FEDERAL UNIVERSITY OF UBERLÂNDIA
ELECTRICAL ENGINEERING FACULTY – CAMPUS PATOS DE MINAS
ELECTRONICS AND TELECOMMUNICATIONS ENGINEERING

DANYEL GUSTAVO GOMES DOS REIS

AN ANALYSIS OF AV1 CODEC EFFICIENCY AND FIDELITY

Patos de Minas, MG
2019

DANYEL GUSTAVO GOMES DOS REIS

AN ANALYSIS OF AV1 CODEC EFFICIENCY AND FIDELITY

Undergraduate thesis submitted in partial fulfillment of the requirements for the major of Electronics and Telecommunications Engineering of the Electrical Engineering Faculty of the Federal University of Uberlândia – Campus Patos de Minas.

Adviser: MsC. Gustavo Nozella Rocha

Patos de Minas, MG
2019

DANYEL GUSTAVO GOMES DOS REIS

AN ANALYSIS OF AV1 CODEC EFFICIENCY AND FIDELITY

Undergraduate thesis submitted in partial fulfillment of the requirements for the major of Electronics and Telecommunications Engineering of the Electrical Engineering Faculty of the Federal University of Uberlândia – Campus Patos de Minas.

Adviser: MsC. Gustavo Nozella Rocha

Patos de Minas June 10, 2019

Evaluation Board

MSc. Gustavo Nozella Rocha (Adviser)

Dr. Karine Barbosa Carbonaro (1st member)

Dr. Pedro Luiz Lima Bertarini (2nd member)

ABSTRACT

Video has become a part of our daily lives, being present on TV, movies, online streaming, entertainment, news, our memories... Throughout the years, new and more efficient methods to store and share video have been developed. They all have the same goal: deliver the best quality as effortlessly as possible. This study aims to study the elements that make a video file, as well as the methods used to compress it. After describing the characteristics of the currently most used codecs (AVC, VP9, HEVC, and AV1), a comparison of their efficiency will be made. Every test will be run multiple times to guarantee reliability.

Key words: AVC, VP9, HEVC, AV1, codec

RESUMO

Vídeo se tornou uma parte do nosso cotidiano, estando presente na televisão, filmes, *streaming online*, entretenimento, notícias, memórias... Ao longo dos anos, novos e mais eficientes métodos para armazenar e distribuir vídeo foram desenvolvidos. Todos com o mesmo objetivo: entregar a máxima qualidade de imagem com a maior portabilidade possível. O objetivo deste trabalho é estudar os elementos que constituem um arquivo de vídeo, assim como as técnicas utilizadas para sua compressão. Após descrever as características de cada um dos principais *codecs* utilizados atualmente (AVC, VP9, HEVC e AV1), é feita uma comparação da sua eficiência. Todos os testes serão realizados múltiplas vezes para garantir um resultado preciso.

Palavras chave: AVC, VP9, HEVC, AV1, *codec*

FIGURE LIST

TABLE LIST

ABBREVIATIONS AND ACRONYMS LIST

4K	Video file with 2160p resolution
ADST	Asymmetrical Discrete Sine Transform
ALF	Adaptive Loop Filtering
AOMedia	Alliance for Open Media
AV1	AOMedia Video 1
AVC	Advanced Video Coding
AVCHD	Advanced Video Coding High Definition
AVI	Audio Video Interleave
BT.2020	ITU-R Recommendation BT.2020
CRT	Cathode Ray Tube
CTB	Coding Tree Block
DCT	Discrete Cosine Transform
DST	Discrete Sine Transform
DVD	Digital Video Disc
FLV	Flash Video
Full HD	Video file with 1080p resolution
GUI	Graphics User Interface
H.264	AVC
H.265	HEVC
HD	High Definition
HDR	High Dynamic Range
HEVC	High Efficiency Video Coding
IDTX	Inverse transform rows with identity and columns with identity
ISO/IEC	International Organization for Standardization/International Electrotechnical Commission
ITU-T	International Telecommunication Union
JPG	Joint Photographic Experts Group
LTS	Long Term Support
MCP	Motion-Compensated Prediction
MKV	Matroska Video Files
MOV	Apple QuickTime Movie
MP3	MPEG-1 Audio Layer III

MP4	MPEG-4 Part 14
MPEG	Moving Picture Experts Group
MPEG LA	Moving Picture Experts Group Licensing Authority
MPEG-1	Moving Picture Experts Group Part 1
MPEG-2	Moving Picture Experts Group Part 2
MSE	Mean Squared Error
MTS	Advanced Video Coding High Definition
NTSC	National Television System Committee
PAL	Phase Alternating Line
PC	Personal computer
Pixel	Picture Element
PSNR	Peak Signal to Noise Ratio
QP	Quantization Parameter
RAM	Random Access Memory
RGB	Red-Green-Blue color space
SAO	Sample Adaptive Offset
SB	Superblock
SSIM	Structural SIMilarity
TV	Television
UHD	Ultra High Definition
VP8	Video Project 8
VP9	Video Project 9
WHT	Walsh-Hadamard Transform
WMV	Windows Media Video
X264	H.264 free and open-source software library
X265	H.265 free and open-source software library
Y4M	YUV4MPEG2 format
$Y'C_BC_R$	Luma-Chroma color space

SYMBOLS LIST

Bit	Smallest unit of information
Byte	8 *bits*
dB	Decibel
Gbps	*Gigabit* per second
Mbps	*Megabit* per second
MB/s	*Megabyte* per second
MP	*Megapixel*

TABLE OF CONTENTS

CHAPTER 1 – INTRODUCTION

The creation, storage, and transmission of video files has become part of the everyday person's routine. That becomes clear observing the growth of this technology through services such as YouTube and Netflix. In 2006, video file traffic accounted for 12% of everything sent on the internet. This number continued to grow, until in 2010, when it became the main kind of traffic. In 2016, this number was as high as 79% and could reach 90% by 2020 (Qz, 2016).

At the same time as media consumption grows, so does the search for more efficient ways to transmit it. Terminology such as HD is no longer esoteric and has become common when referring to cable and online streaming. An uncompressed Full HD video file has a bit rate of up to 1.5Gbps, which would make its transmission pretty much impossible. Aiming to reduce file sizes, video compression is utilized.

During the process of compression, there are two steps: encoding and decoding. The former refers to converting the video file to a compressed format, thus reducing its size, while the latter consists of recovering the original file so it can be reproduced. During the encoding, each frame of the video is divided into blocks and a prediction service based on the Discrete Cosine Transform (DCT) is used to remove information while minimizing quality loss. To decode a video file, the software does the reverse process, reconstructing each frame from the data received (Vcodex, 2007).

For this process, it's necessary to use a codec to encode the video. Several solutions have been developed throughout the years, such as H.264 (AVC). The spec was finalized in 2003, and it opened doors to mass distribution of high-quality video files at a reduced size.

Over a decade later, the next-generation alternative to AVC, H.265, was created. Also known as HEVC, its first spec was finalized in 2013, and it has been updated ever since. Initial tests made by Qualcomm showed a file size reduction of 40 to 45 percent compared to AVC, while keeping the same visual fidelity. However, this comes at the expense of processing power, as HEVC is much more demanding (Cnet, 2012).

Another drawback is that HEVC is closed source; in other words, it's necessary to pay a licensing fee to use it. These royalties are also complex and divided across four different conglomerates (Ozer, 2018).

Hoping to resolve this situation, several groups, including Google, Microsoft, Mozilla, Netflix, Amazon, Cisco, and Intel, have united to create the Alliance for Open Media (AOMedia) and came up with an open source alternative to HEVC. This project, named AV1 (AOMedia Video 1), is still in development, but it shows promising results. Throughout this

paper, its principles, as well as its advantages and drawbacks compared to competing video codecs, will be discussed.

1.1 OBJECTIVES

1.1.1 Main Objective

Aiming to show the advantages of AV1, the new open source codec developed by AOMedia, this paper compares its efficiency and performance with the most popular options available today: AVC, VP9, and HEVC.

1.1.2 Detailed Objectives

- Study the principles of video encoding and decoding;
- Study the origin and development of the AV1 codec;
- Analyze the features and principles of the most popular codecs currently used;
- Compare their efficiency and visual fidelity;
- Compare their performance in difference scenarios;
- Show the advantages of an open source codec.

1.2 JUSTIFICATIONS

There is a clear necessity for better means of transmission. In all scenarios, a robust codec offers a series of advantages, from reduced storage needs and network traffic to better user experience, thanks to improved visual quality, not to mention legal matters due to simplified licensing terms and royalties. Given the boom in video distribution, it is no surprise that so much effort has been put into the development of next generation codecs, and AV1 is a very promising candidate.

1.6 FINAL NOTES

On Chapter two, the concept of video and its structure, the elements that make up a video file, and the restrictions imposed by its transmission will be studied. Additionally, the principles on which video compression is based, as well as the improvements introduced by

new codecs over time will be discussed. Finally, an in-depth analysis of AV1 and what it offers compared to the competition will be provided.

On Chapter three, the methods used to encode the uncompressed samples and the tools used will be described. Moreover, the metrics used to compare them will be explained.

On Chapter four, the results obtained will be discussed and the conclusions drawn from them, elaborated.

CHAPTER 2 – VIDEO CODECS

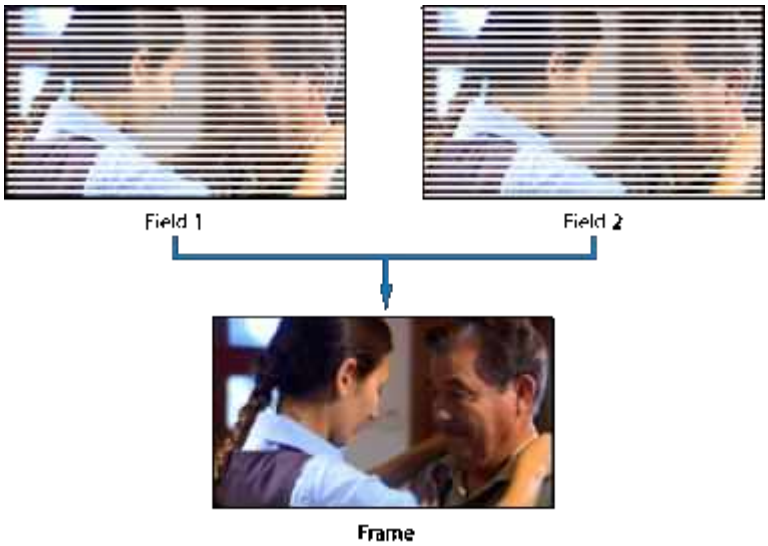
In order to understand how the AV1 codec works and its advantages, first it is necessary to study what the characteristics of a video file are and how a codec works.

2.1 VIDEO CHARACTERISTICS

A video file consists of a medium through which it is possible to record, copy, reproduce or transmit a sequence of images (frames) accompanied or not by audio. The technology was initially developed for CRT (Cathode Ray Tube) TVs, and it has been improved ever since (HiDef Audio and Video Sales and Installation, 2015).

Initially, the transmission was made by means of interlaced video. In this type of transmission, only half the lines of each frame are reproduced at once, alternating between even and odd lines, as seen in Figure 1. This was necessary to reduce flickering (abrupt changes of brightness caused by a low refresh rate). Alternating between even and odd lines allows doubling of the refresh rate without increasing the bandwidth. For old TV transmissions, each field was refreshed 15 times per second (HiDef Audio and Video Sales and Installation, 2015).

Figure 1 – Odd and even fields in interlaced video



Source: Apple (2010)

With digital video, such tricks are not necessary, and progressive video became standard. For this type of transmission, all lines are reproduced at the same time (HiDef Audio and Video Sales and Installation, 2015).

Each frame is composed by pixels. They are the smallest units of information in an image. The word *pixel* means Picture Element. By and large, the more pixels there are, the more details are stored. *Resolution* refers to how many pixels the image is composed of. If an image is said to have a resolution of 800x600, that means each line has 800 pixels and each column has 600 pixels, or 480,000 pixels in total. Digital cameras usually adopt MP (Megapixels) as a unit. A 12MP camera can shoot images with 12 million pixels (Ultimate Photo Tips, 2010).

The same concept goes for video. One of the most important characteristics of a video file is the frame size, that is, the resolution of each frame. For example, each frame in a Full HD video file has 1920 horizontal pixels and 1080 vertical pixels, or 1080p (1080 lines of progressive video) (Harrington & Krogh, 2012).

The aspect ratio is the ratio between the number of horizontal and vertical pixels. The first TV transmissions were made in a 4x3 aspect ratio. Nowadays, the 16x9 format is widely used by HD transmissions, including HD (1280x720), Full HD (1920x1080), or even 4K (3840x2160). Movies, however, tend to follow the 2.35:1 ratio (Harrington & Krogh, 2015).

The frame rate equals how many frames are shown every second during reproduction. Even though we perceive video as moving, it consists of a sequence of static images in rapid succession. This is possible because our brain can fill the gaps between them, giving the impression of movement. For a smooth transition, it is necessary to have a frame rate of at least 24 frames per second. Movies make use of 24 frames per second while TV uses either 24 (PAL) or 30 (NTSC) frames per second. Slow motion video ranges from 120 to thousands or even millions of frames per second (Harrington & Krogh, 2015).

The bit rate of a video file tells how much information is stored per second. Everything else kept the same, the higher the bit rate, the better the image quality (Harrington & Krogh, 2012). Another factor that has a big impact on the bit rate is color depth, which refers to how many bits of information are used to represent each pixel. In the RGB standard, each color component is represented by 8 bits, for a total of 24 bits per pixel. HDR video, though, often uses 10 or more bits for each component. The bit rate is calculated by Equation 1 (IPFS, 2017).

$$Bitrate = colordepth \times verticalresolution \times horizontalresolution \times framerate \quad (1)$$

2.1.1 Container and Video Codec

The main goal of video compression is to allow the transmission of video at the smallest bit rate without visible loss to image quality. An uncompressed 1080p video has a bit rate of 1.39Gbps (24 bits x 1920 pixels x 1080 pixels x 30 frames per second) or 180MB/s. Such a value would render transmission near impossible. To avoid this issue, compression techniques are employed to greatly reduce file size, minimizing the quality loss.

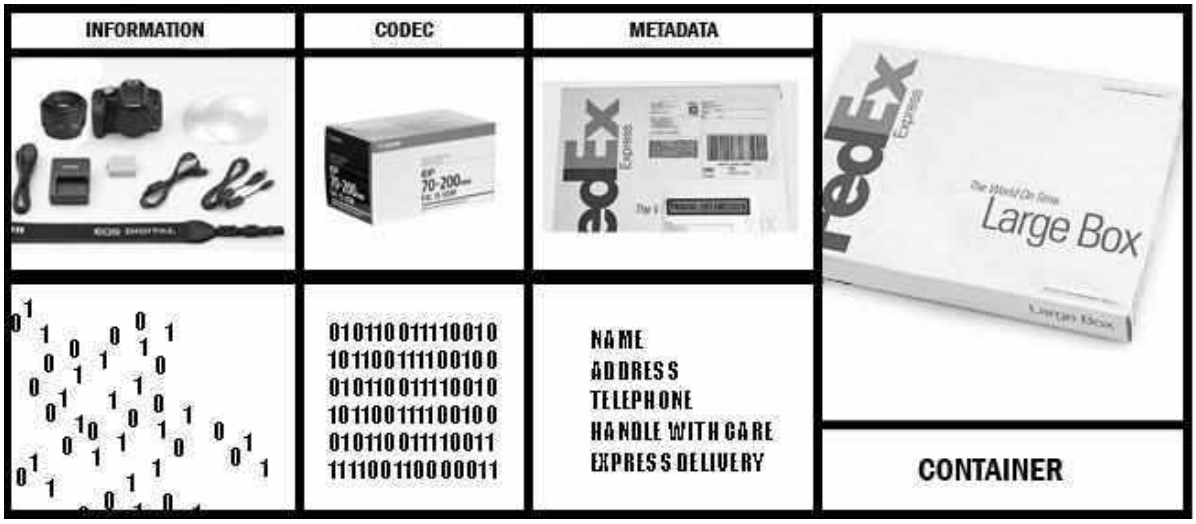
Before diving into details, it is important to distinguish container and codec. For an OS open and understand an image, document or song, it is necessary for it to know what the file extension is. This allows the right software to be associated with the right files. The container corresponds to what file extension the video is saved as (Wolfcrow, 2013).

On the other hand, the codec is the algorithm that was used to encode the video. The word *codec* comes from *coder* and *decoder*. In other words, it is what allows for both compression and reproduction of the data stored (Fitzer, 2009).

The specifications of a video file (frame size, aspect ratio, frame rate, and bit rate) are saved as metadata. This information is used to decide which codec will handle the decoding process (Wolfcrow, 2013).

Figure 2 shows the differences among information, codec, metadata, and container. The information (captured video) is the gift; the codec organizes this information in a comprehensive way in order to save space; the metadata works as the label, specifying where to send the package and how it should be handled; and the package is the container, allowing the content to be stored and transported (Wolfcrow, 2013).

Figure 2 – Representation of the information, codec, metadata, and container



Source: Wolfcrow (2013)

There are dozens of different containers in use nowadays. The most popular ones are .MP4, .MOV, .MTS, and .MKV. Other examples such as .FLV, .AVI, and .WMV have fallen into disuse.

MPEG-4 (.MP4) has become extremely popular to share videos online. Introduced in 1998, it usually uses AVC for video encoding and AAC for audio. The combination of the two allows the transmission of high-quality video files at a comparatively small size (Motion Elements, 2013).

Apple's QuickTime Movie (.MOV) was developed by Apple so that it would be the default format for its ecosystem. Initially, it was only supported on Apple devices, but after QuickTime was also released for PCs, the format became universal. It offers great image quality at the expense of large files (Cpapciak, 2012). Until recently, the video encoding was done using AVC until it was updated to HEVC.

AVCHD (.MTS) was developed by Sony in association with Panasonic for movies and video recordings. Files in this format are usually too big to be shared, and they require lots of processing power to be decoded, making its usage restrictive (Cpapciak, 2012). One of its uses is in Blu-ray discs. Even though MPEG-2 was first used for HD video encoding, it's not optimized for high bit rates. Thus, MPEG-4 AVC was adopted as the standard. For comparison, Blu-ray movies can be up to 26.7Mbps. The audio encoding is done with AC-3 (CoolUtils, 2009).

Matroska Video File (.MKV) was created in Russia in 2002 by Matroska, aiming to be the container of the future. A lot of features were implemented, such as fast skipping, menus and chapter support, multiple audio and video channels, online streaming compatibility, subtitles, and error correction. Furthermore, it's an open source project (Matroska, 2016).

2.2 VIDEO COMPRESSION

This paper aims to compare the image quality of different codecs. This becomes problematic because the perception of what looks better is subjective. One of the factors that changes that perception is how clearly the subjects can be seen. The observer's opinions might change whether they are focusing on a particular area or the picture as a whole (Richardson, 2003).

To avoid that issue, objective metrics must be used, as they yield precise and consistent results. One of the most used is the Peak Signal to Noise Ratio (PSNR), obtained through the

Mean Squared Error (MSE). The MSE is calculated by Equation 2, where *n* is the number of data points and *y* is the vector observed values (Richardson, 2003).

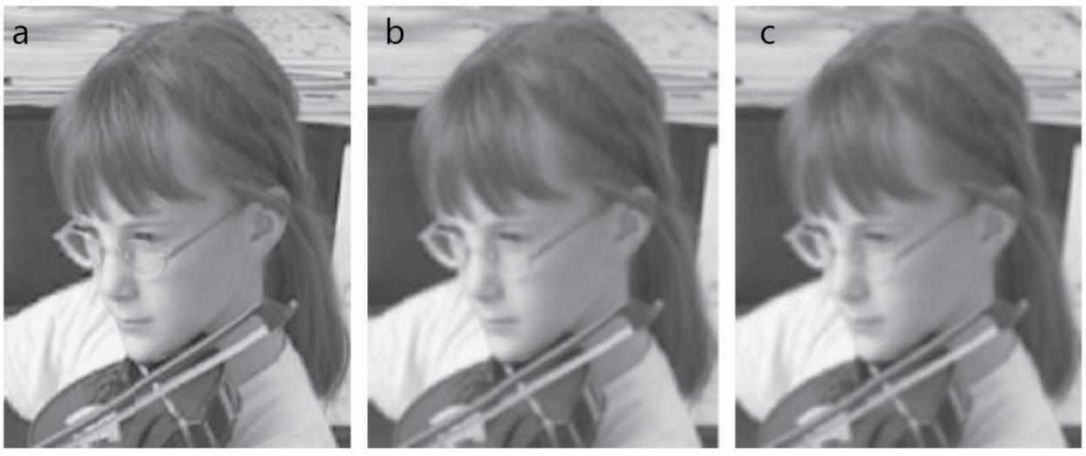
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - y'_i)^2 \quad (2)$$

In simple terms, the PSNR refers to the difference between the original frame and the encoded frame. It's a logarithmic scale, therefore expressed in dB, calculated by Equation 3, where *n* is the number of bits in the image sample (Richardson, 2003).

$$PSNR_{dB} = 10\log_{10}\frac{(2^n - 1)^2}{MSE} \quad (3)$$

Figure 3 shows a crop of the original frame compared to two modified versions of the same scene. Different levels of blur were applied, in a way in which the second image was better preserved (PSNR=30.6dB), while the third image is the blurriest (PSNR=28.3dB). In this case, it's easy to notice that the higher PSNR corresponds to greater image quality (Richardson, 2003).

Figure 3 – Comparing the original image (a), PSNR=30.6dB (b), and PSNR=28.3dB (c)



Source: Richardson (2010)

Although the PSNR is a good objective indicator of quality, it's not an absolute value for subjective quality, as demonstrated by Figure 4. Keeping the girl's face in focus while blurring the background results in a pleasant image, even though its PSNR is inferior to previous

examples (27.7dB). It's also important to note that a reference image is needed to calculate the MSE (Richardson, 2003).

Figure 4 – Image with the blurred background (PSNR=27.7dB)

Source: Richardson (2010)

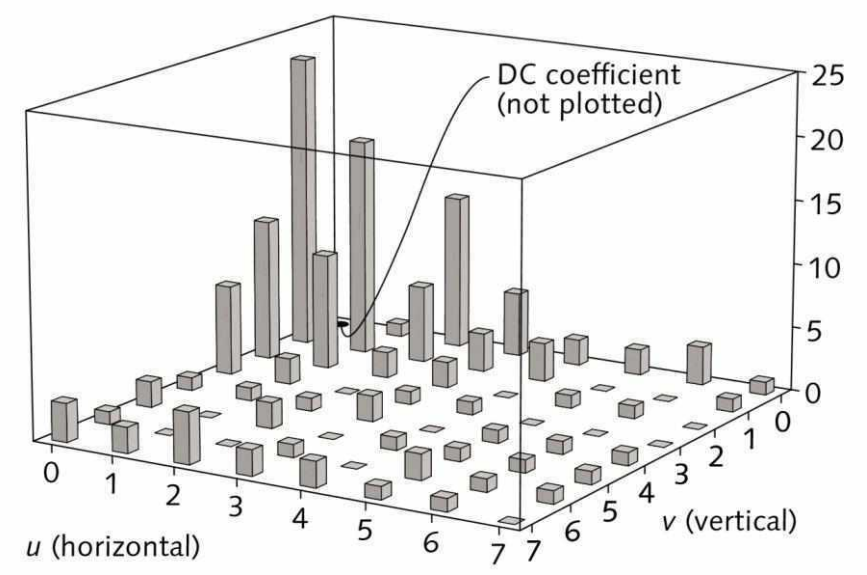
2.2.1 Video Compression Principles

A video is composed of a series of images shown in rapid sucession to create the impression of movement. A JPG image is composed of thousands or millions of pixels, which are composed of three color components with 8 bits each. These images are usually reproduced at a rate of 24 frames or more per second. Without compression, the transmission of all this information would be almost impossible (Poynton, 2012).

Lossless image compression allows reduction of size without any loss in quality. However, this process depends on the characteristics of the data. In order to obtain better results, it's necessary to compress the images, introducing information loss. One of the most used methods makes use of the Discrete Cosine Transform (DCT). Both JPEG, which is used in stills, as well as MPEG-2 and AVC, which are used on TV and Blu-ray movies, use DCT (Poynton, 2012).

The JPG format, adopted in 1992, can be used for black and white or colored images. To convert an image to JPG, the R'G'B' color space is first converted into $Y'C_BC_R$ and then subsampled to 4:2:0 before applying a linear spatial transform (DCT). The image is often divided in chunks of 8x8 pixels. Figure 5 shows the visual representation of the DCT, with the DC coefficient not plotted due to its scale. Noticeably, most of the values are close or equal to zero. Compression is achieved by discarding those values, which often goes unnoticed (Poynton, 2012).

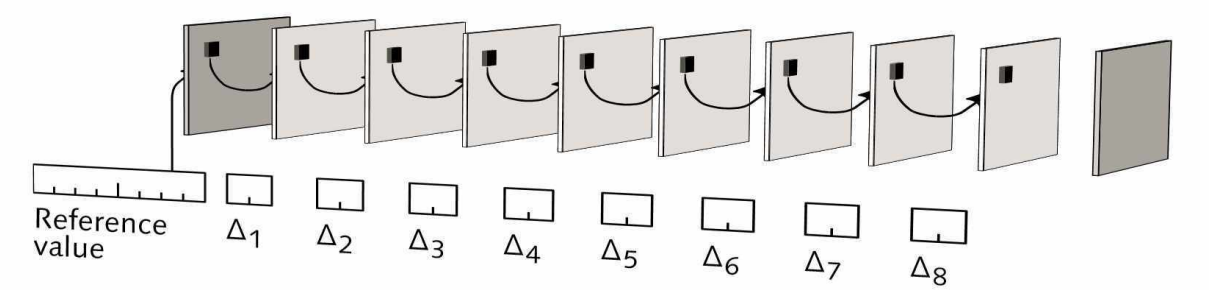
Figure 5 – Visual representation of the linear spatial transform.



Source: Poynton (2012)

In most cases, except when there a change of scenes, there is little variance between one frame and the next. It's possible to take advantage of these redundancies and exponentially increase the compression rate. This is the main principle MPEG (Moving Picture Experts Group) is based on. Two techniques are implemented: intraframe and interframe compression. Intraframe refers to compression within a single frame, called reference frame. Interframe compression is applied to a sequence of similar frames, called Group of Pictures. Instead of transmitting the entirety of every frame, the first image is used as a reference and only the pixel variations between frames are saved. Figure 6 shows this process. In a group of 9 images, only the first frame is fully stored. For the following frames, only the pixel variations, represented by the moving block, are saved (Poynton, 2012).

Figure 6 – Interframe compression for a Group of Images



Source: Poynton (2012)

Rapid camera movements may cause great variation between frames, drastically reducing the compression efficiency. To avoid this problem, MPEG uses Motion-Compensated Prediction (MCP). Movement vectors are used to predict movement, allowing the encoder to move the necessary pixels in the reference frame. The result is compared to the actual frame, and the residual difference is saved so that the decoder can use it (Poynton, 2012).

Moreover, 3 types of frames can be identified in a Group of Images: I, P, and B frames. I frames are the reference frames, to which intraframe compression is applied. P frames are the frames predicted based on the previous one. Meanwhile, B frames are predicted both from the previous and next frames, allowing an increase in the compression rate even further. However, B frames require a buffer, making it impractical for real-time applications, such as video calls (Poynton, 2012).

The principles used by MPEG originated with MPEG-1, which is used in multimedia applications. It was optimized for progressive video at a resolution of 352x240 at 30 frames per second. Due to its limitations, the bit rate was locked at 1.5Mbps. Aiming for better image quality, MPEG-2 was developed. This new codec, which is used for TV transmissions and DVD movies, offers support for progressive and interlaced video (Poynton, 2012).

2.3 AVC

H.264, or Advanced Video Coding (AVC), is one of the most important methods to compress video, being used in phones, online streaming, Blu-ray discs, video call apps, and more. Initially published in 2003, the codec aims to increase the compression efficiency offered by MPEG-2, as well as improve its flexibility. The encoder follows the standards set by the International Telecommunication Union (ITU-T) and the International Organization for Standardization/International Electrotechnical Commission (ISO/IEC), detailed in its specification. Those specifications generate the AVC bitstream, which is used by the decoder to reproduce the original content (Richardson, 2003).

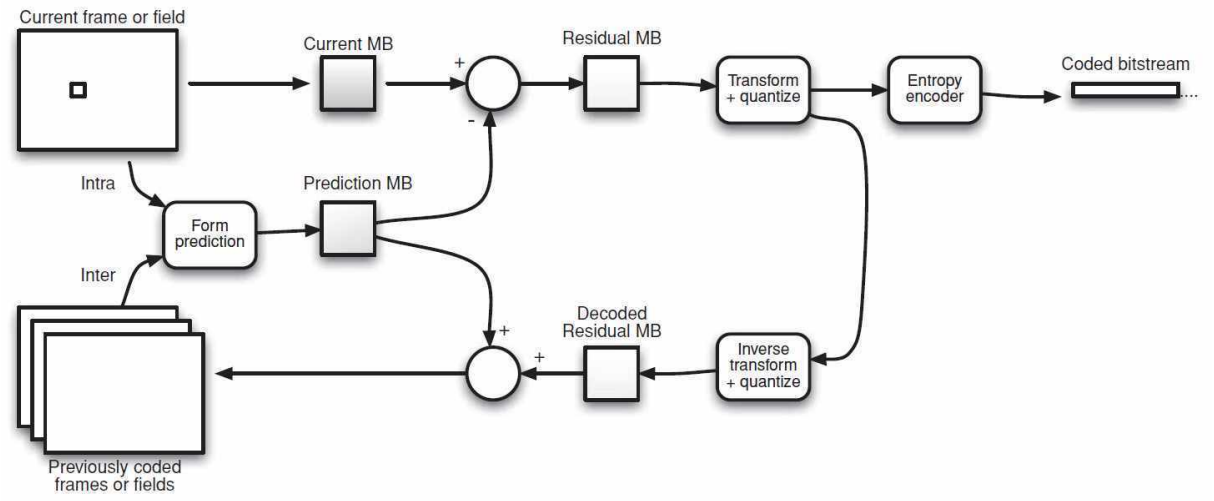
When the AVC bitstream is created, the encoder uses a set of tools so that the decoder can read the resulting information. This is possible through profiles, standards set by the official specification (Richardson, 2003).

Figure 7 shows the block view of an AVC encoder. Each frame is divided into smaller areas, called macroblocks, of 4x4 or 16x16 pixels. Each one of these is predicted, using information from the current frame (intra) or neighbor frames (inter), and then they are compared to the next frame's macroblock. The residual difference between them is saved. The

information of all encoded frames in a video file, alongside its metadata, forms the bitstream. At the same time, the quantized information is resized and inverse transformed so that it can be used with the predicted macroblock in order to reconstruct the original frame. That frame is then stored for future predictions (Richardson, 2003).

The difference between the macroblocks is obtained through an approximated version of the DCT. When the DCT is applied to a macroblock, it results in a series of coefficients. These coefficients are then quantized; in other words, each element is divided by an integer, the quantization parameter (QP), reducing its precision by increasing the numbers of values equal to zero. A higher QP results in higher compression at the expense of image quality. Similarly, a lower QP reduces the compression rate, but it allows better image quality (Richardson, 2003).

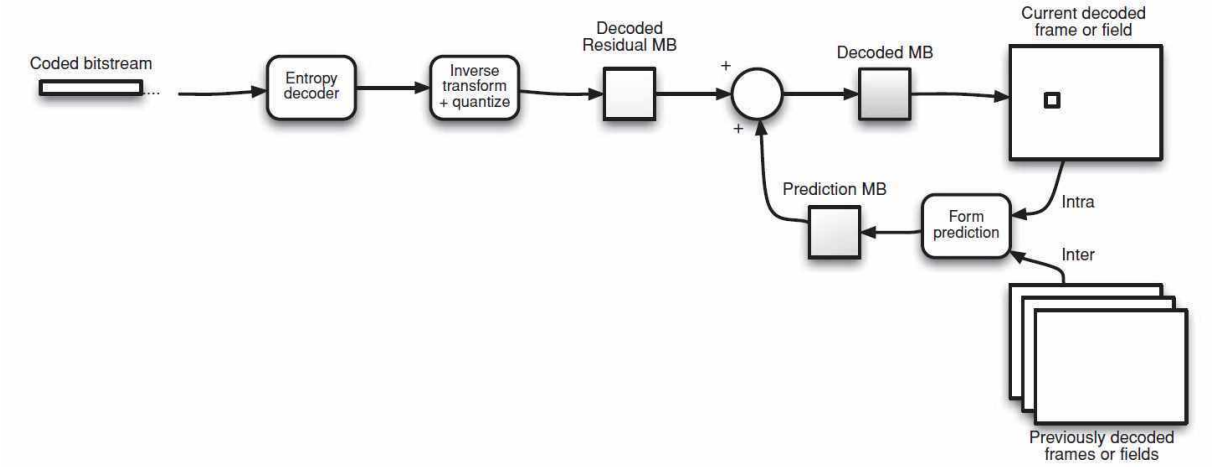
Figure 7 –AVC Encoder



Source: Richardson (2010)

Figure 8 shows a block view of an AVC decoder. The bitstream is received, then quantized and inverse transformed, then the residual macroblock information is decoded. Then the decoder predicts all the frames plus the macroblock information. The result is a frame close to the original (Richardson, 2003).

Figure 8 – AVC Decoder



Source: Richardson (2010)

All these operations require memory and processing power. To ensure the file can be reproduced properly, each video file and device has a profile and level information embedded. The former specifies which tools are necessary for decoding and the latter specifies which resolution and bit rate are supported. For example, a device with a Main profile level 2.2 can decode files with a resolution up to 720x576 pixels at a bit rate up to 4Mbps (Richardson, 2003).

2.4 VP9

The evolution of the internet has set the requirements for video distribution. As demand has increased, more and more efficient alternatives have been required. With that in mind, Google started the WebM project in 2010, an open media format made for online content based on VP8 for video and Vorbis for audio. The container was based on MKV (Mukherjee, et al., 2015).

The demand for high quality video kept increasing, and in 2011, Google started the development of the VP8 successor, VP9, a completely open source project (Mukherjee, et al., 2015).

The main goal of VP9 is to increase compression efficiency. To achieve this goal, it makes use of prediction blocks of variable size. These superblocks can be up to 64x64 pixels and they can be broken into smaller blocks through recursive partitioning. (Mukherjee, et al., 2015) Figure 9 shows all allowed sizes. A 64x64 superblock may keep its original size, or it can be divided in two horizontally or vertically, or be divided into four equal blocks as little as 4x4 pixels (Sonnati, 2016).

The first frame of a group of images uses intraprediction, while the remaining frames use interprediction. For intraprediction, up to 10 different prediction modes are used, depending on the block size. These modes include DC prediction, true-motion, horizontal, vertical, and directional (for 27-, 153-, 135-, 117-, 63-, and 45-degree angles measured counterclockwise from the horizontal axis). All blocks follow the same transform, depending on their size, for both encoding and decoding. If a 4x4 transform is applied, all 4x4 blocks inside a 16x16 superblock go through prediction, encoding, and prediction before starting the next transform. For interprediction, one of 4 modes are used. NEARESTMV and NEARMV are used for movement vectors already encoded, ZEROMV is used when there is no movement within a certain block, and NEWMV is used for new movement vectors (Mukherjee, et al., 2015).

Additionally, VP9 supports sub-pixel interpolation with 1/8 pixel precision. It's possible to change that value to 1/4 with a flag. For other values, motion blur is usually used (Mukherjee, et al., 2015).

Figure 9 – Superblock recursive partitioning

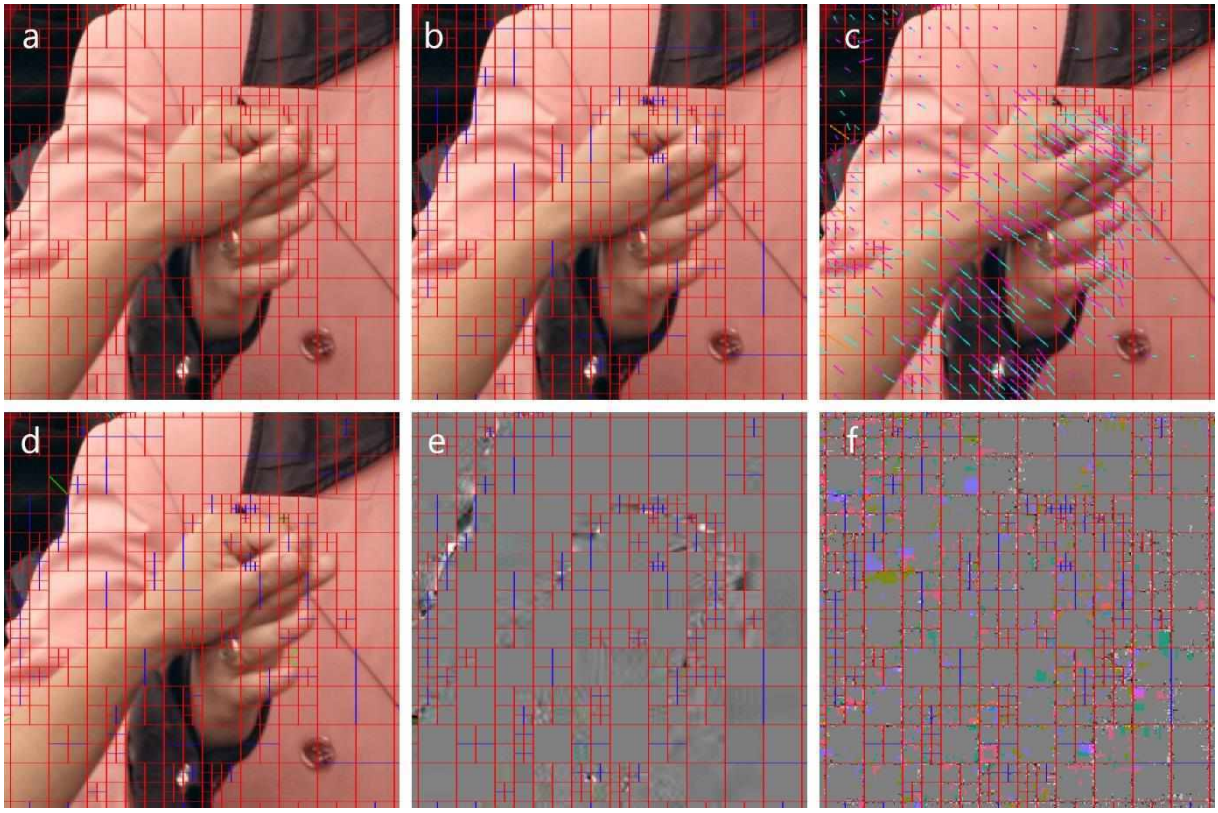
Source: Sonnati (2016)

VP9 uses one of 3 transforms for residue encoding: Discrete Cosine Transform (DCT), Asymmetrical Discrete Sine Transform (ADST) or Walsh-Hadamard Transform (WHT). A combination of DCT and ADST is used according to the prediction mode (Rbultje, 2016). A 4x4 WHT is only used for lower quantization values, in order to obtain a lossless encoding (Mukherjee, et al., 2015).

Due to this diversity of sizes, a loop filter is used to eliminate artefacts on the block's boundaries. During the reconstruction process, the loop filter scans all superblocks. Then it goes through all the smaller neighbor blocks starting at the top and to the left. By default, the filter is applied only to 32x32 and 64x64 blocks (Mukherjee, et al., 2015).

Figure 10 shows each step applied to a frame section. Figure 10.a shows the superblock decomposition in red; Figure 10.b shows the transform blocks, which is applied to any blocks that aren't squares; Figure 10.c shows the movement vectors in cyan, magenta, and orange; Figure 10.d shows the intraprediction mode in green; Figure 10.e shows the residual coding, the differences between the predicted frame and the original with the contrast amplified 10 times; and Figure 10.f shows the loop filter effects with the contrast amplified 100 times in order to highlight artefacts (Rbultje, 2016).

Figure 10 – Superblock decomposition (a), transform blocks (b), movement vectors (c), intraprediction (d), residual coding (e), loop filter (f)



Source: Gnome (2016)

2.5 HEVC

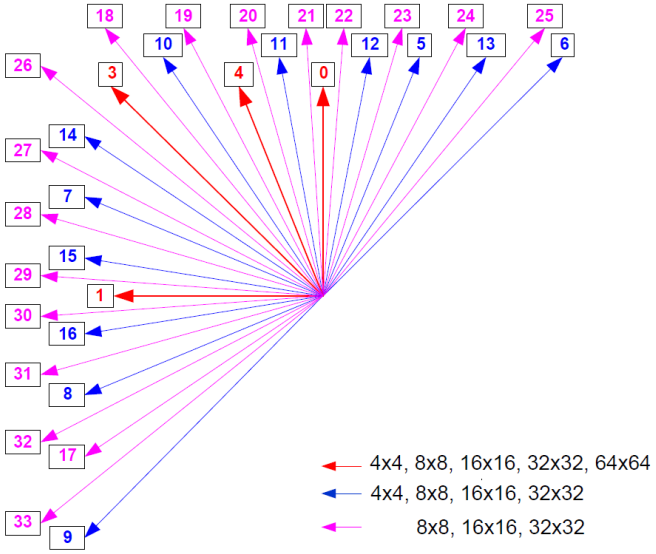
In January 2013, the specification for the next generation codec, H.265, also known as High Efficiency Video Coding (HEVC), was finalized. In tests using Full HD video, it was

possible to attain a 36% reduction in the bit rate compared to AVC at the same PSNR. In visual tests, that difference was up to 70% (Bordes, Clare, Henry, Raulet, & Viéron, 2012).

Many of the principles applied in HEVC are based on AVC. However, several tools have been improved and optimized for ultra-high definition video, such as 4K. Some of these improvements are a new partitioning scheme, 10-bit precision support, a parallel bitstream, and better intra- and intercoding (Bordes, Clare, Henry, Raulet, & Viéron, 2012).

During the encoding process, the first frame of a group of images is divided into blocks following rules set by the codec. HEVC uses a Coding Tree Block (CTB), allowing for blocks with 16, 32, or 64 samples. The frame is encoded exclusively with intracoding using one of 35 prediction modes, 33 of which are directional as shown in Figure 11. For comparison, AVC has 8 directional prediction modes (Sullivan, Ohm, Han, & Wiegand, 2012).

Figure 11 – Intraprediction directional prediction modes



Source: Bordes (2012)

The other frames are usually encoded with temporal prediction using movement vectors. The difference is that, previously, these vectors were calculated through the median of the three neighbor blocks. Since HEVC has blocks of variable sizes, a 64x64 block may have eight neighbor blocks. To aid that situation, a technique called movement vector competition decides which vector will be used (Bross, Helle, Lakshman, & Ugur, 2014).

Next, a linear spatial transform is applied on the differences between the predicted blocks and the originals. The result is then scaled, quantized, entropy coded, and transmitted with the prediction information (Sullivan, Ohm, Han, & Wiegand, 2012).

During the decoding process, the decoder uses the received predicted information to recreate frames identical to those used in the encoding. At the same time, the difference between frames previously calculated is quantized and inverse scaled so that it can be applied to the predicted frames. Then a loop filter is applied to reduce artefacts introduced by the quantization. Finally, the reconstructed frame is stored in a buffer so that it can be used to predict the following frames (Sullivan, Ohm, Han, & Wiegand, 2012).

Other tools are used to improve image quality, such as deblocking, Sample Adaptive Offset (SAO), and Adaptive Loop Filtering (ALF). The deblocking filter reduces artefacts on the boundaries between blocks; the SAO reduces distortions in the image, compensating the pixel shift according to its position; and the ALF reduces the MSE between the reconstructed frames and the original (Bordes, Clare, Henry, Raulet, & Viéron, 2012).

Because of all the new features introduced, HEVC's decoding is much more complex than AVC's. HEVC decoding counts with parallel techniques to optimize the process. After the decoder reads the bitstream, the decoder divides the CTB into a rectangular grid of units called *tiles*. Then a process called *wave front* allows the decoding of each one of these blocks into different threads in a synchronous way (Sonnati, 2014).

2.6 AV1

The Alliance for Open Media (AOM) consists of a group of hardware, software, content production, streaming services and videoconferences, and browser companies. The full list includes Adobe, Allegro, Amazon, AMD, Amlogic, Apple, Argon Design, ARM, Ateme, BBC, Bitmovin, Broadcom, Chips & Media, Cisco, Facebook, Google, Hulu, IBM, Intel, Ittiam, Microsoft, Mozilla, Netflix, NG Codec, Nvidia, Polycom, Realtek, Sigma, Socionext, Veri Silicon, VideoLAN, Vidyo, and Xilinx. The goal of the AOM is to develop a new open source video standard prepared for existent and future video transmission demands (Egge & Bebenita, 2018).

The fruit of this union is the AOMedia Video 1 (AV1) codec. Some of its main features include interoperability, scalability, low computational footprint, optimization for the web and hardware, flexibility for commercial and domestic use, and real-time transmission (Layek, et al., 2017).

One of the new codec priorities is UHD video. As a result, AV1 supports higher bit rates, wide color gamut, and high frame rates. It is also optimized to allow the reproduction of a 4K video file at 60 frames per second within a browser using a high-performance computer.

It’s also possible to reproduce HDR video with 10- or 12-bit color depth using the BR.2020 color space (Ozer, 2017).

One of the first services to adopt the new codec will be YouTube. This may happen as soon as 6 months after the bitstream is finalized. Currently, the website uses VP9 for high-resolution video. The new codec should be beneficial for UHD and HDR video, considering they require a higher bit rate. Moreover, the first devices natively compatible with AV1 may be launched within a year after the bitstream is finalized (Ozer, 2017).

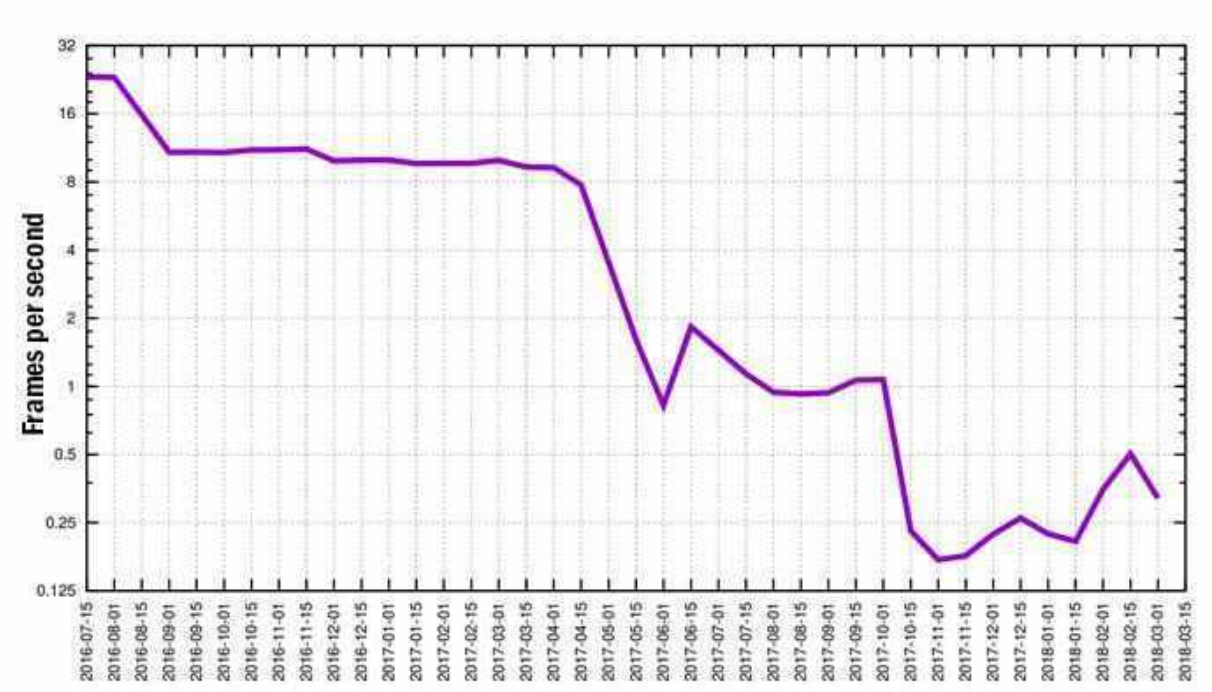
Another service looking to adopt the new codec is Netflix, but Netflix has some requirements. Netflix's director of encoding technologies, David Ronca, said in an interview that “We will be satisfied with 20% efficiency improvement over HEVC when measured across a diverse set of content and would consider a 3-5x increase in computational complexity reasonable” (Ozer, 2017).

In simple terms, AV1’s source code has been updated with experiments originated from different members of the group. These experiments may or may not be activated by default. As new experiments are being added, the compression efficiency, as well as the codec’s complexity, is increasing. Figures 12 and 13, respectively, show the compression and the complexity history as new experiments are added. The first graph shows that around December 2016 there was a 30% increase in the compression rate compared to the initial code. The second one shows that the codec was able to encode in real time (30 frames per second) at first, but that by the end of 2017, the value had dropped to less than 0.25 frame per second (or one frame every four seconds). New experiments have been added to improve this metric since then. It is also interesting to note that there was a sudden spike in the compression rate on February 15, 2015. This was due to a regression introduced in the code, which was corrected for the next build. The same spike is also present on the complexity graph (Egge & Bebenita, 2018).

Figure 12 – AV1 compression history

Source: Mozilla Hacks (2018)

Figure 13 – AV1 complexity history



Source: Mozilla Hacks (2018)

Among the new technologies integrated into AV1, the ones worth mentioning include a new high-level syntax, new adaptive multi-symbol entropy coding, more block sizes, more

transforms types, more references, spatial and temporal scalability, lossless mode, chroma subsampling, more prediction modes, and a new loop filter (Egge & Bebenita, 2018).

Traditionally, the bitstream is encoded according to the binary symbols' probability. The multi-symbol entropy coder, introduced by Mozilla, allows for groups of up to 16 symbols to be encoded at the same time. This allows for a more precise prediction of these symbols and the transmission of more information to the hardware decoder at lower frequencies (Egge & Bebenita, 2018).

AV1 has support for prediction blocks of up to 128x128 pixels. This superior block size is advantageous for UHD video. Furthermore, these blocks can be divided into rectangular blocks with aspect ratios of 1:2 (4x8 pixels), 2:1 (8x4 pixels), 1:4 (4x16 pixels), or 4:1 (16x4 pixels). The transform blocks range up to 64x64 pixels, following these same ratios (Egge & Bebenita, 2018).

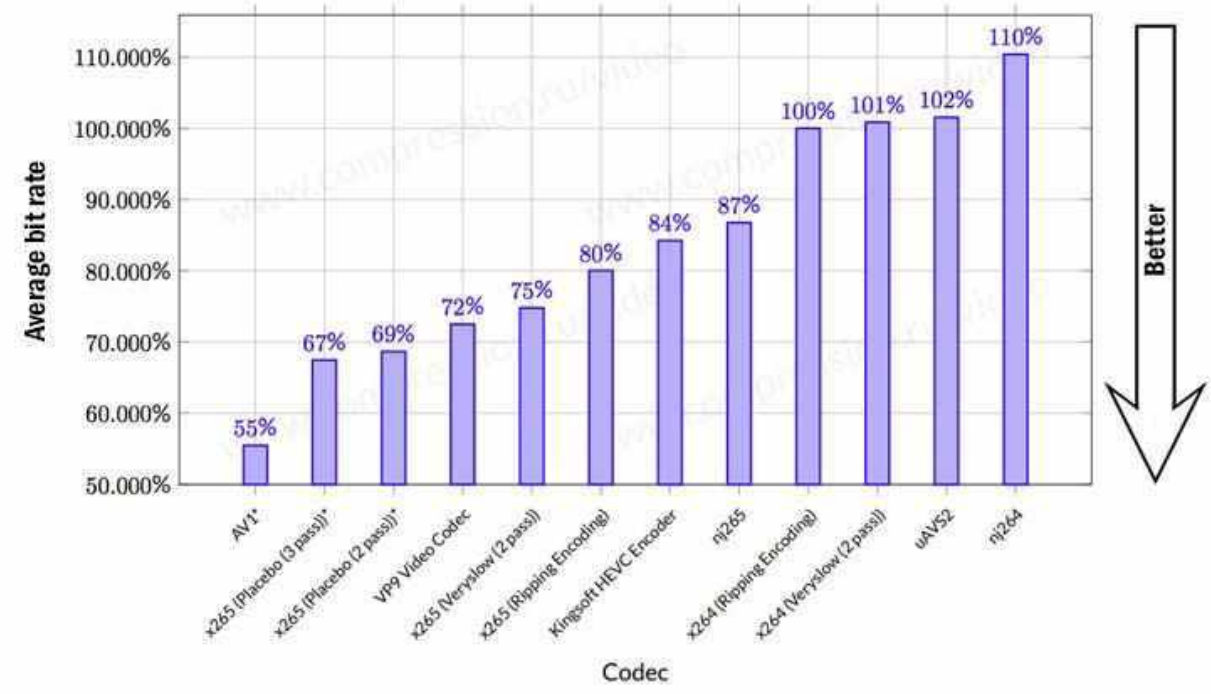
While HEVC uses the DCT to calculate the residual information and VP9 uses a combination of the DCT and ADST, AV1 introduces two new options: an identity transform (IDTX) and Discrete Sine Transform (DST). This allows for up to 16 possible combinations for each block, the combination is chosen based on the movement vectors used (Egge & Bebenita, 2018).

The prediction modes were also improved. For intraprediction, new delta and main vectors were introduced, for a total of 65. These vectors allow for a more precise prediction of the next frame. The reference sample is now obtained from the top-right superblock neighbor. Additionally, a new tool allows the derivation of the chroma intra residuals from the luma intra residuals. For interprediction, up to seven reference frames can be used, and the overlapped block motion compensation reduces discontinuities at block edges. There were also improvements to the vector movements estimations (Sethuraman, Rajan, & Patankar, 2017).

In addition, a combination of new loop filters was implemented. A directional de-ringing, conditional low pass, and loop restoration filters work to remove artefacts caused by the quantization, transform, and block processing (Sethuraman, Rajan, & Patankar, 2017).

All these features combined resulted in a higher compression rate, even though the code complexity was also increased. Figure 14 shows the average bit rate for each codec maintaining the same Structural Similarity (SSIM). On this test, AV1 offers the smallest files without loss of quality.

Figure 14 – Average bit rate for the same SSIM



Source: Moscow State University (2017)

2.7 Licensing/Royalties

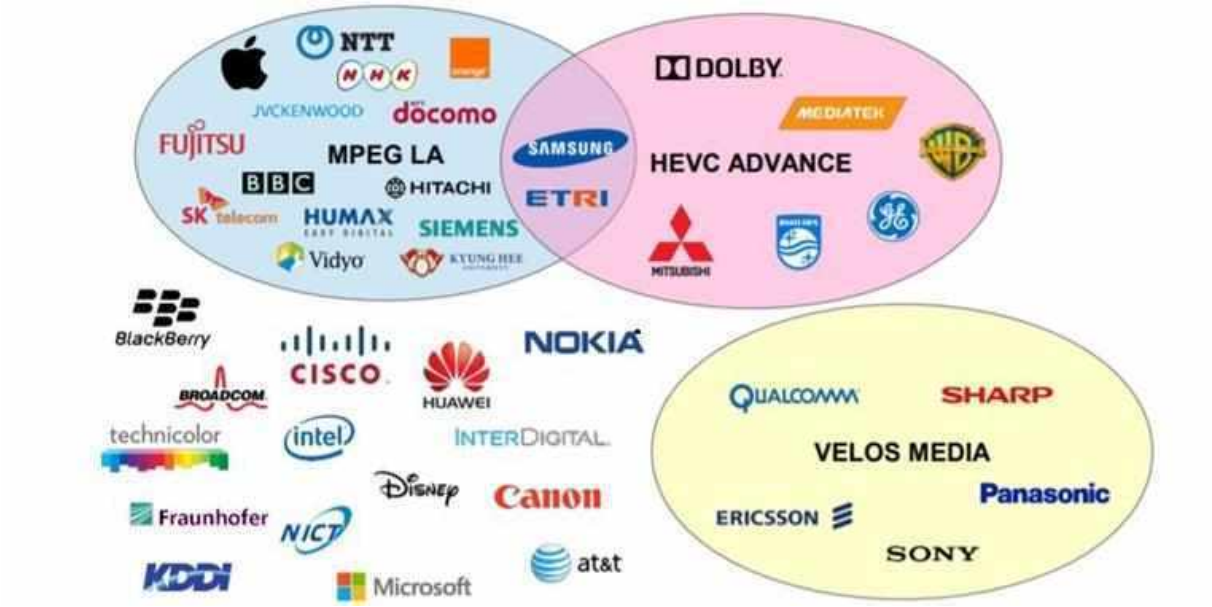
One of the drawbacks in the HEVC implementation is the confusing licensing terms. For over 30 years, the MPEG LA (Moving Picture Experts Group Licensing Authority) has managed the patents licensing for commercial use of MPEG-2 and AVC. MPEG-2 was widely used on HDTV and DVD. Likewise, AVC was also very popular, being used for most online applications, as well as for mobile and storage. Both MPEG-2 and AVC had comprehensive licensing terms, and most of the patents were held by MPEG LA. They also own the MP3 (1992) and MP4 (2001) formats (Chiariglione, 2018).

This system worked because the MPEG-2 and AVC standards were promptly adopted by the market due to their noticeable evolution compared to the previous standards and the lack of competition. When billions of devices already use said standards, it is beneficial for the device's patent owners to allow MPEG LA to manage their use through royalties (Chiariglione, 2018).

However, HEVC followed a different path. A total of 45 companies, divided into three different licensing groups, have patents over the codec. Moreover, some companies did not affiliate with any group (Chiariglione, 2018). Figure 15 shows the situation: if a certain company wants to use HEVC with a product or service, it needs to license it with one of the

three groups pictured. The situation is aggravated when none of the three has total ownership of the patents needed. For example, after licensing the technology with MPEG LA and adopting the HEVC standard on its products, a company may unintentionally use a patent registered by HEVC Advance. The company then has to either pay royalties to the second group or face legal action.

Figure 15 – HEVC licensing groups



Source: Divideon (2018)

Aiming to avoid situations like this, AV1 was developed as completely open source. Benchmarks show that AV1 has similar performance to that of HEVC, with none of the licensing issues. Table 1 shows what this means. While different groups charge different royalties regarding HEVC, AV1 is open for all.

Table 1 – AV1 and HEVC licensing terms comparison

Video Format	**Licensor**	**Codec Royalties**	**Codec Royalties Exceptions**	**Codec Royalty Annual Cap**	**Content Distribution Fee**
HEVC	MPEG LA	US $0.20/unit	First 100k units/year	US $25 million	Zero
	HEVC Advance	**Region 1**: - US $0.40 (mobile) - US $0.80 (PC) - US $1.20 (TV) **Region 2**: - US $0.20 (mobile) - US $0.40 (PC) - US $0.60 (TV)	US $25,000/year	US $40 million	**Content free to users:** - Zero **Paid by title model:** - US $0.025/paid title **Paid subscriber model:** - US $0,005/month/subscriber - US $25 million/year/business model
	Technicolor	Tailor-made agreements			Zero
	Velos Media	???	???	???	???
	Others	???	???	???	???
AV1	AOMedia	Zero	N/A	N/A	Zero

Source: Mozilla Hackers (2018)

CHAPTER 3 – METHODS

For this paper, all the encoding was done on a Core i5 4690K with 16GB of RAM PC running Windows 10 using FFMPEG. For the PSNR calculation, MATLAB R2019a was used.

3.1 Compression tools

The selected video files for this paper include CIF reference files, as well as 1080p and 4K samples. The encoding process was done using FFMPEG running on Windows.

The reference files *akiyo*, *coastguard*, *foreman*, *mobile*, and *pedestrian* were downloaded from the xiph.org website (Xiph.org, 2006). The *ShakeNDry* file was downloaded from the Ultra Video Group website (Ultra Video Group, 2013). All of them are uncompressed samples in the .Y4M format with 4:2:0 subsampling. The first four have 300 frames (10 seconds at 30FPS) at a resolution of 352x288 pixels, resulting in a bit rate of 35.64Mbps (43.5MB file size). *Pedestrian* is a 1080p sample composed of 375 frames (15 seconds at 25FPS), resulting in a bit rate of 607.5Mbps (1.08GB file size). Finally, *ShakeNDry* is a 4K sample composed of 300 frames (10 seconds at 30FPS), resulting in a bit rate of 2.78Gbps (3.47GB file size).

Two versions of FFMPEG were used for the encoding process: the latest stable build (v4.1.1), which was launched February 21, 2019, and the latest nightly build at the time (v20190430-19af948), which was launched May 1, 2019 (FFMPEG, 2019). To ensure fair results, all the fixed bit rate samples were encoded using the latest nightly build using the default settings.

FFMPEG is a command-line tool, which means it does not offer a GUI (Graphic User Interface). In order to use FFMPEG, it is necessary to execute commands (scripts) with the correct parameters. The FFMPEG command line has three main components: input, parameters, and output. The input is denoted by *-i*, followed by the parameters. Everything that cannot be interpreted as a parameter is considered as the output file path. Additionally, the user can specify global options before the input, such as *-y* to overwrite output files without asking, for example (FFMPEG, 2019).

The interface chosen for this was Windows PowerShell. It allows the execution of .PS1 scripts and copying of the resulting log for further analysis. The script can be written using Notepad and then saved with the .PS1 extension. By default, PowerShell does not allow unsigned scripts to be run. This, however, can be changed by running the command "*Set-ExecutionPolicy RemoteSigned*" then pressing A (for Yes to All) (Dummies Insider, 2009).

The scripts for a two-pass 50kbps bit rate for AV1, AVC, HEVC, and VP9 have the following structure, respectively:

```
FFMPEG -y -i INPUT -c:v libaom-av1 -strict experimental -b:v 50k -pass 1 -an -f matroska NUL
FFMPEG -i INPUT -c:v libaom-av1 -strict experimental -benchmark -b:v 50k -pass 2 -c:a libopus OUTPUT

FFMPEG -y -i INPUT -c:v libx264 -b:v 50k -pass 1 -an -f mp4 NUL
FFMPEG -i INPUT -c:v libx264 -benchmark -b:v 50k -pass 2 -c:a aac OUTPUT

FFMPEG -y -i INPUT -c:v libx265 -b:v 50k -x265-params pass=1 -an -f mp4 NUL
FFMPEG -i INPUT -c:v libx265 -benchmark -b:v 50k -x265-params pass=2 -c:a aac OUTPUT

FFMPEG -y -i INPUT -c:v libvpx-vp9 -b:v 50k -pass 1 -an -f webm NUL
FFMPEG -i INPUT -c:v libvpx-vp9 -benchmark -b:v 50k -pass 2 -c:a libopus OUTPUT
```

FFPMEG indicates the path at which ffmpeg.exe file is located, INPUT is the path to the input file, and OUTPUT is the path to the output file.

For the first part of this analysis, the codecs are compared at an average bit rate. In order to ensure the best results, two-pass encoding is used. During one-pass encoding, the codec encodes the video as it goes. Two-pass encoding allows the codec to read the entire video file first and then creates a temporary file with all the information it needs to better allocate the data. This allows for smaller files with better quality throughout. In order to do this, two command lines must be used. The first one has the *-pass 1* parameter and the output set to NUL. The second line has the *-pass 2* parameter and the output file path (FFMPEG, 2019).

The other parameters used are *-c:v* to specify the library to be used (libaom-av1 for AV1, libx264 for AVC, libx265 for HEVC, and libvpx-vp9 for VP9); *-benchmark* includes benchmark information at the end of the encoding process including real time (rtime), system time (stime), and user time (utime) used; *-b:v* specifies the bit rate; *-an* skips audio encoding (used on the first pass since audio doesn't need to be two-pass encoded); *-f* forces the encoder

to use a certain video format (.MKV for AV1, .MP4 for AVC and HEVC, and .WEBM for VP9); and *-c:a* specifies the audio codec (libopus for AV1 and VP9, and AAC for AVC and HEVC). AV1 also requires the *-strict experimental* parameter since it is still considered experimental at this stage. For HEVC, the *-x265-params* parameter is also necessary to specify encoding options exclusive to it (FFMPEG, 2019).

For the second part, AV1 will be evaluated in two ways. Since AV1 is still in development, optimizations are still being added to the code, with a big focus on speed. In order to observe the length of those optimizations, an older build of the AV1 library (v20190221-cde3872) will be compared to the most recent library available (v20190425-9feb158). At the same time, the effects of the *-cpu-used* parameter on render time and quality will also be compared. This parameter is an integer ranging from zero to eight (one being default), and it sets the tradeoff between speed and quality. Higher values mean faster encoding with worse quality and vice versa (FFMPEG, 2019).

The script for a two-pass encoding with *-cpu-used* set to three and a bit rate of 150kbps is as follows:

```
FFMPEG -y -i IPNUT -c:v libaom-av1 -strict experimental -b:v 150k -
pass 1 -an -f matroska NUL
FFMPEG -i INPUT -c:v libaom-av1 -strict experimental -benchmark -cpu-
used 3 -b:v 150k -pass 2 -c:a libopus OUTPUT
```

3.2 Compression analysis

In total, 170 encoded samples were analyzed. Four CIF videos were encoded for each of the four codecs across eight different bit rates, as well as one 1080p and one 4K videos across three different bit rates. Additionally, a CIF video and a 1080p video were encoded using every possible *cpu-used* value in both the latest stable and nightly builds on FFMPEG.

The first metric compared was the encoding time for each codec (rtime), found in the PowerShell log after running each command line. The PSNR for each encoded file was calculated using Matlab R2019a. The code used is described below:

```
% Reading video files
ORIGINALfile=VideoReader('ORIGINAL');
ENCODEDfile=VideoReader('ENCODED');

% Reading video frames
while hasFrame(ORIGINALfile)
    ORIGINAL=readFrame(ORIGINALfile);
end
while hasFrame(ENCODEDfile)
    ENCODED=readFrame(ENCODEDfile);
end

% Calculating PSNR
PSNR_ENCODED=psnr(ENCODED,ORIGINAL)
```

For this example, ORIGINAL refers to the uncompressed sample (.Y4M) and ENCODED to the encoded file (with the corresponding format). The first part of the code reads the video file and stores its information in the object. The second part uses a loop that goes through each frame of the video, reading its contents. Finally, the *psnr* function compares the information in the encoded video with the original and returns the peak signal-to-noise ratio.

CHAPTER 4 – RESULTS

The results are divided into two sections. In the first section, the encoding time and the PSNR for all video samples is compared according to the codec and bit rate used. Then the same metrics are compared using different values for the *cpu-used* parameter in both builds of FFMPEG. The idea is to compare how AV1 stacks to competing codecs and what progress have been made in its recent development.

4.1 Average bit rate across AV1, AVC, HEVC, and VP9

The first video sample compared is *akiyo*. It consists of 300 frames (10 seconds at 30FPS) at 352x288 pixels. The original video file is 44.5MB. Table 2 shows the encoding time (rtime) for each codec at a set bit rate, in kbps. Table 3 shows the PSNR for the same scenarios.

Table 2 – Encoding time (in seconds) for each codec at a set bit rate for *akiyo*

	50	**100**	**150**	**200**	**250**	**300**	**350**	**400**
AV1	153.997	223.422	268.651	315.178	356.686	404.149	439.676	477.64
AVC	0.192	0.235	0.257	0.272	0.287	0.296	0.301	0.314
HEVC	0.946	1.113	1.29	1.413	1.533	1.644	1.752	1.839
VP9	11.003	10.868	12.309	13.691	15.552	16.054	17.217	18.002

Source: The author

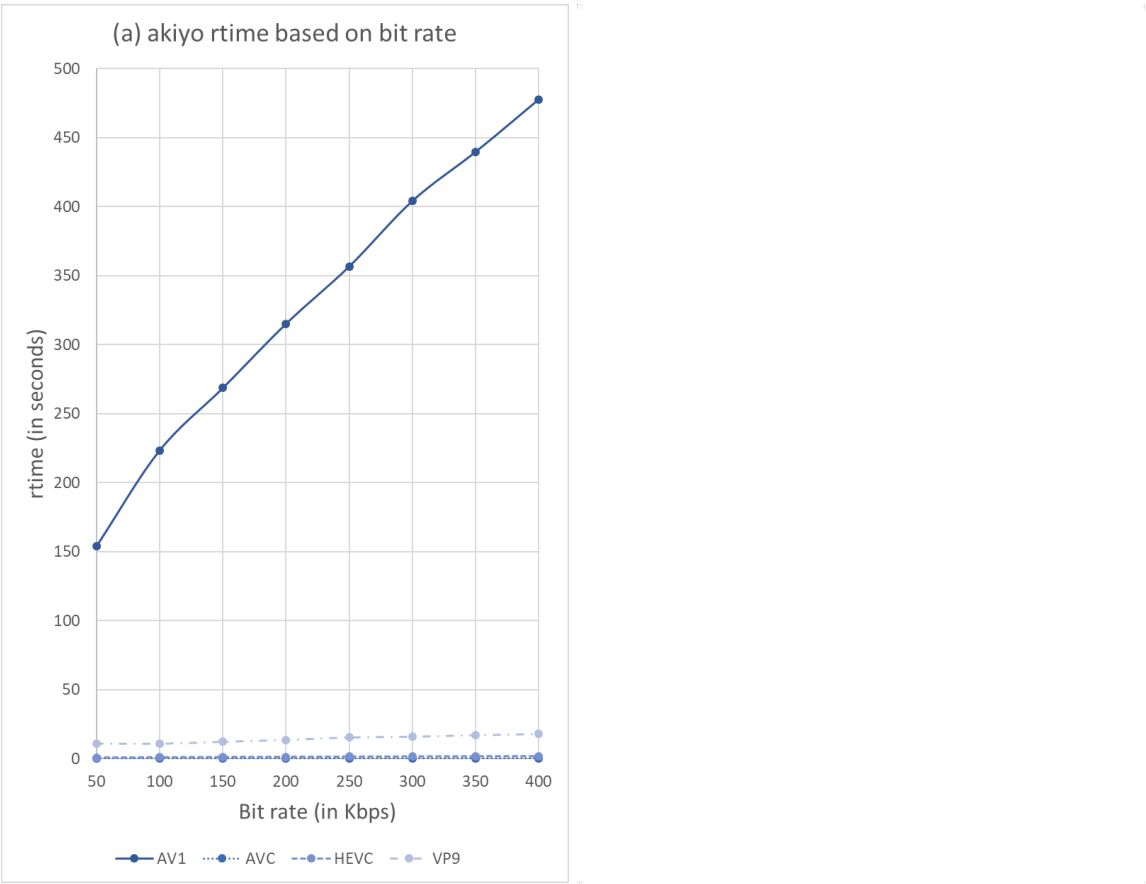
Table 3 – PSNR (in dB) for each codec at a set bit rate for *akiyo*

	50	**100**	**150**	**200**	**250**	**300**	**350**	**400**
AV1	39.3722	41.964	43.5318	44.7936	45.8136	46.3091	46.9452	47.4143
AVC	34.6596	36.5891	37.7608	38.6593	39.1788	39.6725	40.0184	40.2827
HEVC	36.5421	39.3039	40.3433	41.2506	41.9478	42.5509	43.2042	43.7159
VP9	37.4903	39.2243	39.9915	40.6619	41.2057	41.6589	41.8513	42.2463

Source: The author

The data from Tables 2 and 3 is shown in Figures 16.a and 16.b, respectively.

Figure 16 – Encoding time (a) and PSNR (b) for each codec at a set bit rate for *akiyo*



Source: The author

Next one is *coastguard*. It consists of 300 frames (10 seconds at 30FPS) at 352x288 pixels. The original video file is 44.5MB. Table 4 shows the encoding time (rtime) for each codec at a set bit rate, in kbps. Table 5 shows the PSNR for the same scenarios.

Table 4 – Encoding time (in seconds) for each codec at a set bit rate for *coastguard*

	50	**100**	**150**	**200**	**250**	**300**	**350**	**400**
AV1	208.692	311.057	400.779	482.657	505.104	549.293	577.641	603.601
AVC	0.256	0.312	0.376	0.389	0.431	0.431	0.474	0.473
HEVC	1.074	1.314	1.49	1.663	1.79	1.918	2.035	2.106
VP9	13.772	17.827	20.525	24.432	27.035	29.5	31.71	33.731

Source: The author

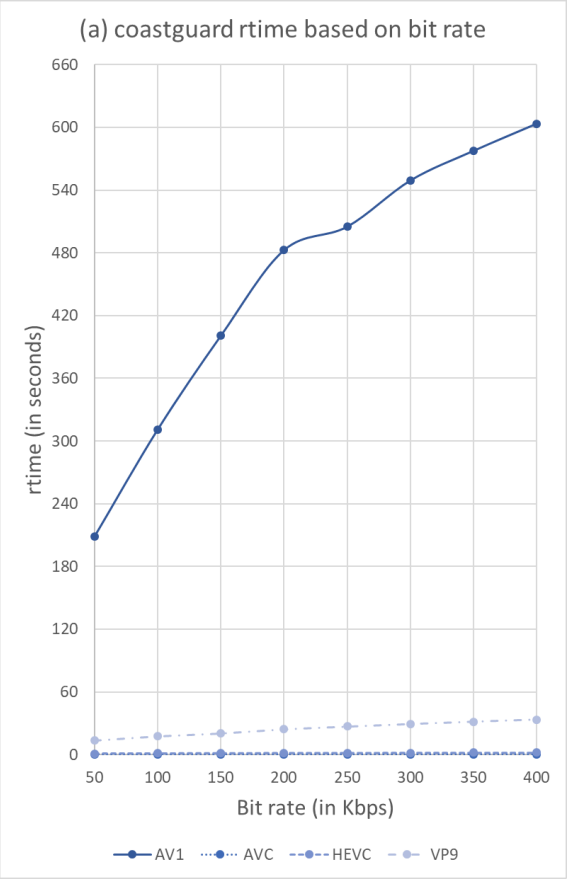
Table 5 – PSNR (in dB) for each codec at a set bit rate for *coastguard*

	50	**100**	**150**	**200**	**250**	**300**	**350**	**400**
AV1	27.3267	29.4802	30.7044	31.7267	32.6204	33.5853	34.2505	34.7957
AVC	23.3351	25.581	27.0528	28.0589	28.8616	29.4013	30.0291	30.5025
HEVC	25.0818	27.0304	28.2164	29.213	29.7656	30.4087	30.9513	31.4682
VP9	26.3562	28.6477	30.1853	31.4302	32.2468	32.8644	33.91	34.5458

Source: The author

The data from Tables 4 and 5 is shown in Figures 17.a and 17.b, respectively.

Figure 17 – Encoding time (a) and PSNR (b) for each codec at a set bit rate for *coastguard*



Source: The author

The next example is *foreman*. It consists of 300 frames (10 seconds at 30FPS) at 352x288 pixels. The original video file is 44.5MB. Table 6 shows the encoding time (rtime) for each codec at a set bit rate, in kbps. Table 7 shows the PSNR for the same scenarios.

Table 6 – Encoding time (in seconds) for each codec at a set bit rate for *foreman*

	50	**100**	**150**	**200**	**250**	**300**	**350**	**400**
AV1	176.943	252.217	312.621	384.206	413.396	451.464	444.927	476.337
AVC	0.232	0.301	0.354	0.384	0.405	0.437	0.456	0.475
HEVC	1.272	1.544	1.715	1.876	1.994	2.116	2.226	2.318
VP9	12.849	14.874	17.23	19.531	21.857	23.047	23.036	24.978

Source: The author

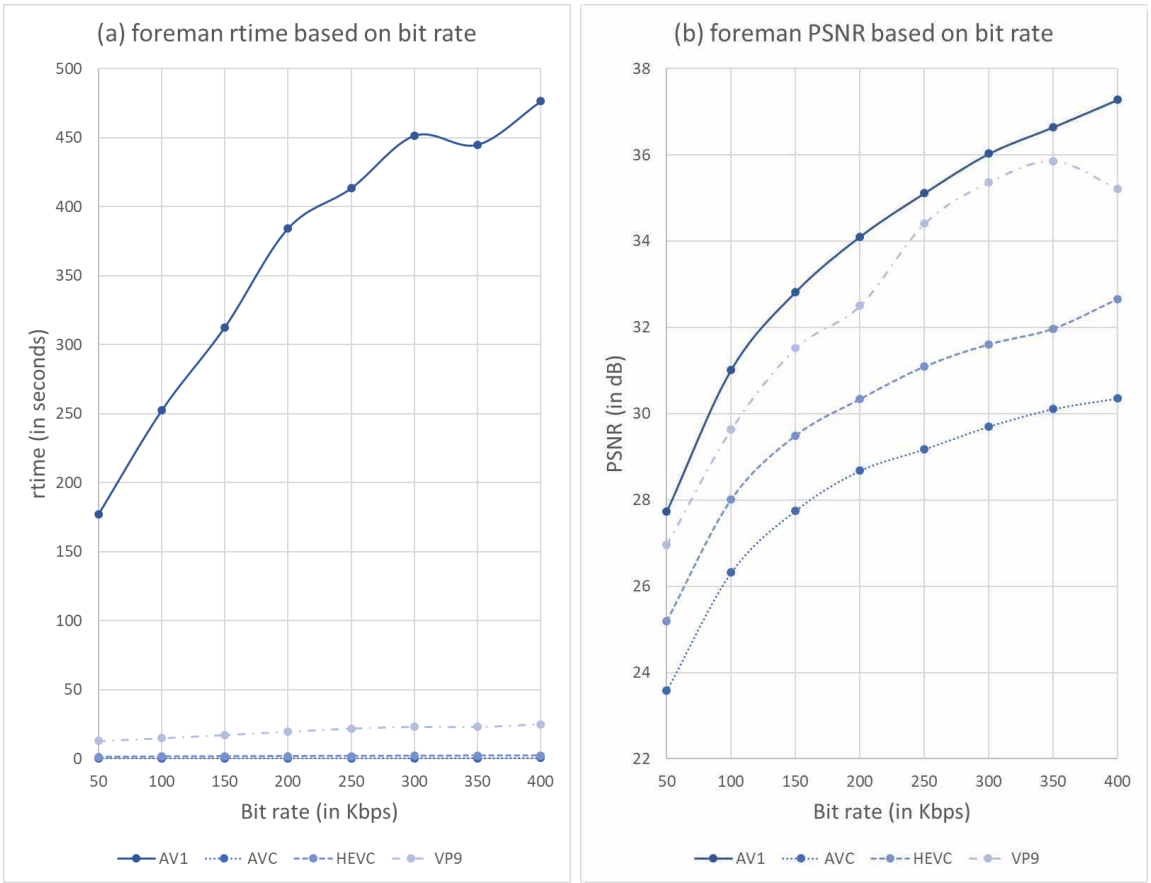
Table 7 – PSNR (in dB) for each codec at a set bit rate for *foreman*

	50	**100**	**150**	**200**	**250**	**300**	**350**	**400**
AV1	27.7318	31.0194	32.8241	34.1054	35.1137	36.0271	36.6426	37.2767
AVC	23.5813	26.3153	27.755	28.6781	29.1765	29.7016	30.1096	30.3531
HEVC	25.1945	28.0088	29.4918	30.3374	31.0927	31.6072	31.9721	32.6616
VP9	26.96	29.6355	31.5152	32.5045	34.4079	35.3638	35.8583	35.2164

Source: The author

The data from Tables 6 and 7 is shown in Figures 18.a and 18.b, respectively.

Figure 18 – Encoding time (a) and PSNR (b) for each codec at a set bit rate for *foreman*



Source: The author

The last CIF sample is *mobile*. It consists of 300 frames (10 seconds at 30FPS) at 352x288 pixels. The original video file is 44.5MB. Table 8 shows the encoding time (rtime) for each codec at a set bit rate, in kbps. Table 9 shows the PSNR for the same scenarios.

Table 8 – Encoding time (in seconds) for each codec at a set bit rate for *mobile*

	50	**100**	**150**	**200**	**250**	**300**	**350**	**400**
AV1	285.68	291.855	312.238	349.721	393.803	425.681	469.946	509.256
AVC	0.316	0.304	0.333	0.411	0.401	0.437	0.44	0.47
HEVC	1.048	1.243	1.443	1.582	1.69	1.784	1.896	1.994
VP9	27.82	20.879	21.852	21.822	24.965	26.159	28.015	30.084

Source: The author

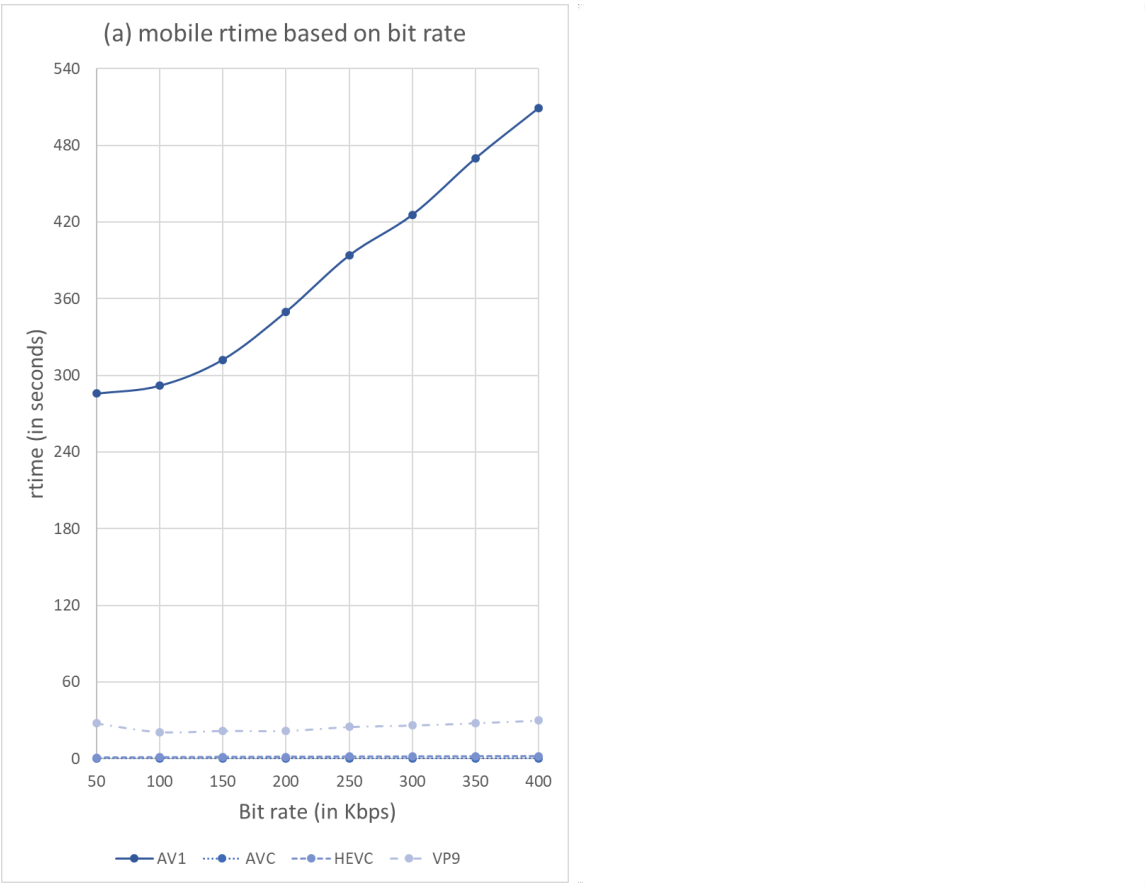
Table 9 – PSNR (in dB) for each codec at a set bit rate for *mobile*

	50	**100**	**150**	**200**	**250**	**300**	**350**	**400**
AV1	22.4874	25.0012	26.8218	28.2892	29.2279	30.1135	30.8433	31.7534
AVC	17.7252	20.6418	21.9809	23.1153	23.7466	24.5309	25.0966	25.6819
HEVC	18.7205	22.3892	23.956	25.0148	25.7741	26.5088	27.0488	27.568
VP9	18.6924	22.4326	24.4011	25.4598	26.6643	26.9475	27.4415	27.9349

Source: The author

The data from Tables 8 and 9 is shown in Figures 19.a and 19.b, respectively.

Figure 19 – Encoding time (a) and PSNR (b) for each codec at a set bit rate for *mobile*



Source: The author

A common trend can be observed. In all cases, AV1 offers the best image quality at the expense of encoding time. AVC and HEVC are both very fast encoders, even though HEVC is, on average, four to five times slower than AVC. Then there is VP9, which is a degree of magnitude greater. At default settings, VP9 is slower than HEVC and it often has better image quality, sometimes matching AV1, which is considerably slower. It is so slow that it is impractical for any real-time applications.

It can also be observed that AV1 sits on top of all image quality comparisons for the same bit rate. VP9 also performs well, albeit inconsistently. In some situations, VP9 shows results similar to AV1, while in other tests it more closely matches HEVC. HEVC has a very competitive performance, offering satisfactory image quality at a fraction of the encoding time. Finally, AVC is the worst performer in all situations. However, it is the fastest, proving itself as the best option for situations where speed is the most important factor.

The true definitive advantage of AV1 is in situations where encoding time is not a concern. Those situations include video sharing and streaming. When a video is uploaded to YouTube, for example, it needs to be encoded on a handful of different presets and then it is

reproduced millions or even billions of times. Similarly, streaming services, such as Netflix, only need to encode the videos once, before distributing them. The same episode will be streamed by millions of people. In both cases, the bandwidth savings far overcompensates the slower encoding time.

While all the samples above had similar results, nobody would consider 288p video acceptable nowadays. More and more, Full HD video has become the standard for quality. For that reason, a 1080p sample, *pedestrian*, is also analyzed. It consists of 375 frames (15 seconds at 25FPS) at 1920x1080 pixels. The original video file is 1.08GB. Table 10 shows the encoding time (rtime) for each codec at a set bit rate, in kbps. Table 11 shows the PSNR for the same scenarios.

Table 10 – Encoding time (in seconds) for each codec at a set bit rate for *pedestrian*

	1000	**1500**	**2000**
AV1	3275.16	3932.911	4566.42
AVC	6.555	7.637	8.391
HEVC	21.441	23.478	25.217
VP9	116.263	125.38	131.715

Source: The author

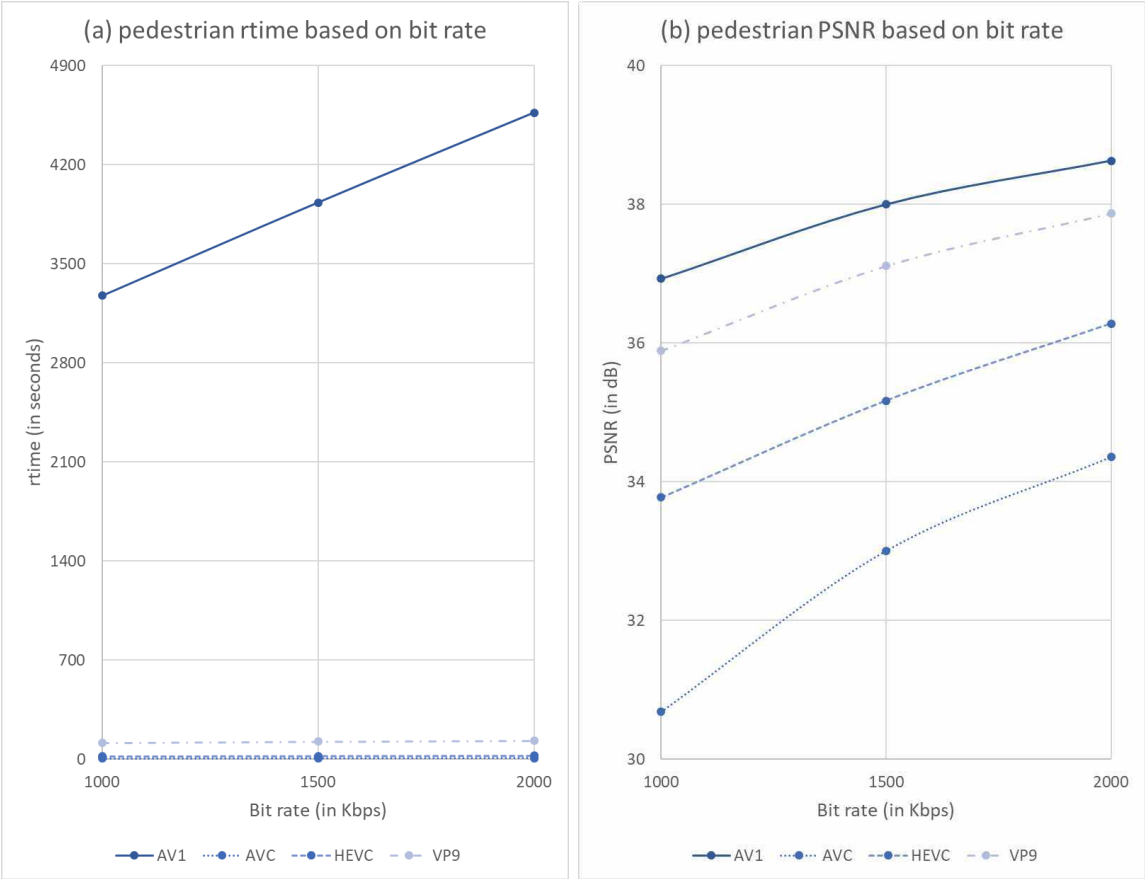
Table 11 – PSNR (in dB) for each codec at a set bit rate for *pedestrian*

	1000	**1500**	**2000**
AV1	36.9286	37.9988	38.6272
AVC	30.6865	33.009	34.3593
HEVC	33.775	35.1712	36.2793
VP9	35.8905	37.1111	37.8653

Source: The author

The data from Tables 10 and 11 is shown in Figures 20.a and 20.b, respectively.

Figure 20 – Encoding time (a) and PSNR (b) for each codec at a set bit rate for *pedestrian*



Source: The author

This sample shows how prohibitive AV1 is for everyday use. A 15-second video can take up to one hour and 15 minutes to encode at a higher bit rate. This is over 300 times real-time performance. The other codecs appear flat by comparison. Meanwhile, looking at the PSNR, the results are in line with the previous comparisons: AV1 at the top followed by VP9 with HEVC in third and AVC in a distant last position.

Probably the most important comparison here is between AV1 and HEVC, considering both are fighting to conquer the title of next-generation codec. Encoding time aside, it is interesting to observe how much better compression AV1 can offer. Figures 21 and 22 show the same frame (at the 5-second mark) in both videos at a bit rate of 1Mbps.

Figure 21 – 125th frame of *pedestrian* encoded in AV1 at 1Mbps

Source: The author

Figure 22 – 125th frame of *pedestrian* encoded in HEVC at 1Mbps

Source: The author

The difference is visible. AV1 offers a much smoother image, while HEVC shows noticeable compression blocks. The ground has a lot of noise in HEVC's image. Areas with movement also look more degraded with HEVC while AV1 depicts more detail on static areas.

Aiming for the future, both HEVC and AV1 had a lot of their development focused on UHD video. Therefore, a 4K sample, *ShakeNDry*, was also included. It consists of 300 frames (10 seconds at 30FPS) at 3840x2160 pixels. The original video file is 3.47GB. Table 12 shows the encoding time (rtime) for each codec at a set bit rate, in kbps. Table 13 shows the PSNR for the same scenarios.

Table 12 – Encoding time (in seconds) for each codec at a set bit rate for *ShakeNDry*

	4000	**6000**	**8000**
AV1	11627.362	18128.738	24580.59
AVC	16.662	19.482	21.923
HEVC	59.877	66.669	83.883
VP9	355.877	416.845	527.905

Source: The author

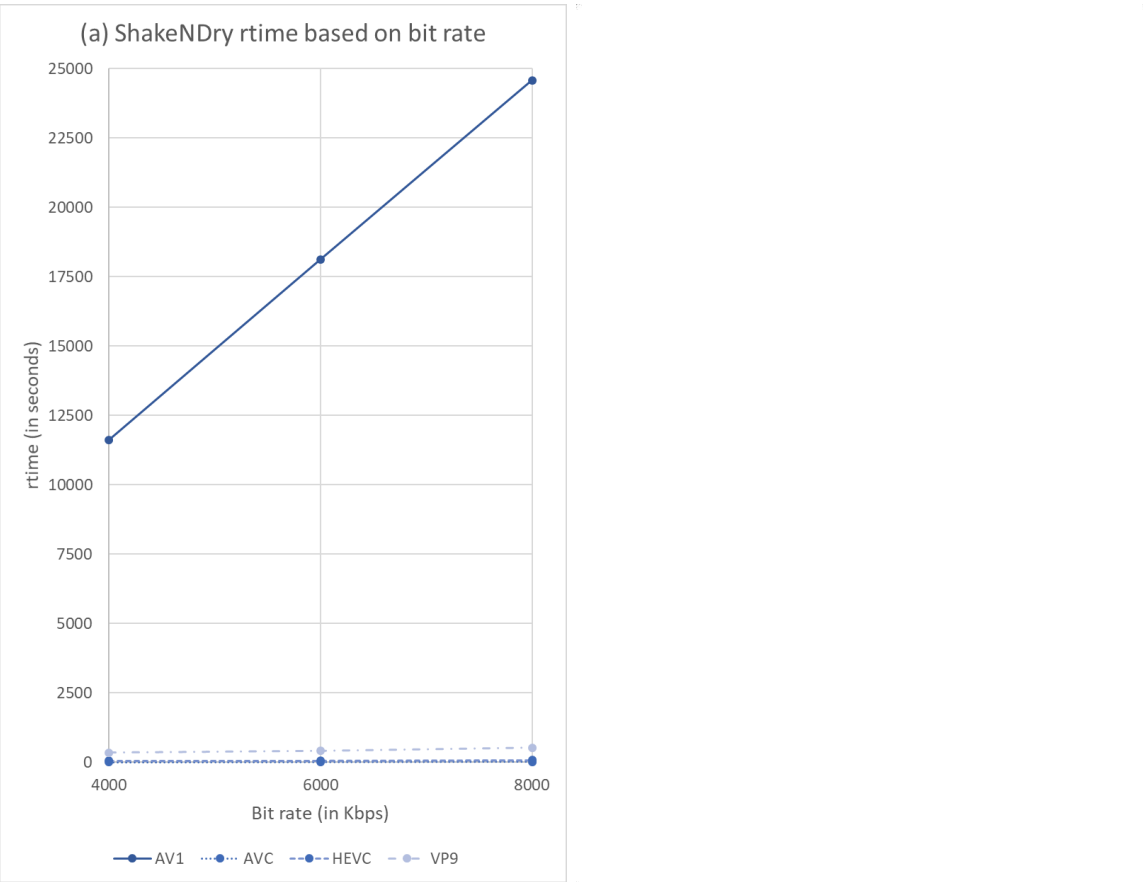
Table 13 – PSNR (in dB) for each codec at a set bit rate for *ShakeNDry*

	4000	**6000**	**8000**
AV1	36.6146	36.9789	37.4337
AVC	33.128	33.9255	34.4364
HEVC	34.7181	35.0872	35.4637
VP9	35.8309	36.1482	36.4011

Source: The author

The data from Tables 12 and 13 is shown in Figures 23.a and 23.b, respectively.

Figure 23 – Encoding time (a) and PSNR (b) for each codec at a set bit rate for *ShakeNDry*



Source: The author

As with pedestrian, the encoding time for AV1 is ludicrous. A single frame of a 4K video at 4Mbps takes nearly 40 seconds to encode. Doubling the bit rate brings the value to over 80 seconds. At this resolution, decoding also starts to become an issue. Due to more complex algorithms, decoding AV1 video requires more processing power and, consequently, more battery power.

HEVC has been in development for much longer and its bitstream has already been finalized. Moreover, several devices already have hardware support: Intel 6th-generation Skylake CPUs or newer, AMD 6th-generation Carizzo APUs or newer, AMD Fiji GPUs or newer, NVIDIA GM206 GPUs or newer, Qualcomm Snapdragon 805 SoCs or newer, Samsung Exynos 5 Octa 5430 SoCs or newer, and Apple A8 SoCs or newer (TechSpot, 2016). All these devices can take advantage of hardware decoding, which means faster decoding with significantly lower battery consumption. As mentioned before, the first devices are expected to offer hardware support for AV1 one year after the bitstream is finalized.

As a result, AV1 is restricted to software decoding at this time. More powerful devices can handle the load, but power-restricted devices such as low-power laptops and older

smartphones may struggle. More efficient decoders and hardware support are expected to greatly improve the situation.

As far as image quality goes, Figures 24 and 25 show how both codecs handle UHD video. They were taken at the 2-second mark at the bit rate of 6Mbps.

Figure 24 – 60th frame of *ShakeNDry* encoded in AV1 at 6Mbps

Source: The author

Figure 25 – 60th frame of *ShakeNDry* encoded in HEVC at 6Mbps

Source: The author

The visual differences between AV1 and HEVC are less stark in this example. AV1 again looks softer while HEVC has more noise, probably due to better deblocking algorithms with AV1. This results in more refined detail of the water particles, observed on HEVC, while AV1 displays a more visually pleasing background.

Finally, it is interesting to note just how efficient both codecs are. At 4Mbps, the encoded files are just a little over 4MB in size. In comparison, the original uncompressed file is almost 800 times larger.

4.2 AV1 performance improvements and the cpu-used parameter

The second part of this comparison is focused on AV1. Since the codec is still in development, constant improvements are being made. There are two key aspects to be examined: the performance improvements and the effect of the *cpu-used* parameter in the speed and efficiency of the codec.

The two builds compared include the following AV1 libraries: 20190221-cde3872 (February 21, 2019 – Stable) and 20190425-9feb158 (April 25, 2019 – Nightly). The samples chosen for this comparison were *foreman* and *pedestrian*. In both cases, the samples were encoded using both builds varying the *cpu-used* parameter from zero to eight.

The encoding time (in seconds) and PSNR (in dB) for foreman are shown in tables 14 and 15, respectively.

Table 14 – Encoding time (in seconds) for each cpu-used integer for *foreman*

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Stable	1454.74	516.94	247.27	148.49	109.59	91.34	85.14	67.67	58.64
Nightly	1102.92	313	223.32	105.44	82.44	68.99	64.24	51.81	44.2

Source: The author

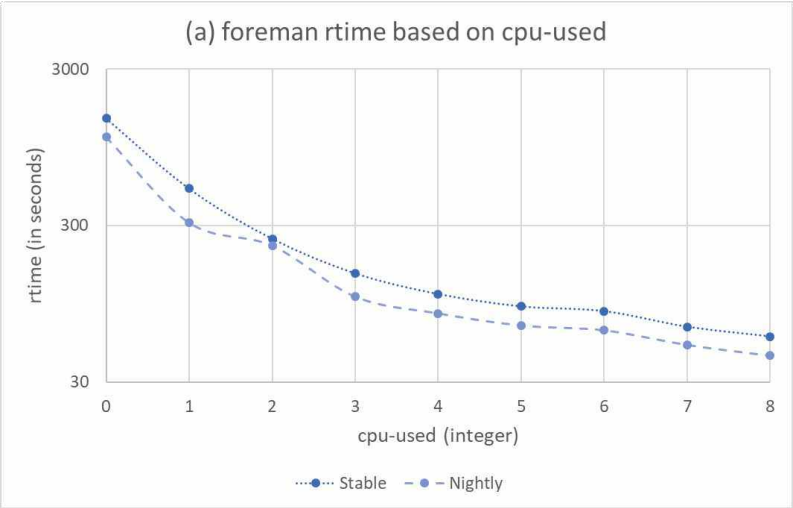
Table 15 – PSNR (in dB) for each cpu-used integer for *foreman*

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Stable	32.854	32.857	32.775	32.73	32.742	32.51	32.638	31.863	31.152
Nightly	32.812	32.842	32.799	32.721	32.677	32.493	32.557	31.73	31.047

Source: The author

The data from Tables 14 and 15 is shown in Figures 26.a and 26.b, respectively.

Figure 26 – Encoding time (a) and PSNR (b) for each cpu-used integer for *foreman*



Source: The author

Likewise, the same results for *pedestrian* are shown in tables 16 and 17, respectively.

Table 16 – Encoding time (in seconds) for each cpu-used integer for *pedestrian*

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Stable	9713.2	4065.3	2684.1	1829.4	1323.5	1148.6	1089.5	877.7	767.3
Nightly	8514.2	3877.8	2552.1	1221.2	974.6	826.6	774.6	629.0	550.1

Source: The author

Table 17 – PSNR (in dB) for each cpu-used integer for *pedestrian*

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Stable	38.02	37.97	37.95	37.89	37.82	37.78	37.72	37.40	37.32
Nightly	38.12	38.00	37.94	37.92	37.87	37.79	37.78	37.68	37.65

Source: The author

The data from Tables 16 and 17 is shown in Figures 27.a and 27.b, respectively.

Figure 27 – Encoding time (a) and PSNR (b) for each cpu-used integer for *pedestrian*

Source: The author

The most challenging aspect of AV1 in all tests so far has been how slow it is compared to the other codecs. Two factors may improve this situation: codec optimizations and encoding parameters.

Since its inception, the AV1 code has been gradually altered with new experiments aiming to improve its compression efficiency. The side effect of this has been complexity. AV1 managed to reach stellar levels of visual quality at the expense of time and processing power.

After the bitstream was frozen, the focus shifted to optimizing the existing tools. FFMPEG was one of the first tools to pick up AV1 support, starting with v4.0 (April 20, 2018). The two versions tested (v4.1.1 and v20190430-19af948) are only two months apart but the improvement is visible. On average, the newer library is 30 to 35 percent faster.

The other alternative to speed up the encoding process is changing the *cpu-used* parameter. The tradeoff between speed and quality is set to one by default, which translates to efficient but slow compression. Setting the value to three offers about three times as fast encoding with little loss in quality. Setting the value to five makes the encoding process 4.5 times faster, even if image quality suffers a bit. Setting values higher than six may speed things up even further, but there is a steep drop in quality. From these results, it is evident that setting *cpu-used* in the three to five range seems to be the sweet spot for most applications. It offers sizeable gains in performance while bringing minor depreciation of image quality.

CHAPTER 5 – CONCLUSIONS

After studying the key aspects and features of the most popular codecs as well as their performance and efficiency, it is clear to see that AV1 is very promising. Across the board, it offers the best possible image quality, making it possible to transmit and store higher definition video at a smaller size. Its main limitation is performance; it still demands a lot of processing power, and it lacks hardware support.

Fortunately, future updates to its library will likely help optimize the encoding and decoding process, with custom alternatives already showing up. As of now, its use in real-time applications is unconceivable as its encoding time and processing power requirements are sky high. Situations where these aspects are not a factor may benefit of AV1's capabilities, with YouTube being one of the first to implement the new codec. Until then, adjustments to the encoder parameters may help AV1 fit users' needs, given that the default settings are heavily focused on quality.

Looking at the competing codecs, AVC is clearly showing its age, as it renders results far behind all the others. However, since it is the least complex of the group, its speed is unmatched. VP9 also shows promising results, but it has failed to achieve the widespread use Google hoped it would achieve. Finally, while HEVC has had a troubled start with obscure patents and licensing terms, it shows an attractive balance between speed and quality.

REFERENCES

Bordes, P., Clare, G., Henry, F., Raulet, M., & Viéron, J. (2012). An overview of the emerging HEVC standard. *IEEE*, 4.

Bross, B., Helle, P., Lakshman, H., & Ugur, K. (2014). Inter-picture prediction in HEVC. *High Efficiency Video Coding (HEVC): Algorithms and Architectures*, 113-140.

Chiariglione, L. (2018). A crisis, the causes and a solution. Retrieved from http://blog.chiariglione.org/2018/01/

Cnet. (2012). Qualcomm shows horsepower of next-gen H.265 video. Retrieved from https://www.cnet.com/news/qualcomm-shows-horsepower-of-next-gen-h-265-video/

CoolUtils. (2009). What is MTS? Retrieved from https://www.coolutils.com/Formats/MTS

Cpapciak. (2012). Most common types of video files and containers. Retrieved from http://www.dvdyourmemories.com/blog/types-of-video-files-containers/

Dummies Insider. (2009). How to create and run a PowerShell script. Retrieved from https://www.dummies.com/computers/operating-systems/windows-xp-vista/create-run-powershell-script/

Egge, N., & Bebenita, M. (2018). Mozilla TechSpeakers: AVI video codec. Retrieved from https://docs.google.com/presentation/d/1lwjqzCnKCrJbPVillqJJp91bDQLAHh14DV69t6cLTn0/edit#slide=id.g25d7c72cd9_0_10

FFMPEG. (2019). Ffmpeg documentation. Retrieved from https://ffmpeg.org/ffmpeg-all.html

Fitzer, M. (2009). What is a codec? Retrieved from https://www.videomaker.com/article/f6/14743-what-is-a-codec

Harrington, R., & Krogh, P. (2012). Video file format overview. Retrieved from https://www.dpbestflow.org/Video_Format_Overview

Harrington, R., & Krogh, P. (2015). Video signal attributes. Retrieved from https://www.dpbestflow.org/node/624

HiDef Audio and Video Sales and Installation. (2015). Video. Retrieved from http://hidefnj.com/video.html

IPFS. (2017). Uncompressed video. Retrieved from https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/Uncompressed_video.html

Layek, A., Thai, Q., Hossain, A., Thu, T., Tuyen, P., Talukder, A., . . . Huh, E.-N. (2017). Performance analysis of H.264, H.265, VP9 and AV1 video encoders. *IEEE*, 322-325.

Matroska. (2016). What is Matroska? Retrieved from https://www.matroska.org/technical/whatis/index.html

Motion Elements. (2013). What you need to know about the 5 most common video file formats. Retrieved from https://www.motionelements.com/blog/articles/what-you-need-to-know-about-the-5-most-common-video-file-formats

Mukherjee, D., Bankoski, J., Grange, A., Han, J., Koleszar, J., Wilkins, P., . . . Bultje, R. (2015). The latest open-source video codec VP9 – An overview and preliminary results. *SMPTE Motion Imaging Journal*, 4.

Ozer, J. (2017). AV1: A status update. Retrieved from http://www.streamingmedia.com/Articles/Editorial/Featured-Articles/AV1-A-Status-Update-120214.aspx

Ozer, J. (2018). The future of HEVC licensing is bleak, declares MPEG chairman. Retrieved from http://www.streamingmedia.com/Articles/Editorial/Featured-Articles/The-Future-of-HEVC-Licensing-Is-Bleak-Declares-MPEG-Chairman-122983.aspx

Poynton, C. (2012). *Digital video and HD: Algorithms and interfaces* (2nd ed.). MK.

Qz. (2016). The internet has been quietly rewired, and video is the reason why. Retrieved from https://qz.com/742474/how-streaming-video-changed-the-shape-of-the-internet/

Rbultje. (2016). Overview of the VP9 video codec. Retrieved from https://blogs.gnome.org/rbultje/2016/12/13/overview-of-the-vp9-video-codec/

Richardson, I. E. (2003). *The H.264 Advanced Video Compression standard* (2nd ed.). Wiley.

Sethuraman, S., Rajan, C., & Patankar, K. (2017). Analysis of the emerging AOMedia video coding format for OTT use-cases.

Sonnati. (2014). H265 – part I: Technical overview. Retrieved from https://sonnati.wordpress.com/2014/06/20/h265-part-i-technical-overview/

Sonnati. (2016). Does VP9 deserve attention? – part I. Retrieved from https://sonnati.wordpress.com/2016/06/03/does-vp9-deserve-attention-part-i/

Sullivan, G., Ohm, J.-R., Han, W.-J., & Wiegand, T. (2012). Overview of the High Efficiency Video Coding (HEVC) standard. *IEEE transactions on circuits and systems for video technology, 22*(12), 1649-1668.

TechSpot. (2016). Guide to HEVC/H.265 encoding and playback - TechSpot. Retrieved from https://www.techspot.com/article/1131-hevc-h256-enconding-playback/

Ultimate Photo Tips. (2010). What is a pixel? Retrieved from http://www.ultimate-photo-tips.com/what-is-a-pixel.html

Ultra Video Group. (2013). Ultra Video Group. Retrieved from http://ultravideo.cs.tut.fi/#testsequences

Vcodex. (2007). An overview of H.264 Advanced Video Coding. Retrieved from https://www.vcodex.com/an-overview-of-h264-advanced-video-coding/

Wolfcrow. (2013). What is a video container or wrapper? Retrieved from https://wolfcrow.com/blog/what-is-a-video-container-or-wrapper/

Xiph.org. (2006). Xiph.org video test media [derf's collection]. Retrieved from https://media.xiph.org/video/derf/