DEGREE PROJECT

Video compression optimized for racing drones

Henrik Theolin

Computer Science and Engineering, master's level
2018

Luleå University of Technology
Department of Computer Science, Electrical and Space Engineering

LULEÅ
UNIVERSITY
OF TECHNOLOGY

Preface

To my wife and son always! Without you I'd never try to become smarter.

Thanks to my supervisor Staffan Johansson at Neava for providing room, tools and the guidance needed to perform this thesis.

To my examiner Rickard Nilsson for helping me focus on the task and reminding me of the time-limit to complete the report.

Abstract

This thesis is a report on the findings of different video coding techniques and their suitability for a low powered lightweight system mounted on a racing drone. Low latency, high consistency and a robust video stream is of the utmost importance. The literature consists of multiple comparisons and reports on the efficiency for the most commonly used video compression algorithms. These reports and findings are mainly not used on a low latency system but are testing in a laboratory environment with settings unusable for a real-time system. The literature that deals with low latency video streaming and network instability shows that only a limited set of each compression algorithms are available to ensure low complexity and no added delay to the coding process. The findings resulted in that AVC/H.264 was the most suited compression algorithm and more precise the x264 implementation was the most optimized to be able to perform well on the low powered system. To reduce delay each frame needs to be divided into sub-frames so that encoding and decoding may be done in parallel independently of other sub-parts of the frame. This also improves error propagation when used together with an All-Intra (AI) mode that doesn't utilize any motion prediction techniques.

Contents

Acronyms

AC Arithmetic Coding. 12, 17, 19, 21, 22, 25, *Glossary:* arithmetic coding

ADST Asymmetric Discrete Sine Transform. 20, 22, 25, 26, *Glossary:* asymmetric discrete sine transform

AI All-Intra. ii, 33, 34, 42, 51, 52, *Glossary:* all-intra

AMP Asymmetric Motion Partitions. 34, *Glossary:* asymmetric motion partition

AMVP Advanced Motion Vector Prediction. 19, *Glossary:* Advanced Motion Vector Prediction

ASIC Application-specific integrated circuit. *Glossary:* application-specific integrated circuit

BEC Boolean Entropy Coding. 15, 18, 21, 24, 26, *Glossary:* boolean entropy coding

CABAC Context-Adaptive Binary Arithmetic Codes. 16, 19, 23–26, *Glossary:* Context-Adaptive Binary Arithmetic Codes

CAVLC Context-Adaptive Variable-Length Codes. 16, 24, 26, *Glossary:* Context-Adaptive Variable-Length Codes

CB Coding Block. 19, *Glossary:* coding block

CBR Constant Bitrate. 7, 33, 45, *Glossary:* constant bitrate

CDEF Constrained Directional Enhancement Filter. 22, 25, *Glossary:* constrained directional enhancement filter

CTB Coding Tree Block. 7, *Glossary:* coding tree block

CTU Coding Tree Unit. 4, 19, 20, 25, *Glossary:* coding tree unit

CU Coding Unit. 19, *Glossary:* coding unit

DCT Discrete Cosine Transform. 5, 7, 14–16, 18–22, 24, 25, *Glossary:* discrete cosine transform

DPCM Differential Pulse Code Modulation. 15, *Glossary:* differential pulse code modulation

DWT Discrete Wavelet Transform. 17, *Glossary:* Discrete Wavelet Transform

EC Entropy Coding. 14–17, 19, 26, *Glossary:* entropy coding

EED End-To-End Distortion. 28, *Glossary:* end-to-end distortion

FEC Forward Error Correction. 28, *Glossary:* forward error correction

FHD Full High-Definition. 11, 23, 27, 32, 33, 37, 39, 46, 54, *Glossary:* full high-definition

FM Frequency Modulation. 8, *Glossary:* frequency modulation

FPGA Field-Programmable Gate Array. 29, *Glossary:* field-programmable gate array

FPMC Fractional Pixel Motion Compensation. 12, 15, *Glossary:* fractional pixel motion compensation

FPV First Person View. 8, 9, 37, *Glossary:* first person view

GPGPU General Purpose Graphical Processing Unit. 29, *Glossary:* general purpose graphical processing unit

HC Huffman Coding. 12, 14, 15, 24, 26, *Glossary:* Huffman coding

HD High-Definition. 42, 46, 52, *Glossary:* high-definition

HT Hadamard Transform. 16, 24, *Glossary:* Hadamard transform

IDCT Inverse Discrete Cosine Transform. 14, *Glossary:* Inverse Discrete Cosine Transformation

JVT Joint Video Team. 53

MB Macroblock. 4, 14–18, 28, *Glossary:* macroblock

MV Motion Vector. 3, 12, 15, 18, 19, 21–23, 25, *Glossary:* motion vector

NAL Network Abstraction Layer. 29, *Glossary:* network abstraction layer

OBMC Overlapped Block-based Motion Compensation. 17, *Glossary:* Overlapped Block-Based Motion Compensation

PB Prediction Block. 16, 19, *Glossary:* prediction block

PSNR Peak Signal-to-Noise Ratio. 6, 28, 31, 32, 36, *Glossary:* peak signal-to-noise ratio

RA Random Access. 33, 34, *Glossary:* random access

RLE Run-Length Encoding. 12, 14, *Glossary:* run-length encoding

SAO Sample Adaptive Offset. 20, 25, *Glossary:* sample adaptive offset

SB Superblock. 17, 20–22, 25, *Glossary:* superblock

SSIM Structural SIMilarity. 31, *Glossary:* structural similarity

SVC Scalable Video Coding. 24, 27, *Glossary:* scalable video coding

TB Transform Block. 14, 19, 22, *Glossary:* transform block

UEP Unequal Error Protection. 28, *Glossary:* unequal error protection

UHD Ultra High-Definition. 28, 32, 33, *Glossary:* ultra high-definition

VBR Variable Bitrate. 33, *Glossary:* variable bitrate

VMAF Video Multi-method Assessment Fusion. 31, 36, 42, 43, *Glossary:* video multi-method assessment fusion

WHT Walsh-Hadamard Transform. 18, 24, *Glossary:* Walsh-Hadamard transform

Y-PSNR Luma Peak Signal-to-Noise Ratio. 31, *Glossary:* luma peak signal-to-noise ratio

Glossary

Advanced Motion Vector Prediction Utilizes a vector competition where a list of candidate Motion Vectors (MVs) that are derived from neighboring blocks and from blocks of temporal frames. 1, 19

All-Intra Only uses i-frames prediction and no motion predictions for the decoder to decode each frame independent of previous frames. ii, 1

Application-Specific Integrated Circuit An integrated circuit designed to perform a specific task. 1

Arithmetic Coding Symbol compression technique that encodes an entire message of symbols into a fractional number between 0.0 and 1.0. Highly efficient with small alphabets. 1, 12

Asymmetric Discrete Sine Transform Transforms a signal into frequency domain using real valued coefficients of sine functions. 1, 21

Asymmetric Motion Partition Allows for different sized and non-square shapes when using motion prediction. useful for irregular shaped objects where a square or symmetric shape cannot be used for an accurate representation. 1, 34

B-Frame Bidirectional predicted picture, a frame that can use data from previous and forward frames to decode. 6, 12, 14, 16, 17, 29, 34, 45

Boolean Entropy Coding A kind of arithmetic coding that compresses a sequence of boolean values that has well estimated probabilities of being a certain value. 15

Chroma The term for two color difference signals. 4, 7, 11, 15, 17–19, 25

Codec A device, algorithm or technique that performs video encoding and decoding. 4, 10, 14–17, 42, 43, 45, 46, 52, 53

Coding Block Part of a subdivided coding tree block. 1, 19

Coding Tree Block Luma and chroma sample blocks that serves as roots of a block partitioning quadtree structure. 1, 4

Coding Tree Unit The basic processing unit in HEVC codec, similarly to Macroblock (MB) was for earlier codecs. 1, 19

Coding Unit A subblock from a partitioned Coding Tree Unit (CTU) that can be of variable or same size. 1, 19

Constant Bitrate A coder tries to maintain the bitrate at a constant value for an average window size by altering quality of the video stream to match the specific rate. 1, 33

Constrained Directional Enhancement Filter A non-linear low-pass filter that uses the direction of edges and patterns for filtering with a high degree of control over filtering strength. Reduces the ringing artifacts and designed to be easily vectorizable [1]. 1, 22

Context-Adaptive Variable-Length Codes Encodes each symbol into variable bit lengths with different probabilities where the probabilities can be varied for better coding efficiency. 1, 16

Differential Pulse Code Modulation Exploits that the difference between neighboring pixels is small this technique stores the difference value between a pixel and it's most likely prediction. 1, 15

Digital Artifact A visual anomaly in the appearance of an image caused by for example compression techniques. 10

Discrete Cosine Transform Transforms a signal into frequency domain using real valued coefficients of cosine functions. 1, 14

Discrete Wavelet Transform Samples the wavelets of a sequence so both frequency and time information is captured and represented as a sequence of coefficients on an orthogonal basis. 1, 17

End-To-End Distortion The distortion of a signal that is experienced at the decoder. 2, 28

Entropy Coding Technique to encode symbols with variable length unique codes to reduce the amount of bits needed to be sent though a channel. 2, 14

Field-Programmable Gate Array A configurable integrated circuit that consists of an array of programmable logic blocks that can be made to solve and problem which is computable. 2, 29

First Person View Relates to the effect that though a display the user is shown the image from the "eyes" of the object. For a drone the camera image in front of the body is displayed in such a way that the user is placed in the cockpit. 2, 8

Forward Error Correction A technique for reducing errors in an unreliable communication channel by adding redundancy to each message sent. 2, 28

Fractional Pixel Motion Compensation Gives better compression ratio where the motion vectors are interpolated from a subsample with varied precisions of for example half, quarter and one-eight where a smaller fraction is better in terms of coding efficiency. 5, 6, 15

Frequency Modulation A transmission technique that alters the frequency with of a signal to relay information. 2, 8

Full High-Definition A resolution of 1920 x 1080 pixels. 2, 11

General Purpose Graphical Processing Unit By making use of Graphical Processing Units that are high-performance many-core processors capable of very high computation and data throughput to perform computation for a specific application often resulting in a speedup in performance compared to optimized CPU implementations . 2, 29

Golden Frame Special reference frame that consists of a buffer with the last decoded i-frame. 14, 15, 18, 20

Hadamard Transform Similarly Discrete Cosine Transform (DCT) transforms into frequency domain, using 4x4 matrices, slight less efficient in decorrelating signal but does so with lower complexity. 2, 16

Half-Pel Is fractional pixel motion compensation where the motion vectors has half the pixel precision compared to the reference frame. 16, 23

Huffman Coding Depending on probabilities for a symbols appearance in a bitstream the symbol is coded such that the highest probability symbol has the shortest code for representation. 2, 12

I-Frame Intra-coded picture, a reference frame that doesn't require any other frames to decode. 3, 5, 6, 14, 18, 20, 22, 23, 25, 28, 33, 43, 53

Inverse Discrete Cosine Transformation Transforms a signal from frequency domain back to the original sample sequence. 2, 14

JPEG Image compression format that uses block transforms and a quantization step. Lossless mode is available without the quantization step and uses a prediction method for encoding pixels. 14

Luma The image brightness or black and white part of an image. 4, 6, 7, 11, 15, 18, 19, 25

Luma Peak Signal-to-Noise Ratio The Peak Signal-to-Noise Ratio (PSNR) is computed only for the luma component of the signal. Typical values ranges between $30-50$ dB for an 8-bits bit depth where higher value most often represents higher quality. 3, 31

Macroblock Portions of a frame divided into smaller blocks of pixel samples. 2, 6, 14

Motion Vector A vector defined to describe the distance between two blocks on the current predicted frame relative to a reference frame. 2, 12

Network Abstraction Layer A packet of bytes used to provide a network friendly transport system. 2, 29

Overlapped Block-Based Motion Compensation A technique that places blocks of pixels in a way that they overlap each other. Used for motion estimation where pixels that lay in the overlapped area has multiple motion vectors associated to it and are combined using weights. 2, 17

P-Frame Predicted picture, a reference frame that uses data from previous frames in order to decode. 6, 12, 14–18, 20–23, 33, 43, 53

Peak Signal-to-Noise Ratio A measure of quality typically used for lossy compression techniques to approximate the human perception of an image. 2, 28

Perceived Quality A measure of video quality for the user watching the content. 11

Prediction Block Subblocks partitioned from the macroblocks used for motion estimation and compensation plural. 2, 16

Quantization A technique for containing a set of continuous numbers into a range of discrete values. The level of quantization determined how big range of values to be stored, using different levels for different frequencies improves compression without sacrificing quality. This is a lossy process because the information removed when rounding numbers to fit the chosen range cannot be retrieved exactly. 5, 12, 14–17, 20

Quarter-Pel Is fractional pixel motion compensation where the motion vectors has quarter pixel precision compared to the reference frame. 15, 16, 18, 19, 21–23

Random Access The first picture of a sequence is coded as a i-frame and the remaining are coded as p-frames and b-frames that together forms a group of pictures where the decoder may start decoding from the i-frame without using previous frames as references. 2, 33

RGB Red, green and blue color coding that allows for the colors to be added together in order to display an array of colors. 11, 53

Run-Length Encoding Algorithm for compressing a binary sequence. Instead of coding each binary symbol the number of consecutive 1s or 0s are coded as a pair giving efficient coding for sequences where a lot of symbols are paired together. 2, 12

Sample Adaptive Offset Filters pixels by performing a non-linear amplitude mapping on each pixel in the Coding Tree Block (CTB). 2, 20

Structural Similarity An index to measure the similarities between two images. 2, 31

Superblock Consists of an array of 4x4 pixel blocks. 2, 17

Transform Block Subblocks partitioned from a larger block used for decorrelating pixels using a transform function plural. 3, 14

Ultra High-Definition 4k resolution or 3840 x 2160 pixels. 3, 28

Unequal Error Protection An error protection method that varies the redundant data sent with each data packet based on how important that package is. 3, 27, 28

Variable Bitrate A coder varies the bitrate to achieve higher quality and compression ratio for a sequence. More complex than Constant Bitrate (CBR) and due to the bitrate fluctuation caused by the difference in highly vs. low detailed images this setting is not well suited for real-time applications. 3, 33

Video Multi-method Assessment Fusion A video quality metric that combines multiple elementary quality metrics to make use of the specific advantages that the other metric has in certain situations. 3, 31

Walsh-Hadamard Transform Similarly DCT transforms into frequency domain, using 4x4 matrices, slight less efficient in decorrelating signal but does so with lower complexity. 3, 18

YUV Y is the luma U and V are color difference signals, chroma. U and V only apply in analog video, the digital counterpart is C_B and C_R although the terms U and V are commonly used for naming in digital video as well. 11, 14, 17, 53

1 Introduction

There is an increasing market with racing drones that are controlled by a pilot on the ground, these drones are usually equipped with a camera and a video transmitter. The pilot can receive this transmission and display it using special First Person View (FPV) goggles to achieve an immersive experience. The quality of the video is far from good and is using old technology which needs improvement. The system currently used by FPV pilots consists of a low resolution camera that outputs analog PAL signal with 720 x 576 pixels resolution with 25 Frames Per Second (FPS) or analog NTSC signal with 720 x 480 pixels resolution at 30 FPS. To transmit this video a Frequency Modulation (FM) transmitter is used that takes the PAL or NTSC signal as an input. Using this technology the latency is almost nonexistent and the size of the components are very small, the transmitter can weigh below 10 grams, with a camera a total weight of 20 grams is not unusual. Receiving the signal on the ground is achieved with goggles that have a built-in FM receiver and displays the video with low-resolution screens. Another good attribute is the gradually fading image quality when flying far away or behind obstacles, this lets the pilot know when to start flying closer without crashing due to lost video. The terminology for describing data compression and computational complexity differs in existing literature, in this thesis coding efficiency corresponds to the compression ratio achieved by the coding technique while coding performance corresponds to the time spent on the coding process.

1.1 Problem Statement

While the current low-resolution analog video transmission system that exists performs well enough for pilots the limits lay in not only video quality but also in the number of pilots that may fly and stream video simultaneously. By using the 5GHz frequency band which ranges from $5.650 - 5.925$GHz and using a FM transmission results in up to 8 pilots transmitting simultaneously without interfering each others video stream. The transmission is also very susceptible to interfering noise that drastically reduces the range and transmission bandwidth. Many advantages are gained by using a more advanced transmission system and higher quality and resolution video streams. A higher quality video stream set high requirements on the bandwidth available thus the video compression technique plays a central role in achieving high-quality low bandwidth video stream that maximizes the utilization of the transmission channel. There exists a large variety of different compression techniques and choosing an optimal tool for this specific use-case is a task that needs considerable research and analysis. The trade-off between video quality and compression efficiency need to be thoroughly investigated and error concealment or correction methods for increasing the perceived quality in high packet loss situations should be considered. To understand the latency chain of a system from the capturing of an event to the user receives the information each part need to be investigated to understand fully how to improve the delay.

1.2 Delimitations

This project will research existing coding algorithms, their compression efficiency and computational complexity. New methods for compressing data will therefore not be researched in depth and only the most commonly used coders will be considered when the research is conducted.

New hardware designs to improve the performance of a coding process will not be performed, lightweight existing board computers will be used such as a Raspberry Pi[2] running a Linux distribution. The GStreamer multimedia framework[3] will be used to evaluate coders using software implementations of the coder algorithms. FFmpeg[42] coding software will be used for evaluating coding performance and quality.

The transmission technique will only be evaluated after the video compression research is finished to a satisfactory degree. The existing WiFi technology of a board computer will be used initially when conducting tests on the platform.

The latency of the system consists of many parts from camera to display the main focus will consist in evaluating the video coders contribution to the delay added.

1.3 Expected Contributions

By the end of the thesis, it's expected that readers may gain knowledge in the area of video coding techniques and with the specific application of zero-latency video streaming gain an understanding of the existing video coding algorithms, their pros and cons for using on a fast-moving FPV drone. The goal is to vastly simplify choosing what parameters to use for a specific coder and know what performance may be expected on a low-cost platform with detailed tests that shows the coding performance and video quality.

1.4 Terminology

There exists some inconsistency in the terminology used when describing different video codecs and their capabilities. In some literature, coding performance is used to describe the compression made on the video sequence in relation to the original file while in others coding performance is describing the time taken to perform the coding process. Similarly, coding efficiency is used differently. For this thesis all references to coding performance in relating to the computational time taken to perform a coding process while coding efficiency is related to the compression achieved relative to the original video file.

2 Background

As the quality of video keeps improving at a fast rate the viewers become more aware of how a good quality video should look like lower resolution video might not be good enough to meet up to the higher standards of the modern viewer. This is a conundrum to be solved where the industry that keeps improving also need to continue moving forward to satisfy the demands of an ever more selective customer. Higher resolution and better quality video increase the amount of data that needs to be transferred over different networks and therefore good compression techniques needs to be applied to limit the bandwidth footprint left behind in the wake of video streaming. Using more advanced video compression techniques always comes at a price in terms of complexity and adding a delay of the coding process. The usage of a low-resolution camera and sending the video stream using an FM transmitter is severely outdated and produce far from an immersive experience when flying a racing drone. This could be improved by using a high-resolution camera and compressing the video stream in such a way that wireless transfer is possible. The difficulty lay in producing this compressed video stream without adding a long delay or produce digital artifacts in the image so that the flight experience is degraded. Video compression is a computing intensive technique and more compressed video stream will most likely result in a higher complexity algorithm and set high requirements on the hardware used to perform this task. For a system that is to be placed on a racing drone weight is of utmost importance where each gram added to the machine will cause more power to be drawn by the electrical motors. Aside from weight, there is also a power requirement added to the equation where a power-hungry computer is less ideal for the task. There is a lot to consider and the sweet spot between video quality versus compression rate needs to be researched. A typical video codec uses many different steps in achieving high compression ratio and each step comes with different complexity thus the algorithms for a few standard video codecs will be researched and evaluated by studying existing literature and performing tests on a system developed to mimic how a real implementation could be done.

3 Video compression

To enable a video stream at a high bitrate to be broadcast through a wireless channel a compression technique is most often required to reduce the number of bits sent and received.

3.1 Coding techniques

Many of the existing coding techniques has been excellently described by the author in [4] and the key features have been summarized in the following sections.

By identifying the human visual capabilities and smart compression techniques redundancy in images can be reduced or removed to allow for fever bits to be used when recording and playing a video. The fact that for an image a lot of redundant data where similar and correlated information between neighboring pixels and frames in a sequence can be reduced or removed in order to achieve high compression rates.

3.1.1 Resolution and Bitrate

An image consists of pixels where each of these pixels is color-coded in RGB format. Each of the three colors can consist of eight bits so for an image with a common Full High-Definition (FHD) resolution and displaying 30 FPS the amount of data is in the GB range for a one second of video. Bitrate is the number of bits sent for a unit of time in video processing this often relates to bits per second in the compressed bitstream.

3.1.2 Luma and Chroma Down-Sampling

The human visual system is more prone to recognize the difference in brightness than color. This fact has been taken advantage of in video compression. It is possible to exploit high correlation in color information to reduce the bitrate without the perceived quality is any worse. For most coding standards the first step in video compression is to subsample the chroma components, which is the color information of an image. The images are captured in RGB space then converted into YUV color space. The ratio between luma and chroma components are describe as 4:4:4, for each 4x2 sample region there are four 4x2 luma samples, 4 U components and 4 V components. By down-sampling the ratio to 4:2:0 as most standards, for each 4x2 sample region there are 2 U components and 0 V components, this is determined to provide sufficient color resolution based on the perceptual quality of the image. A 4:2:0 color scheme results in half the bits used over YUV 4:4:4 or RGB.

3.1.3 Spatial Redundancy

For a still image, neighboring pixels will have similar or same color and brightness information in many cases. When a frame is divided into chroma and luma

components it's clear that the correlation between pixels is high in horizontal and vertical spatial dimension. This creates spatial redundancy and a frame can be divided into smaller blocks to take advantage of this similarity between pixels. Most energy is often concentrated in low-frequency regions in the frequency domain of a frame due to the rate of change in the spatial dimension. Transforming the signal into the frequency domain is, therefore a necessary step to take advantage of this information. The transform itself doesn't provide any compression of data but primes the sequence for the quantization process that usually can be tuned with different levels of data compression which is explained thoroughly in [5]. A coarser quantization will provide better compression ratio at the cost of image quality loss. It's also possible to predict what the next block should contain given already decoded blocks to further reduce the amount of information to be sent.

3.1.4 Temporal Redundancy

For a sequence of frames that are captured at 30 FPS, there will often be very little differences between each frame. The idea is to take advantage of this and represent the next image as the difference between a reference frame and the current frame and only send the necessary bits to reconstruct the frame without losing any information. Often a block-based motion estimation is performed and the sizes of the blocks matter to the performance of the coding efficiency wheres a uniform area would perform better using larger block sizes and the opposite for a highly varied area.

3.1.5 Motion Prediction and Compensation

Further minimizing the amount of information is often done by implementing motion prediction to blocks of pixels in the frame. By predicting where the blocks will be in the frame by sending MV that are subsampled from a residual frame, Fractional Pixel Motion Compensation (FPMC). The vectors define a search window for the reference frame and the best match is determined. Three different types of frames are commonly used for prediction, p-frames, i-frames and b-frames. These frames help in achieving higher compression ratio but b-frames adds a delay due to the fact that it needs forward frames to decode.

3.1.6 Statistical Redundancy

Determining that there is a statistical difference in the probabilities that a certain symbol is to be sent a technique called Run-Length Encoding (RLE) can be used and is especially useful to implement with the quantized coefficients after disregarding the high-frequency information. Entropy coding is commonly used to further reduce the statistical redundancy using RLE. Different algorithms such as Huffman Coding (HC)[6] and Arithmetic Coding (AC)[7] for certain symbol sequences. The better encoding may be achieved when the probability distribution is unknown, not independent and not identically distributed. When

the probability of certain occurrence of an event is larger than one half, AC can offer much better coding efficiency although the algorithm is more computational complex then HC coding. The probability of a symbol and event may not be static but may be updated throughout the coding process for more efficient coding.

3.1.7 Software and Hardware Coding

Software-based encoding offers a flexible way of choosing different parameters and bitrates for encoders. It's easy to implement and a large variety of tools are available for development. The biggest drawback is that the performance worse than the hardware counterparts due to the computational complexity of video encoding. Hardware coding can achieve much faster processing but are usually limited to a specific set of parameters that are possible to modify.

3.1.8 Summary

This section different techniques used for compressing a video into a bitstream has been discussed. From the fact that the human eye is less prone to notice color differences in images to similarities between a sequence of images. Many techniques work together to achieve a higher compression ratio without reducing the visually perceived quality. Spatial redundancy takes advantage of the similarities between pixels in a single frame while temporal redundancy uses multiple frames and predicts motion vectors so that the next frame may be created without needing to send all pixel information.

3.2 Different Video Coding Standards

There are a lot of different video coders developed for specific purposes. A few standards have been set by the industry to simplify usage and provide a uniform usage of video compression. This thesis will focus on the later generation of codecs with the exception of MJPEG.

3.2.1 MJPEG

Though the information about MJPEG is somewhat limited and isn't specified in any international standard the authors of [4] and [8] touches the technology behind this video coding technique.

MJPEG uses i-frame coding only, each frame is compressed independently of each other. Each frame is encoded as a JPEG image and sent as a sequence to the decoder.

Spatial Redundancy is reduced with a DCT based coding is used on $8x8$ pixel blocks called MBs where a two-dimensional forward DCT is applied on each block. The decorrelated signal is then quantized by a uniform quantization process.

Temporal Redundancy isn't reduced in the MJPEG codec, this makes this compression technique less sensitive to fast random movements at a reduced coding efficiency.

Statistical Redundancy is reduced using Entropy Coding (EC). The high-frequency coefficients from quantization are moved to the end of the sequence to improve the efficiency of RLE and a variable-length HC is used do encode the AC-coefficients.

3.2.2 VP6

The VP6 codec was released in 2003 by On2 Technologies, later acquired by Google. It's uses i-frames, p-frames and introduces another frame called golden frame for compressing the video. Note that there are no b-frames so now forward prediction is used. DCT based coding and is a predecessor for later codecs such as VP8, VP9 and AV-1 that are described later in this section. Documentation in form of the Bitstream & Decoder Specification[10] exists contrary the MJPEG codec and is summarized by the authors in [9]. The VP6 codec supports the YUV 4:2:0 format.

Spatial Redundancy is reduced using MB based coding on i-frames, using $8x8$ pixel Transform Blocks (TBs) for forward transform using DCT for decorrelation of the signal. The Inverse Discrete Cosine Transform (IDCT) is slightly modified from a standard version to reduce complexity in decoding. By omitting

small-margin coefficients and grouping zero coefficients the decoding complexity is reduced.

Temporal Redundancy uses p-frames or from the golden frame as references for the MVs. The MV can be calculated from a predicted frame or a differential between nearest blocks MVs can be used. Two filter alternatives can be applied for FPMC with quarter-pel precision. Bilinear filtering using a 2-tap filter is used with quarter-pel luma sample and ⅛ chroma sample precision. The result of a first pass filtering can be used as input to a second pass if the result holds fractional values to produce a 2-D output. Bicubic filtering using a 4-tap filter and is required if fractional pixel values are required.

Statistical Redundancy is reduced by coding the DCT coefficients and can be done in three levels. Either by predictions of the DC coefficients, coding the AC coefficients or coding zero-runs of the DC and AC coefficients. Entropy Coding is done with two different algorithms, the HC and Boolean Entropy Coding (BEC). While HC is more computational efficient the BEC has higher compression efficiency. A conditional probability distribution is used with respect to a defined context where baseline probabilities are weighted by information from the already decoded data.

Quantization is performed on the DCT coefficients by two separate scalar quantizers, one for the DC coefficient and the other for the 63 AC coefficients.

Filtering is used for reducing blocking artifacts and is done by a prediction loop filter. The prediction block boundaries are filtered before the FPMC. The output of the filtering is stored in a separate buffer and used on the block edges if a motion vector crosses a block boundary.

3.2.3 AVC/H.264

A commonly used video compression codec first released 2003 [12]. By far the most commonly used codec and it's shown in the literature where a vast majority describes the techniques behind this codec such as in [4], [9], [11] and [13]. Improvements of the coding standard is being made continuously and new profiles are implemented to improve and add features. Many profiles are described and they determine what tools the specific codec can use. The coding algorithm uses a lossy-predictive block-based hybrid Differential Pulse Code Modulation (DPCM), motion-compensated prediction, quantization and EC. A wide range of picture formats is supported ranging from low to high resolution, chroma subsampling and color bit-depth.

Spatial Redundancy is reduced by variable block based intra-prediction, where a prediction block is constructed based on previously decoded MBs. A lossy technique where information is lost in the process of coding the residual

signal between the current and predicted block is used to reduce the bit's needed for representing the video. The intra- Prediction Block (PB) may be of sizes 16x16, 8x8 and 4x4 pixels. Small block-sizes of 4x4 or 8x8 pixels are transformed using a modified integer transform based on DCT and the DC components of neighboring blocks are then grouped together into a new 4x4 block that uses a Hadamard Transform (HT) for further decorrelation. Quantization is performed and there are 52 different levels for different quality preferences.

Temporal Redundancy is reduced by motion estimation and compensation on partitioned blocks from the MBs. AVC/H.264 can use both p-frames and b-frames for reference to the prediction calculation. By down-sampling, the reference frame the motion vector accuracy is half-pel for a one-step filtering and quarter-pel using two-step filtering. Reaching a higher compression ratio the process for two-step filtering has higher complexity since the second step filtering requires the result from the first filtering step. Predicting the motion vectors in done from a list of previous frames where more frames will produce a higher memory footprint on the decoder and receive gain in estimation accuracy that results in better compression efficiency.

Statistical Redundancy is reduced by EC. Different variable-length codes are used based on context characteristics. Context-Adaptive Variable-Length Codes (CAVLC) or a more compression effective Context-Adaptive Binary Arithmetic Codes (CABAC) can be used by increasing complexity with a higher coding efficiency of typically $9 - 14\%$ compared to CAVLC as described in [14].

Filtering is used by an in-loop deblocking filter to reduce the artifacts created by all block-based operations used in the coding process. The filtering is complex but a much closer prediction can be obtained with higher coding efficiency.

Error Resilience is also provided in certain profiles of AVC/H.264. Different modes are defined and can be used. A way to order the MBs to make them less sensitive to packet loss. By using a technique for separating the syntax elements unequal error protection is enabled. To compensate for a lost or corrupted slice, part of the frame, a lower fidelity part may be resent increasing redundancy.

Parallel Processing uses a slice-based threading method that divides a frame into slices that may be encoded and decoded independently, This is especially useful for low latency applications, each slice can be encoded and decoded as soon as it reaches the coder without needing to wait for the entire frame adding a latency of a fraction of a frame instead of at least one frame.

3.2.4 DIRAC

An open source and royalty free video codec developed by the BBC 2008 that is specified in [15] and described in [9] and [7]. Designed to be a simple and flexible

competitor to AVC/H.264 and uses a less common Discrete Wavelet Transform (DWT) for decorrelating the signal and motion compensation. The DIRAC codec supports chroma subsampling for YUV 4:4:4, 4:2:2 and 4:2:0 format and 8, 10, 12 and 16 bit formats.

Spatial Redundancy is reduced by decorrelating the signal using 2D DWT on an entire frame at once. This allows for lower resolution data to be extracted at the decoder using low complexity. Fine details are better preserved compared to block-based transformation schemes. Vertical and horizontal components are divided into high and low frequencies by repeated filtering. With still images the Wavelet transform is more efficient then it's block-based counterparts. There are different types of wavelet filters available that are supported and there is a trade-off between complexity and quality.

Temporal Redundancy is reduced by motion estimation and motion compensation. Motion estimation uses both p-frames, and b-frames where each frame can be predicted from at most two reference frames. DIRAC uses a hierarchical approach for creating motion vectors where current and reference frames are downsampled in steps using a 12-taps down conversion filter. A picture is divided into Superblocks (SBs) and predictions may be calculated for each subblock. Overlapped Block-based Motion Compensation (OBMC) is used for motion compensation to avoid blocking artifacts. The data is padded such that there exists an exact number of MBs both vertically and horizontally. Motion prediction is done at ⅛ precision by allowing sub-pixel motion compensation although the precision is determined by the chosen bit-rate.

Statistical Redundancy is reduced by EC that is applied in three steps. Binarization, context modeling and AC. Binarization is made to provide a bitstream that can be used more efficiently by the following AC. Context modeling predicts whether a coefficient is small by looking at its neighbors and parents. AC is then performed on the statistical model to compress further into the bitstream.

Quantization are done on sub-band signals using a rate-distortion optimization algorithm. The first steps of the quantization process are twice as wide as the uniform-quantization and allows for coarser quantization on smaller values.

3.2.5 VP8

VP8 is a successor to the older video codec VP6 is described by a data format and decoding guide in [16] and summarized in [4]. VP8 uses block-based transformations, intra-prediction, motion estimation and motion prediction. The format used in VP8 is exclusively YUV 4:2:0 8-bit picture that is partitioned into MBs of sizes 16x16 pixels and is further divided into 4x4 pixels subblocks.

Spatial Redundancy is reduced by decorrelating the signal using DCT and Walsh-Hadamard Transform (WHT) on the 4x4 blocks of pixels. The DCT is applied on the luma and chroma subblocks while WHT is used on 4x4 size blocks that consists of average intensities of the 4x4 luma subblocks from a MB. Intra-prediction uses already coded MBs above and to the left of current MB and each block is predicted independently of each other.

Temporal Redundancy is reduced by motion estimation and motion compensation. Three different types of p-frames can be used for inter prediction, previous frame, golden frame and altRef frame. A received and decoded i-frame is a golden frame and altRef frame and may optionally replace the most recent of these. The prediction is calculated from all previous frames up to the last i-frame and is thus not tolerant to dropped frames, the golden frame and altRef frames may be used by the decoder to partially overcome the problem with dropped frames. MVs describing predicted blocks displacements are made with quarter-pel precision and the by comparing from a sorted list of MVs from the nearby MBs the best-suited vector for the specific MB is chosen.

Statistical Redundancy is reduced by applying BEC. The boolean coder uses 8-bit probabilities so that the probabilities easily can be represented using a small number of unsigned 16-bit integers. In the VP8 data stream, the probabilities of a bool symbol being zero are not close to one half so the coding efficiency gain of a BEC is big.

In-loop filtering is a computational complex process but needed to reduce blocking artifacts from the compression techniques used. Due to the high complexity, there is also an alternative simple filter that may optionally be used. The filter is applied on an entire frame at once when all MBs has been reconstructed, and the result from the filtering is used in the prediction process of subsequent frames. The simple filter applies only on luma edges to reduce the number of edges that are filtered. A threshold of difference between two adjacent pixels along an edge is determined where the filter is not applied. This can produce certain artifacts depending on the level of this threshold and the level of quantization done by the encoder. The normal filter is a refinement of the simple filter with same properties but is applied to both luma and chroma edges. When the edge variance in between the pixels a larger area around the edge is filtered.

Error Resilient mode is used to enable the encoder to use golden frame and i-frames to quickly recover in the scenario of lost frame packets, thus there are no enhanced error resilient methods but the coder makes use of the tools specified to enhance the recovery when a failure occurs.

3.2.6 HEVC/H.265

Following the earlier standard AVC/H.264, HEVC/H.265 is further improved in terms of coding efficiency and to be able to make use of parallel processing architectures. Complete specification of the bitstream and decoding process is available in [18] and is summarized by the authors of [4] and [17]

The partitioning method in HEVC can be described as a quad-tree or coding tree. The root of the tree consists of a CTU that can be of varied sizes from 16x16, 32x32 or 64x64 samples. A CTU can be further partitioned four equally sized Coding Units (CUs) which in turn also may be individually partitioned into smaller blocks. The luma and chroma samples within a CU are called Coding Block (CB). The CB may be coded either by intra-prediction or by inter-prediction compensation prediction, for intra-prediction a CB may also be split into multiple TBs and for inter-prediction the luma and chroma blocks may be split into PBs. Larger sizes of CTUs will increase the coding efficiency but also increases complexity.

Spatial Redundancy is reduced with help of a prediction method that is derived from neighboring samples, it's performed on the TBs and allows for arbitrary block sizes. There are many modes for predicting the samples to produce a more accurate prediction for many different types of image contents. To improve the continuity between the block boundaries a light post-processing filtering is applied to the boundary samples for some of the modes. Minimizing the overhead added by the intra-prediction a coding step is used that sorts the three most probable mode candidates and uses a CABAC bypassed code word for the rest, less likely modes. HEVC uses varied sizes of DCTs based on the TBs sizes, moreover an additional integer based DCT is used on the 4x4 luma intra prediction residual blocks.

Temporal Redundancy is reduced by motion estimation and motion compensation. Predictions of MVs are calculated using neighboring blocks and earlier coded pictures that usually correlates with the current MV. Due to the different sizes of blocks when for example a large PB is next to several small size PBs a technique called Advanced Motion Vector Prediction (AMVP) is used for reducing the number of possible MVs and choosing the most probable one. MVs uses quarter-pel accuracy and fractional values need interpolation at integer-value positions using filters. Using block based inter-prediction on different sizes of blocks as in the structure of the quad-tree results in an over-segmentation when certain objects move against still background for example. By using a block merging technique where the leaves in the quad-tree may be merged together and allowing for merged blocks to reuse the motion parameters from the neighboring blocks improves the efficiency of these situations.

Statistical Redundancy is reduced by EC. A CABAC is used and it's an improved version from that used in AVC/H.264. The same steps are taken, binarization, context modeling and AC. To increase the throughput of the coder

the high dependency of data for the coding is reduced so that parallel processing becomes easier and increasing performance for hardware and multiple CPU architecture implementations. Measures for decreasing dependency is done by grouping bypassed coded bins that can be coded faster than regular bins and be processed if they occur consecutively.

Quantization level can be varied in many different quality settings. Many images have varied content where some part consists more color and brightness variations thus a frequency dependent quantization step sizes are possible on different parts of an image to make use of this attribute.

In-loop filtering is done using two different filters, an in-loop deblocking filter and a Sample Adaptive Offset (SAO). To enable parallel filtering there is no dependency between the block edges for the deblocking filter. SAO is applied after deblocking and reduces ringing artifacts that may occur when using larger block size operations.

Parallel Processing is improved by allowing each picture to be partitioned into tiles where the tiles can be decoded independently. A slice is a row of CTUs and is defined in a wavefront parallel processing mode where rows can be decoded in a parallel manner. After the process of decoding the first row has made a few decisions the decoding of the next row can be started and so forth. Processing within a slice can also be made in parallel. Only the first slice contains the full header so all other slices are dependent that the decoder has access to the first one. The decoder can then decode the slices as soon as they are received without waiting for next row to arrive.

3.2.7 VP9

VP9 is the successor to VP8 which was discussed earlier and was developed as an open source alternative to AVC and HEVC, VP9 became available 2013 [20] and the bitstream and decoding specification is defined in [19] and summarized by [4]. A frame is partitioned into blocks of sizes 64x64 called SB. These blocks can be further partitioned into one, two or four smaller blocks. The partitioned blocks may also be partitioned to a minimum of 4x4 similarly to HEVC as a quad-tree structure. Two types frames are used i-frames and p-frames where the p-frame consists of three different types, the previous, golden frame and altRef frame.

Spatial Redundancy is reduced by intra-prediction together with residue coding. Predictions are performed based on the previously decoded neighboring blocks and are performed on the same blocks as the transformation blocks. Two 1-D arrays are used for storing the reconstructed pixels from neighboring blocks. Integer transform using DCT is applied on the partitioned blocks that differ in size from 4x4 to 32x32 samples.

In some cases, an Asymmetric Discrete Sine Transform (ADST) may be applied for better transformation when the prediction shape is such that the samples near the block boundaries are better predicted with small error.

Temporal Redundancy is reduced by motion prediction and motion compensation. A MVs can point to any of the three p-frames and is chosen from a sorted list of candidate vectors that are calculated from already decoded surrounding blocks. If there isn't enough information previous decoded frame may be used for calculating MVs. Quarter-pel pixel precision is achieved for motion compensation by applying one of three different 8-tap filters, Lagrangian interpolation filter, DCT-based interpolation filter or a smoothing non-interpolation filter. A frame uses three reference frames for prediction which it chooses from a list of eight references. To allow for quick bit-rate variances the reference frames are scalable to different resolutions if needed.

Statistical Redundancy is reduced by BEC. A small set of unsigned 16-bit integers and an unsigned 16-bit multiplication operation is used. The probabilities can be changed in the frame header and are coded using AC and by keeping track of how many time each type of syntax element has been encoded the BEC may adjust the probabilities at the end of each frame.

In-loop filtering is used to reduce blocking artifacts from block-based processes. Due to the difference in sizes, a flatness detector is implemented to reduce computations needed for flat areas of a picture. 4 filters of different widths that are applied according to edge pixel differences to threshold values

Parallel performance is enabled by the implementation of tiles. Tiles of dimensions that are multiples of 64x64 consisting of SBs are sent so the encoding and decoding can be processed at different tiles at the same time.

Adjustable Quality in a frame is made possible by a segmentation map where each frame may be divided into up to 8 segments. These segments may specify different quality attributes such as quantizer level, loop filter strength and more.

3.2.8 AV-1

The successor to VP9 by the alliance of open media and is designed to be an open source and royalty free codec [21] and is described in [22]. As of writing this report this codec was still being developed and was unoptimized for encoding and decoding thus the features and techniques may that is written in this report may alter from the finalized version of the codec. The codec design allows for multiple resolution and bit-rates to be scalable in order to support multiple devices with different processing power. The decoder can extract only part of a frame for decoding to be especially useful for Virtual Reality (VR) applications

where the focus only lay on a part of the frame, this method is called large-scale tile decoding. SBs of sizes 128x128 or 64x64 is used that can be partitioned into smaller blocks for prediction and transformation with a quad-tree structure and using same frames as VP9, i-frame and p-frame where p-frames consists of three different types, the previous, golden frame and altRef frame.

Spatial Redundancy uses inter prediction together with traditional transformation based techniques to reduce spatial redundancy. Predictions are used based on already decoded neighbors with 65 different angle modes available. Smooth regions are predicted using a special more suitable mode and chroma samples may be predicted from luma intra residues. Transformations are done by using DCT and an ADST to decorrelate the signal. Different transformations can be applied for the horizontal and vertical plane and the sizes of the TBs can vary. Quantization uses a new kind of optimized quantization matrices and can be either uniform or non-uniform.

Temporal Redundancy is reduced by motion estimation and compensation. MVs are calculated from reference frames and their relative distances and stored into a list of candidate MVs that are used for different modes of motion predictions. The inter predictions are made on blocks that may use overlapped motion compensation that produces modified inter-predicted samples, this is done by blending the samples from the current block with the samples based on motion vectors from nearby blocks. ⅛ or quarter-pel precision is used for MV subsampling.

In-loop Filtering is performed at several steps. An adaptive intra edge filter is applied on the above and left edges of each TB with different filtering strengths depending on the block sizes. Interpolation of the inter predicted blocks are affected by two one-dimensional convolutions, different four tap filters depending on the prediction mode are used first horizontally then vertically to obtain the final prediction block. A loop filter is applied to all vertical boundaries first then on all horizontal boundaries. The size, level and threshold of the filter is varied and adaptable due to the many sizes of TBs. A Constrained Directional Enhancement Filter (CDEF) used for deringing purposes based on the detected direction of blocks and is applied on 8x8 pixels blocks.

Entropy Coding is done by a Non-binary AC that may give symbols eight possible values giving the coding a more complex but adds the ability to process several symbols each clock cycle that improves the performance.

3.3 Coding Algorithm Comparison

A comparison between different coding algorithms were made in tables 1 and 2. These show what tools are used for the different algorithms with the exception of the Dirac and VP6 algorithms, Dirac was excluded since the full

frame transform was deemed non-optimal for the system and VP6 because the later iterations VP8 and VP9 contain similar techniques but with added tools for better compression. Figure 1 displays a timeline when the different coding algorithms were released.

There is a clear trend in newer algorithms with the usage of bigger partition sizes that have a tree structure and are sub-dividable. This is especially effective when dealing with higher than FHD resolution images.

Using multiple sizes of block transform operations the coding efficiency is improved but the complexity is increased with each added block size.

By using i-frame predictions the coding efficiency can be increased for each image and by using more modes with multiple different angles the accuracy of the predictions is higher by at the cost of an increasing complexity.

P-frame predictions highly improve the coding efficiency depending on the accuracy in the MVs predictions and subsampling of the vector values. Higher precision results in better quality of the prediction signal that may improve the overall compression efficiency but it will require more bits to represent the MVs, i.e. quarter-pel is higher precision than half-pel.

EC is a lossless process that improves the coding efficiency quite substantially. Different algorithms are more or less efficient with the CABAC being one of the more complex and also efficient methods.

Filtering is an important step for improving the visual quality though not introducing any coding efficiency by itself it allows for other tools to achieve higher compression ratio and the complexity of the filter is highly correlated to the different block transform sizes where more sizes will require multiple levels of filtering to reduce blocking artifacts. Ringing artifacts that are introduced by the reducing of high-frequency components when performing transformations may also be filtered for improved visual quality.

Parallel processing adds overhead reducing coding efficiency but greatly improves the performance when the coder is implemented on a many-core processing unit or a dedicated hardware chip.

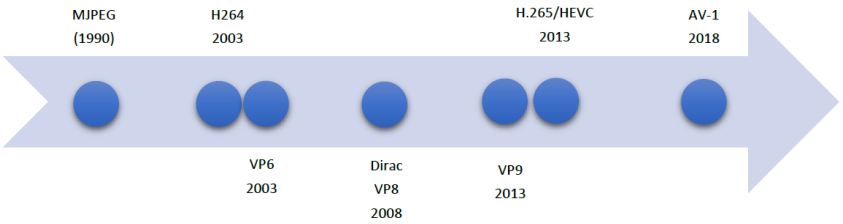


Figure 1: Timeline of the release of different coding standards.

Codec	MJPEG	VP8	H.264/AVC
Partitioning sizes	8x8	16x16 with 4x4 subblocks	16x16, 8x8 and 4x4
Transform	2-D DCT	DCT or WHT on 4x4 blocks	DCT and HT on 4x4 block of transformed DC coefficients
Spatial prediction	None	Intra-frame macroblock prediction	Intra-frame macroblock prediction, 9 modes with eight directional modes
Temporal prediction	None	Predictions from previous frame, 1/4 luma and 1/8 chroma precision and motion vectors predicted from up to three surrounding blocks	Predictions from previous and next frame, 1/4 pixel precision for motion vectors
Entropy encoding	Variable-length HC	BEC	CAVLC or CABAC
In-loop filtering	None	Filter on an entire frame, two different filters available	Filtering on horizontal and vertical edges
Useful features	None	Error recovery	Flexible MB ordering, Scalable Video Coding (SVC) extension, Slice-based threading

Table 1: Comparison of a few coding techniques used by different codecs

Codec	VP9	H.265/HEVC	AV1
Partitioning sizes	64x64 SB, further partitioned into min 4x4	64x64 CTU, further partitioned multiple sizes	128x128 or 64x64 SBs in a quad-tree structure
Transform	DCT or ADST	DCT or integer transform based on DCT	DCT and ADST
Spatial prediction	I-frame MB prediction. 10 modes and six directional predictions	i-frame MB prediction, 35 modes with 33 directional modes	I-frame MB prediction, 65 different angle modes. Chroma can be predicted from Luma intra residues
Temporal prediction	Predictions from previous frames, ⅛ pixel precision and motion vectors predicted from a list of candidate vectors from up to eight surrounding blocks	Predictions from previous and next frame, two lists with 16 frames each are used. MVs from a list of candidate vectors, ⅛ pixel precision	Predictions from previous frames, list of candidate MVs using overlapped motion compensation, ⅛ pixel precision
Entropy encoding	Arithmetic BEC	CABAC	Non-binary AC
In-loop filtering	4 different steps of filtering, flatness detector for less complex filtering	In-loop deblocking and SAO filters both optional	Adaptive intra edge filter with different filtering strength and CDEF
Useful features	AltQ and AltLF segmentation, tiles for encoding/decoding in parallel	Tiles for parallel processing	Scalable for different devices, large scale tile decoding

Table 2: Comparison of a few coding techniques used by different codecs

3.4 Summary

A few of the most commonly used codecs on the market has been presented and some of the technical aspects of each has been highlighted.

Starting with the simplest codec MJPEG that only reduces spatial and statistical redundancy.

VP6 codec was thereafter presented that reduces temporal redundancy based on previous frames using block-based transformation. HC or BEC is used for EC.

AVC/H.264 uses a prediction method to reduce spatial redundancy and can predict motion vectors based on both previous and future frames, the entropy coding is performed by a CAVLC or CABAC that offers significantly better compression ratio than earlier codecs.

Designed as an open-source competitor to AVC/H.264 the DIRAC coded uses a different transformation technique and lacks the intra-prediction for spatial redundancy. Statistical redundancy is done by a kind of CABAC.

A successor to VP6 is the VP8 codec that uses similar techniques on larger block sizes that are partitionable. Intra-prediction, motion estimation and compensation is calculated from previous frames. A simple and normal filter is available for reducing the complexity of the filtering process.

Further research and compression improvements resulted in HEVC/H.265 codec that is a successor to AVC/H.264. The block sizes are enlarged and a quad-tree structure is applied where the prediction and transform blocks are subdivided from the bigger block. A specific block can be coded by intra and/or inter prediction and uses an improved CABAC with higher bit throughput and the ability to be decoded in parallel.

VP9 codec followed after VP8 and uses similar block sizes and quad-tree structure as HEVC/H.265. Uses same frames as VP8 and VP6 but has a ADST alternative for transformation. Better precision motion prediction and more filtering results in higher coding efficiency and greater complexity. Parallel performance is improved by use of tiles and segments may be used for adjusting quality within a frame.

An alliance of several actors was formed and AV-1 emerged as a successor to VP9 and designed to be the next step after VP9, HEVC/H.265 with higher compression ratio. Larger block sizes that are subdivided into varied sized blocks for transformation and prediction stored as a tree structure. Spatial prediction also used for predicting chroma samples based on the luma residues further improves coding efficiency.

4 Related work

Wireless video transmission is not a new technology and comparisons between different coding standards has been made extensively. Improvements and optimizations are made both in the wireless transfer protocols and in video compression specialized for a wireless channel that expects packet loss and signal deterioration. This section presents an overview of literature and research that has been made in the area of wireless video coding from drones, techniques used to improve quality of a video stream over a wireless network, error concealment methods and ways to reduce the latency of a video coding algorithm.

4.1 Drone video transmission

The quest for zero-latency video coding had been sought after for some time when writing this thesis.

One company that has integrated their video transmission system with drones is AMIMON [23] that uses a transmitter that they claim can send delay-free uncompressed video over a radio link on the 5GHz band. Used by a few professional drone pilots but far from widely integrated for most users.

DJI [24] that is a company mainly focusing on video platform drones has developed a system that can achieve 50ms latency for 480p resolutions though it is unclear if that is the screen to screen latency or only the video transmission latency.

An open source project[25] made with the focus on using cheap hardware such as the Raspberry Pi [2] reducing latency and improving robustness by altering the WiFi protocol so that packages are sent arbitrarily without association. It claims to be able to transmit a FHD video stream at around 100ms when using the hardware accelerated video coding of the Raspberry Pi.

4.2 Video coding over wireless networks

Video coding over a wireless network limits the amount of bandwidth and set a requirement for a high compression rate. While packets may be lost and due to the latency requirement, the delay of retransmitting new packages would be too long therefor an error resilient coding technique that can help to recover lost or corrupted data helps to improve the perceived video quality without adding a large delay.

A technique that provides the transmission to use lower temporal and spatial resolutions or reduced quality provides graceful degradation for an unreliable transmission channel. The H.264 codec implements this as an extension called SVC[26]. Though adding an overhead by sending more data and complexity in the decoder the ability to transmit during lossy transmission environments is a very good attribute for the system proposed in this thesis.

Temporal concealment where the correlation between a lost and it's neighboring frames is a popular error concealment technique. For example the

AVC/H.264 uses a couple of techniques for countering packet losses, A slice-structured coding where no intra-frame prediction is performed between different slices and with small packet sizes the probability of a bit-error hitting a packet is smaller. Flexible MB ordering maps different patterns of MBs into slices and data partitioning that is efficient for using together with prioritization, Unequal Error Protection (UEP) or Forward Error Correction (FEC)[27]. The VP9 codec implements an error resilient mode that allows all frames to be decoded independently of previous frames.[19]

Another approach uses a form of error resilience packets that are sent at a time intervals to add redundancy to i-frames and prediction information in [28] The result is an improvement in PSNR in comparison to a frame copy method that conceals a lost frame using the previously received frame.

When using a temporal prediction method high compression efficiency is achieved but the coding process will be vulnerable to lost packets through a wireless channel due to the error propagation where the next prediction will suffer from the lost information. Typically an intra-refresh method was used that resets the temporal prediction by sending an i-frame for reference. This method decreases the coding efficiency and a proposed framework delivered in [29] that allows for more options to counter the error propagation. A soft reset joint intra-inter prediction mode that controls the dependency on previous frames using adjustable weights for a controlled trade-off between compression and resilience. The idea was that if the encoder can accurately estimate the End-To-End Distortion (EED) and make use of a number of modes achieving better control of the error propagation.

An efficient method for constraining error propagation and error concealment distortion is proposed in [30] and is based on frame level rate-distortion analysis. The method was shown to increase PSNR from the method used in AVC/H.264. Though these methods were proven to outperform older technology in the specific applications there was little discussion about added overhead and complexity to the compression algorithms.

For a robust real-time video stream in Ultra High-Definition (UHD), the HEVC codec mainly focuses on high compression rate and takes little consideration of the video transmission according to the authors in [31] where they propose three methods for improving performance and robustness. Picture prioritization, error concealment mode signaling and tile-based video parallel processing. A moderate video quality gain was achieved using the first two methods while the third proved to improve the decoding speed quite substantially.

4.3 Low latency video coding

Achieving a low latency video coding for a real-time video requires that both the encoder and decoder to add as little delay as possible when performing the computations. As the compression efficiency is steadily improved with each new coding standards the complexity also increases and requiring higher performance hardware to enable real-time coding performance.

Near zero latency coding can be achieved using the right hardware components such as Field-Programmable Gate Arrays (FPGAs) as done in [32] that performs capture to display latency of 20.54 ms while using the AVC/H.264 coder. The authors point out that latency for a coding process is highly dependent of entire frame or where a large number of video lines are buffered. Such a part of the coding process is the bit-rate averaging buffer that works to maintain a specified bit-rate over a period of time, an averaging period that a decoder stream buffer must match to successfully decode the video stream. The size of this buffer is highly correlated with the quality of the video for a constant bit-rate video.

As the complexity of a codec algorithm is increased the ability to perform parts of the process in parallel helps the codec to achieve fast coding times when implementing a codec on massively parallel computing architectures such as a General Purpose Graphical Processing Unit (GPGPU), multi-core CPU or a hardware solution. By altering the in-loop deblocking filter that is a highly computational intensive part of the VP9 codec in a way that the authors in [33] proposed the complexity was reduced allowing for easier implementation on a GPGPU reducing the coding time without reducing visual quality.

Using a Raspberry Pi with a camera for low latency streaming applications by making use of the hardware acceleration implemented to improve the coding performance has been performed by many enthusiasts such as [34] that shows that real-time encoding is achieved at High-Definition (HD) resolution with just a few easy steps.

A more extensive research using a Raspberry Pi is made in [35] where video streaming over a distributed Internet of Things system was researched. The conclusion was that while reaching an end-to-end delay of 181ms the video coding accounted for 90% of that delay. Coding was done by AVC/H.264 coder due to the hardware capabilities of the Raspberry Pi and also the byte stream format that is defined in the AVC/H.264 that packets the coded data as Network Abstraction Layer (NAL) units.

A technique to reduce the latency of a coding process is by slicing a frame as can be done in AVC/H.264 and the theoretical gain is discussed in [36] and while the latency is theoretically reduced the coding efficiency is lower with the increased overhead that each slice introduces.

The authors of [37] explains the overall latency for a video conferencing application in order to propose a sub-frame based data flow to reduce the overall latency from versus a frame based version. They highlight the importance of a video codec to avoid b-frames to reduce buffering on the decoder side and reduce the size of the bitrate buffer that ensures that a certain target average bitrate is met. An error resilient is also advised to enable to help the decoder to recover and conceal the errors that might occur in high packet loss situations. The conclusion was that by dividing a frame into smaller sub-frames the latency could be reduced from 33ms to 2ms for that specific system.

The latency for a system from an event occurring to that event being displayed on a screen or received to be analyzed by an autonomous system is presented in [38]. All parts that contribute to a delay was analyzed and a fo-

cus on the delay caused by the camera refresh process that is correlated to the frame rate output. A high frame rate camera was used and a frame skipping method proposed to utilize the low delay achieved when using a high refresh rate and also reducing the bitrate by skipping frames that were similar to the last frame. The authors also propose a preemption mechanism that flushes the encoder buffer and shortens the waiting time of frames that differs largely from the previous. The first method provided a bitrate reduction of up to 40 times versus sending all frames captured by the camera while reducing the latency for the system from around 100ms with a low frame rate camera to around 20ms using the high frame rate camera.

4.4 Summary

Existing technology and research in the area of racing drone video compression was summarized. Hardware solutions that transmit low latency video over distance exist but the main focus in these solutions are based on the wireless protocol delivering the data. Research has been performed extensively and modern video codecs use error correction and concealment methods to increase the perceived video quality over error-prone networks. To be able to achieve close to zero latency video coding most systems use hardware-based solutions or hardware-accelerated onboard chips, the downside of using these methods is the limited flexibility that the codecs offers. Usage of a Raspberry Pi as the platform for the video capture, coding and transmitting of data has been performed and again the focus has been mainly on improving the wireless protocol to optimize for low latency and improving the link robustness. Mostly used codec is the AVC/H.264 due to its relative simplicity and easy implementation in both software and hardware accelerated systems. No uses of SVC to send multiple low fidelity streams alongside the higher fidelity stream has been found, this might be due to the increased complexity and low availability of hardware accelerated coders.

5 Codec Comparisons

Different coding techniques were compared in terms of their compression ratio and computational performance by subjectively reading and analyzing existing literature on the subject. Measuring the quality comparison from the original source to the lossy coded output is proven to be a topic that has created many discussions on which technique provides the most accurate result. A widely used technique that compares each pixel between two images is PSNR or Luma Peak Signal-to-Noise Ratio (Y-PSNR). A method that tries to predict the perceived quality is Structural SIMilarity (SSIM). Some methods use several different methods to such as Video Multi-method Assessment Fusion (VMAF) that tries to achieve a better prediction of the perceived quality [39]. Next to these calculated objective scores of quality is also the method to use subject viewing tests where video sequences are shown to non-expert test subjects that assess the quality of the video. Subjective tests are the preferred method by many and the result may take priority over the objective testings that take advantage of effects that are visually noticeable but are not reflected in the objective tests [40].

Many comparisons are made using a quality setting removes optimizations that reduce the metric scores such as PSNR and SSIM. This allows coders to achieve a higher benchmark score without the actual perceived visual quality is any higher or the coding times are too high for practical use, especially for a real-time system as is proposed in this thesis. Many comparisons also use the proposed reference software provided by Joint Video Team (JVT) that lacks optimizations for better coding performance thus only the amount of bits that are compressed is relative in these tests.

5.1 Video coding comparisons

For live game streaming an extensive comparison in made by [41] where the coder implementations of AVC/H.264, HEVC/H265 and VP9 codecs was compared. The implementations used were from the FFmpeg [42] library which is commonly used and the presets for each encoder was chosen to optimize speed over quality. x265 coding efficiency was found better than both x264 and VP9 for the chosen settings, compared to x264 around 20% bitrate savings and VP9 around 27%. The x264 coder was found to be more efficient for low to medium complexity videos at lower resolutions compared to VP9. The encoding time was found to be higher for x265 and VP9 compared to x264 where the x265 was approximately 2.6 times slower while VP9 was about 4 times slower. It was also found that the coders performed differently for varied complexities of video sequences.

Subjective quality without regard to the coding performance of HEVC/H.265, AV1 and VP9 were evaluated by [43] where high-quality video broadcasts using different bitrates and coders was shown to subjects and assessed. Tests showed that the compression efficiency of AV1 was improved from VP9 while the efficiency of HEVC/H.265 was slightly higher than that of AV1 and required lower

bitrate to achieve the same subjective quality.

A comparison between VP8 and AVC/H.264 in [44] revealed that the quality level is comparable from low to high definition and compression ratios between 1 to 40. The encoding speed for the x264 implementation of AVC/H.264 was found to be much faster for same quality however at the time the optimizations to the VP8 coder was still under progress.

A large-scale comparison of AVC/H.264, HEVC/H.265 and VP9 was performed in [45] using a large set of video sequences to evaluate coding efficiency in many different scenarios. Ranging from fast movement, scene changes and animation coders perform better or worse as noted in previously reported findings where some research prove to be in favor of a coder depending on the sequences chosen for evaluation. This comparison was performed with video on demand services in mind, therefore sub-optimal presents for a real-time stream were used. The result of the coding efficiency was in favor of HEVC/H.265 closely followed by VP9 and lastly AVC/H.264.

The Dirac codec was compared to AVC/H.264 in [46] and the authors used low-resolution video sequences with constant bitrate for both coders to evaluate the quality, compression efficiency and relative speed of the encoding. The result showed that the visual quality of AVC/H.264 was better then Dirac but due to Dirac's low complexity the coding performance was better. This was against the AVC/H.264 JM 17.1 reference software from JVT and not the more optimized x264 implementation. The authors concluded that the H.264 achieved better overall results in the comparison.

A project from Moscow State University aimed at delivering annual video compression reports released a comparison between 10 video codecs implementations [47], among them, were the AV1, x265 based on HEVC/H.265, x264 based on AVC/H.264 and VP9. In their 2018 report, they showed an advantage in both compression ratio and encoding speed to an HEVC/H.265 coder over an AVC/H.264 coder with a bitrate improvement for most cases. The parameters were set to achieve similar visual quality so the encoder speed presets differed for AVC/H.264 at "fast" and for HEVC/H.265 was chosen as "ultrafast". The encoding speed could be improved in the AVC/H.264 encoder at the cost of worse quality. In their comparison from 2017 using highest quality settings AV1 showed better compression efficiency for same quality compared to AVC/H.264 and HEVC/H.265 although the AV1 encoding speed was a lot slower than the others, during writing of this report the AV1 coder lacked speed optimizations which might partly explain the slow performance of the coder.

HEVC/H.265, VP9 and AVC/H.264 was compared for a set of UHD sequences in [43]. The comparison in compression efficiency consisted of both an objective and a subjective part where both methods showed that HEVC/H.265 required lower bitrate to achieve similar quality. The test also showed that AVC/H.264 was superior to VP9 in coding efficiency based on the subjective method.

An objective comparison of VP9 and HEVC/H.265 was performed using the reference software for high-resolution video streams in [48]. Best quality encoder parameters were used to achieve the highest possible PSNR and the

tests showed that HEVC/H.265 achieved higher PSNR for the same bitrate on FHD and UHD resolutions and was only beaten on HD at 5Mbit/s bitrate. The authors did note that VP9 is an open-source standard which it benefits from.

5.2 Compression efficiency

The compression efficiency of each codec differs heavily depending on the chosen setting and tools used when coding the video stream. As later codecs may provide more tools to increase the coding efficiency they also entail a higher complexity requiring more powerful hardware. Commonly used for real-time streaming is CBR that reduce the complexity of the coding process versus Variable Bitrate (VBR) and for a network connection where signal degradation is expected a good practice is to use a bitrate of approximately 80% of the bandwidth available. In reality this bitrate will vary even while using CBR due the the large compression differences many coding algorithms use for p-frames and i-frames where a p-frame is usually many times smaller then an i-frame. This sets a requirement on the coding algorithm to be able to achieve compression ratio corresponding to a bitrate of around 4000 kbit/s for a high quality, high resolution and at least 30 FPS on a wireless connection that exists on the proposed system. To achieve this level of compression the MJPEG coder falls short where a FHD stream at 30 FPS produces a bitrate of around 40 Mbit/s, 10 times the requirement. For the other coding algorithms VP8, AVC/H.264 can produce 4000 Kbit/s[41] bitrate while VP9, HEVC/H.265 and AV1 also manages this but at even better quality [43].

5.3 Coding performance

The time it takes for a video stream to be encoded and decoded is highly correlated with the efficiency of the coder. A highly efficient coder is most often also a low-performance coder that requires high complexity to achieve the high compression rate. Different aspects of the coding algorithm may add higher complexity at a little gain in visual quality and research is made to improve the implementations of such parts of the different coders. One such part is the deblocking and deringing filter that most coders use in different varieties from simple to more complex, the visual quality gain when using a deblocking filter is usually high and make up for the increased complexity. The cost of the increased block sizes, different sizes of TBs and PBs in later codecs such as VP9, HEVC/H.265 and AV1 make filtering more complex so that all edges within blocks are filtered.

The computational complexity of HEVC/H.265 was modeled and tested in [49] and showed that the complexity varied highly depending on the encoding configuration, compared to AVC/H.264 High Profile the complexity of HEVC/H.265 was concluded to range from $9 - 502\%$ more complex. The main contributions for the high complexity w determined to be the transform and quantization that added up to 43.2% of encoding time consumption when using AI mode. Using Random Access (RA) configuration resulted in a significant

complexity increase of the motion estimation and utilized around 58.6% of the encoder time consumption, the decoder complexity was dominated by the inverse transform and filters, 15.9% and 12.9% respectively in AI configuration and for RA configuration, motion compensation utilized 24.8% and the filter used up 12.4%. The interactivity between different tools of a coding algorithm makes complexity estimations harder where each part most often is estimated separately. A test of the complexity of HEVC/H.265 was performed with a sequence of predefined encoder configurations. It was determined that tools such as the Hadamard Motion Estimation, Asymmetric Motion Partitions (AMP) and filters should be the first to be enabled for a complexity constrained system. The efficiency gain when using AMP is also verified in [50] that showed a coding time increase by 14% while increasing the coding efficiency slightly for a video conferencing scenario.

5.4 Discussion

A deep analysis of the different coding algorithms and implementations have been presented from the existing literature. Comparisons of implementation and algorithms show that both AVC/H.264 and HEVC/H.265 has modes for low latency situations even though the algorithms have tools such as b-frames which per definition adds at least one frame of latency. The low latency modes have inactivated these frames to prevent this added latency and must be enabled by the decoder to prevent unnecessary buffering of frames. By reviewing the encoding speed of the different implementations AVC/H.264 and HEVC/H.265 were clearly better than both VP8 and VP9. AV-1 codec performance was during the writing of this report greatly unoptimized and not usable for this implementation. Even though the coding efficiency of the HEVC/H.265 encoder is higher than the AVC/H.264 the increased complexity needed to achieve this compression improvement sets high requirements on the hardware platform and tests that showed favorable encoding speeds for HEVC/H.265 used a faster encoding preset to achieve this.

5.5 Summary

Most of the comparisons found in the literature have no constraint in encoding performance and mainly focus on the best possible quality for lowest bitrate. As this is optimal for a streaming service that requires encoding once and multiple decoding requests such as Video On Demand (VOD) services. The comparison made for a live gaming application shows a requirement on the coding performance and AVC/H.264 gained the advantage in a higher encoding speed for the low latency settings. Among the sequences used were Counter-Strike and Need for Speed which shows similarities to the camera movements of a drone in flight, for these sequences HEVC/H.265 showed a bitrate reduction over AVC/H.264 of approximately 30% while the encoding run-time was around 3.5 times longer for HEVC/H.265. By following the development of the implementations of the coding algorithms it's clear that as an algorithm becomes more mature, better

optimizations are made. Comparing the results from [47] it was noticed that the x265 implementation of HEVC/H.265 performed better than x264 in the last comparison while the results from [41] showed the opposite in terms of encoding speed. This could be explained in the parameters used for the different tests but also the relatively new and unoptimized x265 encoder implementation.

6 Research Methodology

The methodology used for this thesis was to conduct a literature study on the existing published research on low latency video streaming and streaming over wireless networks. This was performed in order to gain knowledge on the video coding algorithms, their complexity and efficiency in terms of bitrate reduction, coding time and perceived quality on a wireless transportation media. The study started with gathering information about techniques used for reducing the bitrate of images and video streams. From the gathered information a few different coding algorithms were selected depending on popularity and availability for using an optimized implementation for testing on a low power board computer. The different tools that were of most interest for achieving high compression ratio were researched and explained for each algorithm so that a comparison of the tools used could be made.

Research containing coding efficiency comparisons were gathered and studied and due to the large number of existing studies, the content of each was sorted with respect to the relevance and similarities of the goals of this thesis. Many papers containing algorithms for improving certain aspects of a specific coder were determined as having low relevance. Since most comparisons that related to the coding efficiency were performed using unoptimized reference encoders that mainly focus on achieving highest possible PSNR or similar quality measurement and only a few of these were deemed relevant enough to be considered. The most relevant papers found considered low latency streaming either for video conferencing or live video game streaming. The complexity of coding algorithms was also deemed relevant to a deeper comparison of the tools used and their quality/complexity trade-off could be made which was vital to the success of the thesis, these papers were fewer in the amount so searching for relevant information proved to be a time-consuming task.

6.1 Coding Performance and Efficiency

A comparison between codecs was performed first on a laptop where the performance and quality loss for a set of coding algorithms using similar settings. Four different video sequences were used with varied content, frame-rate and resolution to gain better comparison data. An open-source software FFmpeg [42] was used for evaluating coding performance in terms of frames encoded per second. For quality assessment of these different codecs and bitrates, the tool VMAF was used. The best performing codec was then used for an implementation on a system that mimicked the drone-ground setup that is sought after.

6.2 Latency Measuring

Measuring the latency of a video coding process can be made in multiple ways. A commonly used method is done by setting a camera in front of a display showing a stopwatch with millisecond resolution. The measurement error is highly dependent on capturing time of the camera and the display refresh rate.

For a 60Hz, 16.7ms frame period display and a 30 FPS a 33ms frame period camera the error can be as big as 49.7ms [38]. The stopwatch accuracy may also present problems where the digits are updated during the capturing process that yields that reading the results prove difficult and inaccurate. A more advanced measuring technique is done by using a diode and a photoresistor. The camera is set up facing the diode and the photoresistor is facing a screen that receives the video stream of the diode. An oscilloscope is the used to measure the outputs of both components and the time delay between the diode and the photoresistor is then measured. A software may also be used for measuring the time for an encoder finish encoding a video sequence. For measuring encoding in real-time the issue consists in that each frame is divided into multiple blocks that are encoded and decoded in parallel. There are no good methods to measure the encoder time using the proposed software for a complete frame but only the separate buffers that are created.

6.3 Network Instability

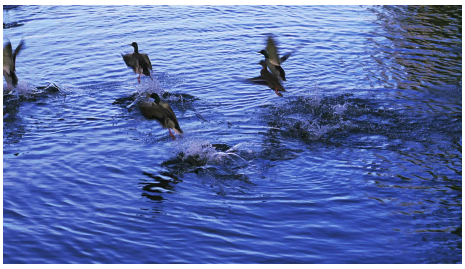
Testing how a streamed video sequence deteriorates in quality and how error propagates was done using a GStreamer packet called netsim. This allowed for simulating probabilities that a data packet is dropped, duplicated, delayed or reordered. This tool was useful to evaluate the error correction performance of the video codec.

6.4 Video Sequences

Using a video file stored as a raw format on disk and measuring the time to process the sequence is a good way to compare the performance of different coding algorithms. For higher resolutions such as FHD, the devices hard drive reading capabilities quickly becomes a bottleneck limiting the encoding performance. The solution was therefore, to limit the resolution and video sequence length so that the memory allocation was sufficient so that the CPU could read the file directly from memory and not be hindered by the slow reading speed of the persistent storage, this was also more similar to the real-time streaming scenario where no persistent storage is used. Four different video sequences were used for comparing performance and quality. These were chosen to have similar characteristics as a video from a racing drone with varied frame-rate and resolution. Bike in figure 2a and 2b is featuring FPV of a mountain-biker going downhill with fast ambient movement as well as camera movements. Ducks in figure 2c is captured at a higher frame-rate of 50 FPS and contains a lot of temporal movement when all ducks go airborne simultaneously. The last sequence is a Drone in figure 2d flying over a city.

(a) Bike 720p30.

(b) Bike 1080p30.



(c) Ducks 720p50.

(d) Drone 720p30.

Figure 2: Still image from video sequences used for comparisons.

6.5 Summary

This section has been focusing on the research methodology used. It was clear that there were several parts involved when determining the suitability for different video codecs when low latency, bad network capabilities and limited hardware were taken into account. Testing the performance and efficiency therefore, had to be split into different sections and evaluation was done independently of each other.

7 System components

The components used to mimic a real-life testing environment is presented in this section.

7.1 Transmitter and Receiver

The system used a Raspberry Pi 3b+ which is a single board computer as the transmitter of the video stream with a Broadcom BCM2837B0 1.4 GHz 64-bit processor, 1GB ram, dual channel WIFI IEEE 802.11.b/g/n/ac, CSI camera port and more. The board ran a Linux distribution and the GStreamer multimedia framework for video streaming. Camera used is a Raspberry Pi camera module v2 that supports FHD at 30 FPS, HD at 60 FPS resolutions among others. For receiver a laptop with an Intel I7-5700HQ CPU at 3.5GHz, NVidia GTX 965M video card, 8GB RAM and WIFI card supporting IEEE 802.11ac running Ubuntu 16.04 and GStreamer 1.4.

7.2 Drone

The main goal of this system is that it should be able to be mounted on a small racing drone with a total weight of around 500-700 grams as is exampled in figure 3. The drone the system was intended for was powered by a 4 cell lithium-polymer battery with a maximum voltage of 16.8V and capacity of 1400mAh. Four motor controllers (ESCs) each capable of producing maximum 20 amperes continuous current draw connected to four electrical motors results in a flight time of around $3 - 4$ minutes during regular flight operations.

Figure 3: Image over an example drone

7.3 Display

An Acer 27" G276HLAbid LED display used running on FHD resolution with a refresh rate of 60 Hz was used for visual quality assessment.

7.4 Latency measurements

Latency was measured by a stopwatch with millisecond resolution was used for determining the end-to-end latency for the system as can be seen in figure 4. A Canon EOS550D camera was used to capture the stopwatch and the received frame of the stopwatch time with a shutter speed of 1/4000s.

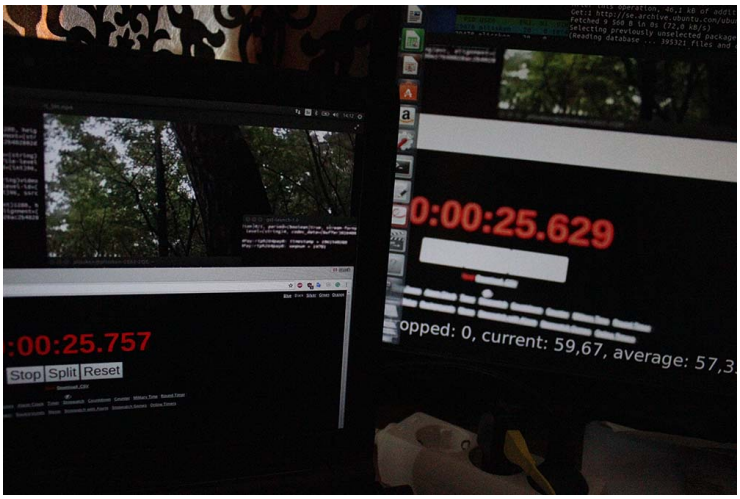


Figure 4: Image over the latency measurement system

7.5 Software

The GStreamer multimedia framework as mentioned earlier was used for coding the video stream and packet onto an RTP stream. This framework consists of multiple encoder and decoder implementation for several different standards such as AVC/H.264, HEVC/H.265, VP8 and VP9 enabling a simple test setup that well suited for the chosen platform.

8 Experiment methods

The experiments consisted of two parts, one where four different video codecs were compared in terms of coding performance using FFmpeg software and the laptop described in section 7 as evaluation platforms. The other experiment consisted of an implementation on the Raspberry Pi and the laptop described in section 7.1 to better evaluate real-life performance and quality of the video stream. The four codecs that were decided to be evaluated was VP8, VP9, x254 and x265 where even though VP8 and x264 are predecessors to VP9 and x265 it was determined that due to their lower complexity and highly optimized implementation they deserved to be evaluated even though it was expected that the compression efficiency would be worse.

8.1 Testing parameters

As the coding algorithms were to be implemented on a low powered system the settings for low complexity were chosen for the different codec, also as low latency is a requirement a preset for this was also initially chosen. For x264 and x265 the parameters "ultrafast" and "zerolatency" and for VP8 and VP9 the parameters "deadline=realtime" and "cpu-used 8" were used and are described in appendix C. For the second part of the experiments, similar presets were used on the Raspberry Pi and due to limitations in hardware the resolution was needed to be capped at HD running at 30 FPS. Using the hardware accelerated capabilities on the Raspberry Pi would have made it possible to run at higher resolutions but with a more limited set of parameters to be tuned. A total of 16 latency test were performed varying quality and resolution settings while an additional 8 test were performed varying key-frame interval and simulating drop-probabilities on network packets. All these parameters used are described in appendix D. Parameters that were chosen to be tuned was the quantizer min and max value that is determined by the constant rate factor where the lower value will produce higher quality video at a higher bitrate. As mentioned in section 5.3 quantization added up to 43.2% of the encoding time consumption for AI mode for HEVC/H.265. The other parameter tuned was the key-int-max that determines the interval between keyframes for a complete decode refresh. This was useful for network instability situations where the motion predictions may cause error propagation in the sequence.

8.2 Quality

The initial test using the laptop was done in part to determine the quality loss for different bitrate modes for each of the video codec for a set of video sequences. Though a subjective quality experiment was the preferred method a simpler method using VMAF was chosen as a good substitute that isn't as time-consuming while is a good measurement of the perceived quality. The tool for determining the VMAF score is grading the differences between the original

and the compressed file between 0 and 100 where 100 represents no difference in quality.

8.3 Performance

The performance was measured by encoding video sequences and determining the average frames per seconds and that was achieved for each sequence and codec. This was performed using the "benchmark" option in FFmpeg when encoding the video sequence. This command displays real, system and user time of the CPU for an encoding process. Memory consumption is also displayed but was not used for this experiment when this was not deemed to be of significant importance. Each video sequence was encoded five times with the same settings and a median of these was calculated to provide more accurate performance results.

8.4 Latency measurement

Latency was only measured on the implemented system on the Raspberry Pi when streaming to the laptop. Though there exists more advanced and accurate methods as discussed in earlier section the method of using a simple stopwatch was determined to be good enough to notice if the latency will have a small or big impact on the user experience. To compensate some with the inaccuracy of the chosen method 6 different measurements were made and the median of these was calculated. Due to the limitation in processing power of the Raspberry Pi when using the software encoder only a limited set of parameters were possible to alter without increasing the complexity to a degree where the frame rate of the video sequence became lower than 30 FPS that's needed for a visually smooth video.

8.5 Perceived video quality

Due to a lack of observers, a good measurement of the perceived quality was not obtained. Though the VMAF tool is a quantifiable measurement tool that assesses the perceived quality it's not perfect for all cases and may give an inaccurate representation the differs from the users' opinion. It was however decided that this tool was good enough with the additional visual examination of the author. A more complicated test when implementing a packet loss in the video stream to evaluate the error concealment performance of a video was performed by visual inspection of a video stream when introducing a probability that a network packet is dropped. 10, 20 and 30% drop probabilities were tested with different key-frame-max values to determine how the x264 implementation handled network instability. A delimitation of the x264 implementation of the AVC/H.264 encoder is the lack of several key techniques for dealing with error concealment and error redundancy. This limited the testing to only deal with a number of p-frames between each i-frames. A value of 1 means that all frames are i-frames and may be decoded independent of each other and will

severely decrease the coding efficiency. This is however, a common approach in dealing with network instability to send a decoder refresh frame that may be independently decoded as mentioned in section 4.2.

8.6 Summary

The two parts of the experiment done to evaluate performance, quality and latency have been described and the methods used to perform the results were explained. Sub-optimal methods for latency and quality assessments were used due to their simplicity and good enough results for this proposed system.

9 Results

This section deals with the conclusions derived from the literature study and the experiments on the proposed system. The main part of the study resulted in a vast increase in the knowledge of the workings of different video coding techniques but only a limited part dealt with low latency over low-quality wireless networks.

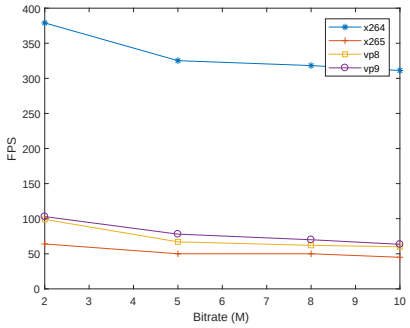
9.1 Results from Literature Analysis

While there existed many studies and scientific researches about video compression there was a limited set that dealt with close to zero latency solutions. Most compression techniques that are being developed are using predictions from previous and next frames. Though the previous frames don't add a delay the error propagation properties when decoding based on previous frames is a large issue when dealing with unstable network connections with a high probability of packet loss. The literature shows ways to deal with this issue and the improvements that may be achieved in visual quality by using different techniques. Where the compression techniques that utilize forward frames to improve the motion predictions adds entire frames in latency depending on the chosen number of frames. While this has a great effect on the compression efficiency achieved it may not be used when latency is a priority. A discussed in section 5.3, CBR is commonly used for real-time streaming and techniques for optimizing quality for a sequence of frames. Commonly used for encoding is a multi-pass CBR where the encoder analysis the entire sequence before determining optimized quality settings to match the chosen bitrate. Real-time encoding has to use a different technique that only looks a short window and the size of this window is a big contribution to the overall latency of the coding process as discussed in section 4.3. This technique only adds as much latency as the chosen size of the buffer there is however, a trade-off to consider where the coder will struggle to achieve the reference bitrate if the buffer for controlling this is too small, resulting in large variations in video quality. Performance wise the literature is hard to interpret for many reasons, one being that the reference software used as a tool for analyzing different techniques of the coding algorithm is unoptimized and the real-life implementation of the same algorithm may achieve results that vastly differs from the reference software. Another factor was the presets chosen that are not usable in a low-latency scenario. The consistent performance assessment was that among the four most popular coding algorithms, VP8, VP9, H.264 and H.265, H.264 achieved the highest performance results and was deemed most suitable for a low powered system. For a more powerful system or hardware solution, H.265 would be the best option where most of the features from H.264 and several more advanced techniques are also implemented improving the coding efficiency. Similar as with H.264 the limitations when dealing with a low latency system there can be no b-frames used for increasing the coding efficiency which reduces the compression difference between the two codecs. The largest issue with both these codecs is the error

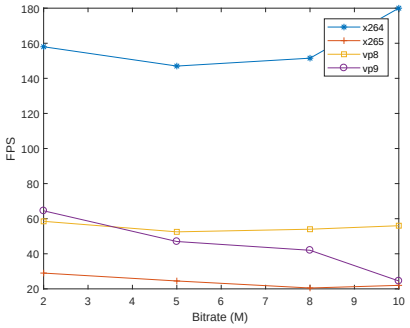
concealment and error correction properties where the current implementations lack tools to enable these techniques putting large limitations in the usages over an unstable network.

9.2 Coding Performance and Quality

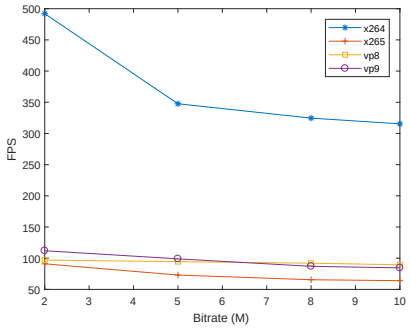
By analyzing the results from the experiments it was clear that x264 codec was far superior in terms of coding performance as is shown in the graphs of figure 5. This was to be expected from the results of the literature study where similar comparisons were made only this test was performed with low latency settings. The performance cap spanned between 3-5 times faster then VP9 which was the second fastest. The quality aspect that is shown in the graphs of figures 6, 7, 8 and 9 of the comparisons shows that x265 produces best quality for the same bitrate but with the slowest coding speed. Figure 7 shows that VP9 and especially x264 struggled to produce good quality when encoding a FHD sequence at 2M bitrate while unexpectedly VP8 performed a lot better and x265 produced the best quality. At 5M bitrate the quality was drastically improved for all codecs and even x264 produces quite a good quality result. For 8 and 10M bitrates the quality for all codecs are good but the performance gap between x264 and the others is big in favor of x264. As for the fast movement HD sequence in figure 6 it can be seen that x264 produced fairly good quality at 2 MB bitrate but falls far behind the best quality producing codec x265. For 5M bitrate and above the visual quality was good for all codecs and again the x264 produced good quality at a much better performance. The higher frame rate sequence in figure 8 with a lot of temporal information where the ducks wings moves fast and ripples in the water all codecs struggled at 2 and 5M bitrates and it was first at 8 and 10M bitrates where all but x264 started to achieve good quality video. Last sequence in figure 9 of a drone flying over a city in shows results similar to figure 6. The visual quality differences can be viewed in section A.



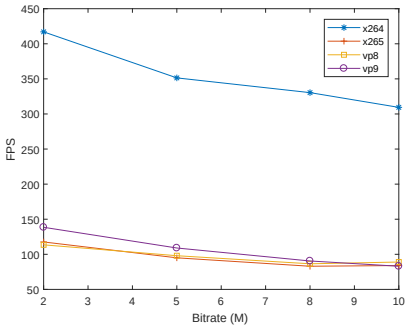
(a) Bike 720p.



(b) Bike 1080p.



(c) Ducks 720p.



(d) Drone 720p.

Figure 5: Performance comparisons for different bitrates.

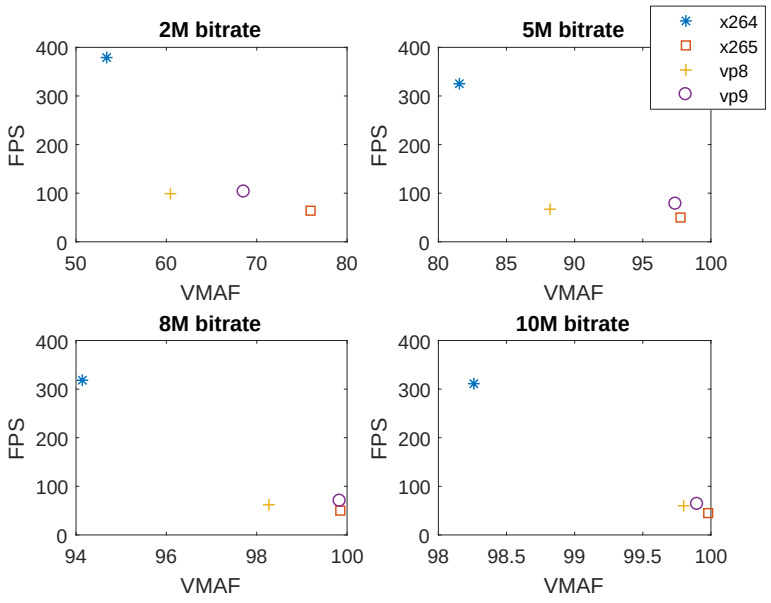


Figure 6: Bike sequence 720p performance vs quality graph.

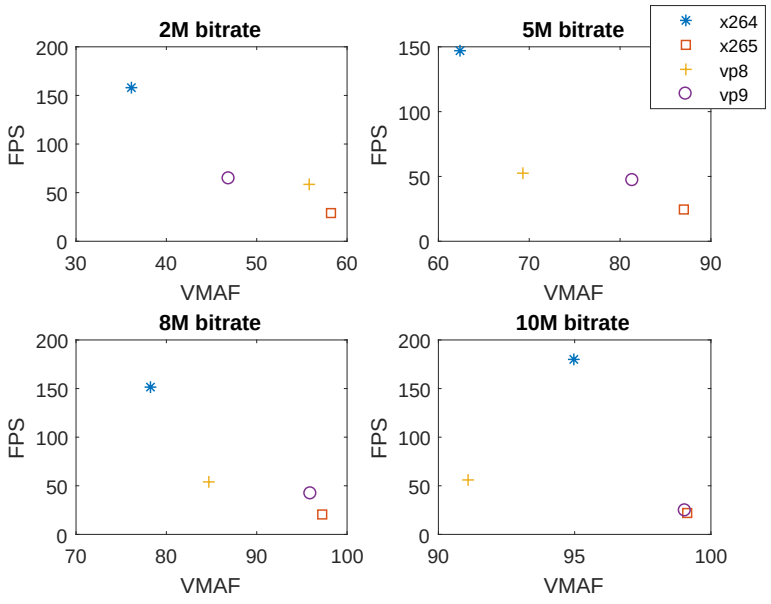


Figure 7: Bike sequence 1080p performance vs quality graph.

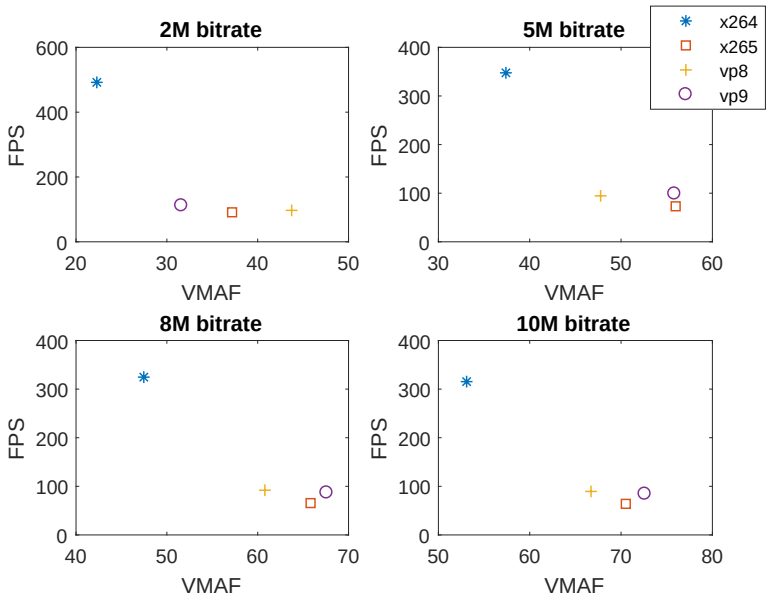


Figure 8: Duck sequence 720p performance vs quality graph.

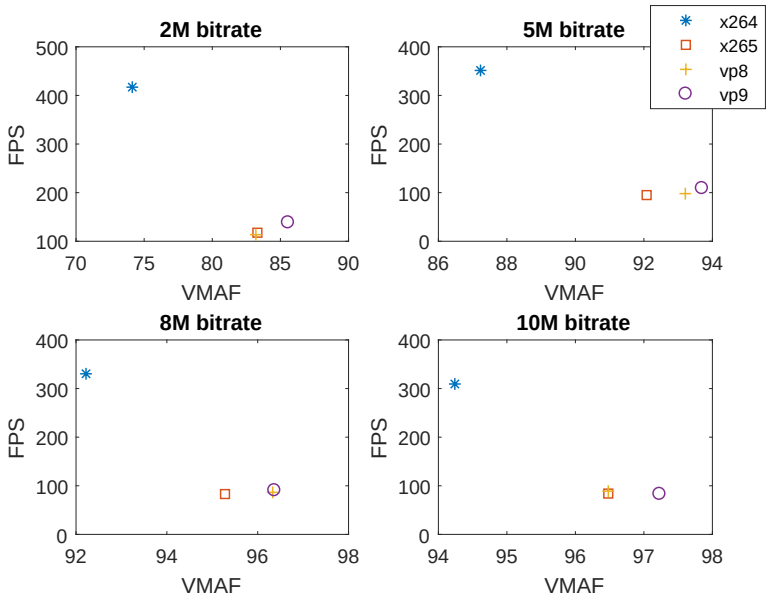


Figure 9: Drone sequence 720p performance vs quality graph.

9.3 Latency results

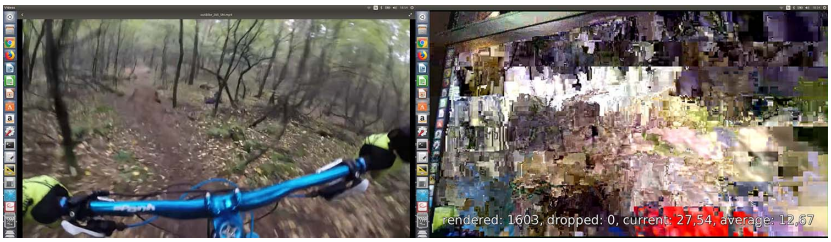
Varying the quantization parameter proved to have a low impact on the measured latency for the system where almost all measurements were around 100ms. The exception was the high latency measured with setting 6 where an aggressive quantization was used for high quality. This could be explained with that even low fidelity, flat regions of the image were being heavily quantized thus adding unnecessary complexity to both encoder and decoder. At the other end, a lower latency was measured for setting 16 where a lower resolution and higher frame rate was used. This is in line with the result in section 4.3 where the camera capture delay is related to the number of lines read by the sensor and the shutter speed that determines the cameras capture rate.

Test Setting	Latency (ms)
Setting 1	114
Setting 2	104
Setting 3	125
Setting 4	104
Setting 5	117
Setting 6	287
Setting 7	104
Setting 8	104
Setting 9	109
Setting 10	108
Setting 11	120
Setting 12	121
Setting 13	117
Setting 14	116
Setting 15	115
Setting 16	91

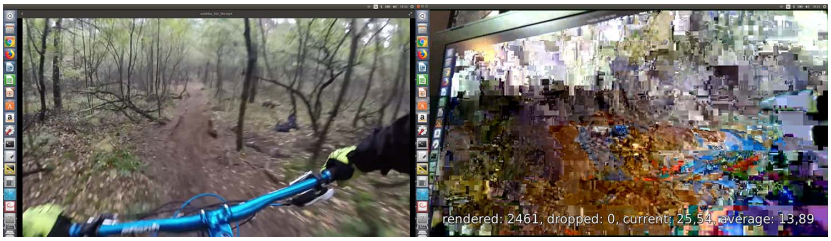
Table 3: Latency measurement for transmission of coded video stream

9.4 Network instability results

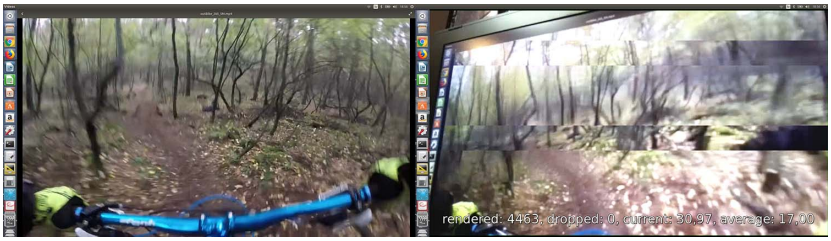
The network instability test with simulated network packet drop probabilities of 10, 20 and 30% showed that even at the lowest drop-rate a key interval above 30 frames i.e. one keyframe per second resulted in a non-distinguishable image where all moving parts of the frame were corrupted as is visible in figure 10a and 10b. When using AI mode it's possible to interpret the image while slices with missing packets are distinguishable but not corrupted and propagated as each frame is independently decoded as seen in figure 10c. The option sliced-threads that was used help with error propagation over an entire frame where each frame is divided into horizontal slices that are decoded independently. This can be seen in that entire rows are corrupted when attempting to decode a slice with missing packets but other rows are unaffected by these corruptions however, using motion prediction the error propagation is too great so all slices are corrupted using this technique.



(a) Key frame interval 60.



(b) Key frame interval 30.



(c) Key frame interval 1.

Figure 10: Packet drop probability 10%

9.5 Summary

Results from the literature study and experiments have been presented in this section. The results show that the complex problem of transmitting a video stream using high compression codec is a large field of study. The limitations of research made about real-time systems where close to zero latency is required are harder to find and most low-latency systems are made for the sub 1 second latency which is not of great interest for the proposed system implementation. Analyzing the study to determine the codecs performance and tools available to achieve good compression and quality showed that the AVC/H.264 algorithm was best for the low-performance system proposed. Performance and quality tests comparing the four most popular codecs confirmed this assessment where the x264 implementation of AVC/H.264 performed around $3-5$ times faster encoding rates than the others. Measurements on the system consisting of a Raspberry Pi 3B+ as a transmitter and a laptop as receiver resulted in an end-to-end latency of around 100ms using HD resolution at 30 FPS. The network instability testing showed that for as low as 10% packet drop probability a keyframe interval of 30 and above the video sequence became non-distinguishable and only for the test using AI mode the resulting video stream became watchable.

10 Discussion

A lot of research is currently being made improving compression techniques to further reduce the bandwidth requirements when transmitting and storing video. The most popular usage is by streaming video from a web player this only requires the video file to be encoded once and can thereafter be decoded multiple times. By 2021 82% of all IP traffic will be video according to [51] where presently live video consists of 3% of this usage, by 2021 live video is estimated to account for 13% of the video traffic. Although this is a major increase there might still be an unbalanced focus where the research in this area improves techniques that aren't optimal for low latency streaming. Even so, the usage of AVC/H.264 provides a low complexity fast performing tool to achieve decent video compression. With faster hardware which is delivered almost at a yearly basis it's not far from being able to use the HEVC/H.265 codec that delivers even better compression ratio even on cheap low powered solutions. By addressing the lack of error correction and error concealment in the current implementations of the codecs a much higher compression efficiency would be possible to achieve when adding p-frames that can be as much as 10 times smaller than i-frames. The main issue when using p-frames is the fast movements of the flying drone that will actually reduce the compression gains delivered from motion prediction techniques. More testing using these techniques would be preferable but due to the limitations on the Raspberry Pi, it was complex enough using the least complex settings on the x264 encoder and it was not possible to compile and run the unoptimized reference software provided by JVT. Choosing the right camera with a high refresh rate and shutter speed will improve the end-to-end latency for the system. As the case with using the Raspberry Pi camera module v2 and the Raspberry Pi 3B+ board the photons from the camera sensor needs to be converted first into RGB and then converted into YUV for the encoder to be able to perform the compression techniques. This would be much more efficient if all these conversions were performed even before sent from the camera and would simplify the system to only the package and transmission of the encoded video sequence to the receiver. Sadly it's not as easy as to purchase a camera that provides an encoded output, the right parameters need to be chosen to enable sub-frame encoding and decoding such as that parts of frames may be processed in parallel independently as was explained in section 4.3. The choice of the display will also affect the latency of the system where a refresh synchronization such as [52] or [53] that matches the frame rate of the display with the processing unit and eliminating situations where the processing using need to wait for the display refresh adding latency.

11 Future works

There are still no systems low powered, affordable systems that can achieve FHD resolution at near-zero latency. Hardware solutions can come very close but are expensive and hard to come by. It's proven that around 100ms can be achieved using cheap hardware and as technology progresses this will improve further. For a sub 100ms system a high rate camera with hardware encoder optimized for low latency is required. Further improvements to the video quality can be reached if more advanced error concealment and error correction methods are implemented to the x264 and x265 encoders allowing for more compression efficient motion estimation to be used. Much work can still be done in this area and the user experience could improve a lot with more advanced error handling. Little focus has been made on the wireless transport protocol where the WiFi controllers that existed on the Raspberry Pi and laptop were used. While delivering fast communication speeds this standard may add latency to the delivery if there is high traffic on the chosen channel. A larger issue is that when the connection is lost to be able to reconnect to the transmitter and receiver need to authenticate and wait for a reply before re-transmission is possible adding latency and a broken video stream. By implementing a more robust point-to-point communication protocol latency and connection interrupts can be addressed and optimized for the specific system. As was explained in section 4.1 simple alterations to the WiFi protocol is possible to allow for continuous data transmission even without a connection requirement.

References

[1] Jean-Marc Valin and Steinar Midtskogen. "The AV1 Constrained Directional Enhancement Filter (CDEF)". In: *CoRR* abs/1602.05975 (2017). arXiv: 1602.05975. URL: http://arxiv.org/abs/1602.05975v3.

[2] Raspberry Pi Foundation. *Raspberry Pi*. URL: https://www.raspberrypi.org/ (visited on 05/04/2018).

[3] GStramer. *GStreamer open source multimedia framework*. URL: https://gstreamer.freedesktop.org/ (visited on 06/07/2018).

[4] Shahriar Akramullah. *Digital Video Concepts, Methods and Metrics*. Apress, 2014. ISBN: 1430267135.

[5] Alan C. Bovik. *The Essential Guide to Image Processing*. Burlington: Elsevier, 2009. ISBN: 978-0-08-092251-5.

[6] D. A. Huffman. "A Method for the Construction of Minimum-Redundancy Codes". In: *Proceedings of the IRE* 40.9 (Sept. 1952), pp. 1098–1101. ISSN: 0096-8390. DOI: 10.1109/JRPROC.1952.273898.

[7] Khalid Sayood. *Introduction to data compression*. 3. ed. San Francisco, Calif.: Morgan Kaufmann, 2006. ISBN: 0-12-620862-X.

[8] Ee-Leng. Tan and Woon-Seng. Gan. *Perceptual Image Coding with Discrete Cosine Transform*. Singapore: Springer Singapore, 2015. ISBN: 978-981-287-543-3.

[9] K.R. Rao, Do Nyeon. Kim, and Jae Jeong. Hwang. *Video coding standards [Elektronisk resurs] : AVS China, H.264/MPEG-4 PART 10, HEVC, VP6, DIRAC and VC-1*. Dordrecht: Springer Netherlands, 2014. ISBN: 9789400767423.

[10] *VP6 Bitstream & Decoder Specification*. Document version 1.02. On2 Technologies, Inc. 2006.

[11] Tarek. Elarabi, Ahmed. Abdelgawad, and Magdy. Bayoumi. *Real-Time Heterogeneous Video Transcoding for Low-Power Applications*. Cham: Springer International Publishing, 2014. ISBN: 978-3-319-06071-2.

[12] MPEG. *MPEG-4 video standard, MPEG web site*. URL: https://mpeg.chiariglione.org/standards/mpeg-4 (visited on 04/23/2018).

[13] Youn-Long Steve. Lin et al. *VLSI Design for Video Coding : H.264/AVC Encoding from Standard Specification to Chip*. 1st. Boston, MA: Springer US, 2010. ISBN: 978-1-4419-0959-6.

[14] Xiaohua. Tian, Thinh M. Le, and Yong. Lian. *Entropy Coders of the H.264/AVC Standard : Algorithms and VLSI Architectures*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. ISBN: 978-3-642-14703-6.

[15] *Dirac Specification*. Version 2.2.3. BBC. Sept. 2008.

[16] Paul Wilkins et al. *VP8 Data Format and Decoding Guide*. RFC 6386. Nov. 2011. DOI: 10.17487/RFC6386. URL: https://rfc-editor.org/rfc/rfc6386.txt.

[17] Vivienne. Sze, Madhukar. Budagavi, and Gary J. Sullivan. *High Efficiency Video Coding (HEVC) : Algorithms and Architectures*. Cham: Springer International Publishing, 2014. ISBN: 978-3-319-06895-4.

[18] *High efficiency video coding, Recommendation ITU-T H.265*. Version 5. ITU. Feb. 2018. DOI: 11.1002/1000/13433. URL: http://handle.itu.int/11.1002/1000/13433.

[19] Adrian Grange, Peter de Rivaz, and Jonathan Hunt. *VP9 Bitstream & Decoding Process Specification*. version 0.6. Google inc. Mar. 2016.

[20] webM. *webM web site*. URL: https://www.webmproject.org/vp9 (visited on 04/27/2018).

[21] Peter de Rivaz and Jack Haughton. *AV1 Bitstream & Decoding Process Specification*. The Alliance for Open Media. Mar. 2018.

[22] AOM. *Alliance for Open Media web site*. URL: https://aomedia.org/av1-features/get-started/ (visited on 04/27/2018).

[23] AMIMON. *Connex Prosight*. URL: https://www.amimon.com/fpv-market/prosight-product-page-2/ (visited on 05/03/2018).

[24] DJI. *DJI GOGGLES RACING EDITION*. URL: https://www.dji.com/dji-goggles-re?site=brandsite&from=nav (visited on 05/03/2018).

[25] befinitiv. *Wifibroadcast – Analog-like transmission of live video data*. URL: https://befinitiv.wordpress.com/wifibroadcast-analog-like-transmission-of-live-video-data/ (visited on 05/03/2018).

[26] H. Schwarz, D. Marpe, and T. Wiegand. "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard". In: *IEEE Transactions on Circuits and Systems for Video Technology* 17.9 (Sept. 2007), pp. 1103–1120. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2007.905532.

[27] T. Stockhammer, M. M. Hannuksela, and T. Wiegand. "H.264/AVC in wireless environments". In: *IEEE Transactions on Circuits and Systems for Video Technology* 13.7 (July 2003), pp. 657–673. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2003.815167.

[28] D. Chen, N. Gadgil, and E. J. Delp. "VPx video coding for lossy transmission channels using error resilience packets". In: *2016 Picture Coding Symposium (PCS)*. Dec. 2016, pp. 1–5. DOI: 10.1109/PCS.2016.7906328.

[29] B. Li, T. Nanjundaswamy, and K. Rose. "An error-resilient video coding framework with soft reset and end-to-end distortion optimization". In: *2017 IEEE International Conference on Image Processing (ICIP)*. Sept. 2017, pp. 1910–1914. DOI: 10.1109/ICIP.2017.8296614.

[30] Weiwei Xu and Yaowu Chen. "Error resilience video coding parameters and mechanisms selection with End-to-End rate-distortion analysis at frame level". In: *Multimedia Tools and Applications* 75.4 (Feb. 2016), pp. 2347–2366. ISSN: 1573-7721. DOI: 10.1007/s11042-014-2409-0. URL: https://doi.org/10.1007/s11042-014-2409-0.

[31] Eun-Seok Ryu and SunJung Ryu. "Robust real-time UHD video streaming system using scalable high efficiency video coding". In: *Multimedia Tools and Applications* 76.23 (Dec. 2017), pp. 25511–25527. ISSN: 1573-7721. DOI: 10.1007/s11042-017-4835-2. URL: https://doi.org/10.1007/s11042-017-4835-2.

[32] Nikos Zervas. *White paper: Video Streaming with Near-Zero Latency Using Altera Arria V FPGAs and Video and Image Processing Suite Plus the Right Encoder*. Tech. rep. Cast, Inc., Jan. 2016, p. 10.

[33] Z. Lei et al. "GPGPU implementation of VP9 in-loop deblocking filter and improvements for AV1 codec". In: *2017 IEEE International Conference on Image Processing (ICIP)*. Sept. 2017, pp. 925–929. DOI: 10.1109/ICIP.2017.8296416.

[34] Dan the IOT man. *Using raspvid for low-latency Pi Zero W video streaming*. URL: https://dantheiotman.com/2017/08/23/using-raspivid-for-low-latency-pi-zero-w-video-streaming/ (visited on 05/16/2018).

[35] Ulf Jennehag, Stefan Forsstrom, and Federico V. Fiordigigli. "Low Delay Video Streaming on the Internet of Things Using Raspberry Pi." In: *ELECTRONICS* 5.3 (2016). ISSN: 20799292.

[36] Dennis Barrett. *Optimizing your video-enabled drone design*. URL: http://mil-embedded.com/articles/optimizing-video-enabled-drone-design/ (visited on 05/16/2018).

[37] M. Mody, P. Swami, and P. Shastry. "Ultra-low latency video codec for video conferencing". In: *2014 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*. Jan. 2014, pp. 1–5. DOI: 10.1109/CONECCT.2014.6740280.

[38] C. Bachhuber et al. "On the Minimization of Glass-to-Glass and Glass-to-Algorithm Delay in Video Communication". In: *IEEE Transactions on Multimedia* 20.1 (Jan. 2018), pp. 238–252. ISSN: 1520-9210. DOI: 10.1109/TMM.2017.2726189.

[39] Zhi Li et al. *Toward A Practical Perceptual Video Quality Metric*. URL: https://medium.com/netflix-techblog/toward-a-practical-perceptual-video-quality-metric-653f208b9652 (visited on 05/17/2018).

[40] T. Daede, A. Norkin, and I. Brailovskiy. *Video Codec Testing and Quality Measurement*. URL: https://tools.ietf.org/id/draft-ietf-netvc-testing-06.html (visited on 05/17/2018).

[41] N. Barman and M. G. Martini. "H.264/MPEG-AVC, H.265/MPEG-HEVC and VP9 codec comparison for live gaming video streaming". In: *2017 Ninth International Conference on Quality of Multimedia Experience (QoMEX)*. May 2017, pp. 1–6. DOI: 10.1109/QoMEX.2017.7965686.

[42] FFmpeg. *FFmpeg*. URL: https://www.ffmpeg.org/ (visited on 08/09/2018).

[43] Touradj Ebrahimi Martin Řeřábek. *Comparison of compression efficiency between HEVC/H.265 and VP9 based on subjective assessments*. 2014. DOI: 10.1117/12.2065561. URL: https://doi.org/10.1117/12.2065561.

[44] C. Feller et al. "The VP8 video codec - overview and comparison to H.264/AVC". In: *2011 IEEE International Conference on Consumer Electronics -Berlin (ICCE-Berlin)*. Sept. 2011, pp. 57–61. DOI: 10.1109/ICCE-Berlin.2011.6031852.

[45] Jan De Cock et al. *A large-scale video codec comparison of x264, x265 and libvpx for practical VOD applications*. 2016. DOI: 10.1117/12.2238495. URL: https://doi.org/10.1117/12.2238495.

[46] Aruna Ravi and K. R. Rao. "Performance Analysis and Comparison of the Dirac Video Codec with H.264/MPEG-4, Part 10". In: *Advances in Reasoning-Based Image Processing Intelligent Systems: Conventional and Intelligent Paradigms*. Ed. by Roumen Kountchev and Kazumi Nakamatsu. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 9–34. ISBN: 978-3-642-24693-7. DOI: 10.1007/978-3-642-24693-7_2. URL: https://doi.org/10.1007/978-3-642-24693-7_2.

[47] Dr. Dmitriy Vatolin et al. *Everything about the data compression*. URL: http://compression.ru/index_en.htm (visited on 05/18/2018).

[48] V. Deep and T. Elarabi. "HEVC/H.265 vs. VP9 state-of-the-art video coding comparison for HD and UHD applications". In: *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*. Apr. 2017, pp. 1–4. DOI: 10.1109/CCECE.2017.7946796.

[49] Guilherme. Correa et al. *Complexity-Aware High Efficiency Video Coding*. 1st ed. 2016. Cham: Springer International Publishing, 2016. ISBN: 978-3-319-25778-5.

[50] I. Kim et al. "Coding efficiency improvement of HEVC using asymmetric motion partitioning". In: *IEEE international Symposium on Broadband Multimedia Systems and Broadcasting*. June 2012, pp. 1–4. DOI: 10.1109/BMSB.2012.6264283.

[51] Cisco public. *White paper: The Zettabyte Era: Trends and Analysis*. Tech. rep. Cisco., June 2017, p. 32.

[52] NVidia. *G-Sync*. URL: https://www.geforce.com/hardware/technology/g-sync/technology (visited on 08/20/2018).

[53] AMD. *Radeon FreeSync*. URL: https://www.amd.com/en/technologies/free-sync (visited on 08/20/2018).

[54] MeGui. *x264 parameters explained*. URL: https://en.wikibooks.org/wiki/MeGUI/x264_Settings (visited on 08/20/2018).

[55] x265. *x265 parameters explained*. URL: https://x265.readthedocs.io/en/default/cli.html (visited on 08/20/2018).

[56] Webm. *vpx parameters explained*. URL: `https://www.webmproject.org/docs/encoder-parameters/` (visited on 08/20/2018).

Appendices

A Visual quality comparison for different bitrates and codecs

Figure 11: Bike sequence 720p original file.

(a) H.264.

(b) H.265.

(c) VP8.

(d) VP9.

Figure 12: Bike sequence 720p 2Mbps.

(a) H.264.

(b) H.265.

(c) VP8.

(d) VP9.

Figure 13: Bike sequence 720p 5Mbps.

(a) H.264.

(b) H.265.

(c) VP8.

(d) VP9.

Figure 14: Bike sequence 720p 8Mbps.

(a) H.264.

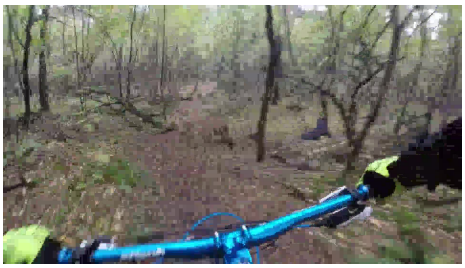
(b) H.265

(c) VP8.

(d) VP9.

Figure 15: Bike sequence 720p 10Mbps.

Figure 16: Bike sequence 1080p original file.



(a) H.264.

(b) H.265.

(c) VP8.

(d) VP9.

Figure 17: Bike sequence 1080p 2Mbps.

(a) H.264.

(b) H.265.

(c) VP8.

(d) VP9.

Figure 18: Bike sequence 1080p 5Mbps.

(a) H.264.

(b) H.265.

(c) VP8.

(d) VP9.

Figure 19: Bike sequence 1080p 8Mbps.

(a) H.264. (b) H.265.

(c) VP8. (d) VP9.

Figure 20: Bike sequence 1080p 10Mbps.

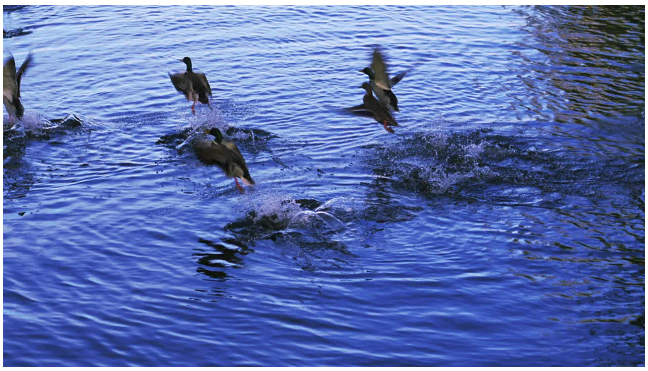


Figure 21: Duck sequence 720p original file.

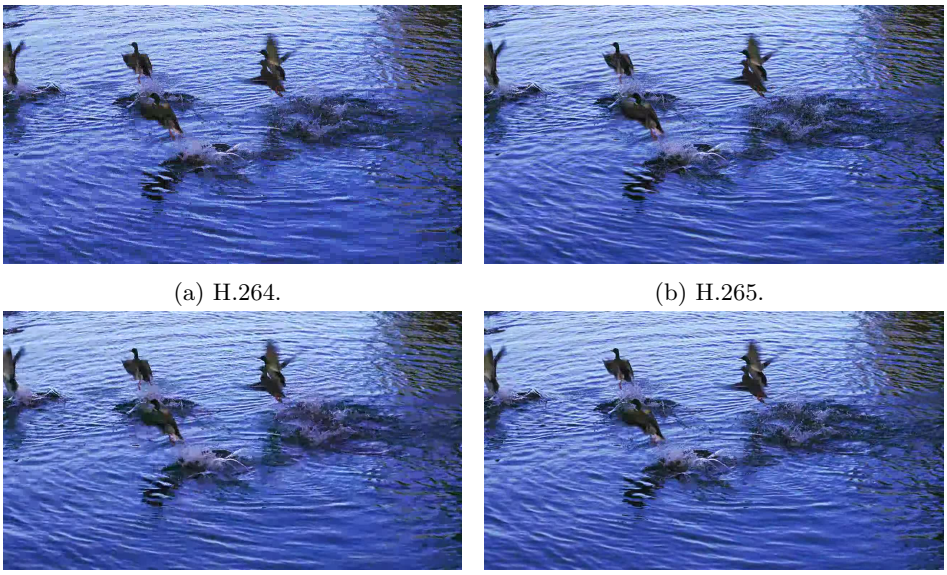
(a) H.264.

(b) H.265.

(c) VP8.

(d) VP9.

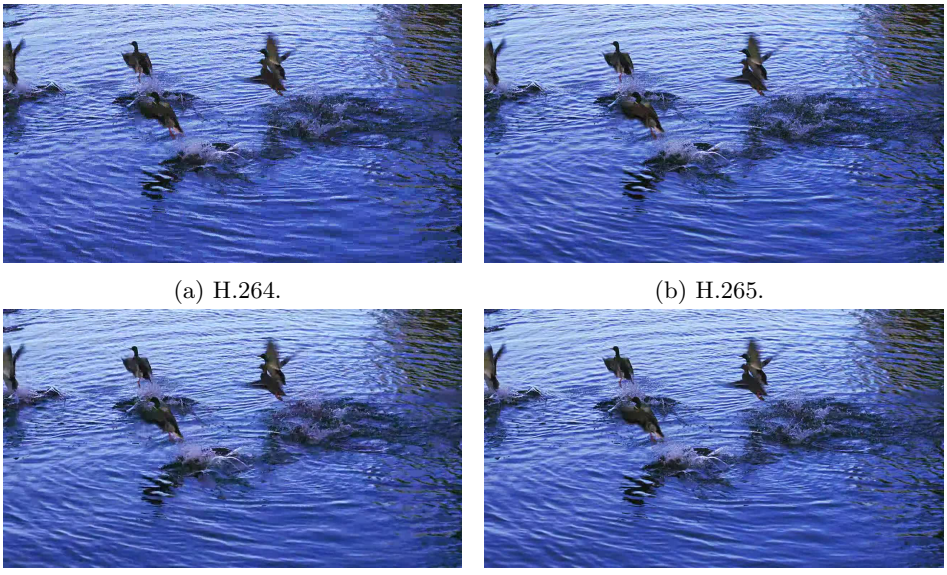
Figure 22: Duck sequence 720p 2Mbps.



(a) H.264. (b) H.265.

(c) VP8. (d) VP9.

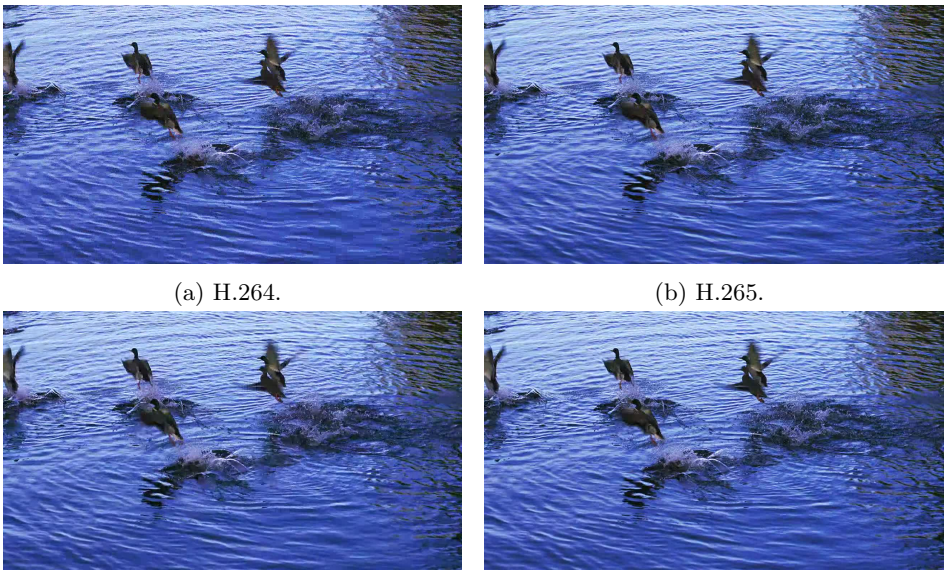
Figure 23: Duck sequence 720p 5Mbps.



(a) H.264. (b) H.265.

(c) VP8. (d) VP9.

Figure 24: Duck sequence 720p 8Mbps.



(a) H.264. (b) H.265.

(c) VP8. (d) VP9.

Figure 25: Duck sequence 720p 10Mbps.

Figure 26: Bike sequence 1080p original file.

(a) H.264.

(b) H.265.

(c) VP8.

(d) VP9.

Figure 27: Bike sequence 1080p 2Mbps.

(a) H.264.

(b) H.265.

(c) VP8.

(d) VP9.

Figure 28: Bike sequence 1080p 5Mbps.

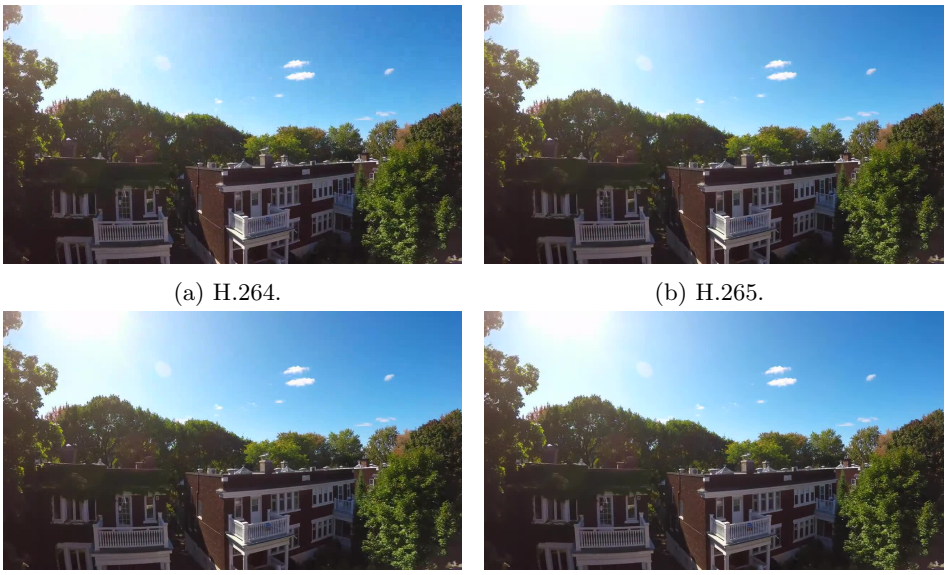
(a) H.264.

(b) H.265.

(c) VP8.

(d) VP9.

Figure 29: Bike sequence 1080p 8Mbps.

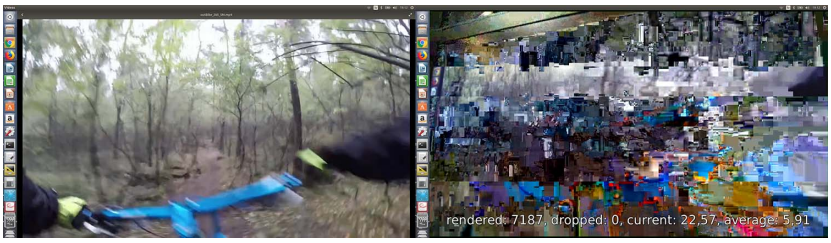


(a) H.264. (b) H.265.

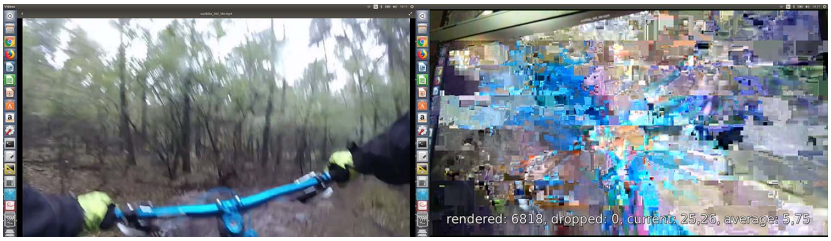
(c) VP8. (d) VP9.

Figure 30: Bike sequence 1080p 10Mbps.

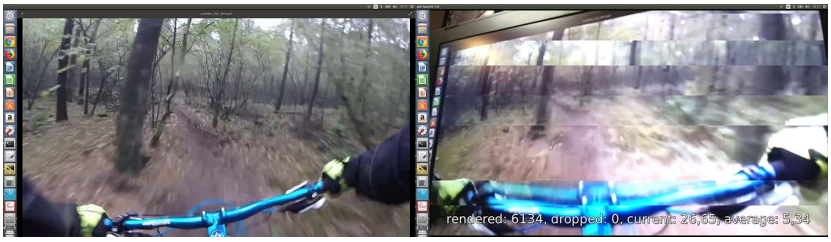
B Network instability simulation



(a) Key frame interval 60.



(b) Key frame interval 30.



(c) Key frame interval 1.

Figure 31: Packet drop probability 20%

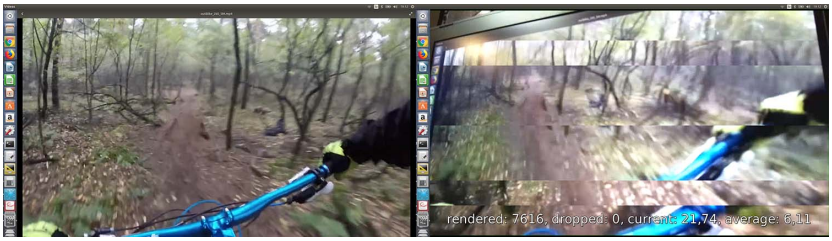


Figure 32: Packet drop probability 30%, Key frame interval 1

C Preset options for video codecs

Presets were chosen to limit the complexity of each encoder and removing filters and forward prediction frames that adds latency to the encoding. Each of the encoder preset options are listed in the following subsections while for more explanations about the different options a link to the encoders documentations.

C.1 x264

Using preset option "ultrafast" and tune option "zerolatency" along with bitrate settings for constant 10M bitrate the options in table 4 was used. The different parameters are explained in [54].

Option	Value
cabac	0
ref	1
deblock	0 : 0 : 0
analyse	0 : 0
me	dia
subme	0
psy	1
psy_rd	1.00 : 0.00
mixed_ref	0
me_range	16
chroma_me	1
trellis	1
8x8dct	0
cqm	0
deadzone	21, 11
fast_pskip	1
chroma_qp_offset	0
threads	3
lookahead_threads	3
sliced_threads	1
slices	3
nr	0
decimate	1
interlaced	0
bluray_compat	0
constrained_intra	0
bframes	0
weightp	0
keyint	250
keyint_min	25
scenecut	0
intra_refresh	0
rc_lookahead	0
rc	cbr
mbtree	0
bitrate	10000
rateol	1.0
qcomp	0.60
qpmin	0
qpmax	69
qpstep	4
vbv_maxrate	10000
vbv_bufsize	10000
nal_hrd	none
filler	0
ip_ratio	1.40
aq	0

Table 4: Options for the Ultrafast preset and Zerolatency tune used for x264 encoder.

C.2 x265

Using same preset option "ultrafast" and tune "zerolatency" as the x264 encoder the resulting options for the x265 encoder can be seen in table 5. The different parameters are explained in [55].

Option	Value
ctu	32
min-cu-size	16
bframes	0
b-adapt	0
rc-lookahead	0
lookahead-slices	8
scenecut	0
ref	1
limit-refs	0
me	dia
merange	57
subme	0
rect	0
amp	0
limit-modes	0
max-merge	2
early-skip	1
recursion-skip	1
fast-intra	1
b-intra	1
sao	0
signhide	0
weightp	0
weightb	0
aq-mode	0
cuTree	0
rdLevel	2
rdoq-level	0
tu-intra	1
tu-inter	1
limit-tu	0
frame-threads	1
rd	2
psy-rd	2.00
tmvp	1
strong-intra-smoothing	1
deblock	1

Table 5: Options for the Ultrafast preset and zerolatency tune used for x265 encoder.

C.3 VP8 & VP9

These two encoders are very similar in parameters used so both and the documentation isn't as explanatory as for x264-x265 encoders. The settings used for the encoders were "-deadline realtime" and "-cpu-used 8" which are explained in [56] and were chosen to produce a good quality vs. speed trade-off.

D GStreamer x264 parameters

Parameters used for the GStreamer pipeline, mainly focusing on the encoder options not the RTP packaging and queue buffers between these.

Setting 1 width=1280, height=720, framerate=30/1, x264enc speed-preset=ultrafast tune=zerolatency threads=8 bitrate=10000 qp-min=15 qp-max=40

Setting 2 width=1280, height=720, framerate=30/1, x264enc speed-preset=ultrafast tune=zerolatency threads=8 bitrate=10000 qp-min=15 qp-max=30

Setting 3 width=1280, height=720, framerate=30/1, x264enc speed-preset=ultrafast tune=zerolatency threads=8 bitrate=10000 qp-min=15 qp-max=20

Setting 4 width=1280, height=720, framerate=30/1, x264enc speed-preset=ultrafast tune=zerolatency threads=8 bitrate=10000 qp-min=8 qp-max=40

Setting 5 width=1280, height=720, framerate=30/1, x264enc speed-preset=ultrafast tune=zerolatency threads=8 bitrate=10000 qp-min=8 qp-max=30

Setting 6 width=1280, height=720, framerate=30/1, x264enc speed-preset=ultrafast tune=zerolatency threads=8 bitrate=10000 qp-min=8 qp-max=20

Setting 7 width=1280, height=720, framerate=30/1, x264enc speed-preset=ultrafast tune=zerolatency threads=8 bitrate=10000 qp-min=25 qp-max=40

Setting 8 width=1280, height=720, framerate=30/1, x264enc speed-preset=ultrafast tune=zerolatency threads=8 bitrate=10000 qp-min=25 qp-max=30

Setting 9 width=1280, height=720, framerate=30/1, x264enc speed-preset=ultrafast tune=zerolatency threads=8 bitrate=10000 qp-min=25 qp-max=51

Setting 10 width=1280, height=720, framerate=30/1, x264enc speed-preset=ultrafast tune=zerolatency threads=8 bitrate=10000 qp-min=15 qp-max=51

Setting 11 width=1280, height=720, framerate=30/1, x264enc speed-preset=ultrafast tune=zerolatency threads=8 bitrate=10000 qp-min=8 qp-max=51

Setting 12 width=1280, height=720, framerate=30/1, x264enc speed-preset=ultrafast tune=zerolatency threads=8 bitrate=10000 qp-min=15 qp-max=51 key-int-max=1

Setting 13 width=1280, height=720, framerate=30/1, x264enc speed-preset=ultrafast tune=zerolatency threads=8 bitrate=10000 qp-min=15 qp-max=51 key-int-max=30

Setting 14 width=1280, height=720, framerate=30/1, x264enc speed-preset=ultrafast tune=zerolatency threads=8 bitrate=10000 qp-min=15 qp-max=51 key-int-max=60

Setting 15 width=640, height=480, framerate=30/1, x264enc speed-preset=ultrafast tune=zerolatency threads=8 bitrate=10000 qp-min=15 qp-max=51 key-int-max=60

Setting 16 width=1280, height=720, framerate=60/1, x264enc speed-preset=ultrafast tune=zerolatency threads=8 bitrate=10000 qp-min=15 qp-max=51 key-int-max=60