The two most common transforms are the DCT and the DWT, the first used by JPEG and the latter used by JPEG 2000. These transformations allows encoders to quantize coefficients in the transform domain, which allows for better rate-distortion than quantization of pixel values.

DCT is the most common transform coding used in video compression. When DCT is used, the frame is divided into rectangular blocks of pixels, before each is transformed separately. The DCT transforms a macroblock from the pixel domain into the frequency domain, and allows for different quantization of the different frequency components. If blocking artifacts are present in a macroblock, they will lead to high frequency components. If the macroblock from the motion compensation corresponds to the DCT macroblocks, the blocking artifacts will not be inside the DCT macroblock, and will not cause high frequency components.

Today, DWT is mostly used in still image compression, and not in video compression. For example in the JPEG 2000 standard. Unlike the DCT-based codecs where the image is divided into macroblocks before transformation, the wavelet transformation is normally applied to the entire image. With block based motion compensation the blocking artifacts will cause high frequency components in all subbands of the wavelet transformation. Therefore, OBMC that does not introduce blocking artifact is a better choice when DWT is applied to the residue image.

2.4 Wavelet based video coding

There has been a lot of research in wavelet based video coding, and a few different approaches are presented in this section, except for the hybrid codec with wavelet transformation discussed in Section 2.3.2.

2.4.1 Wavelet domain motion compensation

In this approach the motion compensation is performed in the transformed domain, instead of the pixel domain as done in hybrid codec. Each frame is first passed through a wavelet decomposition, and then motion estimation is performed on the wavelet coefficients. For example, a video coding system that uses VBSMC for motion compensation in the redundant wavelet domain is proposed in [21].

2.4.2 Three dimensional wavelet transform

When a three dimensional wavelet transform is used for video coding, the standard 2D DWT is expanded to include the temporal axis in the subband decomposition. If there is little motion in the sequence there will be little energy in the high frequency components for the temporal dimension. Therefore, for video sequences with static content, a 3D wavelet scheme will have high energy compaction. However, if there is motion in the sequence, the correlation along the temporal axis can be reduced, resulting in larger coefficients and lower compression. To maintain the high energy compaction in this case, motion compensated temporal filtering (MCTF) can be employed [22].

2.4.3 Motion compensated temporal filtering

MCTF is often used to achieve temporal scalability [23], for example in the Scalable Video Coding (SVC) extension of H.264/AVC [19]. The SVC extension, like regular H.264/AVC, uses DCT for the remaining frames. Whether MCTF is used in combination with DCT or wavelet transforms, the (5,3) wavelet is often used to perform the MCTF. Figure 2.8 shows the MCTF structure for the (5,3) wavelet.

L^n denotes the low-pass frames at decomposition level n, and H^n denotes the high-pass frame. Remember that level 0 is the original frame. For every odd-numbered frame, a high-pass frame is predicted from the adjacent even-numbered frames. The even index frame is updated, using the the two adjacent high-pass

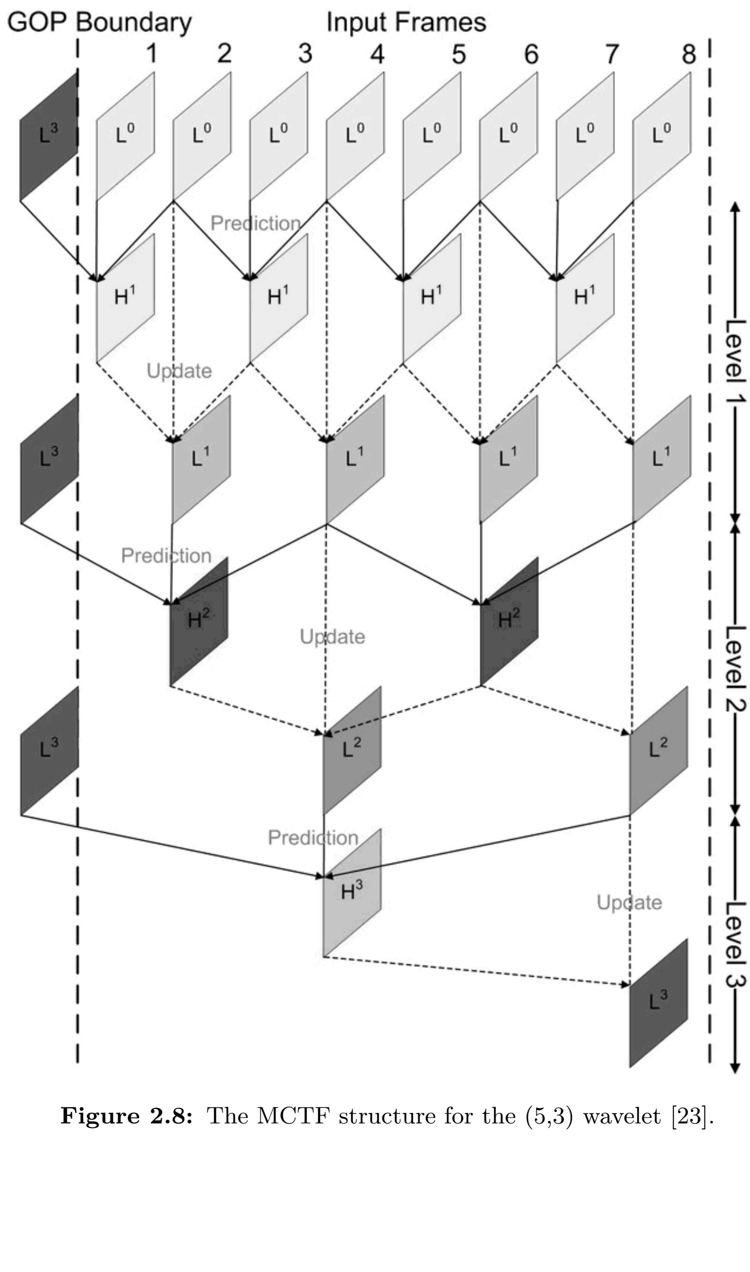


Figure 2.8: The MCTF structure for the (5,3) wavelet [23].

frames to generate a low-pas frame. From these generated low pass frames, a new decomposition level can then be calculated. For each decomposition level, a new level of temporal scalability is provided. Before the prediction and update, motion compensation is performed to reduce temporal redundancy [23].

When MCTF is followed by wavelet decomposition of the frames, the scheme is referred to as 't+2D'. This way the MCTF offers temporal scalability, while the 2D wavelet offers resolution and peak signal-to-noise ratio (PSNR) scalability.

Chapter 3

Method

The following sections contain the methods used in this thesis. First the scenario and the test material is described. Followed by a description of the selected reference codec, and the proposed hybrid codec. Finally, methods for analyzing the estimation error is presented.

3.1 Scenario

With the introduction of high definition flat screen TVs in households, there has been an increased demand of high definition content. It is not possible to meet the demand without a contribution that supports the increase in quality. State of the art TV resolution has 1080 progressive horizontal lines with 60 frames per second (1080p60), which uncompressed has bit rates of approximately 3 Gb/sec. If the video has to be transferred over long distances, or over communication networks with limited bandwidth, compression of the signal may often be necessary. This can either be done with a video codec or a still image codec that codes each frame separately.

T-Vips AS, an Oslo based company, produces video gateways that compress HD content and transfer it over an IP network. These gateways offers both lossless and lossy compression using the JPEG 2000 still image codec to reduce the overall bit rate [24, 25]. This intra-only video coding offer many advantages, for example easier editing due to no predictive frames, higher bit depth than many video codecs, no blocking artifacts, low latency, robustness to transmission errors, and less suffering from generation loss [26]. However, since the gateways utilize JPEG 2000 still image encoding, there is no exploitation of temporal redundancy.

The scenario is that the proposed hybrid video codec is used in the same environments as the T-Vips gateways.

3.2 Test Material

The test material is chosen to represent different types of content with assorted degrees of difficulty. A total of eight sequences are used, five of them are from 'The SVT High Definition Multi Format Test Set', two are provided by the European Broadcasting Union (EBU) and the last one is provided by the Blender Institute as an open source project. All sequences have been retrieved in the resolution 1920x1080. Only sequences with 8 bit per component were used in the tests. These sequences are again combined to form a standard test sequence. One frame from each sequence can be found in Appendix D.

3.2.1 The SVT High Definition Multi Format Test Set

The sequences in the test set are excerpts from the multi-genre TV-program 'Fairy-tale' produced by SVT. The sequences are filmed with a high end professional 65mm camera in 50p, digitized and mastered in 3840x2160p50. The sequences were retrieved in 1080p50 with 16 bits per color plane (RGB), and XnView was used to convert each frame to an 8-bit grayscale image. All the sequences suffers from noise to a certain extent.

Table 3.1: List of SVT sequences.

Name	*Difficulty*	*Frames used*
Crowdrun	Difficult	7111-7140
Parkjoy	Difficult	15720-15749
DucksTakeOff (Ducks)	Difficult	13060-13089
IntoTree	Easy	5119-5148
OldTownCross (OldTown)	Easy	1217-1230

The Crowdrun sequence offers a lot of local movement and sharp edges. This movement results in much occlusion in the form of small areas around the athletes. There is little global motion or larger objects with high velocity in this sequences. In the sequence DucksTakeOff the flapping of the wings of the ducks results in much occlusion, and there is a lot of slower motion caused by the waves in the water. The sequence with the most occlusion is Parkjoy, which has a large tree passing by in front of the scene, in addition to local movement. The sequences IntoTree and OldTownCross offers little occlusion and local movement, but has instead global movement.

3.2.2 EBU sequences

The sequences Horse and Vegies are more static than the SVT sequences, and they are not excerpts from what would be considered ordinary content. They do however represent content with a different type of movement and less noise than the SVT material. The sequences were provided in 1080p50 with a chroma subsampling of 4:2:2 and 10 bits per color component. To reduce the bit depth from 10 bit to 8 bit a crude downsample was performed [27]. XnView was then used to convert each frame to 8-bit grayscale.

Table 3.2: List of EBU sequences.

Name	*Difficulty*	*Frames used*
Horse	Easy	0-29
Vegies	Easy	0-29

In the Horse sequence, most of the frames are static but there is movement in two objects. The first is a toy horse connected to a ball that moves in opposite directions, and the second is a metronome with a moving pendulum. Due to the swinging motion of the objects, there will be some resulting occlusion. The Vegies sequence is a shot of a rotating plate with vegetables, with a static background. Also this sequence will experience some occlusion, arising due to the rotational movement of the vegetables.

3.2.3 Big Buck Bunny

The short computer animated film Big Buck Bunny (Bunny) has been produced by Peach Open Movie Team using the free software tool Blender. The frames selected from the movie are from the opening scene, with a vertical panning of the camera, revealing more and more of the landscape (frame 100-129). Since the sequence is computer animated, there is no noise in the frames and the global motion appears constant. The difficulty is defined as easy for the frames selected. The sequence was retrieved in y4m format and converted first to avi and then to ppm images by using FFmpeg. Again XnView was used to convert the images to 8-bit grayscale.

3.2.4 Standard test sequence

All of the above test sequences have been combined into one single test sequence. Thirty frames from each sequence gives a total of 240 frames. The entire sequence was 8 bit grayscale in the resolution 1920x1080p, and all the material was produced in 50 frames per second.

3.3 Reference codec

To analyze the performance of the proposed video codec, a reference codec had to be chosen. According to the scenario a JPEG 2000 intra-only still image encoder would be preferable for comparison. To simplify the process, the Matlab function *imwrite* was used to create the reference video sequences. In this process, Matlab will read the uncompressed images from the hard drive, then compress and save each image using the JPEG 2000 encoder. The bit depth and resolution of the JPEG 2000 files will match that of the uncompressed sequence, unless otherwise is specified.

When compressing the images, the compressed file size is 1/10 of its original. For an 8 bit grayscale image, the average number of bits needed to encode each pixel will then be 0.8. This will lead to an acceptable compromise between bit rate and quality. The default setting of 6 reduction levels in the wavelet transform, and one tile for the entire image was also used. The following Matlab-code can be used to save an image into a JPEG 2000 file

```
imwrite(matlabmatrix,'iframe.jp2','CompressionRatio',10);
```

where matlabmatrix is an M-by-N matrix representing a grayscale image, or M-by-N-by-3 matrix representing an image with three color components.

3.4 Proposed hybrid codec

The proposed hybrid codec was designed with regards to the scenario presented in Section 3.1. A block diagram is shown in Figure 3.1, with some differences from the generic hybrid encoder: The optical flow can be calculated from uncompressed frames, and an in-loop denoise filer can be applied before motion compensation.

To avoid a long delay, B-frames that are dependent of both preceding and following frames are not allowed. By limiting the use to only P-frames, there will be no need to buffer subsequent frames before decoding. Extra delay, in comparison with an intra only codec, will therefore be limited to estimation of motion vectors and calculation of the residual frames. How much this extra delay will amount to is implementation dependent and can be decreased with extra computational resources. If B-frames were allowed, the extra delay could not be smaller than the time required to buffer the needed frames.

3.4.1 Vector calculation - optical flow algorithm

To calculate the optical flow an existing state of the art algorithm, classic+nl, is used. The algorithm is presented in [1] and at the time of its publication (March 2010), it ranked 1st in both average angular and average end-point errors in the the Middlebury evaluation. However, as of May 2011 the algorithm ranks 5th

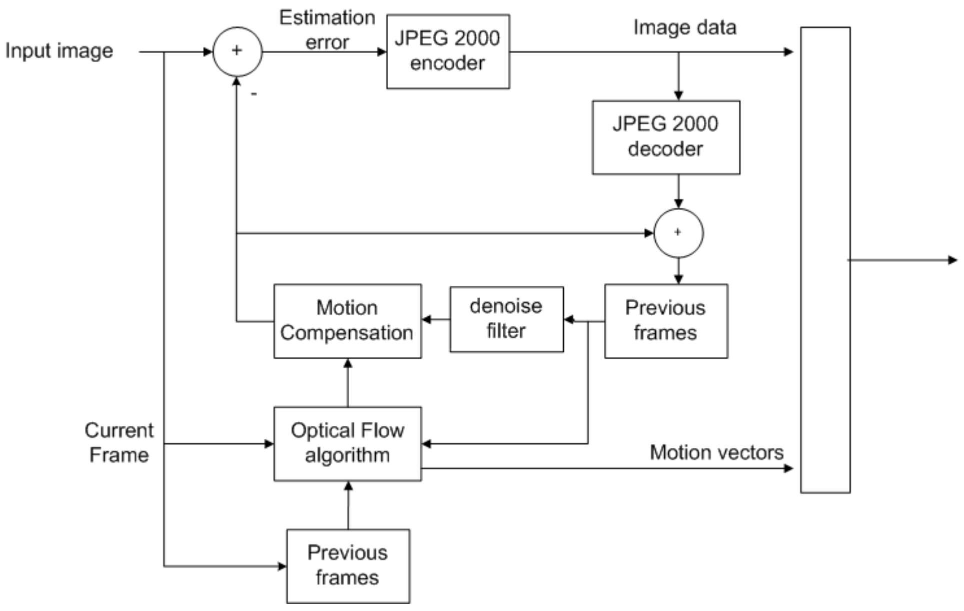


Figure 3.1: Block diagram of proposed hybrid encoder.

in the same tests [28]. The Middlebury evaluation is a database and evaluation methodology for optical flow calculations [29]. Even though the algorithm does not rank 1st anymore, it was chosen because the Matlab implementation has been made available for research purposes [30]. A faster version of the algorithm was used to calculate the results, using default parameters. The following Matlab-code calculates the optical flow between frame I(t) and I(t+1):

```
estimate_flow_interface(I(t), I(t+1), 'classic+nl-fast');
```

If not otherwise specified, the vectors are calculated based on the uncompressed original version of the frames I(t) and I(t+1).

The vector field can be calculated based on the original full resolution image, or on a downsized version. If the flow is calculated on a downsized version, the processing time will decrease, the number of vectors will be fewer, and the resolution of the vector field will not match the resolution of the original image. Bicubic interpolation can then be used to upscale the vector field resolution, such that it matches the resolution of the original image. The Matlab function `imresize` was used for upscaling of the vectors and downsizing of the images.

3.4.2 Vector calculation - mean squared error

In addition to using optical flow to calculate vectors, some vectors were calculated using equation (2.15) where the cost function Q is the mean squared error defined by

$$Q = E[(I(x + \Delta x, y + \Delta y, t + \Delta t) - I(x, y, t))^2] \tag{3.1}$$

and minimized over all valid values of Δx,Δy, andΔt. This was implemented as an exhaustive search in predefined rectangular areas in the previous frame. The frame $I(x, y, t)$ is the decoded version of the original frame, containing some compression errors.

3.4.3 Vector compression

The motion vectors along with the residual frame are needed to decode P-frames. Without compression the optical flow vectors would require 128 extra bit per vector (two vector components with double precision), which would be 16 times higher than an uncompressed 8 bit pixel value. To reduce this size, the vectors are first rounded to the desired accuracy and then the vector field is compressed with lossless data compression. The rounding is performed in the following matter

```
V=round(V*pixelresolution)
```

where V is an $M \times N \times 2$ matrix containing the motion vectors, and pixelresolution is the accuracy of the motion vector. For example, a pixelresolution value of 4 equal a quarter-pixel resolution. The compression process removes information not needed to decode the frames and thus it reducees the entropy of the vector field. The Matlab function `save` (version 7) was used to save the vector field using lossless data compression. The entire matrix was saved in Matlabs .mat format that uses buffered in-memory gzip for lossless compression. The compression ratio could probably be reduced further by using an algorithm and file-structure specially designed for motion vectors, rather than using a general purpose method.

3.4.4 Residual frame calculation

The codec has been implemented in Matlab 7.11.0 (2010b) and consists of a framework with both an encoder and a decoder, such that all frames are first encoded and then decoded. During calculations the internal bit depth was increased to double precision.

The residual frame $I_{residual}$ has been calculated in the following matter. First an estimation of frame I at time $t + 1$ is calculated:

$$I_{est}(x, y, t + 1) = I(x + \Delta x, y + \Delta y, t)) \quad \forall \quad x, y \tag{3.2}$$

where Δx and Δy represents the motion vector pointing from frame $t+1$ to t. If the motion vectors have sub pixel accuracy, bilinear interpolation is used to calculate the sub pixels. For resource and performance purposes the bilinear interpolation is performed on the entire frame I with the Matlab function `imresize`. The pixel values in the original frame I are used when the vector does not have sub-pixel accuracy. After this process the estimation error is calculated:

$$I_{residual}(x, y, t+1) = I_{est}(x, y, t+1) - I(x, y, t+1) \tag{3.3}$$

When the pixel values in $I(x, y, t)$ have a bit depth of 8 and are in the range [0,255], the pixel values in $I_{residual}$ will be in the range [-255,255]. To represent this range more than 8 bits are needed, and with Matlab's implementation of the JPEG 2000 encoder the residual image must then be encoded with 16 bits. Two possible approaches to encode the residual image with a bit depth of 8 have been implemented. The first one is to add 127 to all pixel values, so that the new range becomes [-128,382]. Afterwards all values below 0 is set to 0, and all values above 255 are set to 255. The second approach first scales the values by a factor of x, and then adds 127 to each pixel. Afterwards all pixels are rounded to the nearest integer in the range [0,255]. The scaling factor x is selected as follows:

$$y = \frac{\left\lfloor \frac{127}{max(abs(I_{residual}(x, y, t)))} \cdot 10 \right\rfloor}{10} \quad \text{for each t} \tag{3.4}$$

$$x = \begin{cases} y & \text{if} \quad y < 1 \\ 1 & \text{if} \quad y > 1 \end{cases} \tag{3.5}$$

While equation (3.4) sets a scaling factor with one decimal, equation (3.5) only allows for scaling values up to and including 1. Each frame will then have a scaling value in the range [0.5, 1] with a step size of 0.1.

If a bit depth of 16 is used in the residual image, 32640 is added to each pixel value, so that the pixels are in the range [32385, 32895]. The three different approaches are referred to as '8 bit cutoff','8 bit scaling' and '16 bit'.

3.4.5 Residual frame compression

The estimation error is compressed with the same built in Matlab function as the reference codec, the `imwrite` function. The complementary `imread` function is used by the decoder to decompress the files. To achieve the same bit rate for both 8 bit and 16 bit frames, the 'CompressionRatio' setting for 16 bit frames must be twice that of the 8 bit frames.

3.4.6 Noise

When noise was removed on the entire sequence, the XnView filter 'Median Box 5x5' was used. For in-loop denoise filter the Matlab functions `wiener2` and `medfilt2` was used. The wiener filter was applied with default settings, while the median filter was calculated based on a 5×5 pixel neighborhood.

The XnView filter 'Add laplacian noise' was used when noise was added to the images, because it resembles noise that can be present in some types of content.

3.5 Analysis methods

All tests were performed on greyscale content, to represent the luminance or brightness of the sequences. While colors are an essential part in the visual experience, the human visual system is more sensitive to intensity in luminance than in chrominance. In most video compression system, this is exploited to reduce the overall bit rate [31]. For easier comparison between the reference codec and the proposed hybrid codec, color images was not used in the tests.

Most test are performed on the standard test sequence described in Section 3.2.4. The GOP size is two frames long to prevent propagation of error from one frame to another, unless specified otherwise. Also the in-loop denoise filter is switched off, unless specified otherwise

The test were performed with Matlab 7.11.0.586 (2010b) 64 bit edition, on a computer with a 3.2GHz quad core Intel i7 960 processor, 6 Gb RAM and Windows 7 Enterprise 64-bit edition.

3.5.1 Objective metrics

To evaluate the performance of the proposed hybrid codec the objective metrics peak signal to noise ratio (PSNR) and structural similarity (SSIM) are applied. PSNR calculates the signal to noise ratio in the following way

$$PSNR = 10 \cdot \log \left(\frac{MAX_I^2}{MSE} \right) \tag{3.6}$$

where MAX_I^2 is the maximum signal intensity squared, which is 255^2 for 8 bit images, and MSE is the mean squared error between the compressed and the original image. Since PSNR only considers the signal to noise ratio, and not how errors are perceived by the human visual system, SSIM is designed to take this into account [32]. SSIM is calculated with the following equation:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{3.7}$$

where μ_x and μ_y is the average of x and y, σ_x^2 and σ_y^2 is the variance of x and y,σ_{xy} is the covariance of x and y, and $c_1 = (0.01 * 2^B)^2$ and $c_2 = (0.03 * 2^B)^2$ are variables that stabilizes the division (B is bit depth). The Matlab software MetrixMux was used to calculate PSNR and SSIM [33].

3.5.2 Objective test method

The performance of the proposed hybrid codec is either presented as a difference in PSNR, or as a bit rate fraction, both compared to the reference codec. When the numbers are presented as a difference in PSNR the following course has been taken to calculate them:

1. The standard test sequence is encoded and decoded with the reference codec, as described in Section 3.3.
2. The standard test sequence is encoded and decoded with the proposed hybrid codec. The residual frames will have the same bit rate as the reference codec.
3. Difference in PSNR and SSIM for each sequence, along with the confidence intervals, are calculated with the Matlab script found in Appendix B. Only P-frames are considered.

Positive and negative value will indicate increase and decrease in PSNR, respectively. Remember that the value is compared to the reference codec and that the bit rate of the vectors are not taken into consideration.

To include the bit rate of the vectors, an overall reduction in bit rate has been calculated. A distortion control has been implemented so that each frame compressed by the proposed codec will have the same PSNR as the one encoded by the reference codec. The bit rate is then calculated in the following manner:

$$E\left[\frac{residual_{bitrate} + vector_{bitrate}}{reference_{bitrate}}\right] \tag{3.8}$$

With equation (3.8) only the bit rate of P-frames are considered. A rate below 100% indicates an overall improvement, while a rate above indicates poorer performance.

3.5.3 Analysis of residual frames

To obtain more detailed information about the estimation error and the contents of residual frames, two different approaches are used. The first is to evaluate the distribution of the pixel values in a frame. Figure 3.2 shows a histogram created by the Matlab function `imhist`.

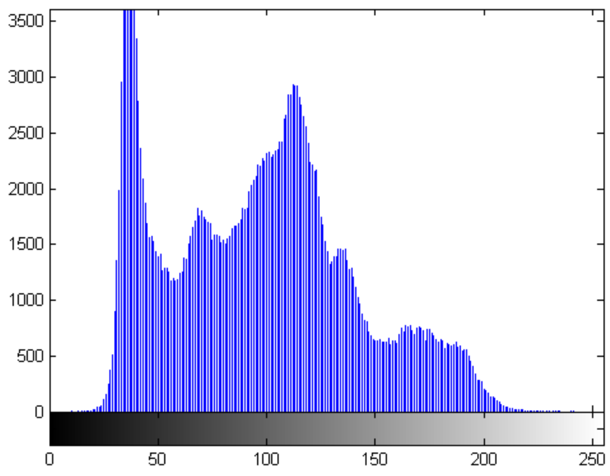


Figure 3.2: Histogram of pixel values for grayscale version of Lena, resolution 512x512.

While a histogram will show the distribution of the pixel values, it does not necessary represent how much it can be compressed by JPEG 2000 at a certain distortion level. To analyze the residual frame with regard to the encoding performed in JPEG 2000 the following process is performed on both the original uncompressed frame and the estimation error:

1. The pixel values are scaled to the range $[-\frac{1}{2}, \frac{1}{2}]$, according to the irreversible path in Section 2.2.2.
2. A two-level wavelet decomposition is performed using the filter taps in Table 2.1.
3. For each subband the mean, minimum value, maximum value, standard derivation and entropy[1] is calculated for the coefficients.
4. Each subband is weighted according to number of coefficients, and the average entropy is calculated.

There are several reasons this approach will not accurately represent the JPEG 2000 encoding.

- The entropy function does not accurately reflect the arithmetic encoder in JPEG 2000, as neither the central dead zone, nor the spatial distribution of the coefficients are taken into account.
- JPEG 2000 employs truncation with the EBCOT scheme.
- Only a two-level decomposition is performed.

[1]The Matlab function entropy uses 256 levels (bins) for calculating probability

- The PRCD scheme is not implemented.

The approach will however give information about the wavelet coefficients distribution, and the difference between the original and the residual frame. The Matlab implementation of this process can be found in Appendix B.

Chapter 4

Tests and Results

The following sections contains the results of tests performed, along with a brief discussion of each test. First, a bit depth for encoding of the estimation error is selected. Then the results from a general evaluation of the selected optical flow algorithm is presented. After this, the effect different resolutions and pixel accuracies have is evaluated. Then the overall bit rate of the proposed codec is determined, followed by results for different GOP structures. Finally, the effect of noise and the performance of the in-loop denoise filter is presented.

4.1 Bit depth of residue image

As explained in Section 3.4.4, the dynamic range of a residue image can exceed that of an 8 bit image. The three proposed solutions to this problem have been tested and the results can be seen in Table 4.1. The vector field is calculated using the optical flow algorithm and pixel accuracy of the vectors.

Table 4.1: PSNR (dB) difference for different bitdepths of residue image. Vector field of 1920x1080 with one pixel accuracy.

Method	*Bunny*	*Crowdrun*	*Ducks*	*IntoTree*	*OldTown*	*Parkjoy*	*Vegies*	*Horse*	*Average*
8 bit cutoff	1.58	0.65	-0.96	0.19	0.25	-0.02	-0.56	3.60	0.82
8 bit scaling	1.58	0.84	-0.80	0.19	0.25	0.74	-0.53	3.32	0.89
16 bit	1.83	0.85	-0.62	-0.08	0.017	0.71	-0.33	3.24	0.89

As seen in Table 4.1 there are only small differences in the average PSNR between the different solutions. While ’8bit cutoff’ has an average improvement of 0.82 dB, the two others have a slightly higher average, i.e. 0.89 dB. The only sequence where ’8 bit cutoff’ performs better than the others, is the Horse sequence. There is an even distribution between which one of ’8 bit scaling’ and ’16 bit’ that performs best.

More notable, the visual compression artifacts will be different for the different methods. With the ’8 bit cutoff’ method, areas that experience a high estimation error will also have a corresponding visual degradation. This is easy to perceive for occluded regions in the sequences Ducks and Parkjoy. ’8 bit scaling’ on the other hand, will have an overall reduction in pixel accuracy when the range is scaled. In ’16 bit’ there is no scaling of the values, and no irreversible processing of the pixel values before the residue image is passed to the JPEG 2000 encoder.

To compare ’16 bit’ and ’8 bit scaling’ more closely, the average PSNR difference with a 95% confidence interval is plotted in Figure 4.1.

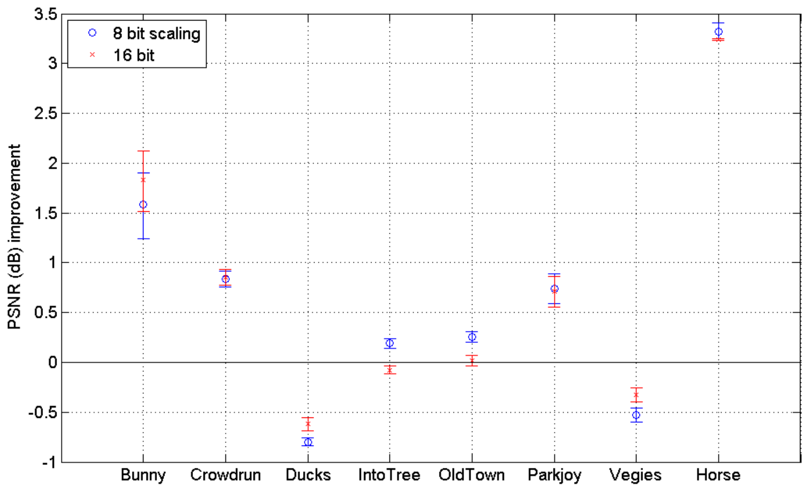


Figure 4.1: PSNR (dB) difference for 1920x1080 vector field with one pixel accuracy.

The results for the ’16 bit’ method and for the ’8 bit scaling’ method are practically the same, with some differences for a few of the sequences. ’8 bit scaling’ performs a little better for the IntoTree and OldTown sequnces, while ’16 bit’ performs better for the Ducks and Vegies sequnces.

The ’16 bit’ residue images would appear to be gray and homogeneous when viewed on a computer screen, and software support for 16 bit images is limited. To ease visual analysis of the residual images, ’8 bit scaling’ is used for encoding of residual images.

4.2 Quality of optical flow algorithm

To determine the quality of the optical flow algorithm, two computer generated images have been coded with the proposed hybrid codec. The first image, shown in Figure 4.2, is a black arrow on a gray background. The second image contains the same arrow, moved five pixels to the right. With these images it is possible to calculate an optimal optical flow.

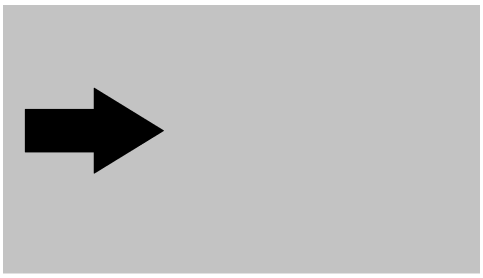


Figure 4.2: Image of black arrow on gray background. Used to calculate quality of the optical flow algorithm

To see how the codec performs when the images suffer from noise, different degrees of laplacian noise have been added, which mostly distorts the gray background. Table 4.2 shows that there is a significant reduction in PSNR as the noise increases. This reduction can be seen for both I-frames and P-frames, and shows the effect that noise has on the performance of the JPEG 2000 encoder.

Table 4.2: PSNR (dB) of arrow sequence with calculated OF vectors. Laplacian noise added with XnView filter 'Add laplacian noise'.

	without noise	*laplacian 1.0*	*laplacian 2.0*	*laplacian 3.0*	*laplacian 4.0*
I-frame	75.7	47.4	42.0	38.4	36.2
Vector + residual	75.8	43.7	39.2	36.6	33.9
Vector	75.1	41.6	35.7	33.3	30.4

There is no perceptual degradation from the uncompressed to the decoded, when the same amount of bits are used to encode the residue image as the original image. The drop in PSNR can be attributed to errors in the noise, which will affect PSNR but not the visual quality. However, with increased noise in the images, there is an increased drop in the performance of the optical flow algorithm. This can be seen in the residual images, where the edges of the arrow suffer from estimation errors, and shows that noise will affect the calculation of the optical flow. This error is not present with the lowest degree of noise.

Table 4.2 also shows the PSNR when no bits are used to encode the residual image, so that the estimation error is not corrected. This will cause some noteworthy visual artifacts: In addition to the error around the edges of the arrow, the noise in the image will move according to the calculated optical flow. In these situations the noise will not behave as noise anymore, but rather as moving texture. Furthermore, the moving noise is already compressed in the anchor frame. This effect would appear unnatural for the human eye, and will cause both a drop in PSNR and a degradation of the subjective quality.

Table 4.3: PSNR (dB) of arrow sequence with optimal OF vectors.

	without noise	*laplacian 1*	*laplacian 2*	*laplacian 3*	*laplacian 4*
I-frame	75.7	47.4	42.0	38.4	36.2
Vector + residual	75.8	43.7	39.2	36.4	33.7
Vector	75.0	41.6	36.6	33.4	30.9

Table 4.3 shows how the encoder performs if the motion vectors are optimal. The same vectors are used for all levels of noise, and are based on the original images without noise. By comparing with the results in Table 4.2, it can be seen that there are only small differences between the calculated optical flow, and the optimal vectors. When the residual frame is encoded, the results are visually indistinguishable from when the vectors are calculated using optical flow. However, when optimal vectors are used and the estimation error is not corrected, the arrow will not suffer from edge errors. For the noiseless images this will cause a small decrease in PSNR, which can be attributed to error propagating from the anchor frame. For the other images, the degradation in PSNR can be attributed to moving noise.

In the previous tests, the optical flow was calculated between two original uncompressed images. However, the motion compensation is performed from the compressed version of the previous frame. Table 4.4 shows the results when the OF vectors are calculated from the compressed version of the previous frame.

Table 4.4: PSNR (dB) of arrow sequence with OF vectors calculated from compressed image.

	without noise	*laplacian 1*	*laplacian 2*	*laplacian 3*	*laplacian 4*
I-frame	75.7	47.4	42.0	38.4	36.2
Vector + residual	75.8	43.8	39.3	36.6	34.0
Vector	75.1	41.2	35.7	32.9	30.6

A comparison between these results and the results presented in Table 4.2, shows that there is little difference of wether the optical flow is calculated from the original image, or the compressed version. It could however be differences for other types of content.

4.3 Resolution and accuracy of vector field

In this section, different settings for the resolution and the accuracy of the vector field are tested on the standard test sequence.

4.3.1 Resolution

The vector field can be calculated based on the original full resolution image, or on a downsized version. Table 4.5 shows the results when different resolutions are used to calculate the flow with one pixel accuracy. Also included in the table, is the results when all the vectors are zero, i.e. when standard differential encoding is performed.

Table 4.5: PSNR (dB) difference: Optical flow calculated with a accuracy of one pixel.

Resolution	*Bunny*	*Crowdrun*	*Ducks*	*IntoTree*	*OldTown*	*Parkjoy*	*Vegies*	*Horse*	*Average*
1920x1080	1.56	0.83	-0.80	0.19	0.25	0.73	-0.54	3.32	0.88
960x540	1.62	0.86	-0.83	0.15	0.21	0.76	-0.57	3.34	0.88
480x270	1.58	0.69	-0.90	0.16	0.22	0.64	-0.53	3.17	0.81
320x180	1.60	0.58	-0.94	0.15	0.24	0.50	-0.56	3.15	0.77
240x135	1.49	0.49	-0.97	0.14	0.24	0.37	-0.58	3.13	0.72
160x90	1.35	0.37	-0.96	0.11	0.22	0.14	-0.84	3.16	0.63
differential	-2.50	-0.03	-0.50	-0.69	-0.33	-1.97	-1.14	3.20	-0.16

From the results it is obvious that optical flow outperforms differential encoding. Differential encoding experiences no improvement in PSNR, except for the Horse sequence where there is an improvement of 3.20 dB.

From the results, it can be seen that for resolutions lower than 960x540, PSNR will decrease with the resolution. Some of the sequences suffer more from the decline than the others, most notable is Parkjoy that falls from 0.73 dB to 0.14 dB, and Crowdrun from 0.83 dB to 0.37 dB. These sequences have in common that there is a lot of local motion in opposite directions for small objects.

4.3.2 Accuracy

Since the optical flow algorithm calculates motion vectors on a sub-pixel level, these sub-pixels can be used instead of the original values when calculating the residual image.

From Table 4.6 it can be seen that one pixel accuracy is outperformed by half pixel accuracy for all sequences. On average one pixel accuracy gives an improvement of 0.88 dB, while half pixel accuracy gives an improvement of 1.32 dB. Quarter pixel accuracy however, gives the same or slightly more improvement on average than

Table 4.6: PSNR (dB) difference: Optical flow calculated from original image with a resolution of 1920x1080.

Accuracy	*Bunny*	*Crowdrun*	*Ducks*	*IntoTree*	*OldTown*	*Parkjoy*	*Vegies*	*Horse*	*Average*
One pixel	1.56	0.83	-0.80	0.19	0.25	0.73	-0.54	3.32	0.88
Half pixel	1.71	1.55	-0.20	0.76	0.70	1.42	0.05	3.46	1.32
Quarter pixel	1.58	1.60	-0.17	0.79	0.72	1.46	0.07	3.46	1.33

half pixel accuracy. This is true for all resolution, and can be seen when comparing Table A.2 with A.3.

The increased accuracy will also lead to an increased bit rate. In Table 4.7 the average improvement along with the bit rate of the vectors are given. Remember that the vector bit rate is given in relation to the total bit rate of the reference codec.

Table 4.7: PSNR (dB) difference and vector size for one pixel, half pixel and quarter pixel accuracy.

	One pixel		*Half pixel*		*Quarter pixel*	
Resolution	*Difference*	*Vector size*	*Difference*	*Vector size*	*Difference*	*Vector size*
1920x1080	0.88	62.16%	1.32	93.87%	1.33	142.92%
960x540	0.88	20.38%	1.29	30.40%	1.33	44.82%
480x270	0.81	6.26%	1.22	9.34%	1.26	13.91%
320x180	0.77	3.13%	1.16	4.74%	1.19	7.10%
240x135	0.72	2.03%	1.10	3.10%	1.15	4.61%
160x90	0.63	1.10%	1.00	1.69%	1.04	2.49%

Increasing the accuracy at a lower resolution gives a better improvement per bit, than a high resolution and lower accuracy. For example, the highest resolution with the lowest accuracy gives 0.88 dB improvement at the cost of 62.16% increase in bit rate, while the lowest resolution and the highest accuracy gives 1.04 dB improvement at the cost of 2.49% increase in bit rate.

4.3.3 MSE correction of OF vectors

Even with quarter pixel accuracy, optical flow algorithm still produces estimation errors. To adjust the vector field calculated by the optical flow algorithm, the endpoint of each vector has been adjusted with a MSE search. The search is performed over 3x3 pixels, where the middle pixel is the original estimate. The difference in PSNR for each sequence is shown in Figure 4.3.

There is an significant improvement in PSNR for all sequences. This can be attributed to less estimation error in areas with noise and at edges. The improved vector field has a bit rate of over 600%, while the original had a bit rate of 62.16%.

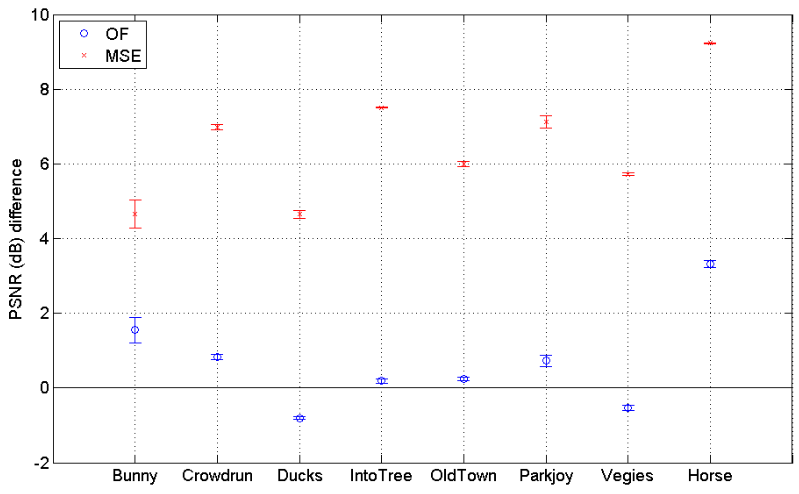


Figure 4.3: Difference in PSNR (dB) for optical flow and optical flow corrected with MSE-search. Vector field of 1920x1080 and one pixel accuracy. Search area for MSE was 3x3 pixels.

This shows that the smoothness of an optical flow field reduces the bit rate of the vectors substantially.

4.3.4 Performance

To compare the performance of the proposed hybrid codec to the reference codec, the standard test sequence has been encoded with the same PSNR for both sequences. The best average performance was achieved when the vector field had a resolution of 240x135, with half pixel accuracy.

Table 4.8: Bit rate for P-frames, given as percentage of I-frame. Vector field of 240x135 and half pixel accuracy.

	Bunny	*Crowdrun*	*Ducks*	*IntoTree*	*OldTown*	*Parkjoy*	*Vegies*	*Horse*	*Average*
Residue	73.0%	79.8%	108.8%	80.4%	78.4%	83.8%	103.1%	23.9%	78.9%
Vector	1.3%	5.5%	5.6%	2.3%	2.0%	4.4%	2.5%	1.1%	3.1%
Total	74.3%	85.3%	114.4%	82.8%	80.4%	88.2%	105.6%	25.1%	82.0%

The proposed hybrid encoder achieves a bit rate of 82.0% of the refererence codec for P-frames, with the same PSNR. When comparing Table A.2 with the reduction in bit rate, it is clear that a higher improvement in PSNR gives a higher reduction in bit rate. For the sequences Vegies and Ducks, which had a decline in PSNR, there was an increase in bit rate for the residue image. Along with the bit rate of the vectors, the total bit rate will be significantly higher than the reference codec. In situations with a bit rate higher than the reference codec, the encoder should switch to intra-only coding.

To assess if there has been a degradation in image quality, the difference in SSIM has been calculated for each sequence. This is shown in Figure 4.4.

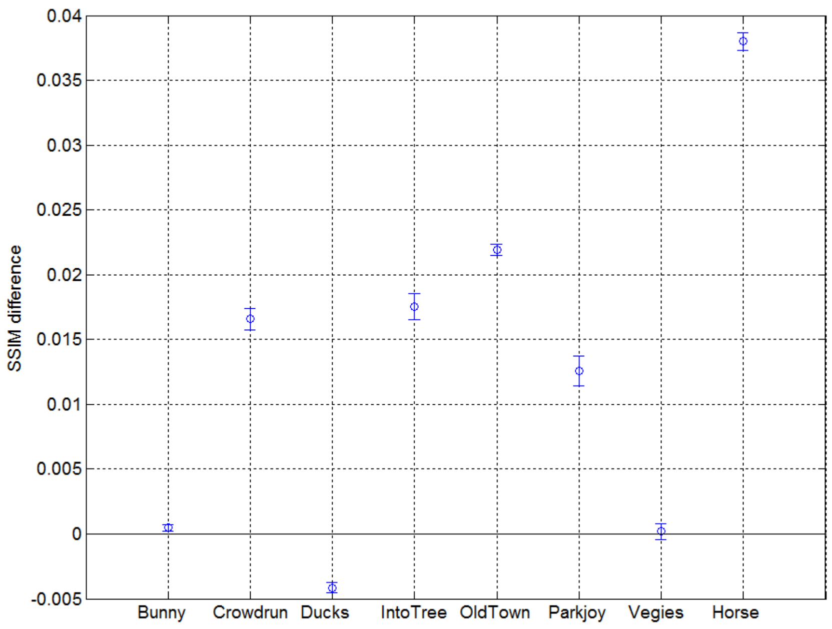


Figure 4.4: Difference in SSIM between proposed hybrid codec and reference codec. Vector field 240x135 and with half pixel accuracy.

There is a slight decrease in SSIM for the Ducks sequence, and some increase for a few of the other sequences. Since both the PSNR and the SSIM practically are the same for the reference codec and the proposed hybrid codec, and the fact that both codecs are wavelet based, it is reasonable to assume that the subjective quality is the same.

4.4 GOP structure

How the GOP is structured is critical for the performance of a hybrid codec. With a longer GOP a higher portion of the frames will be P-frames, which normally will reduce the overall bit rate. The achieved reduction of 18% for P-frames will only cause a total bit rate reduction of 9% if the GOP structure is IPIPIPI. How the proposed hybrid codec responds to a change in GOP structure is therefore crucial to the overall performance of the codec.

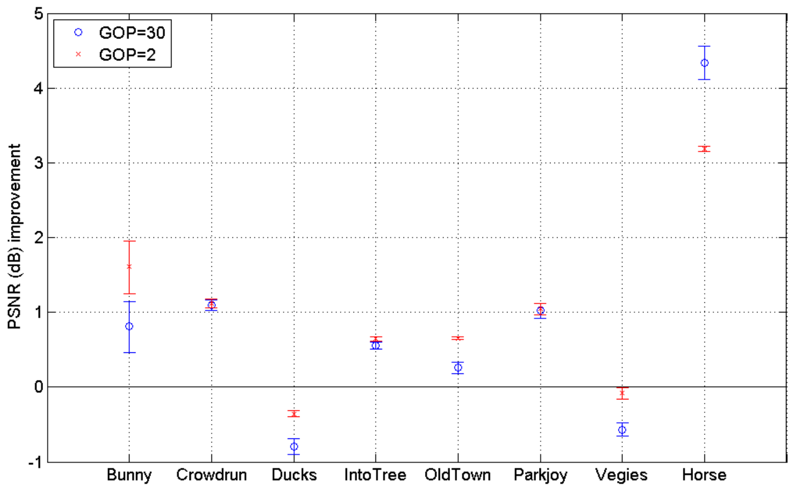


Figure 4.5: Increase in PSNR for sequence with a GOP of 2 and 30 frames, shown for each sequence. Vector field is 240x135 and with half pixel accuracy. The same 14 frames are compared for each sequence.

In Figure 4.5 the average increase in PSNR for the difference sequences are shown for each frame. When the GOP size is increased to 30 frames, the average difference in PSNR is lower or the same for all sequences, except the Horse sequence. A decrease in PSNR is to be expected, as errors not corrected by the residual frame can propagate inside a GOP.

The impact the change in PSNR has on bit rate is shown in Table 4.9. The Horse sequence, which had an increase in PSNR, has a reduction in bit rate from 23.9% to 8.9%. For all the other sequences there is an increase in bit rate, even for the sequences with the same PSNR.

The results when the optical flow is calculated from the decoded image instead of the uncompressed are also shown. There is an increase of 1.3% in bit rate for the residue image, and a decrease in bit rate for the vectors of 0.3%. In total, there is no gain in calculating the optical flow from the decoded image. Results can be found in Table A.6

Even though there is an increase in bit rate for P-frames when the GOP size is increased, there will still be an overall reduction in bit rate. The overall bit rate with a GOP size of 2 is 91%, and 88.3% for 30 frames.

Table 4.9: Bit rate for residue-frames, given as percentage of I-frame, for GOP size of 2 and 30. Vector field of 240x135 and half pixel accuracy. *=Optical flow is calculated from the decoded frames.

	Bunny	*Crowdrun*	*Ducks*	*IntoTree*	*OldTown*	*Parkjoy*	*Vegies*	*Horse*	*Average*
GOP=2	73.0%	79.8%	108.8%	80.4%	78.4%	83.8%	103.1%	23.9%	78.9%
GOP=30	84.6%	82.5%	121.5%	82.9%	89.3%	87.1%	121.7%	8.9%	84.8%
GOP=30*	84.4%	85.8%	119.6%	84.3%	90.4%	93.0%	122.2%	9.5%	86.1%

A good encoder would not use motion prediction if it does not reduce the bit rate. Such a situation may for example occur when there is a scene change, and the encoder would swap the P-frame with an I-frame. If the proposed hybrid encoder switches to I-frames for frames that have no reduction in bit rate, the overall reduction would increase. For a GOP size of 2 the overall bit rate would be 89.8%, and with a GOP size of 30 the rate would be 82.1%.

4.5 Removal of Noise

In the following section some experiments regarding the noise in the images are presented. In Figure 4.6 the difference in PSNR is show when the sequences have been median filtered. It can be seen that the PSNR is higher for the original sequences, except for the OldTown sequence, where the difference is the same.

Since median filtering has a tendency to not only remove noise, but also distort details and edges, this can affect the performance of the optical flow algorithm. When median filtering is applied before encoding, the decoded video will suffer from the same artifacts, which is an undesirable effect.

As shown in Figure 3.1, the removal of noise can be performed in-loop to improve the motion compensation. In Figure 4.7 it can be seen that using median filtering does not improve the results. However, the wiener filter improves the performance for several of the sequences.

Table 4.10: Bit rate for in-loop wiener denoise filter, calculated for GOP size of 2 and 30 frames. Vector field of 240x135 and half pixel accuracy.

	Bunny	*Crowdrun*	*Ducks*	*IntoTree*	*OldTown*	*Parkjoy*	*Vegies*	*Horse*	*Average*
GOP=2 original	74.3%	85.3%	114.4%	82.8%	80.4%	88.2%	105.6%	25.1%	82.0%
GOP=2 wiener	81.4%	85.3%	104.9%	79.1%	77.0%	88.5%	92.4%	48.5%	82.1%
GOP=30 original	85.9%	88.0%	127.1%	85.2%	91.3%	91.4%	124.1%	10.0%	87.9%
GOP=30 wiener	82.1%	86.4%	106.4%	79.7%	79.6%	89.9%	93.2%	54.1%	83.9%

In Table 4.10 the difference in bit rate for each sequence is shown. With a GOP of 2 frames, there was a decrease in bit rate for all sequences, except for Bunny and Horse, which matches the results in Figure 4.7. However, when changing to 30 frames in the GOP, the decrease in bit rate was larger, and the Bunny sequence had

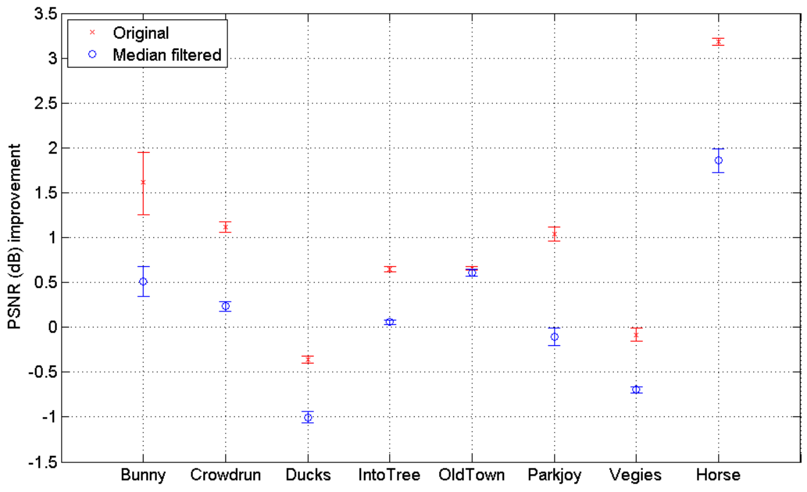


Figure 4.6: PSNR increase with original and median filtered sequences. Vector field is 240x135 and with half pixel accuracy.

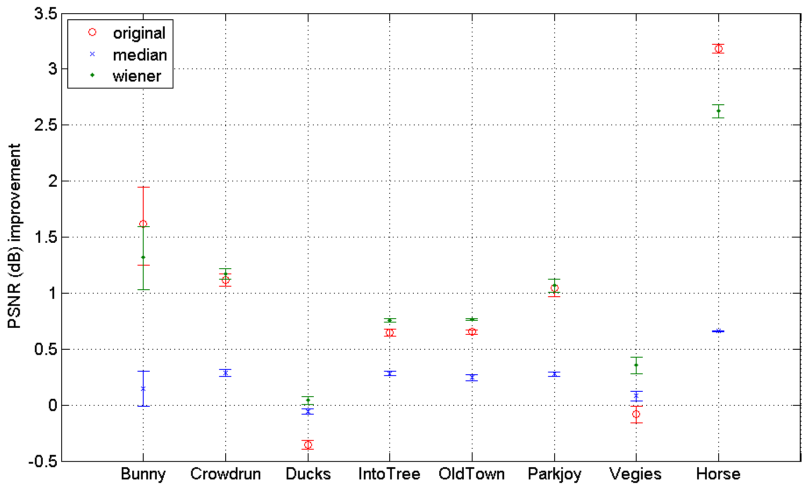


Figure 4.7: PSNR difference with in-loop denoise filter. Vector field is 240x135 and with half pixel accuracy. Filter is either median 5x5 or wiener filter.

a decrease in bit rate. With a GOP of 30 frames the Horse sequence experienced a high decrease in bit rate without the wiener filter, but not with a filter.

Since only the Ducks sequence had a bit rate over 100%, changing to intra-only mode will not reduce the bit rate much. With this feature, the overall bit rate was 83.7%, due to the increase of the Horse sequence.

Chapter 5

Discussion

In the following sections the presented results are further discussed. An analysis of the estimation error is presented, followed by a discussion of optical flows suitability for motion compensation. Then the performance of the proposed hybrid codec is compared to that of the reference codec in the chosen scenario. Finally, there is a brief discussion of the potential for a hybrid encoder with a wavelet transform.

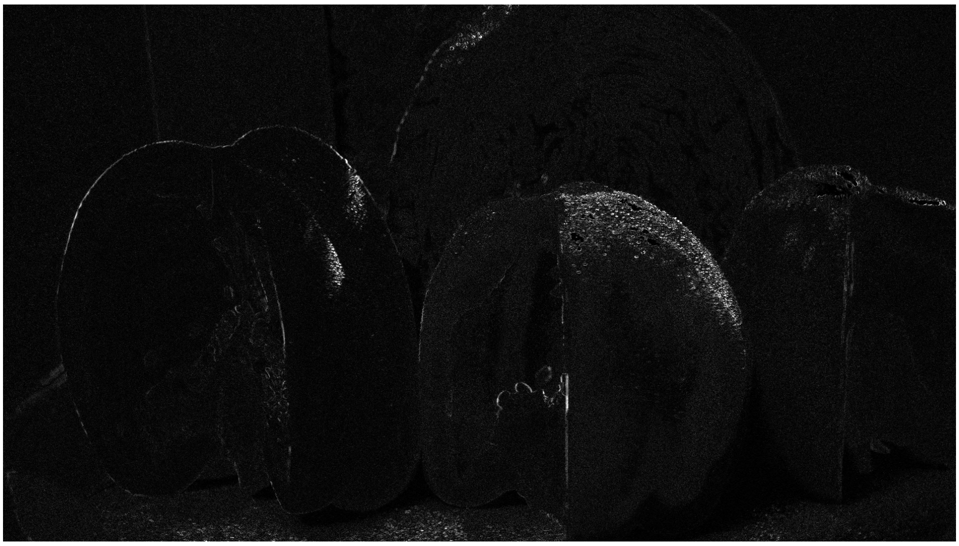
5.1 Analysis of estimation error

Compared to the reference codec, the proposed hybrid codec has a bit rate at 25.1% for the best sequence (Horse), and 114.5% for the worst sequence (Ducks) with the following settings: A GOP size of 2 frames, vector field of 240x135, half pixel accuracy, and the in-loop denoise filter switched off. In this section, the properties of a few residue images are analyzed.

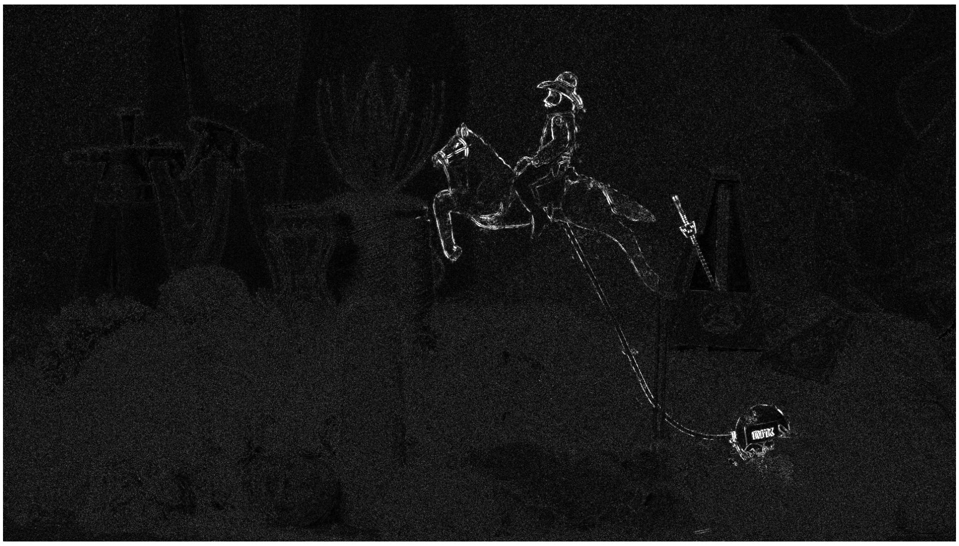
Two interesting sequences to compare, are the Horse and the Vegies sequences provided by EBU, described in Section 3.2.2. The Vegies sequence needed a bit rate of 105% for P-frames, over four times that of the Horse sequence. Figure 5.1 is a visualization of the estimation error.

As presented earlier in Figure 4.1, the scaling of the residue image has a negative effect on the performance for the Vegies sequence. The Vegies frame was scaled with a factor of 0.7, while the Horse frame was scaled with a factor of 0.8. All analysis of the residual images has been performed after the scaling.

From Figure 5.1, it can be seen that both sequences suffer from estimation errors in areas with motion. In the Horse frame there are defined edges between the moving objects and the background, which suffer from estimation error. Also, there is some estimation error for the background. For the Vegies frame, there are some



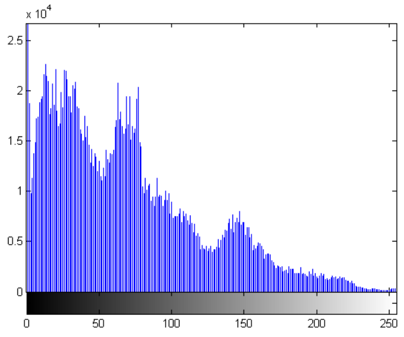
(a) Vegies - frame 13



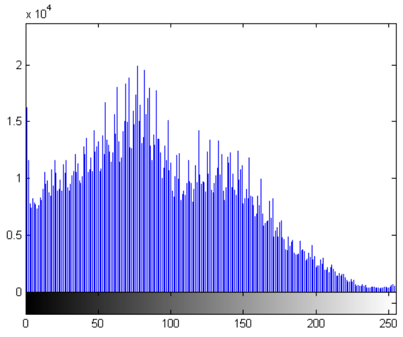
(b) Horse - frame 1

Figure 5.1: Visualization of estimation error where white indicates error. Vector field resolution of 240x160 and half pixel accuracy.

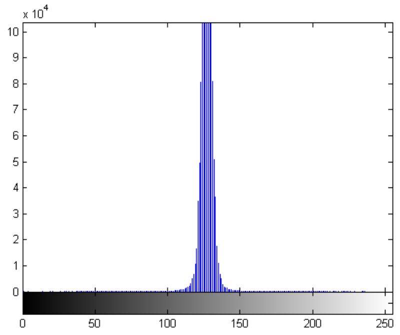
estimation errors on the edges between the objects and the background. There are also some errors one the surface of the vegetables due to the rotational movement.



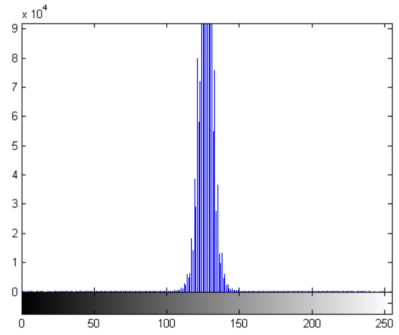
(a) Vegies: original, entropy=7.4, std=53



(b) Horse: original, entropy=7.67, std=54



(c) Vegies: residue, entropy=3.7, std=3.9



(d) Horse: residue, entropy=4.1, std=5.0

Figure 5.2: Histogram of uncompressed and corresponding residue images from the Horse (frame 1) and Vegies (frame 13) sequences. Vector field resolution of 240x160 and half pixel accuracy.

Figure 5.2 shows the distribution of the pixel values in the frames. From the histograms, it is clear that the motion compensations packs the pixels around the value 127 (no estimation error). The motion compensation appears to perform equally well for both sequences. All in all, both sequences seem to suffer from the same amount of estimation error.

Since the distribution of pixel values gives no indication of the difference in performance, an analysis of the wavelet coefficients is necessary. A two-level wavelet decomposition was done on both the original and the residue image. In Figure 5.3 and 5.4 the change in the wavelet coefficients for the different subbands is shown. Details can be found in Appendix C.

The only subband where the reduction in the standard deviation, range and entropy

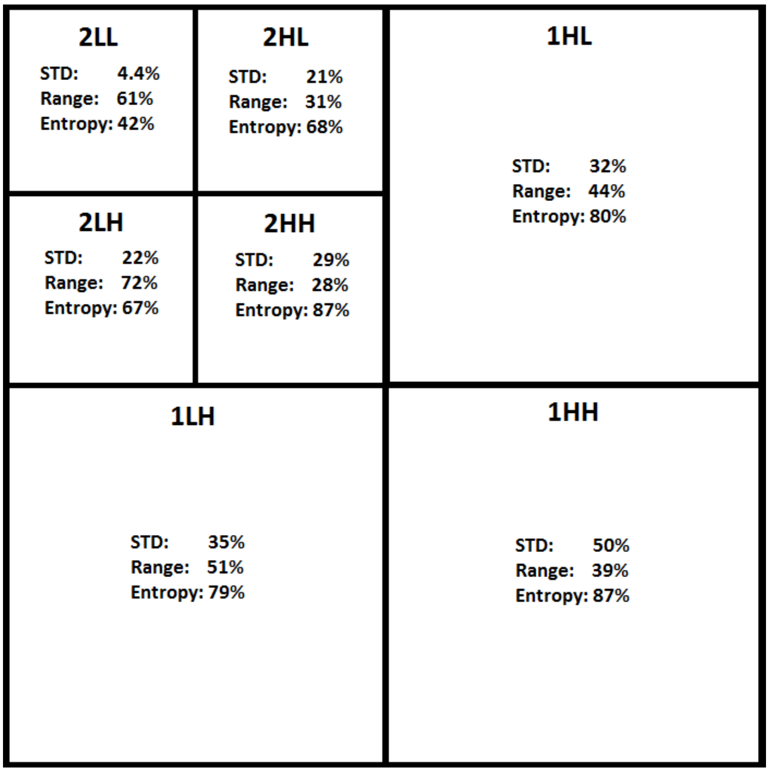


Figure 5.3: Difference in wavelet coefficients for residue and original of Horse frame 1. The ratio between the residue and the original is shown for standard deviation (STD), range of coefficients, and entropy of coefficients.

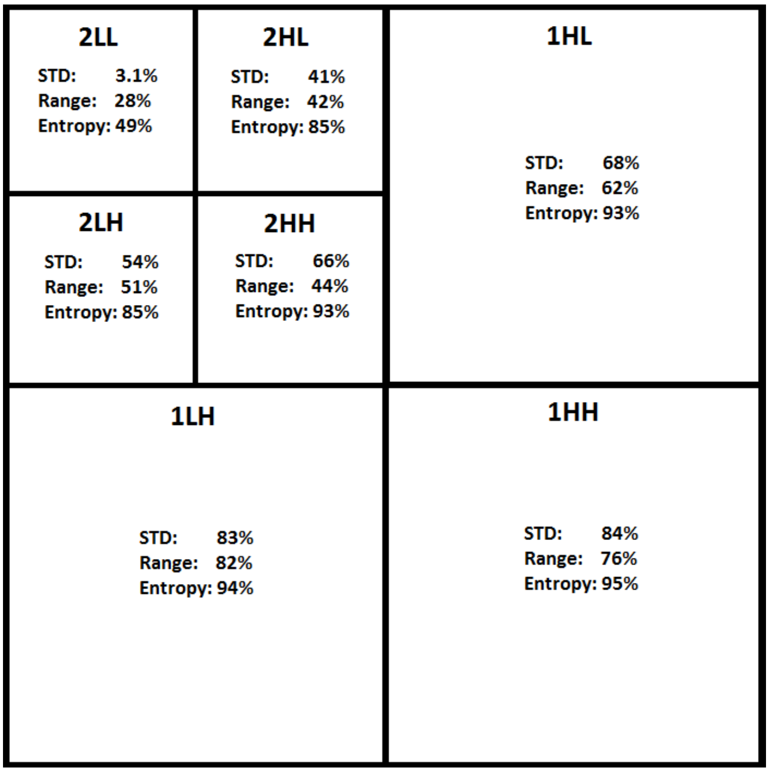


Figure 5.4: Difference in wavelet coefficients for residue and original of Vegies frame 13. The ratio between the residue and the original is shown for standard deviation (STD), range of coefficients, and entropy of coefficients.

are higher for the Vegies frame, is the 2LL sub band. The wavelet coefficients in this subband, representing the low frequency parts of the image, only counts for one-sixteenth of the coefficients. The motion compensation works better for reducing the standard deviance, range and entropy for all the other subbands. One exception is higher decrease of the range for the 2LH sub band. With this analysis it is possible to see that the motion compensation has higher reduction in most of the subbands for the Horse sequence, than the Vegies sequence. However, as described in Section 3.5.3, this analysis does not accurately describe why the Horse sequence achieves better bit rate than the Vegies sequence, it only gives an indication.

Table 5.1: Wavelet analysis - Vector field 240x135 - Half pixel accuracy - GOP size of 2 frames - Range, Entropy and STD given for residue frame, as a percentage of original frame.

	Bunny	*Crowdrun*	*Ducks*	*IntoTree*	*OldTown*	*Parkjoy*	*Vegies*	*Horse*
PSNR (dB)	1.61	1.12	-0.36	0.65	0.65	1.04	-0.08	3.18
Range	80%	69%	79%	70%	61%	102%	76%	47%
Entropy	74%	85%	84%	94%	92%	86%	96%	79%
STD	61%	53%	61%	83%	77%	56%	79%	36%

In Table 5.1 the average difference between the the original frames, and the residue frames, is shown for each sequence. There is no distinct correlation between the increase in PSNR, and the average differences. In general a reduction in range and variance should reduce the quantization errors, and thus increase the PSNR. However, the JPEG 2000 compression scheme is more complication than this approximation.

5.2 Optical flow and motion compensation

In general, optical flow algorithms try to find the actual motion in a sequence of images. This is different from standard motion estimation, that only tries to minimize the estimation error with respect to a cost function.

In Section 5.1 it was shown that the optical flow algorithm reduces entropy of the residual frame, and in Appendix D the estimation error for each sequence is visualized.

5.2.1 Noise

A good optical flow algorithm will not let noise affect the calculations of the flow. As demonstrated in Section 4.2, motion compensation in the presence of noise can lead to estimation error, as the vectors only move the noise around instead of compensating for it. As an example, assume that the pixels in a homogeneous region suffer from independent Gaussian noise with a zero mean, $N(0, \sigma^2)$. If motion compensation is performed between two arbitrary pixels in this region, the residue pixel will then have a distribution of $N(0, 2\sigma^2)$. As there is no correlation in noise, this will reduce the efficiency of the JPEG 2000 encoder, and thus the performance of the proposed hybrid codec.

This effect can be reduced by adding a in-loop denoise filter, which increased the performance for most sequences. This shows that an effective way of addressing the noise problem can reduce the bit rate further.

5.2.2 Occlusion

Regions that have been occluded are obviously difficult to estimate, since these parts of the image are not present in the previous frame. In these situations block based motion compensation would have searched for the best match in available anchor-frames. Optical flow however, will not calculate valid vectors, and thus experience high estimation error. This effect can easily be seen in the estimation error for the sequences Ducks and Parkjoy, and will cause very high frequency components in the residue image, and reduce the performance of the JPEG 2000 encoding.

5.2.3 Performance of optical flow algorithm

In addition to errors in occluded regions, there are errors at the edge of objects, even if the motion is global. This can be seen in the Bunny sequence, where there are errors at the edge between the trees and the sky. These errors are minimized with increased resolution of the vector field, and when interpolation is used to calculate sub-pixel values. The lowest estimation error is achieved when the optical flow vectors are calculated for the original resolution, and with quarter pixel precision. However, the bit rate for the vector field will then be larger than the bit rate for I-frames. To achieve an overall reduction in bit rate it is therefore necessary to use a vector field calculated for downsized images.

In the Middlebury Evaluation there are several others algorithms that perform better than 'Classic+NL', which is the one used in the proposed hybrid codec [28]. Even though the average endpoint error ranges from 0.09 to 0.65 pixels for the highest ranking algorithm (MDP-Flow2), many pixels still have poorer estimation: 16% of the pixels had an endpoint error above 1 pixel, and 9.24% had an endpoint error over 2 pixels. The best optical flow algorithms available, does not have perfect estimation.

5.2.4 Summary

With an improved optical flow algorithm and motion estimation, the estimation error would decrease, resulting in better performance for the JPEG 2000 encoding.

Since the bit rate of the most accurate vector field generated by the 'Classic+NL' algorithm exceeded that of the reference codec, it is unclear what the bit rate would have been for an improved vector field. However, even with a perfect optical flow algorithm, the impact occluded regions and noise have on the estimation error would still be a problem, and must be addressed.

5.3 Evaluation of the proposed codec

To assess the quality and performance of the proposed hybrid codec, it is compared to the performance of the reference codec in the contribution scenario described in Section 3.1.

5.3.1 Delay and complexity

The proposed hybrid encoder has considerable higher complexity than the reference encoder. The calculation of the optical flow is a time consuming process at full resolution, ranging from a few minutes to half an hour depending on the content. Calculating the optical flow for the resolution 240x180 takes between 7 and 15 seconds. Even though this is a software implementation in Matlab, and can be optimized in hardware, the T-Vips TVG450 gateway only has a delay of 60 ms per frame for 1080p50 content [25]. In addition to being a CPU-intensive process, the memory requirements of the optical flow algorithm are also large. Matlab was registered to use over 4GB of RAM during some of the calculations.

Not only the calculation of the vector field, but also the motion compensation requires additional processing time and requirements. To perform motion estimation and compensation, the anchor frame or frames need to be buffered. Extra processing power is also necessary to perform interpolation when sub-pixel accuracy is used.

5.3.2 Error resilianse

The reference codec is very robust against transmission errors. If there is an transmission error, for example in the form of a packet loss, it will only cause a degradation in quality for the corresponding frame. With the GOP structure proposed for the hybrid encoder, a packet loss can cause degradation in quality in all the remaining frames in the GOP. These errors can be minimized by error detection and correction techniques. Regardless, the proposed codec will be more vulnerable to transmission errors than the reference codec.

5.3.3 Bit rate

The proposed hybrid codec achieves an average reduction in bit rate of 11.7% over all sequences, while maintaining the same PSNR and SSIM. If the codec is allowed to switch to intra-only mode when deemed nesecarry, the reduction will increase to 18.8%.

For IP-networks a constant bit rate is often a desired feature. As an example, assume that a constant bit rate is set 10% lower than the bit rate in the reference codec. Only half of the sequences will in this case have the same or improved PSNR. The other half will suffer from a degradation in quality. Therefore, the proposed hybrid codec can not have a constant bit rate that is lower than the reference codec, while matching the quality for all types of content. However, if the codec can switch to I-frames when deemed necessary, the proposed codec will have the same or higher PSNR at the same bit rate, regardless of the content.

5.3.4 Possible improvements

There are several improvements that can be made to the proposed hybrid codec that could reduce the bit rate further. Some of them are presented below.

- As the lossless compression of the vector field uses a generic file format and compression, it is possible to reduce the bit rate of the vector field by changing to a compression algorithm suited for vector fields. Differential encoding of the vector field could also reduce the bit rate further.
- An improved optical flow algorithm, or one designed for motion compensation, can reduce the estimation error, especially the high frequency errors at the edges of objects.
- In addition to optical flow, there should be extra motion compensation in areas with high estimation errors, which often arises from occluded regions.
- Even though JPEG 2000 encoders are very good at maximizing quality at a given bit rate, the knowledge of the residue images could further improve this maximization.
- Either an improvement of the in-loop denoise filter, or find other ways to combat the effects of noise.

5.4 Hybrid video codec with wavelet transform

Instead of using optical flow for motion estimation, it is also possible to use block based techniques. As explained in Section 2.3.1 the block based motion compensation that does not introduce blocking artifacts is the OBMC. Hybrid codecs that combine OBMC and wavelet transformation have been proposed earlier [34, 35]. If the motion estimation is performed in the wavelet domain, blocking artifacts will not be an issue, and techniques that employ wavelet domain motion compensation have been proposed [21]. These techniques have in common that they do not match the performance of the best DCT-based hybrid codecs.

The next generation video codec HEVC, uses VBMC and integer transforms which causes blocking artifacts. These artifacts are however minimized by introducing advanced in-loop deblocking filters. For example, the in-loop deblocking filter found in H.264/AVC improves the subjective quality and the PSNR [36].

In addition to deblocking, several other techniques can be employed by DCT based codecs, that are not practical with wavelet decomposition. One technique is the option to discard encoding of blocks with little information. This is not possible if the transform is performed on larger areas, as is the case in JPEG 2000. Another is the use of intra prediction which can further exploit the spatial redundancy. In fact, by employing these techniques and more, some reports suggest that the H.264/AVC High 4:4:4 Intra Profile perform similar to JPEG 2000, and this is without exploiting the temporal redundancy [37, 38].

The emerging HEVC standard aims at a bit rate half of H.264/AVC with the same quality. From the HEVC standard, it can be seen that the video compression community see most potential in hybrid codecs with assorted block based motion compensation techniques and DCT integer transforms [39, 40].

Chapter 6

Conclusions

A hybrid codec that uses optical flow for motion estimation and JPEG 2000 for coding of the estimation error has been presented. The performance of the motion estimation has been evaluated and the estimation error has been analyzed. The overall quality of the codec has been compared to an intra-only JPEG 2000 codec, in a broadcast scenario.

Motion compensation with optical flow reduces the entropy in the residue images, and most pixels will have a low estimation error. Inaccuracies in the optical flow will often cause estimation errors at the edge of objects, which in turn will create high frequency components in the residue image. In addition to these inaccuracies the optical flow algorithm suffer from high estimation error in occluded regions. As optical flow algorithms try not to take noise into consideration when calculating the flow, the energy of the noise may increase after motion compensation. Promising results were achieved by employing an in-loop wiener filter, to reduce the effect of noise.

Even with an optimal optical flow algorithm the estimation error is not suited for being encoded with a wavelet transform, without addressing the problems of noise and occluded regions.

At the same objective quality as the reference codec there were significant differences between the sequences, tested with the settings that gave the best overall reduction in bit rate. The bit rate ranged from a decrease of 90% to an increase of 27%, with an overall reduction of 11.7%. If the codec switches to intra-only coding when deemed necessary, the reduction increases to 18.8%.

There are several disadvantages with the proposed codec in a contribution environment: Increased delay, high complexity, higher vulnerability to transmission error, and the lack of a constant bit stream. Even with the overall reduction in bit rate the proposed codec is not suited to replace the reference codec in an contribution environment.

6.1 Future Work

There are several improvements that can be made to the proposed hybrid codec that could increase the performance, which are described in Section 5.3.4. It is the author's opinion that future work should focus on more promising alternatives, instead of improving this scheme.

First, wavelet based MCTF provides both temporal, spatial and SNR scalability. No research on how this principle behaves in an contribution environment has been found, and an investigations should therefore be performed.

Second, the new HEVC video codec under development aims to achieve a bit rate half that of H.264/AVC, with the same quality. Interestingly, a 'Low Delay' profile has been proposed that may suite some of the requirements in contribution environments [40]. Investigations into HEVC performance in a contribution environment should also be conducted.

References

[1] Deqing Sun, Stefan Roth, and Michael J. Black. Secrets of optical flow estimation and their principles. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:2432–2439, 2010.

[2] Nielsen Company. Average U.S. Home Now Receives a Record 118.6 TV Channels, According to Nielsen, 2008.

[3] A. Morello and V. Mignone. DVB-S2: The Second Generation Standard for Satellite Broad-Band Services. *Proceedings of the IEEE*, 94(1):210 –227, jan. 2006.

[4] Gary Demos. Layered motion compensation for moving image compression. *SMPTE Motion Imaging Journal*, 2009.

[5] Berthold K.P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(1-3):185 – 203, 1981.

[6] James J. Gibson. *The perception of the visual world.* Houghton Mifflin, 1951.

[7] Bruce D. Lucas and Takeo Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. pages 674–679, 1981.

[8] Andrés Bruhn, Joachim Weickert, and Christoph Schnörr. Lucas/kanade meets horn/schunck: Combining local and global optic flow methods. *International Journal of Computer Vision*, 61:211–231, 2005.

[9] Michael J. Black and P. Anandan. The robust estimation of multiple motions: parametric and piecewise-smooth flow fields. *Comput. Vis. Image Underst.*, 63:75–104, January 1996.

[10] Andreas Wedel, Thomas Pock, Christopher Zach, Horst Bischof, and Daniel Cremers. An improved algorithm for TV-L1 optical flow. *Dagstuhl Motion Workshop*, 2008.

[11] P. Moulin, R. Krishnamurthy, and J.W. Woods. Multiscale modeling and estimation of motion fields for video coding. *Image Processing, IEEE Transactions on*, 6(12):1606 –1620, dec 1997.

[12] Y. Q. Shi, S. Lin, and Ya-Qin Zhang. Optical flow-based motion compensation algorithm for very low-bit-rate video coding. *International Journal of Imaging Systems and Technology*, 9(4):230–237, 1998.

[13] Kunal Mukherjee and Amar Mukherjee. Joint Optical Flow Motion Compensation and Video Compression using Hybrid Vector Quantization. *Data Compression Conference*, 0:541, 1999.

[14] Elena Alshina, Alexander Alshin, and Woo-Jin Han. CE1: Samsung's test for bi-directional optical flow. *Input Document to JCT-VC, JCTVC-D329*, 2011.

[15] P. Schelkens, A. Skodras, and T. Ebrahimi. *The JPEG 2000 Suite*. Wiley-IS&T series in imaging science and technology. Wiley, 2009.

[16] D.S. Taubman and M.W. Marcellin. *JPEG2000: image compression fundamentals, standards, and practice*. Number v. 1 in Kluwer international series in engineering and computer science. Kluwer Academic Publishers, 2002.

[17] Discrete wavelet transform - a 3 level filter bank. `http://en.wikipedia.org/wiki/File:Wavelets_-_Filter_Bank.png`, June 13, 2011.

[18] Jpeg2000 2-level wavelet transform. `http://en.wikipedia.org/wiki/File:Jpeg2000_2-level_wavelet_transform-lichtenstein.png`, June 13, 2011.

[19] ITU-T and ISO/IEC 11496-10 (MPEG-4). Advanced video coding for generic audiovisual services. *Recommendation H.264 (03/10), version 5.0*, March 2010.

[20] ITU-T. Video coding for low bit rate communication. *Recommendation H.263 (01/05)*, January 2005.

[21] A. Suliman and R. Li. Motion compensation in redundant wavelet domain. In *Southeastcon, 2011 Proceedings of IEEE*, pages 390 –394, march 2011.

[22] J.-R. Ohm. Three-dimensional subband coding with motion compensation. *Image Processing, IEEE Transactions on*, 3(5):559 –571, sep 1994.

[23] Wen-hsiao Peng and Chia-yang Tsai and Tihao Chiang and Hsueh-ming Hang. Advances of mpeg scalable video coding standard. In *Knowledge-Based Intelligent Information & Engineering Systems*, pages 889–895, 2005.

[24] T-Vips AS. TVG430 HD JPEG2000 Gateway. `http://www.t-vips.com/sites/default/files/datasheets/datasheet_tvg430.pdf`.

[25] T-Vips AS. TVG450 JPEG2000 Gateway. `http://www.t-vips.com/sites/default/files/datasheet_tvg450.pdf`.

[26] Janne T. Morstøl, Helge Stephansen, and Ivar Rognstad. Can JPEG2000 solve the challenge of HDTV Contribution over IP? `http://www.t-vips.com/sites/default/files/whitepapers/Whitepaper_JPEG2000.pdf`, March 2009.

[27] Processing EBU's YUV files with FFmpeg. `http://www.tribler.org/trac/wiki/FfmpegYuv`, 2010.

[28] Flow accuracy and interpolation evaluation. `http://vision.middlebury.edu/flow/eval/`.

[29] S. Baker, D. Scharstein, J.P. Lewis, S. Roth, M.J. Black, and R. Szeliski. A Database and Evaluation Methodology for Optical Flow. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1 –8, oct. 2007.

[30] `http://www.cs.brown.edu/people/dqsun/`, June 2011.

[31] Peter G. J. Barten. *Contrast Sensitivity of the Human Eye and Its Effects on Image Quality.* SPIE Publications, first edition, 1999.

[32] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13(4):600 –612, april 2004.

[33] MeTriX MuX Visual Quality Assessment Package (v.1.1). `http://foulard.ece.cornell.edu/gaubatz/metrix_mux/`.

[34] Ji Zhongwei, Jiang Wenjun, and Zhu Weile. Wavelet-based video coding using adaptive overlapped block motion compensation. In *Communications, Circuits and Systems and West Sino Expositions, IEEE 2002 International Conference on*, volume 2, pages 1090 – 1093 vol.2, june-1 july 2002.

[35] S.A. Martucci, I. Sodagar, T. Chiang, and Ya-Qin Zhang. A zerotree wavelet video coder. *Circuits and Systems for Video Technology, IEEE Transactions on*, 7(1):109 –118, feb 1997.

[36] P. List, A. Joch, J. Lainema, G. Bjontegaard, and M. Karczewicz. Adaptive deblocking filter. *Circuits and Systems for Video Technology, IEEE Transactions on*, 13(7):614 –619, july 2003.

[37] Francesca De Simone, Mourad Ouaret, Frederic Dufaux, Andrew G. Tescher, and Touradj Ebrahimi. A comparative study of JPEG 2000, AVC/H.264, and HD Photo. In *SPIE Optics and Photonics, Applications of Digital Image Processing XXX*, volume 6696, 2007.

[38] A. Al, B.P. Rao, S.S. Kudva, S. Babu, D. Sumam, and A.V. Rao. Quality and complexity comparison of h.264 intra mode with jpeg2000 and jpeg. In *Image Processing, 2004. ICIP '04. 2004 International Conference on*, volume 1, pages 525 – 528 Vol. 1, oct. 2004.

[39] Gary J. Sullivan and Jens-Rainer Ohm. Meeting report of the first meeting of the Joint Collaborative Team on Video Coding (JCT-VC), Dresden, DE, 15-23 April, 2010. *Output Document Approved by JCT-VC*, 2010.

[40] Ken McCann, Benjamin Bross, Shun ichi Sekiguchi, and Woo-Jin Han. HM3: High Efficiency Video Coding (HEVC) Test Model 3 Encoder Description. *Output Document of JCT-VC (draft000), JCTVC-E602*, 2011.

Appendices

Appendix A

Results

Table A.1: PSNR (dB) difference: Optical flow calculated with one pixel accuracy.

Resolution	*Bunny*	*Crowdrun*	*Ducks*	*IntoTree*	*OldTown*	*Parkjoy*	*Vegies*	*Horse*	*Average*
1920x1080	1.56	0.83	-0.80	0.19	0.25	0.73	-0.54	3.32	0.88
960x540	1.62	0.86	-0.83	0.15	0.21	0.76	-0.57	3.34	0.88
480x270	1.58	0.69	-0.90	0.16	0.22	0.64	-0.53	3.17	0.81
320x180	1.60	0.58	-0.94	0.15	0.24	0.50	-0.56	3.15	0.77
240x135	1.49	0.49	-0.97	0.14	0.24	0.37	-0.58	3.13	0.72
160x90	1.35	0.37	-0.96	0.11	0.22	0.14	-0.84	3.16	0.63
differential	-2.50	-0.03	-0.50	-0.69	-0.33	-1.97	-1.14	3.20	-0.16

Table A.2: PSNR (dB) difference: Optical flow calculated with half a pixel accuracy.

Resolution	*Bunny*	*Crowdrun*	*Ducks*	*IntoTree*	*OldTown*	*Parkjoy*	*Vegies*	*Horse*	*Average*
1920x1080	1.71	1.55	-0.20	0.76	0.70	1.42	0.05	3.46	1.32
960x540	1.76	1.55	-0.23	0.67	0.65	1.44	-0.02	3.39	1.29
480x270	1.72	1.36	-0.26	0.68	0.65	1.31	0.00	3.27	1.22
320x180	1.71	1.22	-0.30	0.67	0.65	1.18	-0.04	3.19	1.16
240x135	1.61	1.12	-0.36	0.65	0.65	1.04	-0.08	3.18	1.10
160x90	1.43	0.99	-0.40	0.62	0.62	0.80	-0.32	3.16	1.00

Table A.3: PSNR (dB) difference: Optical flow calculated with a quarter pixel accuracy.

Resolution	*Bunny*	*Crowdrun*	*Ducks*	*IntoTree*	*OldTown*	*Parkjoy*	*Vegies*	*Horse*	*Average*
1920x1080	1.58	1.60	-0.17	0.79	0.72	1.46	0.07	3.46	1.33
960x540	1.62	1.69	-0.15	0.76	0.69	1.48	0.04	3.41	1.33
480x270	1.58	1.51	-0.18	0.74	0.69	1.35	0.05	3.30	1.26
320x180	1.57	1.32	-0.20	0.73	0.69	1.24	0.03	3.18	1.19
240x135	1.54	1.23	-0.24	0.72	0.70	1.12	-0.02	3.19	1.15
160x90	1.37	1.09	-0.30	0.68	0.67	0.89	-0.24	3.14	1.04

Table A.4: Bit rate - Vector field 240x135 - Half pixel accuracy - GOP size of 2 frames.

	Bunny	*Crowdrun*	*Ducks*	*IntoTree*	*OldTown*	*Parkjoy*	*Vegies*	*Horse*	*Average*
Residue	73.0%	79.8%	108.8%	80.4%	78,4%	83.8%	103.1%	23.9%	78,9%
Vector	1.3%	5.5%	5.6%	2.3%	2.0%	4.4%	2.5%	1.1%	3.1%
SUM	74.3%	85.3%	114.4%	82.8%	80.4%	88.2%	105.6%	25.1%	82.0%
SUM optimal	74.3%	85.3%	100.0%	82.8%	80.4%	88.2%	100.0%	25.1%	79.5%

Table A.5: Bit rate - Vector field 240x135 - Half pixel accuracy - GOP size of 30 frames.

	Bunny	*Crowdrun*	*Ducks*	*IntoTree*	*OldTown*	*Parkjoy*	*Vegies*	*Horse*	*Average*
Residue	84.6%	82.5%	121.5%	82.9%	89.3%	87.1%	121.7%	8.9%	84.8%
Vector	1.3%	5.5%	5.6%	2.3%	1.9%	4.4%	2.5%	1.1%	3.1%
SUM	85.9%	88.0%	127.1%	85.2%	91.3%	91.4%	124.1%	10.0%	87.9%
SUM optimal	85.9%	88.0%	100.0%	85.2%	91.3%	91.4%	100.1%	10.0%	81.5%

Table A.6: Bit rate - Vector field 240x135 - Half pixel accuracy - GOP size of 30 frames - Flow calculated from decoded picture.

	Bunny	*Crowdrun*	*Ducks*	*IntoTree*	*OldTown*	*Parkjoy*	*Vegies*	*Horse*	*Average*
Residue	84,4%	85,8%	119,6%	84,3%	90,4%	93,0%	122,2%	9,5%	86,1%
Vector	1,9%	4,3%	3,9%	2,3%	2,0%	4,1%	2,4%	1,1%	2,8%
SUM	86,4%	90,1%	123,5%	86,6%	92,4%	97,1%	124,6%	10,6%	88,9%
SUM optimal	86,4%	90,1%	100,0%	86,6%	92,4%	97,1%	100,0%	10,6%	82,9%

Appendix B

Matlab code

B.1 parse_results.m

```
function [] = parse_results(PSNR,SSIM,filename,file_descripion,CSV,...
    SSIMerrorbar,PSNRerrorbar)
if(size(PSNR) ≠ [2 ,240])
    warning('PSNR has wrong size, must be [2,240]');
elseif(size(SSIM) ≠ [2 ,240])
    warning('SSIM has wrong size, must be [2,240]');
else

    %Other values not supported without changing code
    frames_pr_sequence=15;
    number_of_sequences=8;

    % 13 degrees of freedom — Students t
    confidence_factor_sequence= ...
        2.145/sqrt(frames_pr_sequence);

    % 119 degrees of freedom — Students t
    confidence_factor_total= ...
        1.980/sqrt(frames_pr_sequence*number_of_sequences);

    sequence_names=char('Bunny', 'Crowdrun', 'Ducks', ...
        'IntoTree', 'OldTown', 'Parkjoy', 'Vegies', 'Horse');

    PSNR_calc=zeros(4,number_of_sequences+1);
    SSIM_calc=zeros(4,number_of_sequences+1);

    %PSNR Calculation
    PSNR=10.^(PSNR./10);
    PSNR_tmp=PSNR(1,:)./PSNR(2,:);
    PSNR_tmp=PSNR_tmp(2:2:end);

    for x=1:number_of_sequences;
```

```
        PSNR_calc(1,x)=mean(PSNR_tmp((x−1)*frames_pr_sequence+1 : ...
            x*frames_pr_sequence));

        PSNR_calc(4,x)=std(PSNR_tmp((x−1)*frames_pr_sequence+1 : ...
            x*frames_pr_sequence));

        PSNR_calc(2,x)=PSNR_calc(1,x)−confidence_factor_sequence * ...
            PSNR_calc(4,x);

        PSNR_calc(3,x)=PSNR_calc(1,x)+confidence_factor_sequence * ...
            PSNR_calc(4,x);
    end

    PSNR_calc(1,end)=mean(PSNR_tmp);
    PSNR_calc(4,end)=std(PSNR_tmp);

    PSNR_calc(2,end)=PSNR_calc(1,end)−confidence_factor_total * ...
        PSNR_calc(4,end);

    PSNR_calc(3,end)=PSNR_calc(1,end)+confidence_factor_total * ...
        PSNR_calc(4,end);

    PSNR_calc=10*log10(PSNR_calc);

    %SSIM Calculation
    SSIM_tmp=SSIM(1,:)−SSIM(2,:);
    SSIM_tmp=SSIM_tmp(2:2:end);

    for x=1:number_of_sequences;
        SSIM_calc(1,x)=mean(SSIM_tmp((x−1)*frames_pr_sequence+1 : ...
            x*frames_pr_sequence));

        SSIM_calc(4,x)=std(SSIM_tmp((x−1)*frames_pr_sequence+1 : ...
            x*frames_pr_sequence));

        SSIM_calc(2,x)=SSIM_calc(1,x)− ...
            confidence_factor_sequence * SSIM_calc(4,x);

        SSIM_calc(3,x)=SSIM_calc(1,x)+...
            confidence_factor_sequence * SSIM_calc(4,x);
    end

    SSIM_calc(1,end)=mean(SSIM_tmp);
    SSIM_calc(4,end)=std(SSIM_tmp);

    SSIM_calc(2,end)=SSIM_calc(1,end)− ...
        confidence_factor_total * SSIM_calc(4,end);

    SSIM_calc(3,end)=SSIM_calc(1,end)+ ...
        confidence_factor_total * SSIM_calc(4,end);

    %Write CSV−file
    if (CSV==1)
        fw = fopen(filename, 'w');

        fprintf(fw,'Description:;');
        fprintf(fw,file_descripion);
        fprintf(fw,'\n\n');
        fprintf(fw,'PSNR\n');
        fprintf(fw,'Sequence;Lower (dB);Mean (dB);Upper (dB)\n');

        for x=1:number_of_sequences
            fprintf(fw,'%s',sequence_names(x,:));   fprintf(fw,';');
            fprintf(fw,'%f',PSNR_calc(2,x));        fprintf(fw,';');
            fprintf(fw,'%f',PSNR_calc(1,x));        fprintf(fw,';');
            fprintf(fw,'%f',PSNR_calc(3,x));        fprintf(fw,'\n');
        end
        fprintf(fw,'Total;');
        fprintf(fw,'%f',PSNR_calc(2,end));      fprintf(fw,';');
        fprintf(fw,'%f',PSNR_calc(1,end));      fprintf(fw,';');
        fprintf(fw,'%f',PSNR_calc(3,end));      fprintf(fw,'\n');


        fprintf(fw,'\nSSIM\n');
        fprintf(fw,'Sequence;Lower;Mean;Upper\n');
        for x=1:number_of_sequences
            fprintf(fw,'%s', sequence_names(x,:));  fprintf(fw,';');
            fprintf(fw,'%f',SSIM_calc(2,x));        fprintf(fw,';');
            fprintf(fw,'%f',SSIM_calc(1,x));        fprintf(fw,';');
            fprintf(fw,'%f',SSIM_calc(3,x));        fprintf(fw,'\n');
        end
        fprintf(fw,'Total;');
        fprintf(fw,'%f',SSIM_calc(2,end));      fprintf(fw,';');
        fprintf(fw,'%f',SSIM_calc(1,end));      fprintf(fw,';');
        fprintf(fw,'%f',SSIM_calc(3,end));      fprintf(fw,'\n');

        fclose(fw);
    end

    %Create and save errorbar plot of SSIM difference
    if (SSIMerrorbar==1)
        errorbar(1:number_of_sequences, SSIM_calc(1,1:end-1), ...
            SSIM_calc(2,1:end-1)-SSIM_calc(1,1:end-1), ...
            SSIM_calc(3,1:end-1)-SSIM_calc(1,1:end-1), 'o');
        hold on;
        grid on;
        plot(0:number_of_sequences+1,zeros(1,number_of_sequences+2),...
            'color', 'black');
        set(gca,'XTickLabel',{' ',sequence_names,' '});
        set(gca,'FontSize',14);
        set(gcf,'Color',[1.0 1.0 1.0]);
        set(gcf,'Position',[100,100,1024,768]);
        ylabel('SSIM difference');
        frame=getframe(gcf);
        [X,map]=frame2im(frame);
        close;
        imwrite(X,'SSIMdiff.png');

    end

    %Create and save errorbar plot of PSNR difference
    if (PSNRerrorbar==1)
        errorbar(1:number_of_sequences, PSNR_calc(1,1:end-1),...
            PSNR_calc(2,1:end-1)-PSNR_calc(1,1:end-1), ...
            PSNR_calc(3,1:end−1)−PSNR_calc(1,1:end−1), 'o');
        hold on;
        grid on;
        plot(0:number_of_sequences+1,zeros(1,number_of_sequences+2),...
            'color', 'black');
        set(gca,'XTickLabel',{' ',sequence_names,' '});
        set(gca,'FontSize',14);
        set(gcf,'Color',[1.0 1.0 1.0]);
        set(gcf,'Position',[100,100,1024,768]);
        ylabel('PSNR (dB) difference');
        frame=getframe(gcf);
        [X,map]=frame2im(frame);
        close;
        imwrite(X,'PSNRdiff.png');

    end

end
end
```

B.2 wavelet_decomp.m

```matlab
function [out] = wavelet_decomp(input_I,input_P)
input_I=strcat(input_I,'.pgm');
input_P=strcat(input_P,'.pgm');

%Read image files
I=(imread(input_I));
P=(imread(input_P));

%Normalization
I=im2double(I);
P=im2double(P);

%Level offset
I=I-0.5;
P=P-0.5;

%Filters in JPEG 2000
Lo_D=[0.0267 -0.0168 -0.0782 0.2668 0.6029 0.2668 -0.0782 -0.0168 ...
    0.0267];
Hi_D=[0.0912 -0.0575 -0.5912 1.1150 -0.5912 -0.0575 0.0912];

%Calculate the 2-level Wavelet transform
[scaledI, verticalI, horizontalI, diagonalI]=dwt2(I,Lo_D,Hi_D);

[scaledI2, verticalI2, horizontalI2, diagonalI2]= ...
    dwt2(scaledI, Lo_D,Hi_D);

[scaledP, verticalP, horizontalP, diagonalP]= ...
    dwt2(P, Lo_D, Hi_D);

[scaledP2, verticalP2, horizontalP2, diagonalP2]= ...
    dwt2(scaledP, Lo_D,Hi_D);

%Concatenate matrix so total mean, max, min, and std can be calculated
scaledI=catmat(scaledI);
verticalI=catmat(verticalI);
horizontalI=catmat(horizontalI);
diagonalI=catmat(diagonalI);

scaledI2=catmat(scaledI2);
verticalI2=catmat(verticalI2);
horizontalI2=catmat(horizontalI2);
diagonalI2=catmat(diagonalI2);

scaledP=catmat(scaledP);
verticalP=catmat(verticalP);
horizontalP=catmat(horizontalP);
diagonalP=catmat(diagonalP);

scaledP2=catmat(scaledP2);
verticalP2=catmat(verticalP2);
horizontalP2=catmat(horizontalP2);
diagonalP2=catmat(diagonalP2);
```

```

%Calculate Mean, STD, Min, Mac, Range and Entropy for all sub bands.

strings='   Mean        STD        Min        Max        Range     Entropy'...
    ;
disp(strcat('Original image:',input_I));
disp(strcat('Residue image:',input_P));

disp('----------------------------------1LL------------------------------------');
disp(strings);

mxI=max((scaledI));
mxP=max((scaledP));
miI=min((scaledI));
miP=min((scaledP));

disp([mean((scaledI)),std((scaledI)),miI,mxI,mxI-miI, ...
    entropy(scaledI)]);

disp([mean((scaledP)),std((scaledP)),miP,mxP,mxP-miP, ...
    entropy(scaledP2)]);

disp('----------------------------------2LL------------------------------------');
disp(strings);

mxI=max((scaledI2));
mxP=max((scaledP2));
miI=min((scaledI2));
miP=min((scaledP2));

I_RANGE=(mxI-miI)*0.0625;
P_RANGE=(mxP-miP)*0.0625;

I_ENT=entropy(scaledI2)*0.0625;
P_ENT=entropy(scaledP2)*0.0625;
STD_ratio=0.0625*(std((scaledP2))/std((scaledI2)));

disp([mean((scaledI2)),std((scaledI2)),miI,mxI,mxI-miI, ...
    entropy(scaledI2)]);

disp([mean((scaledP2)),std((scaledP2)),miP,mxP,mxP-miP, ...
    entropy(scaledP2)]);

disp('----------------------------------1HL------------------------------------');
disp(strings);

mxI=max((verticalI));
mxP=max((verticalP));
miI=min((verticalI));
miP=min((verticalP));

I_RANGE=I_RANGE+(mxI-miI)*0.25;
P_RANGE=P_RANGE+(mxP-miP)*0.25;

I_ENT=I_ENT+entropy(verticalI)*0.25;
P_ENT=P_ENT+entropy(verticalP)*0.25;
STD_ratio=STD_ratio + 0.25*(std((verticalP))/std((verticalI)));
```

```

disp([mean((verticalI)),std((verticalI)),miI,mxI,mxI-miI, ...
    entropy(verticalI)]);

disp([mean((verticalP)),std((verticalP)),miP,mxP,mxP-miP, ...
    entropy(verticalP)]);

disp('-----------------------------2HL-------------------------------');
disp(strings);

mxI=max((verticalI2));
mxP=max((verticalP2));
miI=min((verticalI2));
miP=min((verticalP2));

I_RANGE=I_RANGE+(mxI-miI)*0.0625;
P_RANGE=P_RANGE+(mxP-miP)*0.0625;

I_ENT=I_ENT+entropy(verticalI2)*0.0625;
P_ENT=P_ENT+entropy(verticalP2)*0.0625;
STD_ratio=STD_ratio + 0.0625*(std((verticalP2))/ std((verticalI2)));

disp([mean((verticalI2)),std((verticalI2)),miI,mxI,mxI-miI, ...
    entropy(verticalI2)]);

disp([mean((verticalP2)),std((verticalP2)),miP,mxP,mxP-miP, ...
    entropy(verticalP2)]);

disp('-----------------------------1LH-------------------------------');
disp(strings);

mxI=max((horizontalI));
mxP=max((horizontalP));
miI=min((horizontalI));
miP=min((horizontalP));

I_RANGE=I_RANGE+(mxI-miI)*0.25;
P_RANGE=P_RANGE+(mxP-miP)*0.25;

I_ENT=I_ENT+entropy(horizontalI)*0.25;
P_ENT=P_ENT+entropy(horizontalP)*0.25;

STD_ratio=STD_ratio + 0.25*(std((horizontalP))/ std((horizontalI)));

disp([mean((horizontalI)),std((horizontalI)),miI,mxI,mxI-miI, ...
    entropy(horizontalI)]);

disp([mean((horizontalP)),std((horizontalP)),miP,mxP,mxP-miP, ...
    entropy(horizontalP)]);

disp('-----------------------------2LH-------------------------------');
disp(strings);

mxI=max((horizontalI2));
mxP=max((horizontalP2));
miI=min((horizontalI2));
miP=min((horizontalP2));
```

```

I_RANGE=I_RANGE+(mxI—miI)*0.0625;
P_RANGE=P_RANGE+(mxP—miP)*0.0625;

I_ENT=I_ENT+entropy(horizontalI2)*0.0626;
P_ENT=P_ENT+entropy(horizontalP2)*0.0625;

STD_ratio=STD_ratio + ...
    0.0625*(std((horizontalP2))/ std((horizontalI2)));

disp([mean((horizontalI2)),std((horizontalI2)),miI,mxI,mxI—miI, ...
    entropy(horizontalI2)]);

disp([mean((horizontalP2)),std((horizontalP2)),miP,mxP,mxP—miP, ...
    entropy(horizontalP2)]);

disp('—————————————————————————————1HH———————————————————————————————');
disp(strings);

mxI=max((diagonalI));
mxP=max((diagonalP));
miI=min((diagonalI));
miP=min((diagonalP));

I_RANGE=I_RANGE+(mxI—miI)*0.25;
P_RANGE=P_RANGE+(mxP—miP)*0.25;

I_ENT=I_ENT+entropy(diagonalI)*0.25;
P_ENT=P_ENT+entropy(diagonalP)*0.25;

STD_ratio=STD_ratio + 0.25*(std((diagonalP))/ std((diagonalI)));

disp([mean((diagonalI)),std((diagonalI)),miI,mxI,mxI—miI, ...
    entropy(diagonalI)]);

disp([mean((diagonalP)),std((diagonalP)),miP,mxP,mxP—miP, ...
    entropy(diagonalP)]);

disp('—————————————————————————————2HH———————————————————————————————');
disp(strings);

mxI=max((diagonalI2));
mxP=max((diagonalP2));
miI=min((diagonalI2));
miP=min((diagonalP2));

I_RANGE=I_RANGE+(mxI—miI)*0.0625;
P_RANGE=P_RANGE+(mxP—miP)*0.0625;

I_ENT=I_ENT+entropy(diagonalI2)*0.0625;
P_ENT=P_ENT+entropy(diagonalP2)*0.0625;

STD_ratio=STD_ratio + 0.0625*(std((diagonalP2))/ std((diagonalI2)));

disp([mean((diagonalI2)),std((diagonalI2)),miI,mxI,mxI—miI, ...
    entropy(diagonalI2)]);

```

```
disp([mean((diagonalP2)),std((diagonalP2)),miP,mxP,mxP-miP, ...
    entropy(diagonalP2)]);
disp('--------------------------AVERAGE-----------------------------');
disp('   RANGE I:  RANGE P:   RANGE I/P:');
disp([I_RANGE,P_RANGE,P_RANGE/I_RANGE]);

disp('   ENT.  I:  ENT.  P:   ENROPY I/P:');
disp([I_ENT,P_ENT,P_ENT/I_ENT]);

disp('   STD %:');
disp(STD_ratio);

out=[I_RANGE,P_RANGE,P_RANGE/I_RANGE,I_ENT,P_ENT,P_ENT/I_ENT, ...
    STD_ratio];

end
```

Appendix C

Matlab script output

C.1 Wavelet decomposition of Vegies frame 13

wavelet_decomp.m script applied to frame 13 of Vegies sequence, and the residue image after motion compensation by a 240x135 resolution vector field with half pixel accuracy.

```
>> wavelet_decomp('194','p194')
Original image:194.pgm
Residue image:p194.pgm
----------------------------1LL-------------------------------
   Mean        STD       Min        Max        Range      Entropy
   -0.2303     0.2064   -0.5274     0.5739     1.1014     1.6575

   -0.0020     0.0099   -0.3899     0.2478     0.6377     0.7987

----------------------------2LL-------------------------------
   Mean        STD       Min        Max        Range      Entropy
   -0.2319     0.2044   -0.5128     0.5314     1.0442     1.6258

   -0.0020     0.0063   -0.1690     0.1203     0.2893     0.7987

----------------------------1HL-------------------------------
   Mean        STD       Min        Max        Range      Entropy
   -0.0000     0.0258   -0.8480     0.8756     1.7237     2.2780

   -0.0000     0.0175   -0.5156     0.5457     1.0613     2.1240

----------------------------2HL-------------------------------
   Mean        STD       Min        Max        Range      Entropy
    0.0001     0.0249   -0.5566     0.7480     1.3046     1.9137

    0.0001     0.0101   -0.2975     0.2523     0.5498     1.6254

----------------------------1LH-------------------------------
   Mean        STD       Min        Max        Range      Entropy
   -0.0029     0.0138   -0.2889     0.2994     0.5883     1.7769
   -0.0024    0.0115   -0.2728    0.2073    0.4801    1.6791

----------------------------2LH-------------------------------
   Mean       STD       Min       Max       Range     Entropy
    0.0001    0.0156   -0.3789    0.3667    0.7456    1.9110

    0.0001    0.0084   -0.1862    0.1919    0.3781    1.6339

----------------------------1HH-------------------------------
   Mean       STD       Min       Max       Range     Entropy
    0.0000    0.0237   -0.4191    0.3932    0.8123    2.6274

    0.0000    0.0199   -0.3189    0.2960    0.6149    2.4848

----------------------------2HH-------------------------------
   Mean       STD       Min       Max       Range     Entropy
    0.0001    0.0247   -0.7312    0.8274    1.5586    2.3843

    0.0000    0.0163   -0.2890    0.3921    0.6811    2.2152

--------------------------AVERAGE-----------------------------
   RANGE I:  RANGE P:   RANGE I/P:
    1.0719    0.6577    0.6136

   ENT.  I:  ENT.  P:   ENROPY I/P:
    2.1605    1.9641    0.9091

   STD %:
    0.6913
```

C.2 Wavelet decomposition of Horse frame 1

wavelet_decomp.m script applied to frame 1 of Horse sequence, and the residue image after motion compensation by a 240x135 resolution vector field with half pixel accuracy.

```
>> wavelet_decomp('212','p212')
Original image:212.pgm
Residue image:p212.pgm
----------------------------1LL-------------------------------
   Mean       STD       Min       Max       Range     Entropy
   -0.1382    0.2090   -0.5848    0.6354    1.2202    2.6107

   -0.0021    0.0143   -0.4686    0.4581    0.9267    1.0896

----------------------------2LL-------------------------------
   Mean       STD       Min       Max       Range     Entropy
   -0.1360    0.2026   -0.5511    0.5365    1.0876    2.5919

   -0.0021    0.0089   -0.3096    0.3486    0.6583    1.0896

----------------------------1HL-------------------------------
   Mean       STD       Min       Max       Range     Entropy
   -0.0000    0.0506   -0.8163    0.7835    1.5997    2.8897
   -0.0000    0.0166   -0.2928    0.4092    0.7020    2.3039

----------------------------2HL-------------------------------
   Mean       STD       Min       Max       Range     Entropy
   -0.0001    0.0559   -0.7284    0.9078    1.6362    2.8821

   -0.0001    0.0119   -0.2717    0.2381    0.5098    1.9547

----------------------------1LH-------------------------------
   Mean       STD       Min       Max       Range     Entropy
   -0.0018    0.0468   -0.6185    0.6238    1.2424    2.7151

   -0.0010    0.0162   -0.3215    0.3149    0.6363    2.1620

----------------------------2LH-------------------------------
   Mean       STD       Min       Max       Range     Entropy
    0.0002    0.0649   -0.6966    0.8297    1.5263    3.0922

    0.0001    0.0146   -0.4698    0.6250    1.0948    2.0825

----------------------------1HH-------------------------------
   Mean       STD       Min       Max       Range     Entropy
    0.0001    0.0501   -0.6682    1.0022    1.6704    3.0815

    0.0000    0.0250   -0.3549    0.2945    0.6494    2.6680

----------------------------2HH-------------------------------
   Mean       STD       Min       Max       Range     Entropy
   -0.0002    0.0838   -1.4203    0.9925    2.4127    3.2771

   -0.0001    0.0239   -0.3749    0.3096    0.6845    2.5769

--------------------------AVERAGE-----------------------------
   RANGE I:  RANGE P:   RANGE I/P:
    1.5445    0.6811    0.4410

   ENT.  I:  ENT.  P:   ENROPY I/P:
    2.9121    2.2650    0.7778

   STD %:
    0.3418
```

Appendix D

Images

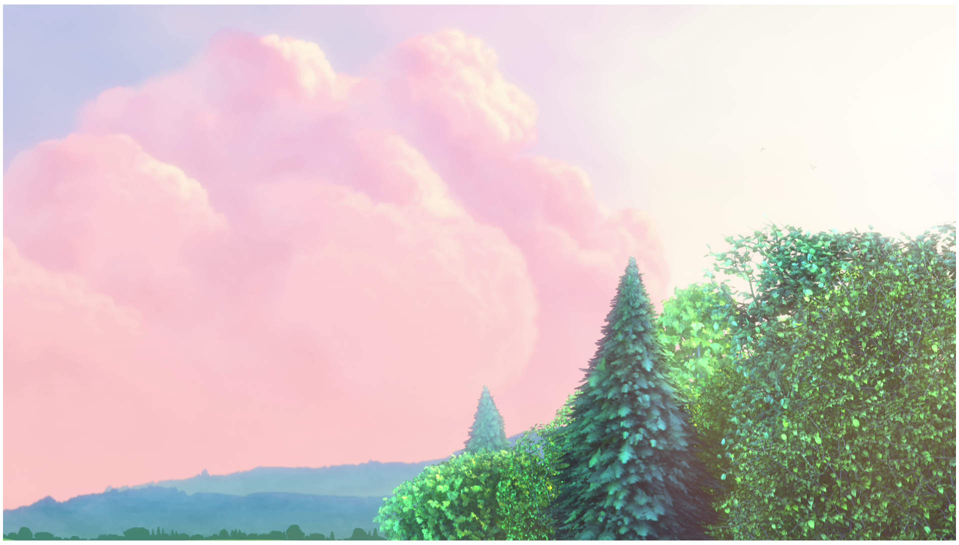


Figure D.1: Frame 115 of Big Buck Bunny (Bunny) sequence.

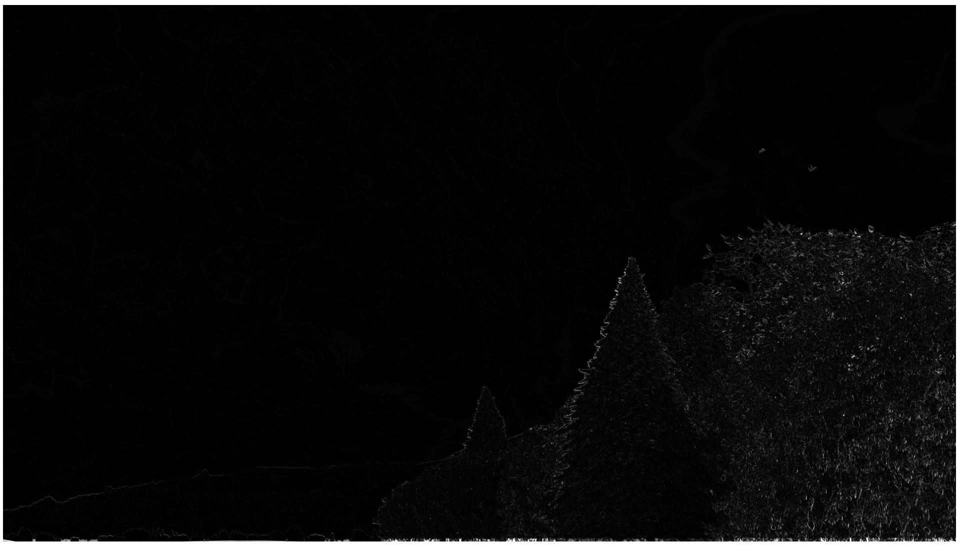


Figure D.2: Estimation error in frame 115 of Big Buck Bunny (Bunny) sequence. White indicates estimation error. Optical flow vector field with a resolution of 1920x1080 and quarter pixel accuracy.

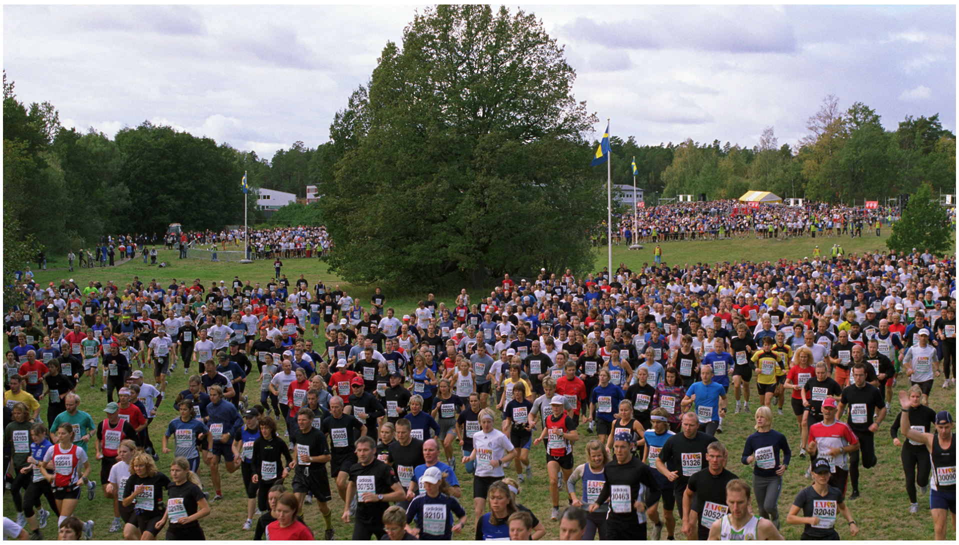


Figure D.3: Frame 7130 of Crowdrun sequence

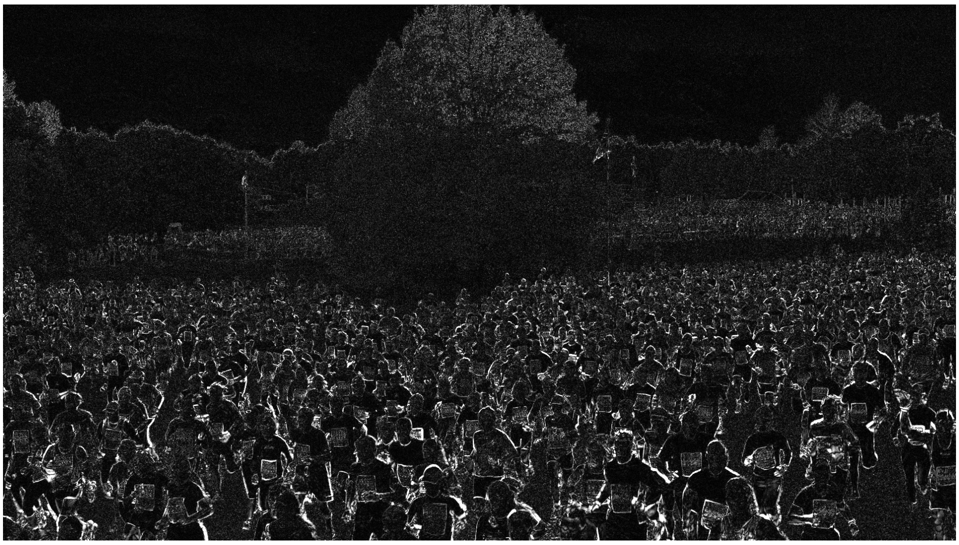


Figure D.4: Estimation error of frame 7130 of Crowdrun sequence. White indicates estimation error. Optical flow vector field with a resolution of 1920x1080 and quarter pixel accuracy.

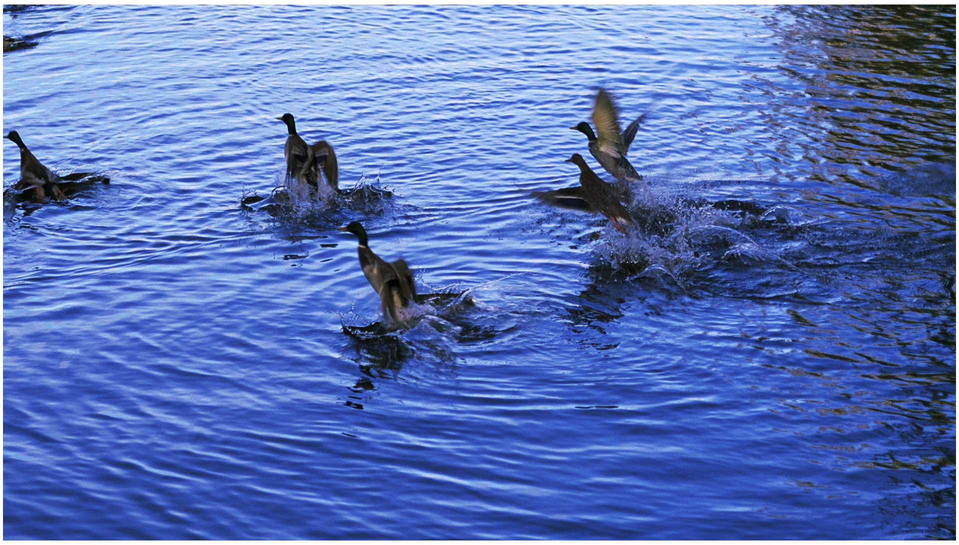


Figure D.5: Frame 13077 of DuckTakeOff (Ducks) sequence.

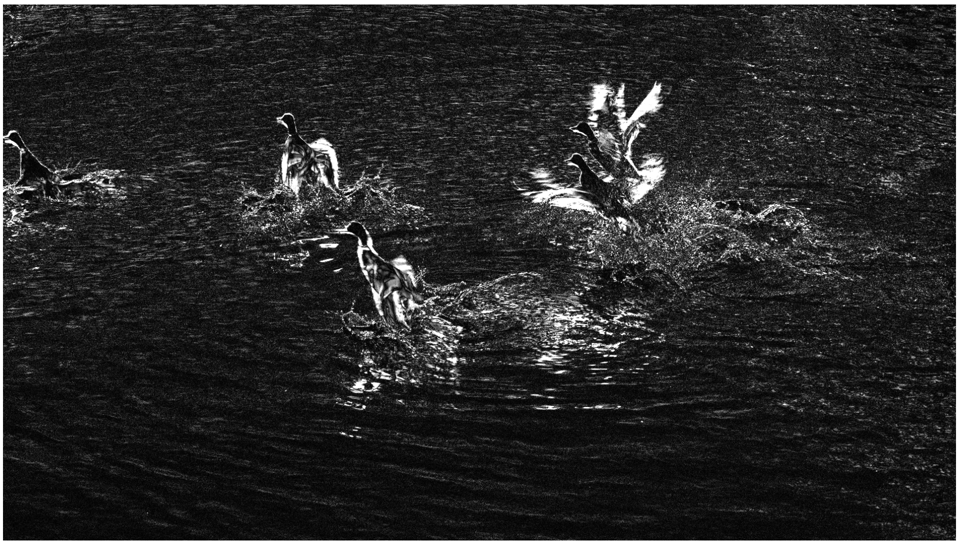


Figure D.6: Estimation error of frame 13077 of DuckTakeOff (Ducks) sequence. White indicates estimation error. Optical flow vector field with a resolution of 1920x1080 and quarter pixel accuracy.

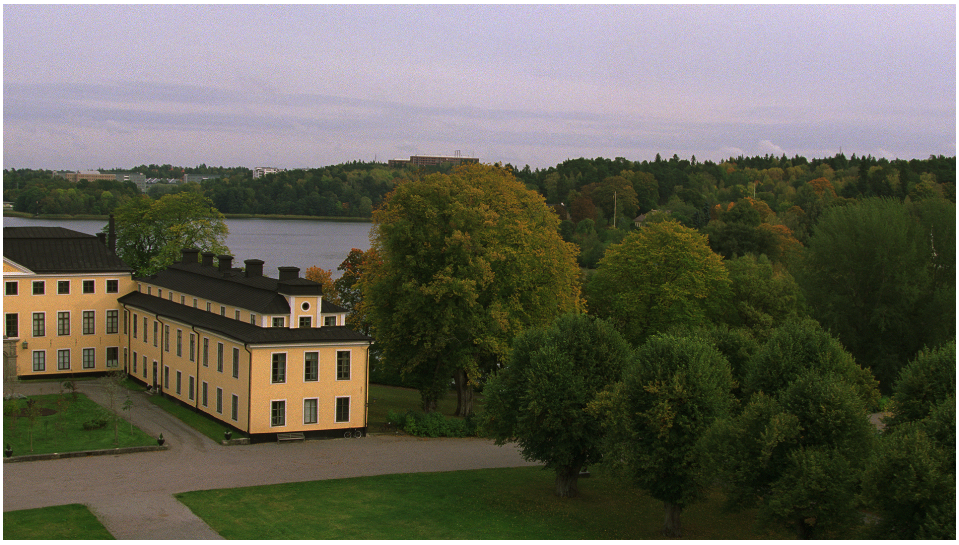


Figure D.7: Frame 5130 of IntoTree sequence

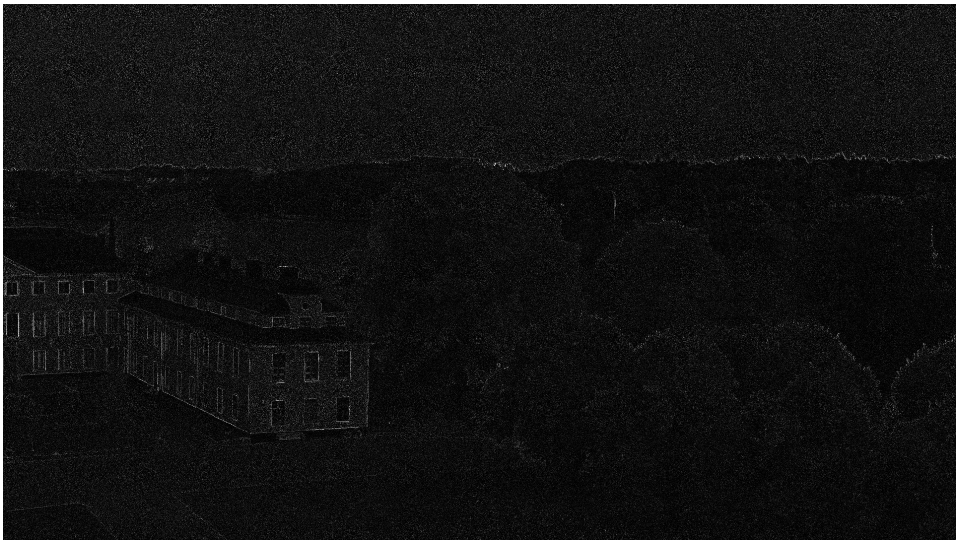


Figure D.8: Estimation error of frame 5130 of IntoTree sequence. White indicates estimation error. Optical flow vector field with a resolution of 1920x1080 and quarter pixel accuracy.

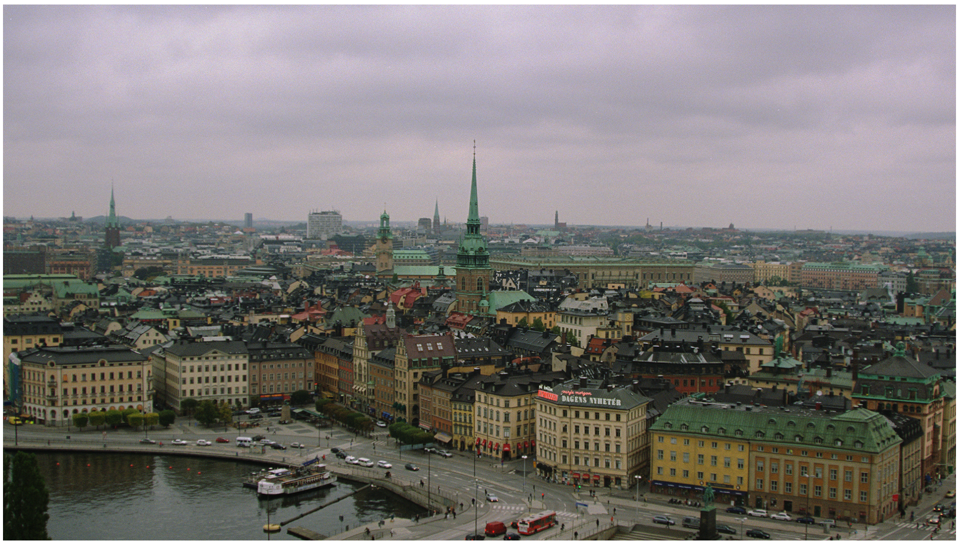


Figure D.9: Frame 1218 of OldTownCross (OldTown) sequence.

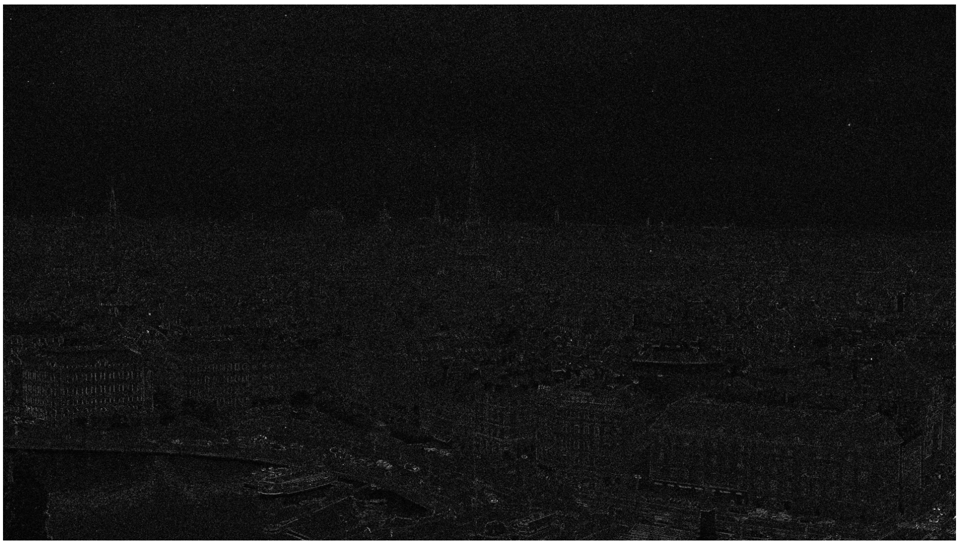


Figure D.10: Estimation error of frame 1218 of OldTownCross (OldTown) sequence. White indicates estimation error. Optical flow vector field with a resolution of 1920x1080 and quarter pixel accuracy.

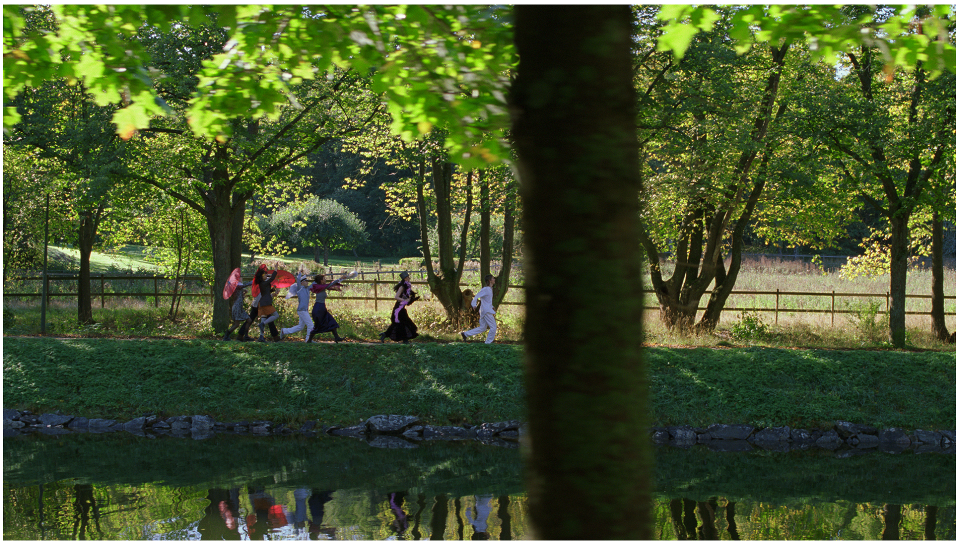


Figure D.11: Frame 15739 of Parkjoy sequence.

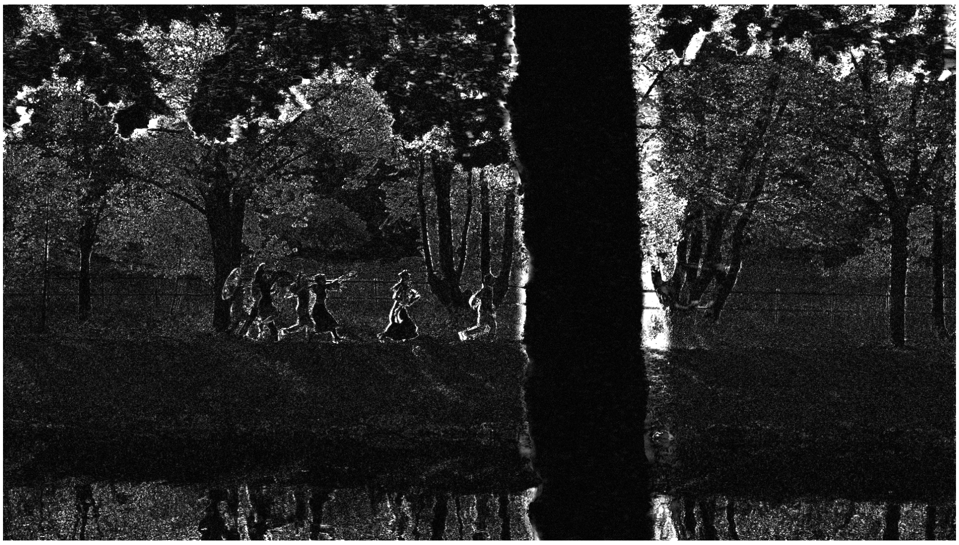


Figure D.12: Estimation error of frame 15739 of Parkjoy sequence. White indicates estimation error. Optical flow vector field with a resolution of 1920x1080 and quarter pixel accuracy.

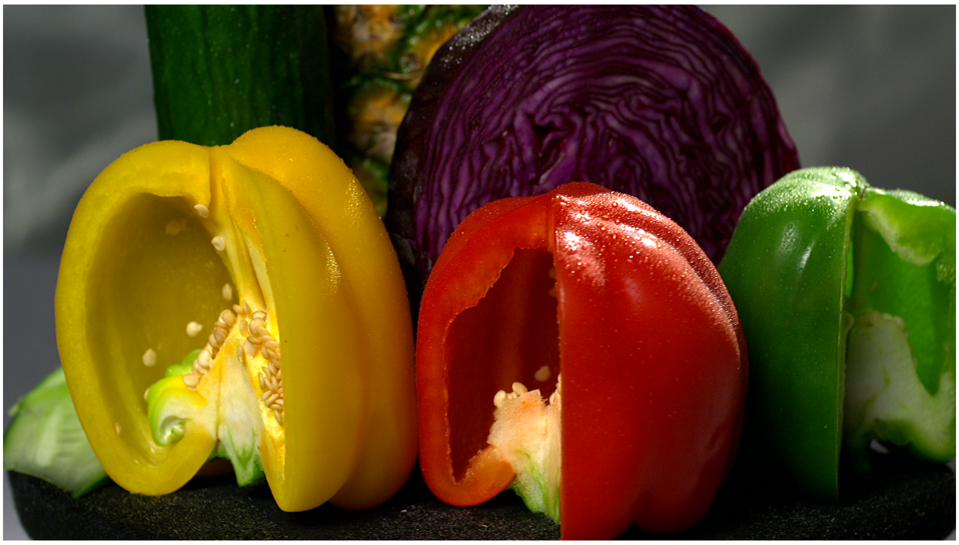


Figure D.13: Frame 13 of Vegies sequence.

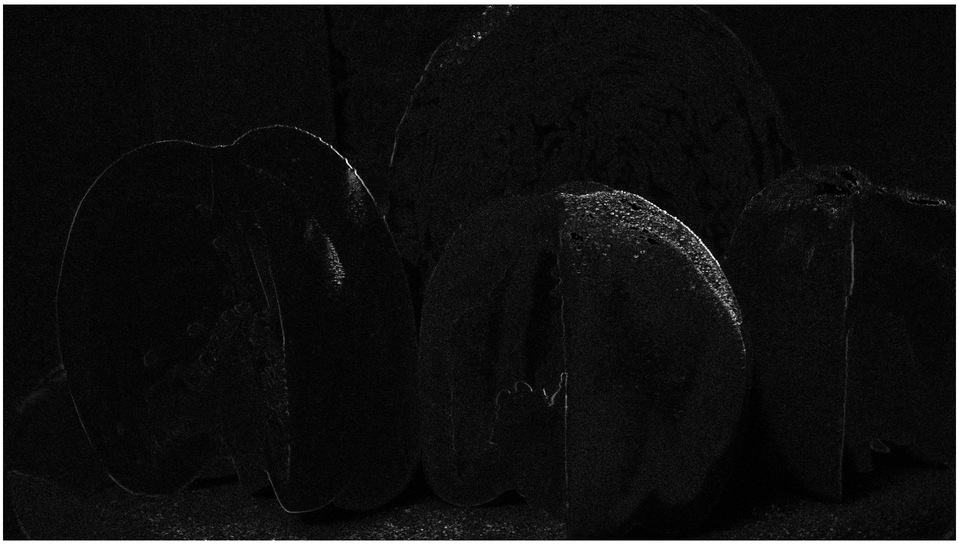


Figure D.14: Estimation error of frame 13 of Vegies sequence. White indicates estimation error. Optical flow vector field with a resolution of 1920x1080 and quarter pixel accuracy.

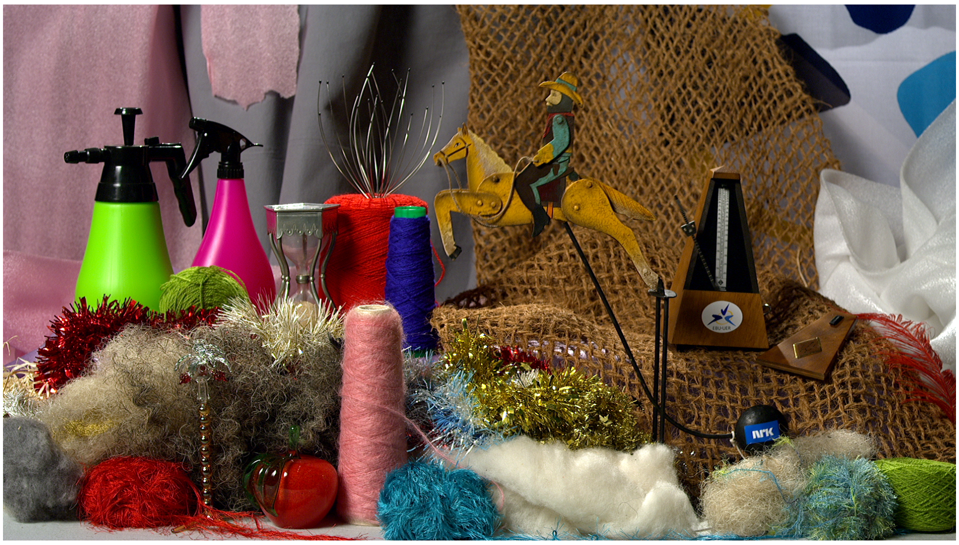


Figure D.15: Frame 1 of Horse sequence.

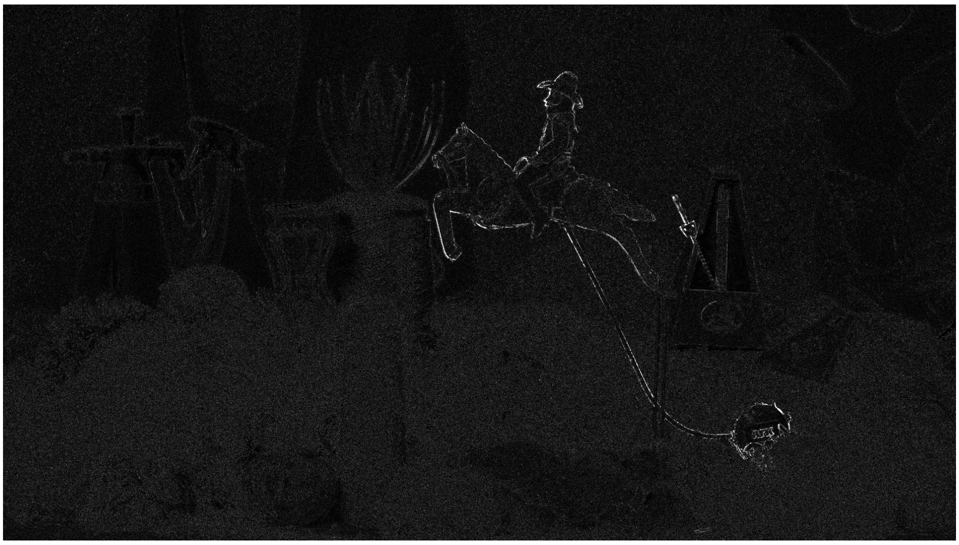


Figure D.16: Estimation error of 115 of frame 1 of Horse sequence. White indicates estimation error. Optical flow vector field with a resolution of 1920x1080 and quarter pixel accuracy.

Appendix E

Zip file

Content:

1. Matlab files
2. Results
3. Figures in higher resolution