Coding of Guiding Data for Video Transcoding

Christopher Hollmann
Abstract

Coding of Guiding Data for Video Transcoding

Christopher Hollmann

In today’s and tomorrow’s internet traffic videos play an increasingly larger role. This is true for both live transmissions like video conferences, and videos stored on internet platforms which the users can access at any time, the so-called Video on Demand (VoD). This thesis focuses on the latter case. Here the provider has several options how to make the content available. Possibilities are for example simulcast, transcoding, or guided transcoding. While each of them has advantages, there are also drawbacks. These can be storage requirements, computational complexity, or a loss of quality.

A different approach named deflation tries to minimize the drawbacks by using less storage than simulcast, being not as computationally expensive as transcoding, and providing a higher quality than guided transcoding. The first step is to estimate the transform coefficients based on the encoded version of the original video, which is decoded and downsized to the wanted resolution. Deflation then calculates the difference between the estimated and original transform coefficients, which are created by encoding the original video at the required resolution. These coefficients contain the actual picture data. They are then written into the bit stream, along with the control data containing information like prediction mode or how the picture is divided into Transform Blocks (TBs). In the inverse operation called inflation these details are parsed from the bit stream and the delta is added to the estimated coefficients, recreating the original values. However, since the deflation uses the identical encoding methods as the most recent video encoding standard HEVC, there is potential for improvement as the encoder can be optimized for deflation and its special cases and coefficient layout. After a thorough analysis to determine the layout of these delta-coefficients several new syntax elements were added and existing ones modified.

These changes were evaluated in various configurations. A key difference to the original scheme is the usage of knowledge gained from the estimated coefficients, which can be applied due to correlations between positions and magnitudes of both coefficient groups. The introduced changes reduce the storage requirements for deflated files by between 1.5 and 3 percentage points compared to the original deflation scheme, with the exact values depending on the configuration and settings of the representation. While there are still many options to improve the deflation scheme, the additions made and described in this thesis proved to be quite successful.
Acknowledgements

I would like to thank my supervisor at Ericsson Research, Kenneth Andersson, for the constant support during the entire process of writing this thesis.

Furthermore, I extend a warm-hearted thanks to Kai Lampka and Philipp Rümmer, my reviewers at the Department of Information Technology at Uppsala University.

In addition, my thanks also go to the team at Ericsson Research, which I was allowed to be a part of, starting at the head of the Visual Technology department, Thomas Rusert, and going all the way to Joakim Borgh, who I had the honour of sharing the office with.

A very special thanks goes to Isabella Svensk, without her understanding, encouragement and lots of tea I could not have completed this thesis.

Tack så mycket!

Christopher Hollmann
Uppsala, October 2016
# Contents

## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
</tr>
</thead>
<tbody>
<tr>
<td>v</td>
</tr>
</tbody>
</table>

## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
</tr>
</tbody>
</table>

## 1. Introduction

1.1. Motivation ................................................................. 2
1.2. Ericsson Research ....................................................... 3
1.3. Outline ................................................................. 3
1.4. Remarks Regarding Notation ...................................... 4

## 2. Background

2.1. Video on Demand ......................................................... 5
  2.1.1. Description of the Use-Case ..................................... 5
  2.1.2. Simulcast .......................................................... 6
  2.1.3. Transcoding ...................................................... 6
  2.1.4. Guided Transcoding ............................................. 7
2.2. High Efficiency Video Coding ....................................... 8
  2.2.1. History .......................................................... 8
  2.2.2. Structure of the HEVC-Encoder-/Decoder ..................... 9
  2.2.3. Inter- and Intra-Picture Prediction ......................... 10
  2.2.4. Transformation and Quantisation ............................. 10
  2.2.5. Context-based Adaptive Binary Arithmetic Coding ........ 11
  2.2.6. Encoding of Transformation Coefficients ................... 11
  2.2.7. Test Sequences ................................................ 15
2.3. Software Tools .......................................................... 15
  2.3.1. HM 16 ............................................................ 15
  2.3.2. D65 .............................................................. 17
  2.3.3. Downscaler ....................................................... 18
  2.3.4. PSNR Static .................................................... 18
  2.3.5. Python Scripts ................................................ 18
  2.3.6. Linux Cluster .................................................. 19
2.4. Related Work ............................................................ 19

## 3. Video on Demand - Deflation and Inflation

3.1. Concept ................................................................. 22
3.2. Differences to Guided Transcoding ................................ 24
3.3. Bottom-Up Approach .................................................. 26
4. Analysis of $\Delta$-Coefficients 27
   4.1. Magnitude of Coefficients ........................................... 27
   4.2. Distribution of the Coefficients .................................. 29
   4.3. Distribution of Bytes Spent on $\Delta$-Coefficients ............ 30
   4.4. Other Evaluated Metrics .............................................. 32
   4.5. Effects of Different Factors on $\Delta$-Coefficients .......... 33
       4.5.1. Top-Down Approach ............................................ 33
       4.5.2. Bottom-Up Approach ........................................... 35
5. Changes Made to Encoder and Decoder 36
   5.1. Coded Delta Flag (CDF) ............................................... 36
   5.2. Maximum Magnitude Is 1 Flag ....................................... 38
   5.3. Significant Coefficient Map ....................................... 40
       5.3.1. Changing Contexts ............................................... 40
       5.3.2. Significance Map Identical Flag ............................... 41
   5.4. Last Significant Coefficient ....................................... 43
6. Evaluation 46
   6.1. Peak Signal-to-Noise Ratio ......................................... 46
   6.2. Encoding Efficiency .................................................. 46
       6.2.1. Coded Delta Flag (CDF) ........................................ 47
       6.2.2. Maximum Magnitude Is 1 Flag ................................ 48
       6.2.3. Significance Map ................................................. 48
       6.2.4. Last Significant Coefficient ................................... 51
   6.3. Discussion of Test Results ......................................... 51
       6.3.1. Principle of Independent Parsing .............................. 53
7. Conclusion 55
   7.1. General Results ...................................................... 55
   7.2. Assessment of Tasks .................................................. 55
   7.3. Future Work ............................................................. 56
A. Acronyms 58
Bibliography 59
List of Figures

2.1. Schematic representation of simulcast ........................................ 6
2.2. Schematic representation of transcoding ................................. 7
2.3. Schematic representation of guided transcoding ......................... 8
2.4. Different scanning orders used in HEVC ................................. 12
2.5. Flow chart describing the encoding process ............................ 14
2.6. Example of the encoding of a 4x4 transformation ....................... 16
   (a). Transformation ..................................................................... 16
   (b). Scan order .......................................................................... 16
   (c). Encoded values ..................................................................... 16
3.1. Schematic representation of deflation ........................................ 23
3.2. Process of generating and storing $\Delta$-coefficients (deflation) .... 25
   (a). Original process ................................................................... 25
   (b). Modified process .................................................................. 25
3.3. Process of parsing $\Delta$-coefficients and storing original coefficients (inflation) 25
   (a). Original process ................................................................... 25
   (b). Modified process .................................................................. 25
5.1. Process of encoding and parsing the CDF ................................. 37
   (a). Deflation ............................................................................ 37
   (b). Inflation .............................................................................. 37
5.2. Process of encoding and parsing the maximum magnitude = 1 flag ... 39
   (a). Deflation ............................................................................ 39
   (b). Inflation .............................................................................. 39
5.3. Context map used for encoding the significant coefficient map ....... 41
   (a). Map for luma ...................................................................... 41
   (b). Map for chroma ................................................................... 41
5.4. Process of encoding and parsing the SigMapIdentical flag .......... 42
   (a). Deflation ............................................................................ 42
   (b). Inflation .............................................................................. 42
5.5. Process of encoding and parsing the LscIdentical flag ............... 45
   (a). Deflation ............................................................................ 45
   (b). Inflation .............................................................................. 45
List of Tables

2.1. Distribution of bytes spent in HEVC ........................................ 12

3.1. Share of $\Delta$-coefficients in deflation .................................. 26

4.1. Magnitude of $\Delta$-coefficients based on total number of coefficients .... 27
4.2. Magnitude of $\Delta$-coefficients based on number of significant coefficients 28
4.3. Probability that all $\Delta$-coefficients are zero for different transformations 28
4.4. Relation between magnitudes of estimated and $\Delta$-coefficients .......... 29
4.5. Magnitude of $\Delta$-coefficients for the bottom-up approach ................. 29
4.6. Normalised maps for 4x4 transformations .................................. 30
   (a). $\Delta$-coefficients ....................................................... 30
   (b). Original coefficients .................................................... 30
4.7. Normalised maps for 8x8 transformations .................................. 31
   (a). $\Delta$-coefficients ....................................................... 31
   (b). Original coefficients .................................................... 31
4.8. Probability that the last significant position is identical for estimated and $\Delta$-coefficients .................................................. 31
4.9. Probability that the significance maps are identical for estimated and $\Delta$-coefficients ................................................. 32
4.10. Average byte distribution for $\Delta$-coefficients ............................ 33
4.11. Probability that all estimated coefficients are zero ......................... 34

5.1. Overview over different context configurations used for encoding the significance map ................................................... 40

6.1. Comparison of PSNR values between different settings ..................... 46
6.2. Effects of the CDF on the deflation ....................................... 47
6.3. Results using the Maximum Magnitude = 1 flag ............................. 48
6.4. Test results for changing the significance map contexts .................... 49
   (a). Test results ............................................................ 49
   (b). Comparison ............................................................ 49
6.5. Results using the SigMapIdentical flag ..................................... 50
6.6. Results using the LscIdentical flag ........................................ 51
6.7. Overall improvement for deflation ........................................... 52
1. Introduction

1.1. Motivation

Videos in the internet are a very big part of people’s lives in today’s world. Starting from news pages and social media portals to platforms specifically made for video sharing, most web sides feature videos in some fashion. The biggest platform of those, the American company YouTube, has, according to their own statistics, over hundreds of millions of hours of videos watched every day generating billions of views [You16]. Storing and displaying these humongous amounts of videos takes a vast amount of hard drives and computational power. Both of these resources are quite expensive, and therefore every provider has to do a trade-off between them.

Using the latest standard High Efficiency Video Coding (HEVC) to encode videos already reduces the required bit rate compared to older standards like H.264/AVC or H.263. However, since the amount of data is so large, every reduction of bit rate or storage requirement is very welcome.

There are different strategies to deal with this trade-off. However, finding the right way is challenging and takes a thorough investigation of the available resources to determine the best decision. This thesis analyses the so called deflation scheme, which can be seen as a special case of guided transcoding. This version has its foundation in the two options simulcast and transcoding. Guided transcoding tries to take the best parts of both simulcast (low coding complexity) and transcoding (low storage requirement) to reduce the required storage space by encoding all lower qualities before being requested by the user. However, only the guiding data is kept and used to increase the speed of the encoding process once the video is requested.

While guided transcoding provides a very solid middle way, it is unfortunate that the coefficients containing the picture data are calculated just to be removed, and, after a user request, calculated again. Deflation tries to avoid this by determining the delta between the coefficients from the original video stream and the estimated coefficients from the lower quality video stream. This difference is then stored along with the guiding data, taking up less space than simulcast while also allowing a very fast recreation of the video. As a side effect the loss of quality introduced by the transcoding scheme is reversed.

The question coming up during this process is how the distribution of these $\Delta$-coefficients looks like and if there is a way to encode them in a more efficient way. While the current encoding provides some improvement, it is designed and optimised to encode the original
1. Introduction

transform coefficients, and not the difference between original and estimated coefficients.

Finding an answer to this question is the main goal of this thesis. By looking at both the distribution and magnitudes of the \( \Delta \)-coefficients a modified scheme is developed to encode these coefficients as efficiently as possible. This new scheme is then compared to the current encoding schemes trying to see if there is an actual improvement.

Out of this motivation the following tasks can be deduced:

- The first step shall be to look into the current state-of-the-art video encoding standard HEVC and see how the transform coefficients are generated and stored.
- Afterwards the \( \Delta \)-coefficients created by the deflation scheme shall be analysed in regards to both magnitude and distribution to find patterns which can be used to make the encoding more efficient.
- The layout of the estimated coefficients shall be compared to the one of the \( \Delta \)-coefficients to find correlations which can be taken advantage of in order to increase the efficiency of the encoding of the \( \Delta \)-coefficients. One way is to take identical values from the estimated coefficients in order to omit the encoding of these values in the \( \Delta \)-coefficients.
- The current usage of entropy encoding contexts shall be verified. If possible, ways using an equal or lower number of contexts shall be found to reduce the complexity of the decoder and encoder.
- An implementation for the bottom-up approach shall be finished to test whether the use of deflation can bring an advantage for upscaling video sequences.

1.2. Ericsson Research

This thesis was carried out in cooperation with Ericsson Research, which is a part of the Ericsson group and centred in Kista, Stockholm. The Visual Technology unit, which provided the hard- and software required for this work as well as a vast amount of knowledge about video encoding, is a driving force in research, development and standardisation of video compression and media delivery technology.

1.3. Outline

This thesis details the evaluation and analysis of the \( \Delta \)-coefficients generated by using deflation on a video stream. New syntax elements were added to improve the efficiency of the encoding. Chapter 2 provides an introduction in the underlying and used technologies and the software basis. A description of existing Video on Demand (VoD) schemes can be found here as well. In chapter 3 the principles of an additional approach for VoD
1. Introduction

called deflation are introduced and described. This scheme is evaluated in this thesis. A thorough analysis of the ∆-coefficients generated by the deflation can be found in chapter 4. Concrete changes made to the encoder and decoder are detailed in chapter 5. This includes an overview about newly introduced and changed syntax elements to improve the encoding efficiency. The impact of the added or changed components are evaluated and discussed in chapter 6. Chapter 7 concludes this thesis and looks back at the task set while also suggesting the next steps in the development.

1.4. Remarks Regarding Notation

To keep the data in this thesis comparable, a default configuration was used. These settings and their notation are described here, along with other notes regarding software and notation.

- **22/22, 720p**: The first value shows the Quantisation Parameter (QP) used for the original (HQ) encoding, the second one is the QP of the LQ encoding. The third value is the resolution, meaning the height of each frame. As there is only a limited number of used resolutions, the options in this thesis are 1920x1080 (1080p), 1280x720p (720p), 960x540 (540p) and 640x360 (360p).

- Unless stated otherwise, statistics are taken from the sequences BASKETBALLDRIVE, BQTERRACE, CACTUS, KIMONO and PARKSCENE with the configuration 22/22, 720p.

- Nearly all the tests and results were created using HM-16.7 as en- and decoder. If a different version was used, it is explicitly stated.

- An overview over used abbreviations can be found in appendix A.

- References to JCT-VC documents can be retrieved via http://phenix.int-evry.fr/jct. By selecting All meetings a list of JCT-VC meetings is shown, also listing the Unique Serial Letter which precedes the document number. For example, document G323 is file 323 from the meeting in Geneva which was set in November 2011.
2. Background

2.1. Video on Demand

2.1.1. Description of the Use-Case

Video transmissions in today’s world are more and more important and take up a bigger and bigger share of data sent through the internet [Cis16]. A very common use-case are videos on online platforms, stored so anybody can watch these at any time. Portals like YouTube\(^1\) provide millions of hours of video content for millions of users according to their own statistics [You16]. However, the providers need to make different levels of quality available. This can be due to

- a slow or metered internet connection between the user and the content provider,
- the users device, which can vary in screen size (TV screen or mobile phone), or
- the user wants to set the quality based on his own preferences.

For example, it does not make any sense for the provider to send a video in 1080p, to a user who uses a mobile phone with a screen resolution of 480p. On the other hand, a user might not want to have a video in 1080p, if he has only a limited amount of data traffic available, and might be satisfied with the video in 720p or an even lower resolution.

The most common video resolutions for on demand videos range between 1080p and 360p, at times other qualities like 240p, 1440p or 2160p are also available [Wik16]. To bring various quality levels to users the portal owner needs to consider very thoroughly how to design his own system.

- The first option is to store all formats and send the requested video in the wanted quality to the user. This method is called *simulcast* and is detailed in section 2.1.2.
- Another variant is to store only the highest available quality and generate lower qualities only when they are requested. This goes by the term *transcoding* and is described in section 2.1.3.
- As always, there is a compromise. *Guided transcoding* takes the best parts of both simulcast and transcoding, its specifics can be found in section 2.1.4.

---

\(^1\)https://youtube.com
2. Background

2.1.2. Simulcast

Simulcast is a way to provide different qualities of videos which focuses on reducing the coding complexity. Here all desired qualities are encoded and stored before the actual user request comes. Once the request arrives, the server can just select the demanded quality and transmit it without any further required computations. As can be imagined, the issue here is the large storage capacity required to save all demanded representations, which has to be provided to be able to save all different qualities. The coding complexity on the other hand is quite low, as there are no operations necessary once the request for a video comes besides sending it. A schematic representation of simulcast is shown in figure 2.1.

2.1.3. Transcoding

Transcoding on the other side tries to reduce the storage demands as much as possible. Therefore, when applying transcoding only the highest quality is encoded and saved. Once the user request is received, the servers actions are based on the request. If the user wants to retrieve the highest quality, it is already present and it will be transmitted. In a case where the user demands a lower resolution, the server has to decode the highest quality, downsize it to the wanted resolution, and encode the smaller version of the video.

This scheme allows the server to save much of the storage capacity simulcast requires at the expense of increasing the required computational complexity, if the user requests a lower quality video. A further downside is that the quality of the transmitted video is lower since not the original video is encoded, but a version of the source which has been encoded and then decoded.
This high computational demand is, however, considered a disadvantage. Especially with popular videos the server will be busy doing the same operations over and over again. Even using a cache for encoded videos in different qualities is not a great solution, assuming a worst case scenario where a video is only requested once it has been overwritten in the cache. An overview over transcoding can be found in figure 2.2.

### 2.1.4. Guided Transcoding

A third way trying to compromise and take only the best parts of both transcoding and simulcast is called guided transcoding and visualised in figure 2.3. This scheme tries to reduce both the computational complexity of encoding the lower resolution of videos on demand and the storage requirements of storing information about all desired qualities of videos.

Guided transcoding does the encoding of low quality videos both before and after the user request. When the video is provided, the encoding is done very similar to simulcast. However, not all data is stored. Only the Side Information (SI) is kept while the actual picture data is removed from the bit stream. The SI contain information like motion prediction, intra-picture prediction, and details about how the frame is divided into coding units, which is expensive and time consuming to calculate. However, as the data of the actual image is not retained, the required storage space is much less. By using these SI the actual encoding process after the user request is faster, as the data which is most expensive to generate is already present.

This coding scheme in the current or former variants has been evaluated in several publications, notably [VCS03] and [WCW12]. [RAYN16] also provides more details about...
2. Background

Figure 2.3.: Schematic representation of guided transcoding

guided transcoding.

2.2. High Efficiency Video Coding

HEVC is a well known topic which has been subject to many papers and publications. This section will provide a short introduction and present some more specific details about those features, which are most relevant to the topic of this work. More complete and elaborate descriptions can be found elsewhere [SOHW12, SBS14, Wie15a, ITU15].

2.2.1. History

HEVC is the current state-of-the-art video codec. It is the successor of the H.264/AVC standard, which was finalised in 2003 [WSBL03], and also known as H.265 [ITU15]. After researching options for improving the coding efficiency and performance for multiple years, the ISO/IEC MPEG issued a call for evidence in 2009 and founded the Joint Collaborative Team on Video Coding (JCT-VC) in cooperation with the ITU-T VCEG along with a call for proposals on video compression technology in 2010 [SOHW12]. During the first meeting of the joint team the name for the new codec was established and the testing phase of different suggested modules from various proposals started. Over
2. Background

the course of three years the standard was tested and completed, leading to its formal public release in 2013 [Sul14].

Comparing the performances of HEVC and its predecessor H.264/AVC, results show that HEVC uses roughly half the bit rate required by H.264/AVC\(^2\) while maintaining a nearly identical visual quality. This holds true for both subjective and objective tests [OSS+12]. HEVC reuses many of the encoding methods applied in prior standards. However, while some elements are quite similar, for example both HEVC and H.264/AVC use prediction from either the same or surrounding frames, others have been remodelled in their structure. A notable example is the transformation applied when calculating the transform coefficients. While H.264/AVC uses an integer transformation matrix, HEVC applies in most cases a Discrete Cosine Transformation (DCT), whereas the predefined integer matrix is an alternative for 4x4 transformations [SOHW12, WSBL03]. Another difference is the scan applied to the transform coefficients. In H.264/AVC a zig-zag scan is used, starting at the top left corner of the transformation block. HEVC on the other hand scans the coefficients from the bottom right corner with either a horizontal, vertical or diagonal scanning order [SM14]. To summarise, most of the differences between the two encoding standards are quite low-level and of a very technical nature, but of only marginal significance to the topic of this thesis.

While the encoding complexity of the HEVC encoder is significantly higher than the complexity of its H.264/AVC counterpart, with a well-chosen configuration it can be optimised to have a reasonably increased complexity with a strongly reduced bit rate [CAAdSC12, BBSF12].

However, as research never stops it becomes clear that HEVC is not the ultimate solution to video encoding and there are different areas for further investigation and improvement. These areas include the range of supported video formats, for example higher bit depths or 3D video, as well as advances in coding efficiency and complexity [SOHW12].

2.2.2. Structure of the HEVC-Encoder/-Decoder

The HEVC encoder and decoder are very complex software applications, describing them in detail would go far beyond the scope of this thesis. Many details of the implementation are thoroughly explained in the specification [ITU15]. In this section a short summary about the general structure will be presented, for more informations the reader is referred to the sources mentioned at the beginning of this chapter.

Frames are divided into square regions, the exact positioning of which is stored in the bit stream for the decoder. These regions are then encoded applying a prediction model using data from either the same frame or surrounding ones. The details of the prediction are written into the bit stream as well to allow the decoder to recreate the frames precisely.

---

\(^2\)The exact values, however, depend on a number of factors, including amongst others the video itself and the encoding configuration.
2. Background

The residual signal, which is the difference between the prediction and the original picture data, is subject to a linear spatial transformation. The resulting transform coefficients are scaled, quantised and entropy-coded into the bit stream. The decoder works in reverse order, adding a loop filter to smooth out artefacts created by the block-wise processing [SOHW12].

2.2.3. Inter- and Intra-Picture Prediction

In HEVC frames can be encoded in two different prediction modes: Inter and Intra. Inter-picture prediction means that coefficients are calculated based on data from different frames. An example is that if in frame 0 a structure is in one corner and in frame 2 the same structure can be found in a different corner, the chance is relatively high that in frame 1, which is shown between 0 and 2, the structure can be found in the middle between the two corners. This can be used for sequences with more than one frame and implies that frames are not decoded in the order they are shown. More details about the precise algorithms can be found in [SOHW12].

Intra-coded frames on the other hand use the prediction coming from the same picture. HEVC supports a total of 35 prediction modes, with 33 being angular modes in addition to the planar and the DC mode. The angular modes are distributed over a half-circle (180°) and denser at horizontal and vertical directions. Planar prediction allows smoother sample surfaces. More specific information is available in [SOHW12, LBH+12].

A main difference is that intra-coded frames can be decoded without any knowledge from previous pictures. This is useful if transmitted data is lost and the decoder would have to wait for a retransmission otherwise. Also, if a user jumps ahead to a point in the video, the decoder can skip decoding all frames up to that point and can just start from the nearest intra-coded frame.

2.2.4. Transformation and Quantisation

HEVC applies, similar to H.264/AVC, transform coding of the residual signal. This residual signal is created based on the difference between the predicted picture data and the actual picture data. The signal is stored in form of a matrix with different allowed sizes ranging from 4x4 to 32x32.

The residual signal is then subject to a transformation. The codec uses an approximated DCT with key symmetric values to increase the calculation speed of the matrix multiplication. For 4x4 luma transformations an alternative transform matrix based on a Discrete Sine Transformation (DST) is available.

The calculated values are quantised afterwards. The used scheme is essentially the same as in H.264/AVC, it is controlled by a Quantisation Parameter (QP) between 0 and 51.
2. Background

Alternatively, also quantisation scaling matrices are supported. Details can be found in [SOHW12].

The resulting coefficients, also known as transform coefficients, are then encoded into the bit stream using entropy coding as detailed below.

In this thesis, the used QPs for encoding the HQ version of the sequence are 22, 26, 30 and 34. These are based on the common test conditions of the Joint Collaborative Team on Video Coding (JCT-VC) for HEVC Scalability Extension (SHVC) [SH16].

2.2.5. Context-based Adaptive Binary Arithmetic Coding

HEVC uses Context-based Adaptive Binary Arithmetic Coding (CABAC) as its coding engine. Context-based Adaptive Binary Arithmetic Coding (CABAC) is a lossless form of compression and uses the estimated probability of binary symbols to achieve compression levels close to the entropy of a sequence. It does so by mapping symbols to codewords with a non-integer length [SB12].

CABAC consists of three main functions: binarisation, context modelling, and arithmetic coding. In the binarisation the syntax elements are converted to binary symbols. To achieve this several different schemes are used in HEVC. Examples are unary coding, Exp-Golomb or fixed length. The scheme to use is dependent on the syntax element to be encoded [Wie15b].

The context model used provides HEVC with an accurate probability estimation for different binary symbols. The context can be switched based on factors like transformation size, luma or chroma, or positioning inside the syntax element. Applying a single context for bins with a similar distribution allows very accurate prediction and therefore efficient encoding.

The principle of arithmetic coding is a recursive interval division. Here an interval is divided based on the probability that the value to be encoded is either 1 or 0. For the next bit, the range is updated with the previous subinterval and the division process repeated. The probability can be based on the context or it can be assumed equal. In the latter case, the bypass coding, the division can be done with a shift, and is very cheap compared to the look up table required for context-coded bins [SM14].

2.2.6. Encoding of Transformation Coefficients

This section describes the encoding of the transformation data. The data comes in square matrices with sizes varying between 4x4, 8x8, 16x16 and 32x32. The values are integer numbers and represent the quantised and transformed difference between the residual signal and the prediction, as described in section 2.2.4. Much research effort has gone into encoding the coefficients in a most efficient way, meaning to use as few bits as possible, since the coefficients take up a large part of the bit stream. This is visualised in table 2.1.
2. Background

<table>
<thead>
<tr>
<th>Syntax element</th>
<th>Bytes spent</th>
<th>Share</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAL body</td>
<td>19 133 252</td>
<td>100.00 %</td>
</tr>
<tr>
<td>Total coefficients</td>
<td>15 086 191</td>
<td>78.85 %</td>
</tr>
<tr>
<td>last_sig_coeff_x_y</td>
<td>2 132 526</td>
<td>11.15 %</td>
</tr>
<tr>
<td>sig_coeff_group_flag</td>
<td>210 480</td>
<td>1.10 %</td>
</tr>
<tr>
<td>sig_coeff_map_flag</td>
<td>7 377 498</td>
<td>38.56 %</td>
</tr>
<tr>
<td>coeff_abs_level_greater1_flag</td>
<td>1 329 546</td>
<td>6.95 %</td>
</tr>
<tr>
<td>coeff_abs_level_greater2_flag</td>
<td>133 138</td>
<td>0.70 %</td>
</tr>
<tr>
<td>coeff_sign_flag</td>
<td>2 878 149</td>
<td>15.04 %</td>
</tr>
<tr>
<td>coeff_abs_level_remaining</td>
<td>1 024 854</td>
<td>5.36 %</td>
</tr>
</tbody>
</table>

Table 2.1.: Distribution of bytes spent in HEVC

The difference between the coefficients and the Network Abstraction Layer (NAL) body originates in the control data, which is used to guide the decoder.

To reduce the number of bits for the case that a transformation has no significant coefficients, meaning coefficients with a value different from 0, a special flag, the Coded Block Flag (CBF) is transmitted. If this flag is 0, the entire transformation is skipped in the encoding. This is one part of the aforementioned control data, which is used as the SI in the guided transcoding scheme as described in section 2.1.4.

The encoding of the coefficients always follows the same procedure. The only difference is the used scanning order, which is based on the prediction. There are three options: horizontal, vertical and diagonal, which are shown in figure 2.4. However, horizontal and vertical scans are only permitted if the TB is intra coded and the prediction is close to horizontal or vertical, respectively. The predecessor to HEVC, H.264/AVC, used a zigzag scan, which was replaced due to its poor interaction with the template-based context models used in HEVC [SJN+12, SJK11].

Larger transformations are divided into 4x4 sub-blocks, which are then evaluated according to the scanning order. The scan always starts at the bottom right corner of the matrix and ends at the top left corner. This is also a change compared to H.264/AVC, which used both forward and backward scans. The key difference here is that H.264/AVC
scanned the significance map starting in the top left corner and added a *last significant coeff* flag after every significant coefficient to signalise whether the end of the transformation is reached [WSBL03]. Afterwards, levels and signs were scanned starting at the last significant coefficient. To reduce the complexity, HEVC uses only one reverse scanning order for all syntax elements [SM14]. This scan can, however, vary as described above, but the same one is used for all parts of the encoding.

Each TB is divided into Coding Groups (CGs) consisting of sixteen coefficients and representing a 4x4 sub-block. Each CG is scanned up to five times, each time retrieving a syntax element required for encoding: *significant_coeff_flag*, *coeff_abs_level_greater1_flag*, *coeff_abs_level_greater2_flag*, *coeff_sign_flag* and *coeff_abs_level_remaining*. The syntax elements are described in more details below. A flow chart providing more information about the exact order in which the encoding happens can be found in figure 2.5. Please note that the block *Encode level and sign* summarises the four steps *encode GT1 flag*, *encode GT2 flag*, *encode sign*, and *encode absolute level remaining*.

- **Last Significant Coefficient**
  The first element to be encoded is the position of the last significant coefficient in the transformation. The coordinates are divided into prefix and suffix, which are signalled separately. As the encoding scheme the prefix uses a truncated unary representation. The suffix is encoded with a fixed length binary code [SJN+12]. The encoder first writes the prefixes for both x- and y-coordinate into bit stream, followed by the suffixes, if at least one of them is necessary. The suffixes are bypass coded, whereas the prefixes use different contexts for each bit. The prefix is used to represent an interval which is dependent on the transformation size while the suffix determines the offset between these intervals.

- **Significance Map**
  The significance map consists of two different parts, which contain all information about the positions of the significant coefficients. The first element is the *Coded Sub-Block Flag* (CSBF), which signals if a sub-block of 4x4 coefficients has any significant values. This flag is not encoded for the last sub-block with a significant coefficient, as it is already implicitly indicated that this block has at least one significant coefficient by the last significant position. Furthermore, it is not encoded for the first sub-block, which is in the top left position, as the chance that there are coefficients with values different than zero is nearly 100 %. For 4x4 transformations this flag is skipped, as there is only one sub-block which has definitely at least one significant coefficient, since otherwise the Coded Block Flag (CBF) would not be set.

After the coded sub-block flag is encoded the remaining positions in the CG are scanned and for each of them one bit is written into the stream. If the coefficient is significant, a 1 is encoded, otherwise a 0. This scan ignores the position of the last significant coefficient, as it is already signalled by the previous element that this coefficient is significant. For other sub-blocks with at least one significant coefficient the entire block is scanned and encoded.
Figure 2.5.: Flow chart describing the encoding process
2. Background

- **Coefficient Greater Than 1 Flag**
  For the first eight significant coefficients of a CG a *greater than 1* flag is encoded as well. These signal whether the absolute value of the respective coefficient is larger than 1. This flag is limited to eight occurrences per sub-block.

- **Coefficient Greater Than 2 Flag**
  The first coefficient which is larger than 1 is encoded with an additional flag, the *greater than 2* flag. Similar to the greater than 1 flag, it is encoded as 1 if the absolute value of the coefficient is larger than 2. If there is no coefficient with a higher absolute value than 1, this syntax element is skipped.

- **Sign Map**
  After the flags for the coefficient levels another map is encoded. This map shows whether coefficients have values below or above 0. For negative magnitudes a 1 and for positive coefficients 0 is encoded. As the signs of the coefficients have a close to random distribution, they are bypass coded.

- **Coefficient Absolute Level Remaining**
  The last syntax element to be encoded is the remaining value of the coefficient. This is done using Golomb-Rice binarisation. All bits in the absolute level are bypass coded [SM14].

An example of the encoding of a transformation matrix is shown in figure 2.6.

2.2.7. Test Sequences

There is a total of 24 test sequences set for testing and developing HEVC. These sequences are grouped into six different classes. An overview can be found in the common test conditions for SHVC [SS16].

The tests executed on the cluster whose results were used in this thesis were based on the five sequences of class B *BasketballDrive*, *BQTerrace*, *Cactus*, *Kimono* and *ParkScene*. Local tests in a reduced configuration were done with *RaceHorses* from class C, primarily to verify the functionality of the code.

2.3. Software Tools

2.3.1. HM 16

The HEVC Test Model (HM) is the tool used to test and develop new functionalities and features of HEVC. This tool, which is freely available at the internet presence of the Fraunhofer Heinrich Hertz Institute [Fra16b], was developed along with the standard and is updated regularly. The first release of the current version (16-0) was published in August 2014. Most of the test and experiments conducted in this thesis were done with
2. Background

(a) Transformation

(b) Scan order

(c) Encoded values

Figure 2.6.: Example of the encoding of a 4x4 transformation, based on [SM14]
2. Background

release 16-7. While new versions were published, it was decided to conduct all tests on the same HM engine to ensure comparability between the test results.

The code is, however, compatible to run on newer versions of the HM without changes, as the interface of encoder and decoder was not modified. The only necessary change is the path to the executables.

2.3.2. D65

The D65 is the main tool base used for this thesis. It covers five different functionalities of which three are required for the deflation scheme. The other two, Residual Pruning and Residual Reconstruct, are features used for the pruning scheme, which is another attempt to enhance the HEVC encoding. The pruning scheme is, however, not a topic covered in this thesis. The other three functionalities are detailed below. As all programs use the same software base, the build process toggles a set of flags in the main header of the library to decide which of the sub-programs to build. The D65 uses C as its programming language. This tool was also a key part of an earlier thesis at Ericsson Research by Harald Nordgren [Nor16].

- **Output in Decoding Order**
  This version of the decoder is used to decode the encoded version of the original video. Its functionality is basically identical compared to the normal HEVC decoder. The only difference is that pictures are not sorted, even though they are encoded out of order. This way the deflation build does not need to apply any changes to the order of input frames as it would otherwise not match the order of the frames in the bit stream from the encoder.

- **Deflation**
  The deflater runs before the video is stored. It takes the encoded downsized original file as an input in addition to the encoded, decoded and downsized video. It parses the original coefficients from the first encoded file. In parallel the decoded file is encoded to get the estimation of the coefficients. The deflater calculates the delta between the two groups of coefficients and stores these along with the SI from the original video file.

- **Inflation**
  The inflater is run by the server once a request for a low resolution video arrives. It parses the ∆-coefficients containing the SI. To regenerate the original video it uses the stored encoded HQ sequence, which is decoded and downsized. While encoding this version it adds the delta from the deflated file to accurately recreate the original video sequence.

More details about the functionality of the deflation and inflation builds can be found in chapter 3.
2. Background

2.3.3. Downscaler

The downscaler is a C++ programme used to downsize videos from a higher to a lower resolution. It uses either a 13 tap filter to interpolate between values for removed pixels. This programme is part of the SHVC package and is provided by the Fraunhofer Institute [Fra16a].

2.3.4. PSNR Static

Similar to the downscaler, the PSNR Static is a tool written in C++. Its purpose is to determine the difference between the signal and the noise appearing in a video frame. This value is averaged over the entire sequence. The Peak Signal-to-Noise Ratio (PSNR) is a common measurement to determine the clarity and quality of a video stream [Bjo01].

2.3.5. Python Scripts

The testing of the deflation scheme is operated by a PYTHON script. There are two main scripts, bsub_transcoding.py and guided_transcoding_modular.py. Their functionalities are described below. A number of other scripts contain data types, definitions and configurations, but are not detailed here as their features are mostly trivial or of only supportive nature. The scripts were originally developed by Nordgren [Nor16] and extended to support the additional demands in this thesis.

- **bsub_transcoding.py**
  This is the so-to-speak main script of the guided transcoding test process. It takes all the configurations and settings and turns them into jobs for the cluster or, if the test is executed locally, the processor. The preferences are then parsed into arguments which are used to call the script actually performing the test. Options include, among others, locations of files, QP, and the resolution. The total number of 41 arguments prevents listing all of them.

- **guided_transcoding_modular.py**
  This file executes a single test cases. After parsing the arguments, all necessary actions to get comparable test results are performed. This consists of several groups of steps. After each segment the PSNR and bit rate are calculated and stored, as these are the values which can be used for comparing different encoding variants.

  The first part is to encode and decode the HQ video to get reference values for the other segments. The next step is to downsize the HQ video and encode and decode this version. Afterwards the deflation is executed. Here no PSNR value can be determined as the file is in no valid format. However, the bit rate is one of the most important figures of the test. The last part is to inflate the generated file. While this does not provide new insights, the PSNR and bit rate are identical to their respective values before the deflation, it is an important sanity test, showing
2. Background

that the video generated from the deflated file is actually the same as it was before the deflation.

A third script, called `gt_bottomup.py`, was written for the bottom-up approach which was reviewed for possible savings and is detailed in section 3.3. It executes basically the same code as `guided_transcoding_modular.py` with a few modifications due to the differences in the process. The key variation is that there is no downsizing as the predictions for the LQ version\(^3\) are based on the same resolution and so several steps are omitted.

As some function calls are using features added in Python3, the scripts cannot run with an earlier Python version.

2.3.6. Linux Cluster

Most of the tests were performed on the Linux cluster. This had the advantage, that tests could be done in parallel without having to wait for one to finish before starting the next one. Especially considering the total number of tasks per test run it was not feasible to execute them on the local machine. Using both bottom-up and top-down approach the number of tests was increased from 120 to 140, which can be calculated from the following equation:

\[
\text{num\_tests} = \text{num\_sequences} \times \text{num\_hq\_qp} \times (\text{num\_resolutions} \times \text{num\_lq\_qp} + \text{num\_bu\_qp})
\]

The cluster consists of a network of machines which are also situated in Kista, Sweden. While the performance of the cluster is highly dependant on the number of active jobs, it was generally no problem of executing tests within a reasonable time frame.

2.4. Related Work

Rusert et al. provide details about an implementation of the guided transcoding approach [RAYN16]. The authors present a detailed overview of the different options to prepare videos for the same use-case as in this thesis. Furthermore, they introduce an additional variant called Pruning, where the transform coefficients are only removed from certain frames depending on their position amongst a group of pictures representing the order in which frames are encoded. Their conclusion is that using guided transcoding provides a significantly lower coding complexity at the cost of moderately higher storage requirements. In their results the authors describe the same gains and rate reductions as Nordgren in his M.Sc. thesis at Lund University.

\(^3\)Even though the LQ version has a lower QP and therefore a higher quality, it is still referred to as the LQ to simplify the terminology and keep it identical to the terminology of the top-down approach.
2. Background

In his thesis Nordgren laid the foundation for this deflation scheme [Nor16]. Furthermore, he provides many details about the performance of the basic deflation and pruning schemes. In his conclusion he writes that pruning has rate reductions of nearly 30 % and doing partial pruning with a level of 2 provides a good balance between bit rate gains and decreased computational complexity. Deflation brings gains of around 20 % and has the advantage of not losing any quality in the process. However, while he presents results regarding the complexity of pruning, the author only provides assumptions about the performance of the deflater, saying it will be slower than pruning since more steps are involved.

The idea of improving the encoder performance by using a control stream was introduced by Van Wallendael, De Cock, and Van de Walle [WCW12]. Here the guided transcoding scheme, referenced to as fast transcoding in the paper, is compared to simulcast, scalable coding, and transcoding. The authors present several conclusions in the paper. Compared to simulcast the bit rate is over 60 % lower when storing a control stream instead of a complete video sequence. When taking storing the high quality reference stream into account, the savings are lowered to around 18 %. While it brings an even stronger positive effect to store the sequence via scalable coding, using a control stream has a lower bit rate when sending the video through the network to the user. Comparing their fast transcoding scheme to the original transcoding approach, the authors describe a much lower computational complexity, as the decoder-encoder cascade has approximately the complexity of two decoders. The deflation scheme evaluated in this thesis is based on the suggested solution, applying the same idea that all intensive processing is done only once for each representation.

Already in 2003 Vetro, Christopoulos and Sun evaluated a number of different video transcoding schemes [VCS03]. The authors looked at various techniques for bit rate reduction, and described the then-new advances, taking a closer look at spatial and temporal resolution reduction architectures. Furthermore, they investigated scalable coding techniques and ways to secure the bit stream against transmission errors by making it resilient. One described option is for example to increase the number of intra-coded blocks within the frame while also reducing the number of blocks for each slice. However, adding features to induce a certain level of resilience always increases the bit rate of the video. The authors concluded that there had been a lot of research on the topic of video transcoding, but no optimal strategy had been found. This is still the case today, as otherwise there would have been no reason for doing the research covered in this thesis.

The HEVC standard which serves as a foundation for the deflation scheme has been the topic of many publications, notably Wien [Wie15a], Sze et al. [SBS14], and Sullivan et al. [SOHW12]. As these publications are very thorough in their descriptions many details which have been omitted in this thesis can be found there. These sources also show why decisions in the design of the HEVC standard were made. While Sullivan, Ohm, Han and Wiegand focus on presenting a summary with only short presentations of the main features, Wien and Sze et al. present in-depth descriptions of all parts including comparisons to H.264/AVC. The very detailed chapters provide insights about various
2. Background

elements like entropy coding or motion prediction. In their entirety both books can be seen as two of the main background knowledge sources for the HEVC part of this thesis.
3. Video on Demand - Deflation and Inflation

3.1. Concept

The deflation scheme is based very closely on the guided transcoding scheme as described in section 2.1.4. However, there are a few substantial differences which are described in this section. A schematic description is shown in figure 3.1.

The main thought behind applying deflation is to regain the loss in quality introduced by the encoder-decoder cascade in both the transcoding and guided transcoding schemes. As a side effect, the transformation coefficients, which are calculated but not needed when using guided transcoding, are put to use to store the difference between original and estimated coefficients.

The procedure is basically a combination of the standard HEVC encoder and decoder. The deflater parses the downsized and encoded original video to get the original coefficients as well as the Side Information (SI). These are then applied to the input video stream. This stream is generated by encoding the original video, decoding it and finally downsizing it to the same resolution. As the SI are already present, this is a very fast process. The coefficients produced in this step are referred to as the estimated coefficients. Both sets of coefficients are compared and the difference, the so-called Δ-coefficients, is stored along with the SI for the inflation process. The encoding, decoding and downsizing are handled by the server using the different utilities described in section 2.3.

Once the user request arrives, the inflation process starts. The inflater parses the deflated file, retrieving the SI. Furthermore, the Δ-coefficients are extracted from the deflated file as well. The SI are then applied to the stored encoded version of the original video, which has been decoded and downsized to the requested resolution. This way the estimated coefficients are created, exactly like in the deflation process. By adding the Δ-coefficients from the deflated file to the estimated coefficients, the original values of the coefficients can be accurately recreated. Using this scheme regains the loss inflicted by basing the estimation of the sequence on the output of an encoder-decoder cascade.
Figure 3.1.: Schematic representation of deflation
3. Video on Demand - Deflation and Inflation

3.2. Differences to Guided Transcoding

While guided transcoding and deflation are based on the same basic idea, there are several differences with effects on file size, bit rate, decoder performance, etc. The main difference is that during the encoding of the LQ video, the coefficients containing the image data are not removed, but compared with the coefficients from the original HQ video and their delta is stored.

There are several differences to guided transcoding in the layout of the coefficients, some due to new cases which have to be handled, some caused by the use of features not available in the normal encoder.

One of the biggest differences in the encoding is that storing the delta between original and estimated coefficients allows that all coefficients in the transformation are insignificant. This means on one hand that the estimation is completely accurate, but on the other hand is this a case which does not occur and thus is not handled in HEVC. This happens, depending on the transformation-specific parameters, with a probability reaching up to around 50%. Statistics providing more details on this can be found in the analysis (section 4.2). HEVC employs a flag to signal this special case, the CBF. However, as the deflater writes this flag into the bit stream before the step of the deflation is reached, it would take quite a lot of changes to the syntax to reuse the flag to signal that there are no significant Δ-coefficients.

Therefore the original deflation scheme handles this case in a different way. If the deflater detects that all coefficients besides the one in the top left corner of the transformation map are insignificant and the coefficient in the top left corner is zero or larger, it is increased by one. This prevents the encoder from trying to encode an empty coefficient map. During the inflation this step is reversed to recreate the Δ-coefficients. This can create an inaccuracy when the coefficient value is equal to the maximal possible value. While the deflater does not increase the magnitude in this case, the inflater will reduce the value by one, creating a difference to the original value. However, as Nordgren wrote, this is a purely theoretical case that did not occur in his tests [Nor16].

Another change to the syntax became necessary as the decision was made to base the contexts used for the encoding of the significant coefficient flags on the estimated coefficients. The process to generate and store the Δ-coefficients in the original deflation scheme is shown along with the changes made in figure 3.2. This adjustment also had to be made in the inflater, which is visualised in figure 3.3. As these figures show, the parsing of the TransformationSkipFlag, which was formerly a part of parsing the coefficients, had to be separated from the original process. This had to be done as this flag is required for the calculation of the estimated coefficients. However, since the coding of this flag is not depending on the estimated coefficients, this was not a problem.

The Δ-coefficients take up quite a bit of the storage required for the deflated file. Table 3.1 shows the details of the required storage taken up by the coefficients. No Deflation describes the original file size, which is restored after the inflation. The Deflation column...
Figure 3.2.: Process of generating and storing $\Delta$-coefficients (deflation)

Figure 3.3.: Process of parsing $\Delta$-coefficients and storing original coefficients (inflation)
### 3. Video on Demand - Deflation and Inflation

<table>
<thead>
<tr>
<th>Settings</th>
<th>No Deflation</th>
<th>Deflation</th>
<th>Side Information</th>
<th>Coefficients</th>
<th>Share</th>
</tr>
</thead>
<tbody>
<tr>
<td>22/22, 720p</td>
<td>6 619 339</td>
<td>4 503 718</td>
<td>1 777 595</td>
<td>2 726 123</td>
<td>60.53 %</td>
</tr>
<tr>
<td>22/24, 720p</td>
<td>4 685 423</td>
<td>3 162 184</td>
<td>1 419 800</td>
<td>1 742 384</td>
<td>55.10 %</td>
</tr>
<tr>
<td>22/22, 360p</td>
<td>2 102 338</td>
<td>1 293 671</td>
<td>585 416</td>
<td>708 255</td>
<td>54.75 %</td>
</tr>
<tr>
<td>22/24, 360p</td>
<td>1 573 319</td>
<td>994 348</td>
<td>492 571</td>
<td>501 777</td>
<td>50.46 %</td>
</tr>
<tr>
<td>34/34, 720p</td>
<td>1 063 152</td>
<td>851 944</td>
<td>506 201</td>
<td>345 743</td>
<td>40.58 %</td>
</tr>
<tr>
<td>34/36, 720p</td>
<td>810 307</td>
<td>648 315</td>
<td>410 415</td>
<td>237 900</td>
<td>36.70 %</td>
</tr>
<tr>
<td>34/34, 360p</td>
<td>379 783</td>
<td>291 548</td>
<td>190 515</td>
<td>101 033</td>
<td>34.65 %</td>
</tr>
<tr>
<td>34/36, 360p</td>
<td>289 084</td>
<td>226 345</td>
<td>154 905</td>
<td>71 440</td>
<td>31.56 %</td>
</tr>
<tr>
<td>22/18, 1080p</td>
<td>62 965 712</td>
<td>55 066 333</td>
<td>8 612 497</td>
<td>46 453 836</td>
<td>84.36 %</td>
</tr>
<tr>
<td>34/30, 1080p</td>
<td>3 593 300</td>
<td>3 283 040</td>
<td>1 414 024</td>
<td>1 869 016</td>
<td>56.93 %</td>
</tr>
<tr>
<td>Average Top-Down</td>
<td>2 190 343</td>
<td>1 496 509</td>
<td>692 177</td>
<td>804 332</td>
<td>53.75 %</td>
</tr>
<tr>
<td>Average Bottom-Up</td>
<td>33 279 506</td>
<td>29 174 687</td>
<td>5 013 261</td>
<td>24 161 426</td>
<td>82.82 %</td>
</tr>
</tbody>
</table>

Table 3.1.: Share of $\Delta$-coefficients in deflation, values in bytes

contains the data on files deflated with the original deflation scheme. The column named Side Information describes the number of bytes spent on the SI. The second to last column, Coefficients, contains the amount of bytes used to encode the $\Delta$-coefficients, as the name already implies. The Share shows the part of the encoded bytes for deflation used to encode the coefficients. Results regarding file sizes for newly introduced and evaluated syntax elements can be found in section 6.2.

### 3.3. Bottom-Up Approach

The bottom-up approach covers the idea to use a lower quality as the estimation of the coefficients, instead of a higher one as applied in the “normal” deflation scheme described above.

In the greater context this idea could be realised by storing only the lowest resolution and then using an upscaler to calculate higher resolutions. However, as there was no upscaler available, this approach was limited to predicting a higher quality, but using the same resolution.

The necessary changes to include this approach in the deflation tests were quite small. While a new Python script had to be written, there were no changes required for the encoder, decoder or deflation software.
4. Analysis of $\Delta$-Coefficients

About this chapter: Some metrics are only described for the 4x4 and 8x8 transformations. These conclusions are valid for larger used transformations like 16x16 and 32x32, as well as when taking the effects mentioned in section 4.5 into account.

Tables regarding the magnitude of the $\Delta$-coefficients show only absolute values. An evaluation showed that there the number of coefficients larger than zero roughly equals the number of coefficients with values lower than zero. Therefore these coefficients are summarised and only considered in their absolute values.

4.1. Magnitude of Coefficients

The very first step of the analysis was to determine the magnitudes of the $\Delta$-coefficients. This can be seen as a metric for the quality of the estimation, as described in section 3.1, since low delta values mean that the estimation is very close to the original value. After a test which printed out the magnitudes of the coefficients, the results for the magnitude were very clear. This is displayed in table 4.1. The table lists the share of coefficients by their magnitude for different transformation sizes. While 4x4 transformations have over 12 % significant coefficients, the share goes down to less than 2 % for 32x32 transformations. Furthermore, it can be observed that nearly all significant coefficients have the magnitude 1, whereas higher magnitudes are extremely rare.

Taking another look at this information by eliminating the insignificant coefficients, which is visualised in table 4.2, shows that even then the number of coefficients with an absolute value larger than one is very low.

Evaluating these results further another interesting observation was made. Considering

<table>
<thead>
<tr>
<th>Transformation</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>&gt;3</th>
</tr>
</thead>
<tbody>
<tr>
<td>4x4</td>
<td>87.81 %</td>
<td>12.18 %</td>
<td>&lt; 0.01 %</td>
<td>0.00 %</td>
<td>0.00 %</td>
</tr>
<tr>
<td>8x8</td>
<td>93.81 %</td>
<td>6.19 %</td>
<td>&lt; 0.01 %</td>
<td>0.00 %</td>
<td>0.00 %</td>
</tr>
<tr>
<td>16x16</td>
<td>96.83 %</td>
<td>3.17 %</td>
<td>&lt; 0.01 %</td>
<td>0.00 %</td>
<td>0.00 %</td>
</tr>
<tr>
<td>32x32</td>
<td>98.16 %</td>
<td>1.84 %</td>
<td>&lt; 0.01 %</td>
<td>&lt; 0.01 %</td>
<td>0.00 %</td>
</tr>
<tr>
<td>Average</td>
<td>94.15 %</td>
<td>5.84 %</td>
<td>&lt; 0.01 %</td>
<td>&lt; 0.01 %</td>
<td>0.00 %</td>
</tr>
</tbody>
</table>

Table 4.1.: Magnitude of $\Delta$-coefficients based on total number of coefficients
4. Analysis of Δ-Coefficients

<table>
<thead>
<tr>
<th>Transformation</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>&gt;3</th>
</tr>
</thead>
<tbody>
<tr>
<td>4x4</td>
<td>99.97 %</td>
<td>0.03 %</td>
<td>0.00 %</td>
<td>0.00 %</td>
</tr>
<tr>
<td>8x8</td>
<td>99.95 %</td>
<td>0.05 %</td>
<td>0.00 %</td>
<td>0.00 %</td>
</tr>
<tr>
<td>16x16</td>
<td>99.95 %</td>
<td>0.05 %</td>
<td>0.00 %</td>
<td>0.00 %</td>
</tr>
<tr>
<td>32x32</td>
<td>99.95 %</td>
<td>0.05 %</td>
<td>&lt; 0.01 %</td>
<td>0.00 %</td>
</tr>
<tr>
<td>Average</td>
<td>99.96 %</td>
<td>0.05 %</td>
<td>&lt; 0.01 %</td>
<td>0.00 %</td>
</tr>
</tbody>
</table>

Table 4.2.: Magnitude of Δ-coefficients based on number of significant coefficients

<table>
<thead>
<tr>
<th>Settings</th>
<th>4x4</th>
<th>8x8</th>
<th>16x16</th>
<th>32x32</th>
</tr>
</thead>
<tbody>
<tr>
<td>22/22, 720p</td>
<td>25.57 %</td>
<td>13.69 %</td>
<td>7.97 %</td>
<td>3.38 %</td>
</tr>
<tr>
<td>22/24, 720p</td>
<td>32.67 %</td>
<td>18.49 %</td>
<td>12.33 %</td>
<td>5.24 %</td>
</tr>
<tr>
<td>22/22, 360p</td>
<td>30.04 %</td>
<td>15.96 %</td>
<td>11.78 %</td>
<td>5.31 %</td>
</tr>
<tr>
<td>22/24, 360p</td>
<td>35.20 %</td>
<td>18.86 %</td>
<td>14.56 %</td>
<td>7.06 %</td>
</tr>
<tr>
<td>34/34, 720p</td>
<td>41.41 %</td>
<td>30.31 %</td>
<td>18.89 %</td>
<td>13.25 %</td>
</tr>
<tr>
<td>34/36, 720p</td>
<td>47.61 %</td>
<td>37.20 %</td>
<td>24.32 %</td>
<td>15.72 %</td>
</tr>
<tr>
<td>34/34, 360p</td>
<td>50.89 %</td>
<td>35.41 %</td>
<td>21.36 %</td>
<td>15.97 %</td>
</tr>
<tr>
<td>34/36, 360p</td>
<td>55.66 %</td>
<td>42.38 %</td>
<td>24.98 %</td>
<td>17.48 %</td>
</tr>
<tr>
<td>22/18, 1080p</td>
<td>6.62 %</td>
<td>2.10 %</td>
<td>1.33 %</td>
<td>0.35 %</td>
</tr>
<tr>
<td>34/30, 1080p</td>
<td>19.35 %</td>
<td>11.96 %</td>
<td>7.10 %</td>
<td>3.38 %</td>
</tr>
<tr>
<td>Average Top-Down</td>
<td>39.88 %</td>
<td>26.54 %</td>
<td>17.02 %</td>
<td>10.43 %</td>
</tr>
<tr>
<td>Average Bottom-Up</td>
<td>12.99 %</td>
<td>7.03 %</td>
<td>4.21 %</td>
<td>1.87 %</td>
</tr>
</tbody>
</table>

Table 4.3.: Probability that all Δ-coefficients are zero for different transformations

the low number of significant coefficients, the next assessed metric was the number of Δ-coefficient maps without any significant coefficients at all. Extracting these numbers revealed that, depending on conditions like QP, resolution, sequence, prediction mode, etc., the chance to have an empty significance map can reach over 50%. An overview over different cases with their respective probabilities is displayed in table 4.3.

While extracting this information yet another observation was made. The analysis showed that the position of insignificant Δ-coefficients is strongly related to the position of insignificant estimated coefficients. To verify this connection, a test was executed to determine the relationship between the magnitudes of the estimated and Δ-coefficients. The results are displayed in table 4.4. This was later used as described in section 5.3 to change the contexts for the encoding of the significance map for Δ-coefficients.

When using the bottom-up approach, the magnitudes of the delta coefficients are higher. This correlates with a reduced number of transformations without significant Δ-coefficients as visible in table 4.3. Table 4.5 shows how the magnitudes of Δ-coefficients are distributed in the bottom-up approach. Results were taken from the configuration 22/18,
4. Analysis of ∆-Coefficients

<table>
<thead>
<tr>
<th>Magnitude estimated coefficients</th>
<th>Magnitude ∆-coefficients</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>92.12 %</td>
</tr>
<tr>
<td>1</td>
<td>85.09 %</td>
</tr>
<tr>
<td>2</td>
<td>83.43 %</td>
</tr>
<tr>
<td>3</td>
<td>78.93 %</td>
</tr>
<tr>
<td>4</td>
<td>79.67 %</td>
</tr>
<tr>
<td>5</td>
<td>80.58 %</td>
</tr>
<tr>
<td>6</td>
<td>77.35 %</td>
</tr>
<tr>
<td>7</td>
<td>79.63 %</td>
</tr>
<tr>
<td>8</td>
<td>79.01 %</td>
</tr>
<tr>
<td>9</td>
<td>78.64 %</td>
</tr>
<tr>
<td>&gt;=10</td>
<td>78.29 %</td>
</tr>
</tbody>
</table>

Table 4.4.: Relation between magnitudes of estimated and ∆-coefficients in 4x4 transformations

<table>
<thead>
<tr>
<th>Transformation</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>&gt;=3</th>
</tr>
</thead>
<tbody>
<tr>
<td>4x4</td>
<td>82.72 %</td>
<td>16.92 %</td>
<td>0.35 %</td>
<td>0.01 %</td>
<td>&lt; 0.01 %</td>
</tr>
<tr>
<td>8x8</td>
<td>85.07 %</td>
<td>14.66 %</td>
<td>0.27 %</td>
<td>&lt; 0.01 %</td>
<td>&lt; 0.01 %</td>
</tr>
<tr>
<td>16x16</td>
<td>91.62 %</td>
<td>8.19 %</td>
<td>0.19 %</td>
<td>&lt; 0.01 %</td>
<td>&lt; 0.01 %</td>
</tr>
<tr>
<td>32x32</td>
<td>95.03 %</td>
<td>4.86 %</td>
<td>0.11 %</td>
<td>&lt; 0.01 %</td>
<td>&lt; 0.01 %</td>
</tr>
<tr>
<td>Average</td>
<td>88.61 %</td>
<td>11.16 %</td>
<td>0.23 %</td>
<td>0.01 %</td>
<td>&lt; 0.01 %</td>
</tr>
</tbody>
</table>

Table 4.5.: Magnitude of ∆-coefficients for the bottom-up approach

1080p.

While these results show that the basic trends among the coefficients are the same, it also shows that there are certain differences. These are described in more detail in section 4.5.2.

4.2. Distribution of the Coefficients

Another very important part of encoding the ∆-coefficients is to analyse the structure of how they appear in the transformations. A first step was to evaluate a sufficient number of transformations trying to find patterns. A test to examine how likely it is that a position has a significant coefficient provided the following results.

The first observation is that both the delta as well as the original coefficients follow the same pattern regarding their layout. This is shown in table 4.6. The ∆-coefficients (table 4.6a) have their highest concentration in the top left corner of the transformation.
4. Analysis of $\Delta$-Coefficients

\begin{tabular}{cccccccc}
0.3477 & 0.2529 & 0.1719 & 0.0702 & 0.7148 & 0.4828 & 0.3032 & 0.1310 \\
0.2370 & 0.1771 & 0.1201 & 0.0454 & 0.4532 & 0.3168 & 0.2056 & 0.0816 \\
0.1596 & 0.1129 & 0.0769 & 0.0281 & 0.2813 & 0.1951 & 0.1276 & 0.0501 \\
0.0641 & 0.0441 & 0.0310 & 0.0108 & 0.1213 & 0.0811 & 0.0537 & 0.0207 \\
\end{tabular}

\begin{tabular}{c}
(a) $\Delta$-coefficients \\
\end{tabular}

\begin{tabular}{c}
(b) Original coefficients \\
\end{tabular}

Table 4.6.: Normalised maps for 4x4 transformations

The same can be said about the original coefficients, displayed in table 4.6b.

This distribution can also be observed in the larger transformations, the values for the 8x8 transformations are visualised in table 4.7a for the $\Delta$-coefficients and table 4.7b for the original coefficients.

One syntax element that was also analysed was the position of the last significant coefficient. As this element takes up a large chunk of the bits written into the bit stream, improving the encoding has quite a big impact on the total performance of the deflation. While analysing the relation between the last significant positions of estimated and $\Delta$-coefficients, the observation was made that it occurs, especially in larger transformations, fairly often that they are, in fact, identical. This is visualised in table 4.8. An attempt to improve the encoding was made based on this data. Details can be found in section 5.4.

Another parameter which was analysed during this thesis was the share of transformations where both the estimated and $\Delta$-coefficients have identical positions for all significant coefficients. An overview is shown in table 4.9. While the data shows that this is not the case very often, a flag was implemented based on this data as well. Looking at the data it is noteworthy that the probability is highest for 32x32 transformations. This is also the transformation where the effect it at its largest, as these have on average a higher number of coefficients than smaller transformations.

4.3. Distribution of Bytes Spent on $\Delta$-Coefficients

As described above, the $\Delta$-coefficients are a large part of the deflated file. This section provides more details regarding which syntax elements are actually using this space. A detailed overview\(^1\) is shown in table 4.10. This data is based on the original deflation scheme. It shows that the largest part of encoding the $\Delta$-coefficients is spent on the significance map and the last significant position. Other elements like the greater than one flag, greater than two flag or the absolute level remaining are taking up nearly no space at all.

This can be explained by the layout of the coefficients and the encoding. The Last

\(^1\)Please note that this overview was created using the HM-16.6. There are some smaller differences in the percentages compared to using HM-16.7, but not enough to warrant a re-run of the test.
4. Analysis of $\Delta$-Coefficients

(a) $\Delta$-coefficients

(b) Original coefficients

Table 4.7.: Normalised maps for 8x8 transformations

<table>
<thead>
<tr>
<th>Settings</th>
<th>4x4</th>
<th>8x8</th>
<th>16x16</th>
<th>32x32</th>
</tr>
</thead>
<tbody>
<tr>
<td>22/22, 720p</td>
<td>9.35 %</td>
<td>19.26 %</td>
<td>33.04 %</td>
<td>48.65 %</td>
</tr>
<tr>
<td>22/24, 720p</td>
<td>8.39 %</td>
<td>19.03 %</td>
<td>32.39 %</td>
<td>50.25 %</td>
</tr>
<tr>
<td>22/22, 360p</td>
<td>8.11 %</td>
<td>17.50 %</td>
<td>33.42 %</td>
<td>49.32 %</td>
</tr>
<tr>
<td>22/24, 360p</td>
<td>7.76 %</td>
<td>17.85 %</td>
<td>32.90 %</td>
<td>49.14 %</td>
</tr>
<tr>
<td>34/34, 720p</td>
<td>6.09 %</td>
<td>12.45 %</td>
<td>22.18 %</td>
<td>33.53 %</td>
</tr>
<tr>
<td>34/36, 720p</td>
<td>4.57 %</td>
<td>9.72 %</td>
<td>18.34 %</td>
<td>30.24 %</td>
</tr>
<tr>
<td>34/34, 360p</td>
<td>4.49 %</td>
<td>12.54 %</td>
<td>23.92 %</td>
<td>34.90 %</td>
</tr>
<tr>
<td>34/36, 360p</td>
<td>3.65 %</td>
<td>10.24 %</td>
<td>21.19 %</td>
<td>31.70 %</td>
</tr>
<tr>
<td>22/18, 1080p</td>
<td>10.98 %</td>
<td>15.58 %</td>
<td>25.90 %</td>
<td>35.31 %</td>
</tr>
<tr>
<td>34/30, 1080p</td>
<td>12.00 %</td>
<td>20.69 %</td>
<td>28.41 %</td>
<td>37.00 %</td>
</tr>
<tr>
<td>Average Top-Down</td>
<td>6.55 %</td>
<td>14.82 %</td>
<td>27.17 %</td>
<td>40.97 %</td>
</tr>
<tr>
<td>Average Bottom-Up</td>
<td>11.49 %</td>
<td>18.13 %</td>
<td>27.16 %</td>
<td>36.16 %</td>
</tr>
</tbody>
</table>

Table 4.8.: Probability that the last significant position is identical for estimated and $\Delta$-coefficients
4. Analysis of $\Delta$-Coefficients

<table>
<thead>
<tr>
<th>Settings</th>
<th>4x4</th>
<th>8x8</th>
<th>16x16</th>
<th>32x32</th>
</tr>
</thead>
<tbody>
<tr>
<td>22/22, 720p</td>
<td>1.14%</td>
<td>1.47%</td>
<td>1.48%</td>
<td>2.01%</td>
</tr>
<tr>
<td>22/24, 720p</td>
<td>1.08%</td>
<td>1.67%</td>
<td>1.66%</td>
<td>2.35%</td>
</tr>
<tr>
<td>22/22, 360p</td>
<td>0.75%</td>
<td>0.98%</td>
<td>1.28%</td>
<td>2.65%</td>
</tr>
<tr>
<td>22/24, 360p</td>
<td>0.72%</td>
<td>0.97%</td>
<td>1.26%</td>
<td>2.01%</td>
</tr>
<tr>
<td>34/34, 720p</td>
<td>1.38%</td>
<td>2.03%</td>
<td>2.20%</td>
<td>3.04%</td>
</tr>
<tr>
<td>34/36, 720p</td>
<td>1.05%</td>
<td>1.52%</td>
<td>1.79%</td>
<td>2.18%</td>
</tr>
<tr>
<td>34/34, 360p</td>
<td>0.76%</td>
<td>1.13%</td>
<td>1.05%</td>
<td>2.10%</td>
</tr>
<tr>
<td>34/36, 360p</td>
<td>0.63%</td>
<td>0.98%</td>
<td>0.95%</td>
<td>1.33%</td>
</tr>
<tr>
<td>22/18, 1080p</td>
<td>1.09%</td>
<td>0.70%</td>
<td>1.21%</td>
<td>1.50%</td>
</tr>
<tr>
<td>34/30, 1080p</td>
<td>2.79%</td>
<td>3.22%</td>
<td>3.13%</td>
<td>3.60%</td>
</tr>
</tbody>
</table>

Table 4.9.: Probability that the significance maps are identical for estimated and $\Delta$-coefficients

**Significant Position** is encoded as a combination of truncated unary and fixed length binary code, so it is unavoidable that it takes up quite some bytes for higher values. The **Significance Map** encodes one bit for each coefficient in a coded sub-block. Especially for larger transformations this requires many bits.

The **Greater Than 1 Flag** is only encoded for significant coefficients, and as the number of these is quite low for the $\Delta$-coefficients, it does not require much storage space. Furthermore, the maximal number of times this flag is signaled is limited to eight per sub-block. This is very similar to the **Greater Than 2 Flag**, which is only encoded if there is a coefficient with a magnitude of more than one among the first eight significant coefficients, meaning that the coefficient has to have the **Greater Than 1 Flag** encoded with 1. This flag is limited to just one occurrence per sub-block.

The **Sign Map** encodes one bit for each significant coefficient, with positive and negative signs being distributed evenly. This leaves very little potential for improvement. Lastly, the **Absolute Level Remaining** takes up very little space as the number of coefficients with magnitudes higher than two or that have to be encoded this way is very low.

4.4. Other Evaluated Metrics

One special case that was noticed during the coding of the $\Delta$-coefficients was that occasionally the estimated coefficients were completely insignificant. It was observed that this occurs mostly in smaller transformations, with a very high number in the bottom-up approach. Details denoting the share of transformations without any significant esti-
4. Analysis of Δ-Coefficients

<table>
<thead>
<tr>
<th>Syntax element</th>
<th>22/22, 720p</th>
<th>22/24, 720p</th>
<th>22/22, 360p</th>
<th>22/24, 360p</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Last Significant Position</td>
<td>737 174</td>
<td>509 797</td>
<td>223 958</td>
<td>164 746</td>
<td>408 919</td>
</tr>
<tr>
<td>26.50 %</td>
<td>28.57 %</td>
<td>31.29 %</td>
<td>32.53 %</td>
<td>29.72 %</td>
<td></td>
</tr>
<tr>
<td>Coded Sub-Block Flag</td>
<td>74 796</td>
<td>49 024</td>
<td>16 031</td>
<td>11 799</td>
<td>37 912</td>
</tr>
<tr>
<td>2.55 %</td>
<td>2.72 %</td>
<td>2.26 %</td>
<td>2.39 %</td>
<td>2.48 %</td>
<td></td>
</tr>
<tr>
<td>Significance Map</td>
<td>1 317 357</td>
<td>800 274</td>
<td>314 224</td>
<td>214 994</td>
<td>661 712</td>
</tr>
<tr>
<td>48.84 %</td>
<td>46.70 %</td>
<td>44.75 %</td>
<td>43.28 %</td>
<td>45.89 %</td>
<td></td>
</tr>
<tr>
<td>Greater Than 1 Flag</td>
<td>60 893</td>
<td>42 860</td>
<td>16 930</td>
<td>12 979</td>
<td>33 415</td>
</tr>
<tr>
<td>2.21 %</td>
<td>2.37 %</td>
<td>2.29 %</td>
<td>2.45 %</td>
<td>2.33 %</td>
<td></td>
</tr>
<tr>
<td>Greater Than 2 Flag</td>
<td>656</td>
<td>460</td>
<td>313</td>
<td>226</td>
<td>414</td>
</tr>
<tr>
<td>0.02 %</td>
<td>0.03 %</td>
<td>0.04 %</td>
<td>0.04 %</td>
<td>0.03 %</td>
<td></td>
</tr>
<tr>
<td>Sign Map</td>
<td>536 493</td>
<td>340 269</td>
<td>137 540</td>
<td>97 204</td>
<td>277 877</td>
</tr>
<tr>
<td>19.85 %</td>
<td>19.60 %</td>
<td>19.36 %</td>
<td>19.30 %</td>
<td>19.53 %</td>
<td></td>
</tr>
<tr>
<td>Absolute Level</td>
<td>504</td>
<td>207</td>
<td>66</td>
<td>35</td>
<td>203</td>
</tr>
<tr>
<td>Remaining</td>
<td>0.02 %</td>
<td>0.01 %</td>
<td>0.01 %</td>
<td>0.01 %</td>
<td>0.01 %</td>
</tr>
<tr>
<td>Total</td>
<td>2 727 835</td>
<td>1 742 994</td>
<td>709 031</td>
<td>502 005</td>
<td>1 420 466</td>
</tr>
</tbody>
</table>

Table 4.10.: Average byte distribution for Δ-coefficients

Estimated coefficients can be found in table 4.11. This knowledge was used to omit the CDF in some cases, as described in section 5.1.

4.5. Effects of Different Factors on Δ-Coefficients

4.5.1. Top-Down Approach

There are quite a lot of different factors playing a role in the layout of the Δ-coefficients. The most notable ones are summarised and their effects on most of the evaluated metrics described here.

- **Transformation Size**
  As one of the biggest factors in the transformations, some data comparing different transformation sizes is already presented above. The number of insignificant coefficients increases with larger transformations. This is mostly due to the generally higher number of coefficients. The magnitude of the Δ-coefficients is mostly independent from the size of the transformation. However, the probability of a transformation having no significant coefficients at all is shrinking when looking at larger transformations. This is also caused by the much higher number of coeffi-
4. Analysis of ∆-Coefficients

<table>
<thead>
<tr>
<th>Settings</th>
<th>4x4</th>
<th>8x8</th>
<th>16x16</th>
<th>32x32</th>
</tr>
</thead>
<tbody>
<tr>
<td>22/22, 720p</td>
<td>15.61%</td>
<td>8.86%</td>
<td>3.48%</td>
<td>1.00%</td>
</tr>
<tr>
<td>22/24, 720p</td>
<td>12.26%</td>
<td>8.17%</td>
<td>3.59%</td>
<td>0.65%</td>
</tr>
<tr>
<td>22/22, 360p</td>
<td>7.77%</td>
<td>5.00%</td>
<td>1.61%</td>
<td>0.44%</td>
</tr>
<tr>
<td>22/24, 360p</td>
<td>6.66%</td>
<td>5.35%</td>
<td>1.98%</td>
<td>0.42%</td>
</tr>
<tr>
<td>34/34, 720p</td>
<td>19.95%</td>
<td>15.77%</td>
<td>13.59%</td>
<td>10.54%</td>
</tr>
<tr>
<td>34/36, 720p</td>
<td>19.24%</td>
<td>15.53%</td>
<td>12.91%</td>
<td>11.74%</td>
</tr>
<tr>
<td>34/34, 360p</td>
<td>12.38%</td>
<td>9.65%</td>
<td>9.97%</td>
<td>10.53%</td>
</tr>
<tr>
<td>34/36, 360p</td>
<td>12.74%</td>
<td>9.73%</td>
<td>9.39%</td>
<td>12.68%</td>
</tr>
<tr>
<td>22/18, 1080p</td>
<td>40.52%</td>
<td>20.38%</td>
<td>9.61%</td>
<td>2.57%</td>
</tr>
<tr>
<td>34/30, 1080p</td>
<td>32.98%</td>
<td>24.89%</td>
<td>17.32%</td>
<td>8.76%</td>
</tr>
<tr>
<td>Average Top-Down</td>
<td>13.33%</td>
<td>9.76%</td>
<td>7.06%</td>
<td>6.00%</td>
</tr>
<tr>
<td>Average Bottom-Up</td>
<td>36.75%</td>
<td>22.63%</td>
<td>13.47%</td>
<td>5.67%</td>
</tr>
</tbody>
</table>

Table 4.11.: Probability that all estimated coefficients are zero coefficients per transformation.

The distribution of significant ∆-coefficients does not vary much when changing the transformation size. While the coefficients are a bit more spread out, the probabilities vary only in a small window. Another factor that is heavily influenced by the size of the transformation is the likelihood of both the estimated and the ∆-coefficients having identical last significant positions. With larger transformations the chance of this being the case is significantly higher.

- **Resolution**
  The resolution of the video to be encoded is also a variable with a strong influence on most parts of the ∆-coefficients. Due to the smaller frames the number of total transformations is lower. The percentage of significant coefficients is slightly lower, with even less absolute magnitudes larger than 1 when the frame size is smaller. Resulting from this the share of transformations without significant ∆-coefficients rises when using a lower resolution.

  The share of transformations with identical last significant positions for estimated and ∆-coefficients is roughly the same when using a different resolution. Values are slightly lower for smaller resolutions, but the difference is quite small. When looking at how many transformations have an identical significance map, it is similar. While the values are a bit lower when using a smaller resolution, it is not a large difference.

- **HQ QP**
  A higher QP reduces the number of transformations and therefore the number of coefficients. This has effects on several of the evaluated parameters. Using a higher QP strongly increases the number of transformations without any significant ∆-coefficients. However, the share of transformations where the last significant
4. Analysis of $\Delta$-Coefficients

• LQ QP
  Using a higher value for the LQ QP influences the deflation in a way very similar to using a smaller resolution. Transformations have fewer significant $\Delta$-coefficients, and the share of transformations with all coefficients being zero rises. The percentage of cases with identical last significant positions for both estimated and $\Delta$-coefficients tends to be a bit lower, but there are cases where it is actually higher. The same can be said about the share of transformations with identical significance maps.

Other factors playing significant roles in the layout and distribution of the $\Delta$-coefficients are for example the prediction mode (inter vs. intra) or the picture component (luma vs. chroma). However, listing details for every part would go beyond the scope of this thesis.

4.5.2. Bottom-Up Approach

While some of the effects described for the top-down approach are exactly the same for the bottom-up approach, there are certain differences. The magnitude of $\Delta$-coefficients is higher when compared to the original approach. This is caused by the prediction being worse for a higher quality. While most are still insignificant, the number of coefficients with absolute magnitudes of one or larger is higher. Even values higher than 3 were observed, which did not occur in any case in the top-down approach. This also correlates to the number of transformations without significant coefficients, which is much lower in comparison, as table 4.3 shows.

The distribution of the coefficients lies between the distributions of the original and the $\Delta$-coefficients for the top-down approach. This is based on the higher number of coefficients as the prediction loses a part of its accuracy. A difference is visible for the share of transformations with identical positions for the last significant coefficient for estimated and $\Delta$-coefficients. While the percentage for small transformations is higher, larger ones have lower chances of their last significant coefficient position being identical compared to the corresponding transformations in the top-down approach. This can be seen from table 4.8. The percentage of transformations with an identical significance map depends highly on the used QP settings. While the share is quite equal to the top-down approach for a low HQ QP, the difference is bigger when using a high value. On average the share is a bit higher for small but nearly equal for large transformation sizes.
5. Changes Made to Encoder and Decoder

This chapter describes the implementation details for several newly added syntax elements. It also includes schematic overviews over the influence these new elements have on the entire encoding and decoding process.

5.1. Coded Delta Flag (CDF)

The first change implemented into the encoding of $\Delta$-coefficients was the idea to encode a flag to signalise whether all coefficients in a transformation are insignificant. Since the percentage of this occurring can be quite high, as described in the previous chapter, there was a chance to actually increase the encoding efficiency and reduce the storage required for the $\Delta$-coefficients.

The procedure to achieve this is fairly straightforward. During the deflation phase, when the $\Delta$-coefficients are calculated, the CDF is set once a significant coefficient is found. If there are no significant coefficients, the flag keeps its initial value of 0. When encoding the coefficients, the first step is to encode the flag into the bit stream. Afterwards, if the flag has the value 0, the remaining steps to encode the $\Delta$-coefficients are skipped altogether. If the value is 1, the remaining steps are executed as before.

The inflation works exactly the same way in the other direction. When parsing the coefficients, the flag is parsed first. If the value is 0, the array with $\Delta$-coefficients is set to zero and the parsing process stopped. Otherwise, the value of the flag is ignored and the remaining parts of the coefficients are parsed as usual.

The CABAC context for this flag is switched based on the transformation size, so the CDF uses four contexts for luma and three for chroma.

An overview over the procedure of encoding the CDF is shown in figures 5.1a and 5.1b for deflation and inflation, respectively. As the results in the evaluation in section 6.2.1 show, using the CDF improved the efficiency of the encoding by around 0.8 percentage points.

Another addition to the CDF was the idea to omit it whenever there are no significant estimated coefficients. In this case it is clear that there must be $\Delta$-coefficients with a
5. Changes Made to Encoder and Decoder

- **Check whether transformation contains significant coefficients**
  - **Encode CDF**
    - **CDF == 1**
      - **Encode \( \Delta \)-coefficients**
    - **CDF == 0**
      - **Skip remaining encoding**
  
- **Parse CDF**
  - **CDF == 1**
    - **Parse \( \Delta \)-coefficients**
      - **Calculate original coefficients**
    - **Skip parsing of \( \Delta \)-coefficients**
  - **CDF == 0**
    - **Set coefficient array to 0**
    - **Skip calculation of original coefficients**

(a) Deflation

(b) Inflation

Figure 5.1.: Process of encoding and parsing the CDF
5. Changes Made to Encoder and Decoder

magnitude other than zero, as otherwise the original coefficients would be completely insignificant. This is, as described in section 2.2.6, handled by HEVC using the CBF.

In the deflater a loop is added before the encoding of the CDF searching a significant value in the estimated coefficients. If a coefficient is found, the loop is stopped and the CDF is encoded. Otherwise the CDF is skipped and the encoder continues with the last significant position. Similarly, the inflater also searches the estimated coefficients for any significant values. If none are found, the value for the CDF is set to one, otherwise the CDF is parsed from the bit stream. Afterwards the inflater continues with the remaining syntax elements, starting with the last significant position.

5.2. Maximum Magnitude Is 1 Flag

Another attempt to improve the encoding was made by introducing a flag to signal whether the maximum absolute value of all coefficients is 1. Using this flag allows the encoder to skip the Greater Than 1 and Greater Than 2 flags as well as the Absolute Level Remaining. As the analysis of the Δ-coefficients showed, nearly all transformations have a maximum magnitude of one. However, the analysis also showed that there are not many bits spent on these syntax elements at all. So while the potential for this improvement was not great, it was very likely that this potential was actually achieved.

There were not many coding changes required to implement this flag. In the deflation, while calculating the Δ-coefficients, the largest magnitude of all coefficients is stored. During the encoding this value is compared to 1 and this comparison written into the bit stream right after the CDF. After encoding the significance map, this value is checked again. If it is 1 the sign map is encoded, otherwise the encoder follows the original procedure with greater than 1 flag, greater than 2 flag, sign map and absolute level remaining. This is shown in figure 5.2a.

The inflation process executes the same steps in reverse order. After parsing the CDF, this flag is read from the bit stream. Once the significance map is parsed, the flag is evaluated. If its value is 1, the next element in the bit stream is the sign map. In case the value is 0, the original parsing process procedure continues with the greater than 1 flag, greater than 2 flag, sign map and absolute value remaining. A schematic overview can be found in figure 5.2b.

Similar to the CDF the context is determined based on the size of the current transformation. A total of seven contexts is used, four for luma and three for chroma.

As described in the evaluation, this feature turned out to not improve the efficiency of the encoding. Slight losses of around 0.3 percentage points occurred using this flag compared to using only the CDF. Therefore future changes were made without it.
5. Changes Made to Encoder and Decoder

(a) Deflation

- Determine maximum magnitude of $\Delta$-coefficients
- Encode maximum magnitude $== 1$
  - maximum magnitude $!= 1$
    - Encode remaining syntax elements
  - maximum magnitude $== 1$
    - Encode significance map
    - Encode sign map

(b) Inflation

- Parse maximum magnitude $== 1$
  - Parse significance map
    - maximum magnitude $!= 1$
      - Parse remaining elements
    - maximum magnitude $== 1$
      - Parse sign map

Figure 5.2.: Process of encoding and parsing the maximum magnitude $= 1$ flag
5. Changes Made to Encoder and Decoder

### Number Contexts

1. Switch based on 1
2. 2 (1/1) No switch
3. 4 (2/2) est. coeff == 0
4. 8 (4/4) magnitude of est. coeff
5. 10 (6/4) est. coeff == 0, position
6. 44 (28/16) est. coeff == 0, otherwise original contexts
7. 84 (54/30) different sets of original contexts, switched on est. coeff == 0

Table 5.1.: Overview over different context configurations used for encoding the significance map

### 5.3. Significant Coefficient Map

As described in section 4.3, the significance map takes up the most space in the coefficient encoding. Due to this fact improving the encoding of the significant coefficient map provides a large potential for improving the general performance of the encoder.

#### 5.3.1. Changing Contexts

The first attempt to improve the encoding of the significance map was made by changing the number of contexts used. While this does not provide very large potential for improvement from a bit rate standpoint, it evaluates changes to the number of contexts and therefore the coding complexity. The original HEVC standard uses a total of 42 contexts (27 for luma and 15 for chroma). A total of six different configurations were tested, an overview can be found in table 5.1. The context switches were mostly based on the knowledge gained from the relation between the magnitude of estimated and $\Delta$-coefficients, as described in section 4.1.

The base case with just one context for each luma and chroma was established to see the total effect of using different contexts at all.

In the case with two contexts each for luma and chroma the switch was made based on whether the estimated coefficient in the identical position is significant or not. This was extended for the encoding scheme with a total of eight contexts, using the magnitude of the estimated coefficient as a switch. Here the following separation was made:

- context 0 $\iff$ magnitude estimated coefficient $= 0$
- context 1 $\iff$ magnitude estimated coefficient $= 1$ or $2$
- context 2 $\iff$ magnitude estimated coefficient $> 2$ and $< 8$
- context 3 $\iff$ magnitude estimated coefficient $\geq 8$

1Read: total (luma/chroma)
5. Changes Made to Encoder and Decoder

A fourth alternative of varying the contexts was made based on a combination of the original variant based on the position and the magnitude of the estimated coefficient. This resulted in six used contexts for luma and four for chroma. A map was used to switch the context based on the positions. The maps are shown in figures 5.3a and 5.3b for luma and chroma, respectively. These maps were used for all coefficients in the top-left sub-block. Coefficients in other sub-blocks were assigned to context 5 or 3, respectively. Context 0 was used for all coefficients which had an insignificant coefficient in the corresponding position of the estimated coefficients.

Two more variants were tested based on the original contexts. In the first test a total of 44 different contexts were used, thereof 28 for luma and 16 for chroma. Here the original 42 contexts were used whenever the estimated coefficient in the same position was significant and one separate context for all positions with insignificant estimated coefficients. This idea was extended to use the same differentiation between contexts for positions without significant estimated coefficients. This resulted in a sum of 84 used contexts.

The methods using between four and ten contexts produced very similar results, reducing the average bit rate by around 0.25 percentage points. Using just a single context is quite costly, whereas using twice the total number compared to the original set of 42 contexts provides the largest gains for all tests. The details on the results can be found in section 6.2.3.

5.3.2. Significance Map Identical Flag

A completely different attempt to improve the encoding was made by implementing a flag to signal that the significance maps for both estimated and Δ-coefficients are identical. This is, as the analysis showed, not very often the case, but due to the large potential gain it was deemed worth trying nevertheless. When the flag is set, it is not necessary to encode any of the last significant position, coded sub-block flags or the significance map as these can be calculated from the estimated coefficients. Especially in larger transformations this can take up a lot of bits. This flag was called SigMapIdentical.

Using this flag changes the process of both deflation and inflation as displayed in figures 5.4a and 5.4b for deflation and inflation, respectively. In the deflation a check
5. Changes Made to Encoder and Decoder

Determine whether significance maps are equal

Encode SigMapIdentical

SigMapIdentical != 1

Encode last significant position

Encode remaining syntax elements

SigMapIdentical == 1

Encode coefficient level flags

(a) Deflation

Parse SigMapIdentical

SigMapIdentical != 1

Parse last significant position

Parse remaining syntax elements

SigMapIdentical == 1

Calculate last significant position

Calculate coded sub-blocks

Calculate significance map

(b) Inflation

Figure 5.4.: Process of encoding and parsing the SigMapIdentical flag
5. Changes Made to Encoder and Decoder

whether the significance maps are identical was added after the encoding of the CDF. This is followed by writing this status bit into the stream. If it is set, the next bits written are the greater than 1 flags from the last coded sub-block. In case the value of the flag is 0, the encoder continues with the last significant position, the significance map of the last sub-block, and so on in the normal encoding order.

For the inflation the process is similar. After parsing the CDF this flag is read out. If it is not set, the usual process continues. Otherwise the last significant position, coded sub-blocks, and significance map are calculated based on the estimated coefficients. The next information parsed out of the bit stream is in this case the greater than 1 flags from the last coded sub-block.

Using this flag did not provide any gains, it also caused only very slight increases in the bit rates of around 0.1 to 0.2 percentage points. In a second implementation this flag was only applied to 32x32 transformations. Here these losses were eliminated, however, it also did not reduce the bit rates. Details on the results can be found in section 6.2.3.

5.4. Last Significant Coefficient

The last element where an attempt was made to improve the encoding through a newly introduced flag is the last significant coefficient. As described in the analysis in section 4.2, it is fairly common that the last significant position is identical for both the estimated and ∆-coefficients.

The process of encoding this flag, it received the working title LscIdentical, is very straightforward. In the deflation, while looking for the last significant position among the ∆-coefficients, the same is determined for the estimated coefficients. If these are identical the LscIdentical flag is set to 1, otherwise it remains at its initial value of 0. This flag is encoded right before the position is written into the bit stream. If the flag is set, the encoding of the position is skipped. Afterwards the encoder continues with the significance map of the last sub-block and the remaining syntax elements.

For the inflation the process works precisely in the other direction. Before the parser tries to read out the last significant position, it reads the LscIdentical flag from the bit stream. If it is set, the last significant position is determined from the estimated coefficients. The next step is then to parse the significance map of the last significant sub-block. Otherwise it reads the last significant position from the bit stream and then continues with the ordinary parsing process.

This flag is encoded using one context per transformation size for each luma and chroma, so the total count is seven. In a different encoding variant only five contexts are used, as this flag is not applied for 4x4 transformations.

If this flag is used in combination with the SigMapIdentical flag, this flag is skipped whenever the SigMapIdentical flag is set. An overview over the modified process can be
5. Changes Made to Encoder and Decoder

found in figure 5.5. Using this flag reduced the average bit rates by around 0.8 percentage points.
5. Changes Made to Encoder and Decoder

(a) Deflation

- Determine whether LSC is identical
- Encode LscIdentical
  - LscIdentical != 1
    - Encode last significant position
    - Encode remaining syntax elements
  - LscIdentical == 1
    - Encode coded sub-blocks
    - Encode remaining syntax elements

- Parse LscIdentical
  - LscIdentical != 1
    - Parse last significant position
    - Parse remaining syntax elements
  - LscIdentical == 1
    - Calculate last significant position from estimated coefficients
    - Parse remaining syntax elements

(b) Inflation

Figure 5.5.: Process of encoding and parsing the LscIdentical flag
6. Evaluation

6.1. Peak Signal-to-Noise Ratio

PSNR is a measurement to determine how clear the picture is, comparing the signal strength to the noise. Since deflation is a loss-less coding scheme, there should be no differences between the picture quality of the sequence before the deflation and after the inflation. While these values were closely monitored, there were never any differences encountered.

Table 6.1 shows the differences for PSNR values for different QPs and resolutions, highlighting the fact that there is no difference between the original video sequence and the inflated version. The table shows only the values for luma, the chroma values are a bit higher, but are also identical for the original and inflated sequences. The PSNR is measured in dB.

6.2. Encoding Efficiency

The main measurement for the efficiency of the encoding is the bit rate. It is directly based on the file size, a smaller file resulting in a lower bit rate, indicating a higher efficiency. As there were several different elements added or changed during the work on this thesis, each of them was evaluated to determine the respective influence on the performance of the total deflation.

Tables show the difference to the original file, so a deflation value of 30 % means that the deflated file takes up 30 % less space or the bit rate is 30 % lower. The improvement is

<table>
<thead>
<tr>
<th>Settings</th>
<th>Original Video</th>
<th>Inflated</th>
</tr>
</thead>
<tbody>
<tr>
<td>QP 22, 720p</td>
<td>40.94</td>
<td>40.94</td>
</tr>
<tr>
<td>QP 22, 360p</td>
<td>40.94</td>
<td>40.94</td>
</tr>
<tr>
<td>QP 34, 720p</td>
<td>34.24</td>
<td>34.24</td>
</tr>
<tr>
<td>QP 34, 360p</td>
<td>32.88</td>
<td>32.88</td>
</tr>
<tr>
<td>QP 22, 1080p</td>
<td>39.36</td>
<td>39.36</td>
</tr>
<tr>
<td>QP 34, 1080p</td>
<td>34.48</td>
<td>34.48</td>
</tr>
</tbody>
</table>

Table 6.1.: Comparison of PSNR values between different settings
6. Evaluation

<table>
<thead>
<tr>
<th>Settings</th>
<th>Original</th>
<th>CDF</th>
<th>Omitting</th>
<th>Improvement</th>
<th>Improvement 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>22/22, 720p</td>
<td>32.68 %</td>
<td>33.24 %</td>
<td>33.41 %</td>
<td>+0.56 %</td>
<td>+0.73 %</td>
</tr>
<tr>
<td>22/24, 720p</td>
<td>33.23 %</td>
<td>33.91 %</td>
<td>34.09 %</td>
<td>+0.68 %</td>
<td>+0.86 %</td>
</tr>
<tr>
<td>22/22, 360p</td>
<td>38.91 %</td>
<td>39.58 %</td>
<td>39.70 %</td>
<td>+0.67 %</td>
<td>+0.79 %</td>
</tr>
<tr>
<td>22/24, 360p</td>
<td>37.47 %</td>
<td>38.23 %</td>
<td>38.35 %</td>
<td>+0.76 %</td>
<td>+0.88 %</td>
</tr>
<tr>
<td>34/34, 720p</td>
<td>21.07 %</td>
<td>21.87 %</td>
<td>22.30 %</td>
<td>+0.80 %</td>
<td>+1.23 %</td>
</tr>
<tr>
<td>34/36, 720p</td>
<td>21.30 %</td>
<td>22.30 %</td>
<td>22.80 %</td>
<td>+1.00 %</td>
<td>+1.50 %</td>
</tr>
<tr>
<td>34/34, 360p</td>
<td>24.60 %</td>
<td>25.68 %</td>
<td>26.07 %</td>
<td>+1.08 %</td>
<td>+1.47 %</td>
</tr>
<tr>
<td>34/36, 360p</td>
<td>23.12 %</td>
<td>24.39 %</td>
<td>24.85 %</td>
<td>+1.27 %</td>
<td>+1.73 %</td>
</tr>
<tr>
<td>22/18, 1080p</td>
<td>13.88 %</td>
<td>14.08 %</td>
<td>14.20 %</td>
<td>+0.20 %</td>
<td>+0.32 %</td>
</tr>
<tr>
<td>34/30, 1080p</td>
<td>9.89 %</td>
<td>10.23 %</td>
<td>10.52 %</td>
<td>+0.34 %</td>
<td>+0.63 %</td>
</tr>
<tr>
<td>Average Top-Down</td>
<td>29.05 %</td>
<td>29.90 %</td>
<td>30.20 %</td>
<td>+0.85 %</td>
<td>+1.15 %</td>
</tr>
<tr>
<td>Average Bottom-Up</td>
<td>11.89 %</td>
<td>12.15 %</td>
<td>12.36 %</td>
<td>+0.26 %</td>
<td>+0.47 %</td>
</tr>
</tbody>
</table>

Table 6.2.: Effects of the CDF on the deflation

given in percentage points, comparing the new elements to the original deflation scheme.

6.2.1. Coded Delta Flag (CDF)

The first new syntax element to be implemented was the CDF. As quite a lot of transformations have no significant $\Delta$-coefficients, it proved to have a very positive influence on the deflation. An overview over different cases can be found in table 6.2. The column named CDF shows the results of encoding the CDF in all cases, while the Omitting column shows the results of encoding the flag only when there are significant estimated coefficients. The improvement column shows the gain of using the CDF in all cases, whereas Improvement 2 shows the difference to using it only for transformations with significant estimated coefficients.

These results indicate that the CDF has advantages in all cases. This effect is stronger when using a low resolution or a high QP. A positive result, even if the impact is smaller, can be observed in the bottom-up approach as well. The lower improvement is caused by the much lower number of transformations without significant coefficients, as already described in the analysis.

Based on these results, it was decided to use the CDF in all cases moving forward. As the idea of omitting the CDF was implemented later during the progress of this thesis, results use the CDF in all cases. Unless explicitly stated, results contain the gain from this new syntax element without omitting.
6. Evaluation

<table>
<thead>
<tr>
<th>Settings</th>
<th>CDF</th>
<th>Max. Magnitude and CDF</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>22/22, 720p</td>
<td>33.24 %</td>
<td>33.28 %</td>
<td>+0.04 %</td>
</tr>
<tr>
<td>22/24, 720p</td>
<td>33.91 %</td>
<td>33.90 %</td>
<td>-0.01 %</td>
</tr>
<tr>
<td>22/22, 360p</td>
<td>39.58 %</td>
<td>39.44 %</td>
<td>-0.14 %</td>
</tr>
<tr>
<td>22/24, 360p</td>
<td>38.23 %</td>
<td>38.03 %</td>
<td>-0.20 %</td>
</tr>
<tr>
<td>34/34, 720p</td>
<td>21.87 %</td>
<td>21.64 %</td>
<td>-0.23 %</td>
</tr>
<tr>
<td>34/36, 720p</td>
<td>22.30 %</td>
<td>22.02 %</td>
<td>-0.28 %</td>
</tr>
<tr>
<td>34/34, 360p</td>
<td>25.68 %</td>
<td>25.20 %</td>
<td>-0.48 %</td>
</tr>
<tr>
<td>34/36, 360p</td>
<td>24.39 %</td>
<td>23.83 %</td>
<td>-0.56 %</td>
</tr>
<tr>
<td>22/18, 1080p</td>
<td>14.08 %</td>
<td>14.01 %</td>
<td>-0.07 %</td>
</tr>
<tr>
<td>34/30, 1080p</td>
<td>10.23 %</td>
<td>10.03 %</td>
<td>-0.20 %</td>
</tr>
<tr>
<td>Average Top-Down</td>
<td>29.90 %</td>
<td>29.67 %</td>
<td>-0.23 %</td>
</tr>
<tr>
<td>Average Bottom-Up</td>
<td>12.15 %</td>
<td>12.02 %</td>
<td>-0.13 %</td>
</tr>
</tbody>
</table>

Table 6.3.: Results using the Maximum Magnitude = 1 flag

6.2.2. Maximum Magnitude Is 1 Flag

When looking at this flag, it has to be considered that this is a high efficiency, but low potential element. Therefore it actually does not surprise that the results show no improvement, but rather a decrease in encoding efficiency. These results are summarised in table 6.3.

Especially when using both a low resolution and a high QP, the losses gets quite large. This is caused by the fact that the coefficients do not take up as much of the bit stream as in other configurations. Encoding a flag for every transformation without much savings proves to be quite costly.

6.2.3. Significance Map

As changes were made to two different aspects of encoding the significance map, they had to be evaluated separately.

Changing the Contexts

Varying the CABAC contexts was assumed to have a rather small impact on the overall performance, but well-chosen contexts can still provide a boost. A total of six different context switches was implemented and compared to the original ones. The results can be found in table 6.4. All tests include the use of the CDF, which is taken as the base level for the comparison. The column named CDF uses the original 42 contexts.
### 6. Evaluation

#### Table 6.4.: Test results for changing the significance map contexts

<table>
<thead>
<tr>
<th>Settings</th>
<th>CDF</th>
<th>1/1 ctx</th>
<th>2/2 ctx</th>
<th>4/4 ctx</th>
<th>6/4 ctx</th>
<th>28/16 ctx</th>
<th>54/30 ctx</th>
</tr>
</thead>
<tbody>
<tr>
<td>22/22, 720p</td>
<td>33.24 %</td>
<td>32.72 %</td>
<td>33.13 %</td>
<td>33.15 %</td>
<td>33.14 %</td>
<td>33.11 %</td>
<td>33.79 %</td>
</tr>
<tr>
<td>22/24, 720p</td>
<td>33.91 %</td>
<td>33.48 %</td>
<td>34.00 %</td>
<td>34.02 %</td>
<td>34.02 %</td>
<td>33.96 %</td>
<td>34.53 %</td>
</tr>
<tr>
<td>22/22, 360p</td>
<td>39.58 %</td>
<td>39.39 %</td>
<td>39.78 %</td>
<td>39.79 %</td>
<td>39.79 %</td>
<td>39.63 %</td>
<td>39.94 %</td>
</tr>
<tr>
<td>22/24, 360p</td>
<td>38.23 %</td>
<td>38.10 %</td>
<td>38.57 %</td>
<td>38.58 %</td>
<td>38.57 %</td>
<td>38.37 %</td>
<td>38.61 %</td>
</tr>
<tr>
<td>34/34, 720p</td>
<td>21.87 %</td>
<td>21.48 %</td>
<td>21.95 %</td>
<td>21.94 %</td>
<td>21.92 %</td>
<td>21.81 %</td>
<td>22.34 %</td>
</tr>
<tr>
<td>34/36, 720p</td>
<td>22.30 %</td>
<td>22.04 %</td>
<td>22.42 %</td>
<td>22.41 %</td>
<td>22.39 %</td>
<td>22.24 %</td>
<td>22.67 %</td>
</tr>
<tr>
<td>34/34, 360p</td>
<td>25.68 %</td>
<td>25.74 %</td>
<td>26.17 %</td>
<td>26.16 %</td>
<td>26.15 %</td>
<td>25.76 %</td>
<td>25.90 %</td>
</tr>
<tr>
<td>34/36, 360p</td>
<td>24.39 %</td>
<td>24.54 %</td>
<td>24.89 %</td>
<td>24.89 %</td>
<td>24.87 %</td>
<td>24.40 %</td>
<td>24.51 %</td>
</tr>
<tr>
<td>22/18, 1080p</td>
<td>14.08 %</td>
<td>13.08 %</td>
<td>13.76 %</td>
<td>13.76 %</td>
<td>13.77 %</td>
<td>13.77 %</td>
<td>14.87 %</td>
</tr>
<tr>
<td>34/30, 1080p</td>
<td>10.23 %</td>
<td>8.97 %</td>
<td>10.32 %</td>
<td>10.32 %</td>
<td>10.30 %</td>
<td>10.29 %</td>
<td>11.54 %</td>
</tr>
</tbody>
</table>

#### Average

| Top-Down  | 29.90 %   | 29.69 %    | 30.11 %    | 30.12 %    | 30.11 %    | 29.91 %     | 30.29 %     |
| Bottom-Up | 12.15 %   | 11.02 %    | 12.04 %    | 12.04 %    | 12.03 %    | 13.20 %     |             |

#### (a) Test results

<table>
<thead>
<tr>
<th>Settings</th>
<th>1/1 ctx</th>
<th>2/2 ctx</th>
<th>4/4 ctx</th>
<th>6/4 ctx</th>
<th>28/16 ctx</th>
<th>54/30 ctx</th>
</tr>
</thead>
<tbody>
<tr>
<td>22/22, 720p</td>
<td>-0.52 %</td>
<td>-0.11 %</td>
<td>-0.09 %</td>
<td>-0.10 %</td>
<td>-0.13 %</td>
<td>+0.55 %</td>
</tr>
<tr>
<td>22/24, 720p</td>
<td>-0.43 %</td>
<td>+0.09 %</td>
<td>+0.11 %</td>
<td>+0.11 %</td>
<td>+0.05 %</td>
<td>+0.62 %</td>
</tr>
<tr>
<td>22/22, 360p</td>
<td>-0.19 %</td>
<td>+0.20 %</td>
<td>+0.21 %</td>
<td>+0.21 %</td>
<td>+0.05 %</td>
<td>+0.36 %</td>
</tr>
<tr>
<td>22/24, 360p</td>
<td>-0.13 %</td>
<td>+0.34 %</td>
<td>+0.35 %</td>
<td>+0.34 %</td>
<td>+0.14 %</td>
<td>+0.38 %</td>
</tr>
<tr>
<td>34/34, 720p</td>
<td>-0.39 %</td>
<td>+0.08 %</td>
<td>+0.07 %</td>
<td>+0.05 %</td>
<td>-0.06 %</td>
<td>+0.47 %</td>
</tr>
<tr>
<td>34/36, 720p</td>
<td>-0.26 %</td>
<td>+0.12 %</td>
<td>+0.11 %</td>
<td>+0.09 %</td>
<td>-0.06 %</td>
<td>+0.37 %</td>
</tr>
<tr>
<td>34/34, 360p</td>
<td>+0.06 %</td>
<td>+0.49 %</td>
<td>+0.48 %</td>
<td>+0.47 %</td>
<td>+0.08 %</td>
<td>+0.22 %</td>
</tr>
<tr>
<td>34/36, 360p</td>
<td>+0.15 %</td>
<td>+0.50 %</td>
<td>+0.50 %</td>
<td>+0.48 %</td>
<td>+0.01 %</td>
<td>+0.12 %</td>
</tr>
<tr>
<td>22/18, 1080p</td>
<td>-1.00 %</td>
<td>-0.32 %</td>
<td>-0.32 %</td>
<td>-0.31 %</td>
<td>-0.31 %</td>
<td>+0.79 %</td>
</tr>
<tr>
<td>34/30, 1080p</td>
<td>-1.26 %</td>
<td>+0.09 %</td>
<td>+0.09 %</td>
<td>+0.07 %</td>
<td>+0.06 %</td>
<td>+1.31 %</td>
</tr>
</tbody>
</table>

#### Average

| Top-Down  | -0.21 %  | +0.21 %  | +0.22 %  | +0.21 %  | +0.01 %   | +0.39 %   |
| Bottom-Up | -1.13 %  | -0.11 %  | -0.11 %  | -0.12 %  | +1.05 %   |             |

#### (b) Comparison

Table 6.4.: Test results for changing the significance map contexts
6. Evaluation

<table>
<thead>
<tr>
<th>Settings</th>
<th>CDF</th>
<th>SigMapIdentical</th>
<th>SigMapIdentical for 32x32</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>22/22, 720p</td>
<td>33.24 %</td>
<td>33.16 %</td>
<td>33.24 %</td>
<td>-0.00 %</td>
</tr>
<tr>
<td>22/24, 720p</td>
<td>33.91 %</td>
<td>33.86 %</td>
<td>33.92 %</td>
<td>+0.01 %</td>
</tr>
<tr>
<td>22/22, 360p</td>
<td>39.58 %</td>
<td>39.49 %</td>
<td>39.58 %</td>
<td>+0.00 %</td>
</tr>
<tr>
<td>22/24, 360p</td>
<td>38.23 %</td>
<td>38.15 %</td>
<td>38.23 %</td>
<td>+0.00 %</td>
</tr>
<tr>
<td>34/34, 720p</td>
<td>21.87 %</td>
<td>21.78 %</td>
<td>21.88 %</td>
<td>+0.01 %</td>
</tr>
<tr>
<td>34/36, 720p</td>
<td>22.30 %</td>
<td>22.19 %</td>
<td>22.30 %</td>
<td>+0.00 %</td>
</tr>
<tr>
<td>34/34, 360p</td>
<td>25.68 %</td>
<td>25.56 %</td>
<td>25.67 %</td>
<td>-0.01 %</td>
</tr>
<tr>
<td>34/36, 360p</td>
<td>24.39 %</td>
<td>24.26 %</td>
<td>24.38 %</td>
<td>-0.01 %</td>
</tr>
<tr>
<td>22/18, 1080p</td>
<td>14.08 %</td>
<td>14.00 %</td>
<td>14.08 %</td>
<td>+0.00 %</td>
</tr>
<tr>
<td>34/30, 1080p</td>
<td>10.23 %</td>
<td>10.26 %</td>
<td>10.30 %</td>
<td>+0.07 %</td>
</tr>
<tr>
<td>Average Top-Down</td>
<td>29.90 %</td>
<td>29.81 %</td>
<td>29.90 %</td>
<td>+0.00 %</td>
</tr>
<tr>
<td>Average Bottom-Up</td>
<td>12.15 %</td>
<td>12.13 %</td>
<td>12.19 %</td>
<td>+0.04 %</td>
</tr>
</tbody>
</table>

Table 6.5.: Results using the SigMapIdentical flag

This data shows that the impact in the top-down approach is quite small, with a difference of just 0.6 percentage points between the worst (1/1 contexts) and best (54/30 contexts) cases. In some settings, for example when using a high QP and low resolution, the performance of the deflater is actually better when using two contexts compared to the 42 original ones. In the bottom-up approach the effect is much larger due to the higher number of coefficients per transformation. Here the difference is 2.18 percentage points, showing that in this case a well-chosen selection of contexts can have a significantly positive impact on the efficiency of the deflation.

**SigMapIdentical Flag**

The effect of this flag was evaluated for two similar cases: Using it for all transformation sizes, and using it only for 32x32 transformations. While it did not have a large impact in both cases, it is notable that using the flag for only 32x32 transformations allowed the deflation to run at the same efficiency as not using it at all, despite it only being active in around 2% of the cases. This finds its reason in both the relatively low number of 32x32 transformations, and the large potential of not having to encode the last significant position, the significance map and the coded sub-block flags. Details on the results can be found in table 6.5. The *Improvement* column compares the implementation without the flag with using the flag only for 32x32 transformations.

As the effects of this flag are minimal, it was not used in further tests. However, as the results proved quite interesting the implementation was kept and disabled via a preprocessor directive.
6. Evaluation

Table 6.6.: Results using the LscIdentical flag

<table>
<thead>
<tr>
<th>Settings</th>
<th>CDF</th>
<th>LscIdentical</th>
<th>LscIdentical for ≥ 8x8</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>22/22, 720p</td>
<td>33.24 %</td>
<td>34.09 %</td>
<td>34.07 %</td>
<td>+0.83 %</td>
</tr>
<tr>
<td>22/24, 720p</td>
<td>33.91 %</td>
<td>34.93 %</td>
<td>34.90 %</td>
<td>+0.99 %</td>
</tr>
<tr>
<td>22/22, 360p</td>
<td>39.58 %</td>
<td>40.16 %</td>
<td>40.14 %</td>
<td>+0.56 %</td>
</tr>
<tr>
<td>22/24, 360p</td>
<td>38.23 %</td>
<td>38.94 %</td>
<td>38.90 %</td>
<td>+0.67 %</td>
</tr>
<tr>
<td>34/34, 720p</td>
<td>21.87 %</td>
<td>22.82 %</td>
<td>22.88 %</td>
<td>+1.01 %</td>
</tr>
<tr>
<td>34/36, 720p</td>
<td>22.30 %</td>
<td>23.18 %</td>
<td>23.25 %</td>
<td>+0.95 %</td>
</tr>
<tr>
<td>34/34, 360p</td>
<td>25.68 %</td>
<td>26.52 %</td>
<td>26.55 %</td>
<td>+0.87 %</td>
</tr>
<tr>
<td>34/36, 360p</td>
<td>24.39 %</td>
<td>25.21 %</td>
<td>25.26 %</td>
<td>+0.87 %</td>
</tr>
<tr>
<td>22/18, 1080p</td>
<td>14.08 %</td>
<td>14.40 %</td>
<td>14.50 %</td>
<td>+0.42 %</td>
</tr>
<tr>
<td>34/30, 1080p</td>
<td>10.23 %</td>
<td>11.40 %</td>
<td>11.50 %</td>
<td>+1.27 %</td>
</tr>
<tr>
<td>Average Top-Down</td>
<td>29.90 %</td>
<td>30.73 %</td>
<td>30.74 %</td>
<td>+0.84 %</td>
</tr>
<tr>
<td>Average Bottom-Up</td>
<td>12.15 %</td>
<td>12.90 %</td>
<td>13.00 %</td>
<td>+0.85 %</td>
</tr>
</tbody>
</table>

6.2.4. Last Significant Coefficient

Another flag that was implemented was the LscIdentical flag. Similar to the SigMapIdentical flag, it was also implemented for two different cases. In the first case it was used for all transformation sizes, in the second case only for transformations larger than 4x4. While the results for the top-down approach were nearly identical, in the bottom-up approach the second implementation provided some advantages. Results can be found in table 6.6. In the Improvement column the effect of the implementation for transformation sizes of 8x8 and larger are evaluated.

This flag has a quite big effect on the deflation, even larger than the CDF. Notable is also the nearly equal increase in efficiency in the bottom-up approach.

6.3. Discussion of Test Results

As the detailed results above already show, there are several ways to improve the performance of the deflation. The first new syntax element, the CDF, had a very positive impact and was used for all further tests. Another new element increasing the encoding efficiency was the LscIdentical flag. Its effect in the top-down approach was very similar to the CDF, but it really improved the performance for the bottom-up approach. Changing the contexts for the significance map also boosted the efficiency of the deflater. While it is not a new element, using the knowledge gained from the estimated coefficients proved to be a good addition to the current scheme.

Using these elements together provides quite an improvement over the original scheme.
### 6. Evaluation

<table>
<thead>
<tr>
<th>Settings</th>
<th>Original Deflation</th>
<th>Version 1</th>
<th>Version 2</th>
<th>Improvement Version 1</th>
<th>Improvement Version 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>22/22, 720p</td>
<td>32.68 %</td>
<td>34.16 %</td>
<td>34.80 %</td>
<td>+1.48 %</td>
<td>+2.12 %</td>
</tr>
<tr>
<td>22/24, 720p</td>
<td>33.23 %</td>
<td>35.19 %</td>
<td>35.71 %</td>
<td>+1.96 %</td>
<td>+2.48 %</td>
</tr>
<tr>
<td>22/22, 360p</td>
<td>38.91 %</td>
<td>40.48 %</td>
<td>40.62 %</td>
<td>+1.57 %</td>
<td>+1.71 %</td>
</tr>
<tr>
<td>22/24, 360p</td>
<td>37.47 %</td>
<td>39.37 %</td>
<td>39.40 %</td>
<td>+1.90 %</td>
<td>+1.93 %</td>
</tr>
<tr>
<td>34/34, 720p</td>
<td>21.07 %</td>
<td>23.40 %</td>
<td>23.80 %</td>
<td>+2.33 %</td>
<td>+2.73 %</td>
</tr>
<tr>
<td>34/36, 720p</td>
<td>21.30 %</td>
<td>23.87 %</td>
<td>24.13 %</td>
<td>+2.57 %</td>
<td>+2.83 %</td>
</tr>
<tr>
<td>34/34, 360p</td>
<td>24.60 %</td>
<td>27.42 %</td>
<td>27.17 %</td>
<td>+2.82 %</td>
<td>+2.57 %</td>
</tr>
<tr>
<td>34/36, 360p</td>
<td>23.12 %</td>
<td>26.21 %</td>
<td>25.83 %</td>
<td>+3.09 %</td>
<td>+2.71 %</td>
</tr>
<tr>
<td>22/18, 1080p</td>
<td>13.88 %</td>
<td>14.30 %</td>
<td>15.41 %</td>
<td>+0.42 %</td>
<td>+1.53 %</td>
</tr>
<tr>
<td>34/30, 1080p</td>
<td>9.89 %</td>
<td>11.89 %</td>
<td>13.11 %</td>
<td>+2.00 %</td>
<td>+3.22 %</td>
</tr>
<tr>
<td>Average Top-Down</td>
<td>29.05 %</td>
<td>31.26 %</td>
<td>31.43 %</td>
<td>+2.21 %</td>
<td>+2.38 %</td>
</tr>
<tr>
<td>Average Bottom-Up</td>
<td>11.89 %</td>
<td>13.10 %</td>
<td>14.26 %</td>
<td>+1.21 %</td>
<td>+2.37 %</td>
</tr>
</tbody>
</table>

Table 6.7.: Overall improvement for deflation

This is shown in table 6.7. Version 1 describes the efficiency of the deflation using CDF (omitted if possible), LscIdentical flag for transformation sizes 8x8 and larger, and 4/4 contexts for the significance map, whereas version 2 uses CDF (omitted it if possible), LscIdentical flag for transformation sizes 8x8 and larger, and 54/30 contexts for the significance map.

Other elements had either not a large impact, or even reduced the efficiency for the deflation. The SigMapIdentical flag did not have a strong influence on the performance of the deflater, even though it allows to skip three different elements when it is set. Adding a flag to signal that the maximum magnitude is 1 decreased the efficiency. Even though it is the case in most of the transformations, the combination of CABAC and syntax elements allows this case to be encoded very efficiently.

The results show that it is possible to optimise the encoder for encoding Δ-coefficients instead of HEVC transform coefficients. The gains in the top-down approach do not vary much between both versions mentioned above and are between 1.5 and 3 %. Depending on the settings both versions can produce better results. Overall version 2 creates slightly more gains at the costs of more CABAC contexts. The lower total gains when using a high HQ QP like 34 are caused by the smaller share of Δ-coefficients, as detailed in table 3.1. The data also shows that deflation is most efficient for smaller resolutions, as the values for sequences in 360p are higher than for 720p.

The spread in the bottom-up approach is much larger. Version 2 brings much more gain compared to version 1. The reason can be found, as described above, in the selection of CABAC contexts for the significance map. The example shows that, when using a high HQ QP like 34, the deflation can be improved by more than 30 % from the original scheme. Another difference compared to the top-down approach is the larger distance...
6. Evaluation

between the two evaluated versions. A similarity to the original approach is that the
deflation is less efficient for higher HQ QPs, as the $\Delta$-coefficients are a smaller part of
the encoded file. While the share is still high compared to the top-down variant, the
different layout of the coefficients causes the efficiency to go down.

6.3.1. Principle of Independent Parsing

However, there is also the principle of independent parsing. This principle describes that
the parsing of the $\Delta$-coefficients can be done independently from the calculation of the
estimated coefficients. It was adapted to allow the use of parallel processing for the
calculation of the coefficients in the decoder. It is only restricted to the actual act of
parsing the coefficients though. It would be okay to have a flag saying that two elements
are identical for the estimated and $\Delta$-coefficients, as the parser would not be influenced,
other than knowing that an element can be taken from the estimated coefficients later
without parsing it from the $\Delta$-coefficients.

On the other hand it is problematic to base the parsing process on direct data from the
estimated coefficients. For example, using the magnitude of an estimated coefficient to
choose a CABAC context violates this principle, as in this case the parser requires the
knowledge of the coefficients magnitude to be able to parse the next bit.

This principle affects several of the introduced or changed syntax elements. The $CDF$
does not create a parsing dependency, similar to the maximum magnitude is 1 flag, as
both focus on special cases of the layout of the $\Delta$-coefficients. However, adding the
omitting option to the $CDF$ causes the dependency as the decision whether to omit
the flag or not is based on the estimated coefficients. Changing the contexts for the
significance map based on the magnitude of the estimated coefficients already served as
the example of the parsing dependency.

The $LscIdentical$ and $SigMapIdentical$ flags also have their contexts determined by the
transformation size and do not create a parsing dependency for their own decoding.
However, both cause a dependency as the position of the last significant coefficient also
indicates how many bits have to be read to determine the content of the significance
map. Likewise, knowing that the significance map is identical does not tell the parser
how many significant coefficients there are, and therefore it is unknown how many of the
next bits are to be interpreted as greater than 1 flags.

This principle is, however, not set in stone and can be ignored based on the preferences
of the application. For example, if parallel processing is not an option, there is no point
in following a costly principle by ignoring syntax elements which provide a gain. If the
available hardware setting allows and improves the performance for parallel processing,
the principle should be maintained.

Another factor in the decision whether to follow this principle is the efficiency of the
deflation. If the performance of the deflater increases by a substantial amount using the
6. Evaluation

syntax elements creating parsing dependencies, it might be worth ignoring the principle. But this is, as already mentioned, depending on the application and environmental parameters.
7. Conclusion

7.1. General Results

The goal of this thesis was to analyse the \( \Delta \)-coefficients created by the deflation scheme presented in chapter 3. Afterwards, this knowledge was applied to improve the encoding efficiency compared to the HEVC encoding scheme, which was used as the base. By combining several new syntax elements and modifying the use of existing ones, it was possible to improve the performance of the encoder by over two percentage points. A big part of enhancing the performance was using the knowledge gained from the estimated coefficients, which can be used due to correlations between the magnitude and positioning of the coefficients.

Introducing the bottom-up approach as described in section 3.3, showed that the \( \Delta \)-coefficients in this case have a layout which is an interpolation between the \( \Delta \)-coefficients in the top-down approach and the original coefficients. This occurs due to the less accurate estimation. However, even in this approach the suggested new and changes to the existing syntax elements provide a significant improvement over the basic deflation scheme.

Using the estimated coefficients and their distribution to improve the encoding efficiency of the \( \Delta \)-coefficients can create a parsing dependency. To provide a better possibility for parallel processing the principle of independent parsing demands that the coefficients can be read from the bit stream without knowledge of specific values of the estimated coefficients. Several of the newly implemented violate this principle, but on the other hand also have a quite large and positive impact on the coefficient encoding. The decision which elements to use and in what configuration depends on the application and the hardware preconditions.

7.2. Assessment of Tasks

While there were a lot of different tasks ranging from adjusting the python scripts to adding new syntax elements to the D65 necessary to complete the required work for this thesis, the general tasks stayed roughly the same since the specification of the thesis.

The first step was to analyse the way HEVC encodes the transform coefficients. This is described in section 2.2.6. The analysis of the \( \Delta \)-coefficients in regard to both magnitude
7. Conclusion

and distribution can be found in chapter 4. Afterwards new syntax elements were intro-
duced to improve the efficiency of the encoding. While the usage of CABAC contexts was
only evaluated for a subset of the syntax elements, it proved that varying the contexts
has quite some impact on the performance of the encoder. Due to this the significance
map could be encoded in a much more efficient way.

The bottom-up approach was implemented up to a certain degree. While using an
upscaler was deemed out of scope as described earlier, using a lower quality to predict a
higher one proved to bring quite some savings.

Summarising it can be said that all tasks were completed to a satisfying degree. While
some smaller parts were left out, the vast majority was executed as planned. However,
in terms in researching the $\Delta$-coefficients, this is not the end of the line and there is
much research left to do. A suggestion what the next steps could look like is described
in section 7.3.

7.3. Future Work

There are several ways the current encoding can be improved. All these ideas will have
to be verified and evaluated for potential gains.

The first step should be to continue with the syntax elements introduced in this thesis.
A suggestion for the further development of deflation is to improve the initial values for
the CABAC contexts of the used flags. While this is a relatively straightforward step, it
should bring a minor improvement. However, the effect should not be overestimated as,
due to the large number of transformations, initial values play a rather small role.

A different way to improve the newly introduced syntax elements is to adjust the decision
when to use which context. In the current scheme context switches are mostly based
on the transformation size, but there might be better and more efficient methods. A
thorough analysis should be conducted before this step can be taken.

More improvements are possible by adding more syntax elements or changing the inter-
pretation of current ones. Another improvement can be based on using the last significant
coefficient of the estimated coefficients to start encoding the significance map of the $\Delta$-
coefficients from this position if the position is nearby or equal. This way it would be
unnecessary to encode the last significant position for the $\Delta$-coefficients at the cost of a
few bits to encode a slightly larger significance map. An advantage would be that this
variant does not create a parsing dependency and therefore can be used to improve the
encoding in all cases. However, as there is quite a risk of increasing the bit rate due to
bad cases, a reasonable set of parameters when to use this would have to be implemented.
One option could be to not use this method for 4x4 transformations, as the position in
this case can be encoded quite cheaply.

A further variant would be to base the scan order on the layout of the estimated coef-
7. Conclusion

coefficients, as it is more likely that significant \( \Delta \)-coefficients are in positions where there are also significant estimated coefficients. For this approach a decision regarding the principle of independent parsing has to be made. When following the principle of independent parsing, a different solution for encoding the last significant position is required, as having only the coordinates but not knowing the scan order does not provide enough information to parse the significance map. If the principle is ignored, the scan order is known and therefore the significance map can be parsed without a problem. As this is quite a difference from the original HEVC encoding, the effects are hard to estimate.

Another part of the development which was deemed out of scope for this project was, as described in section 3.3, using a upscaler to analyse whether using the prediction from a low quality can provide savings for high quality videos. A follow-up evaluation would certainly provide interesting results.
## A. Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVC</td>
<td>Advanced Video Coding</td>
</tr>
<tr>
<td>CABAC</td>
<td>Context-based Adaptive Binary Arithmetic Coding</td>
</tr>
<tr>
<td>CBF</td>
<td>Coded Block Flag</td>
</tr>
<tr>
<td>CDF</td>
<td>Coded Delta Flag</td>
</tr>
<tr>
<td>CSBF</td>
<td>Coded Sub-Block Flag</td>
</tr>
<tr>
<td>CG</td>
<td>Coding Group</td>
</tr>
<tr>
<td>dB</td>
<td>decibel</td>
</tr>
<tr>
<td>DCT</td>
<td>Discrete Cosine Transformation</td>
</tr>
<tr>
<td>DST</td>
<td>Discrete Sine Transformation</td>
</tr>
<tr>
<td>HEVC</td>
<td>High Efficiency Video Coding</td>
</tr>
<tr>
<td>HM</td>
<td>HEVC Test Model</td>
</tr>
<tr>
<td>JCT-VC</td>
<td>Joint Collaborative Team on Video Coding</td>
</tr>
<tr>
<td>MPEG</td>
<td>Motion Picture Expert Group</td>
</tr>
<tr>
<td>NAL</td>
<td>Network Abstraction Layer</td>
</tr>
<tr>
<td>PSNR</td>
<td>Peak Signal-to-Noise Ratio</td>
</tr>
<tr>
<td>QP</td>
<td>Quantisation Parameter</td>
</tr>
<tr>
<td>SHVC</td>
<td>HEVC Scalability Extension</td>
</tr>
<tr>
<td>SI</td>
<td>Side Information</td>
</tr>
<tr>
<td>TB</td>
<td>Transform Block</td>
</tr>
<tr>
<td>VCEG</td>
<td>Video Coding Experts Group</td>
</tr>
<tr>
<td>VoD</td>
<td>Video on Demand</td>
</tr>
</tbody>
</table>
Bibliography


Bibliography


Declaration of Originality

Hereby I assure that this piece of work was created by me independently. I did not use any sources and means but the ones listed and all excerpts which have been taken from other works either literally or analogously are marked as such. This thesis was neither in the same nor a similar version part of a previous academic assignment.

December 2016, Uppsala
Date & Place

Christopher Höllmann