University of Tartu

Faculty of Science and Technology

Institute of Technology

Hendrik Türk

NVENC-BASED VIDEO TRANSCODER PROTOTYPE

Master's Thesis in Robotics and Computer Engineering (30 ECTS)

Supervisors:

Heiki Kasemägi, Senior Research Fellow

Vahur Zadin, Research Fellow

Tartu 2017

Abstract

NVENC-BASED VIDEO TRANSCODER PROTOTYPE

This thesis is focused on creating a prototype for competitive hardware accelerated video media transcoding solution that is capable of converting video media into HEVC standard. Finding a working solution for a product that could compete with existing HEVC capable hardware accelerators and could be developed into a commercial product, is done through research and prototype testing.

In this thesis:

- an overview of the basis of data compression alongside a selection of software and hardware accelerated transcoding solutions is given;
- a research is conducted through communication with various specialist in different fields of video media creation and handling;
- a prototype system is created and tested alongside reference systems using recognized software transcoding libraries;
- suggestions and remarks for future development are formulated based on conclusions made during research and test results analysis.

CERCS: P160 Statistics, operations research, programming, actuarial mathematics; P170 Computer science, numerical analysis, systems, control; P175 Informatics, systems theory

Keywords: video media, encoder, decoder, transcoder, NVENC, Nvidia, GPU, HEVC, AVC, libx264, libx265

Resümee

NVENC-PÕHINE VIDEOTRANSKODEERIJA PROTOTÜÜP

Käesolev lõputöö on suunatud konkurentsivõimelise HEVC standardisse transkodeeriva süsteemi prototüübi loomisele. Lahenduse leidmiseks uuriti videomeedia haldamise valdkonda ning korraldati prototüübi testimine.

Käesolevas töös:

- antakse ülevaade andmete tihendamistehnoloogiate taustsüsteemist ning eksisteerivatest tarkvaralistest ja riistvaralise kiirendusega videomeedia transkodeerimislahendustest;
- uuritakse videomeedia loomise ja käsitlemise taustsüsteemi, suheldes vastavate erialaspetsialistidega;
- luuakse testimiseks prototüüp koos tunnustatud tarkvaralistel teekidel töötavate võrdluslahendustega;
- kogutud informatsiooni ja testide analüüsi põhjal formuleeritakse tähelepanekud ning ettepanekud edasiseks arendustööks.

CERCS: P160 Statistika, operatsioonianalüüs, programmeerimine, finants- ja kindlustusmatemaatika; P170 Arvutiteadus, arvanalüüs, süsteemid, juhtimine; P175 Informaatika, süsteemiteooria

Märksõnad: videomeedia, enkooder, dekooder, transkooder, NVENC, Nvidia, GPU, HEVC, AVC, libx264, libx265

Table of Contents

List of Figures

List of Tables

Acronyms and Abbreviations

API – Application Programming Interface

ASI – Actuator Sensor Interface

ASIC – Application-Specific Integrated Circuit

AVC – Advanced Video Coding, also known as h.264

CB – Coding Block

CMOS – Complementary Metal-Oxide-Semiconductor

CPU – Central Processing Unit

CTB – Coding Tree Block

CTU – Coding Tree Unit

CU – Coding Unit

CUDA – Compute Unified Device Architecture

DCT – Discrete Cosine Transform

DPCM – Differential Pulse Code Modulation

DSL – Digital Subscriber Line

EAS (*Ettevõtluse Arendamise Sihtasutus*) – Enterprise Estonia

FPGA – Field-Programmable Gate Array

FTPS – Frames Transcoded Per Second

GOP – Group of Pictures

GPU – Graphics Processing Unit

GUI – Graphical User Interface

HD video – High-Definition video

HEVC – High Efficiency Video Coding, also known as h.265

IP – Internet Protocol

IP core – Intellectual Property core

ISO – International Organization for Standardization

ITU – International Telecommunication Union

MPEG – Moving Picture Experts Group

NVDEC – hardware-based decoder on Nvidia GPUs (started from Kepler generation)

NVENC – hardware-based encoder on Nvidia GPUs (started from Kepler generation)

OS – Operating System

PB – Prediction Block

PCIe – an input-output interconnect bus standard

PUMPS summer school– Programming and Tuning Massively Parallel Systems summer school

RAW data – uncompressed data

SDK – Software Development Kit

TB –Transform Block

TV – short for television

UHD video – Ultra High Definition video

WPP – Wavefront Parallel Processing

1. Introduction

Video media consumption and creation has become more accessible in increased quality through rapid development of mobile devices, digital cameras, UHD TVs, etc. High-quality media improves user experience, but also requires significantly higher network bandwidth, more computational performance and storage space [1]. Additionally, many video coding standards do not support higher resolutions and bit depths that come with high-quality media [2, p. 7]. For relieving these problems, new coding algorithms that are capable of higher coding efficiency, like HEVC, also known as h.265, and VP9, have been designed [1]. Increased efficiency comes with higher computational complexity, meaning that the encoding process takes less time on similar hardware for AVC encoder than for HEVC encoder [3, 4].

Coding efficiency directly influences the quality of video media that is sent to consumer through limited bandwidth connections [1]. It can be understood that service providers, streamers and other content creators, capable of delivering video media in UHD quality to consumer with capable devices, get an advantage over competitors that use higher bitrates for achieving the same quality. As the number of consumers with capable devices increases, the encoding and transcoding load for devices, used by service providers and content creators, also grows. Hardware transcoding systems, not capable of HEVC coding, have to be replaced and software solutions will have to be upgraded in order to keep up with increased compactional complexity. This forced upgrade creates an opportunity for transcoding device designers and software creators, who can provide competitive solutions for optimal prices, to sell their product in high volumes. Furthermore, because of the high volumes and the need for new solutions, the market situation should be open for newcomers, especially if they are capable of adding some additional value compared to established hardware and software creators.

Enterprises often prefer to use software or black-box-like solutions that are easy to maintain and upgrade [5]. Software transcoders might not be reliable enough for applications like professional grade live streaming, and in power- or space-restricted situations. These needs are typically addressed by using hardware-accelerated black box solutions that are capable of real-time video transcoding. Hardware accelerators allow the computationally demanding transcoding to be offloaded from CPUs that can then be assigned with other tasks.

The purpose of this thesis is to create and test a prototype of a black-box-like video transcoding solution that could compete in price and performance with systems that are capable of real-time

or faster than real-time transcoding from less compact video formats into HEVC standard. The needs for video media transcoding and already existing solutions are being ascertained through primary and secondary research. Prototype is tested with both, YUV RAW and AVC sequences, in order to get an overview of the encoding capabilities of the system. Suggestions for future development of the transcoding solution will be formulated based on the research and test results.

2. Overview of Media Compression

2.1. Digital Video

Video is the visual information captured by a camera that consists of time-varying sequence of pictures [6, p. 2]. Early video cameras were electromechanical until first cathode ray tube cameras were introduced. Recorded signals were stored on analogue video cassettes or analogue laser discs. For processing in digital domain, analogue signals have to be encoded into digital formats. In 1980s, solid-state sensors started to replace the analogue tube cameras [6, p. 2]. Active pixel sensors like CMOS enabled the use of digital video.

Digital video signals can be compressed and subsequently decompressed. Analogue signals are not amendable to compression and have to be converted to digital formats before they can be used in digital workflow [6, p. 2]. Relatively inexpensive integrated circuits, efficient video compression techniques, dense storage media and high-speed communication networks have made all-digital workflow widespread for both everyday consumers and professionals.

2.2. Data Compression

Compact representation of information can be achieved through data compression. It is possible through redundancy that is present in most representations of information [7, p. 423]. This means that there is data that can be removed without losing the essential information. Typically, redundancy can be found through correlation, context, probability, nature of information, characteristics of the user etc. For example, spatially close pixels on an image are generally also close in value [7, p. 423]. Also, high spatial frequency is redundant for some applications, usually because humans cannot see above certain spatial frequency. Redundancy removal has historically been called decorrelation [7, p. 423]. Additionally, in the binary representation, reoccurring parts of code can be substituted with shorter bit lengths. These different redundancies allow compression schemes to be divided into lossless and lossy compression. The general compression structure used in modern image compression standards can be seen on Figure 1 [8].

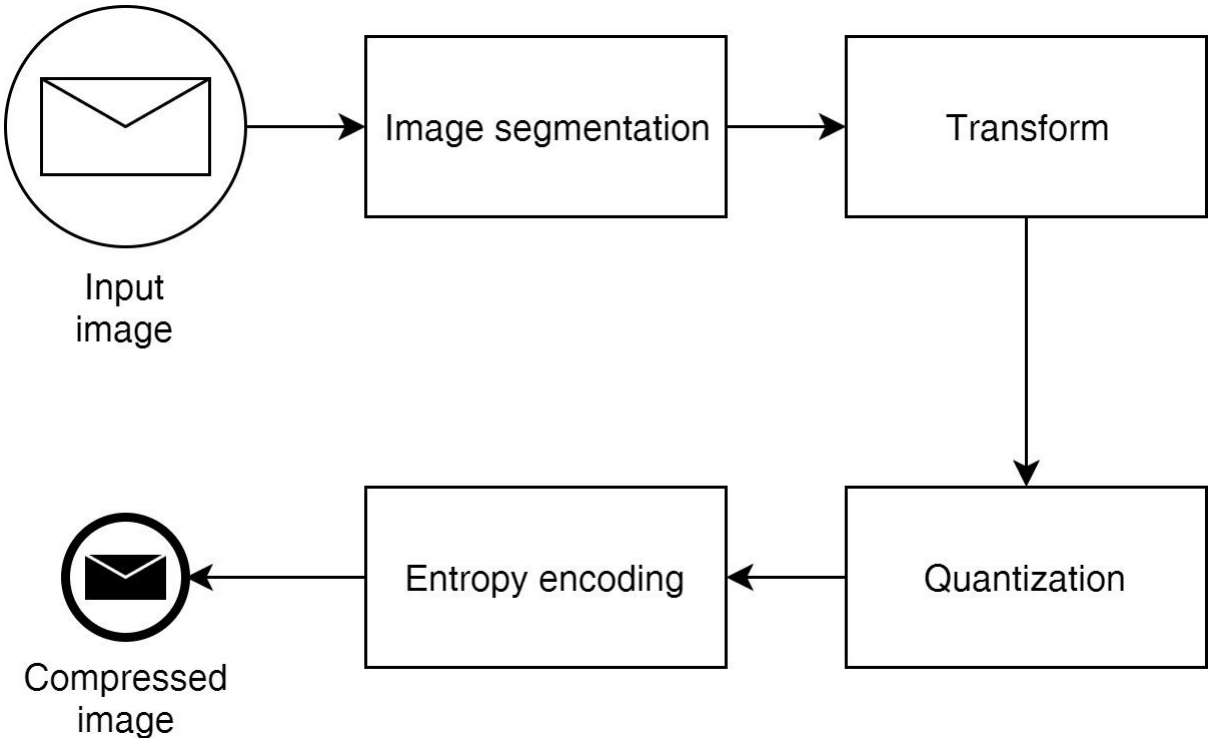


Figure 1. General structure used in modern image compression standards

2.2.1. Lossless Compression

Lossless compression, as its name states, preserves all the information through the compression and decompression operations [7, p. 424]. As no data is lost during lossless compression, the original data can be fully reconstructed from the compression result. This strictly limits the amount of compression available for the target information.

There is a variety of clever techniques for doing lossless compression. It is hard to pinpoint the exact beginning of data compression, however theoretical footing was established by Claude E. Shannon [7, p. 424], who formed the basis of a mathematical representation of the communication process in the 20th century. He described a system, known as entropy coding, where less steps and thus less bits are needed for solving more probable solutions. Representative scheme can be seen on Figure 2. On this figure, it can be seen that more probable solutions A and B are represented by single bit and less probable solutions C and D by two bits. It should be noted that incorrect probability estimates will decrease the compression efficiency of the procedure.

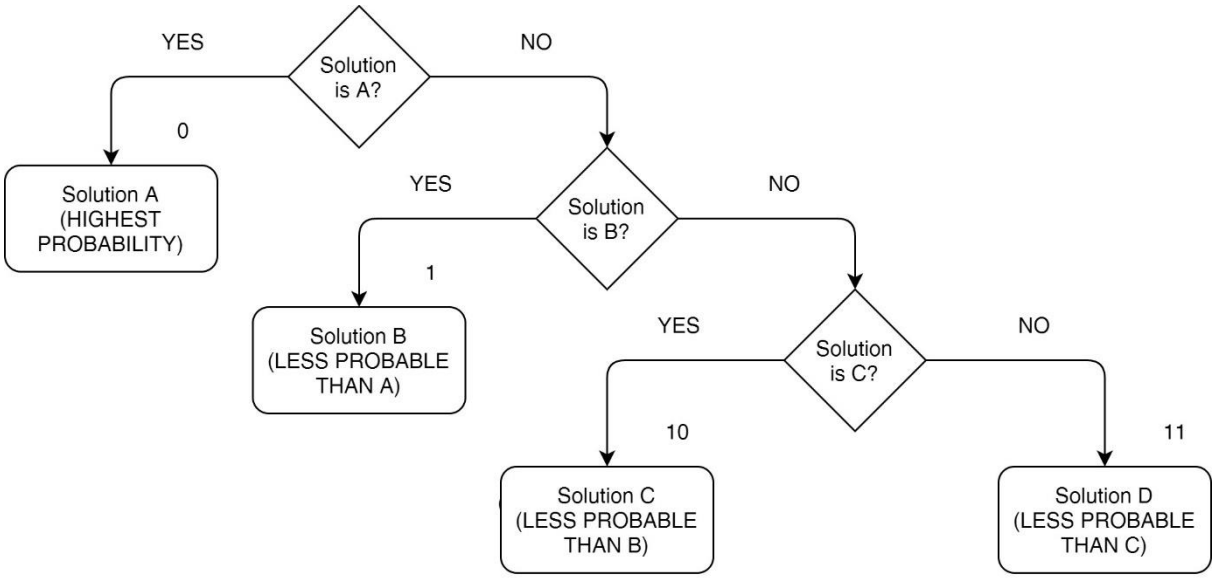


Figure 2. A representative scheme of the idea behind a simple entropy coding lossless compression technique

In order to systematically generate bit sequences with variable lengths for a given source, several coding algorithms like Huffman coding and arithmetic coding have been created. Huffman coding uses three rules for creating the code [7, p. 425]:

1. the codes corresponding to the higher probability solutions cannot be longer than the codes for the lower probability solutions;
2. the code words for two lowest probability solutions have to have the same length;
3. the codes for two lowest probability solutions are identical except for the last bit.

An example on Figure 3 [7, p. 426] visualizes the Huffman code binary tree for a problem with five solutions: A, B, C, D and E. It can be seen that zero bit denotes the left branch and one bit denotes the right side. Furthermore, the code words corresponding to the tree leaves are shown next to the tree. More information on arithmetic and lossless compression can be found from Appendix 2.

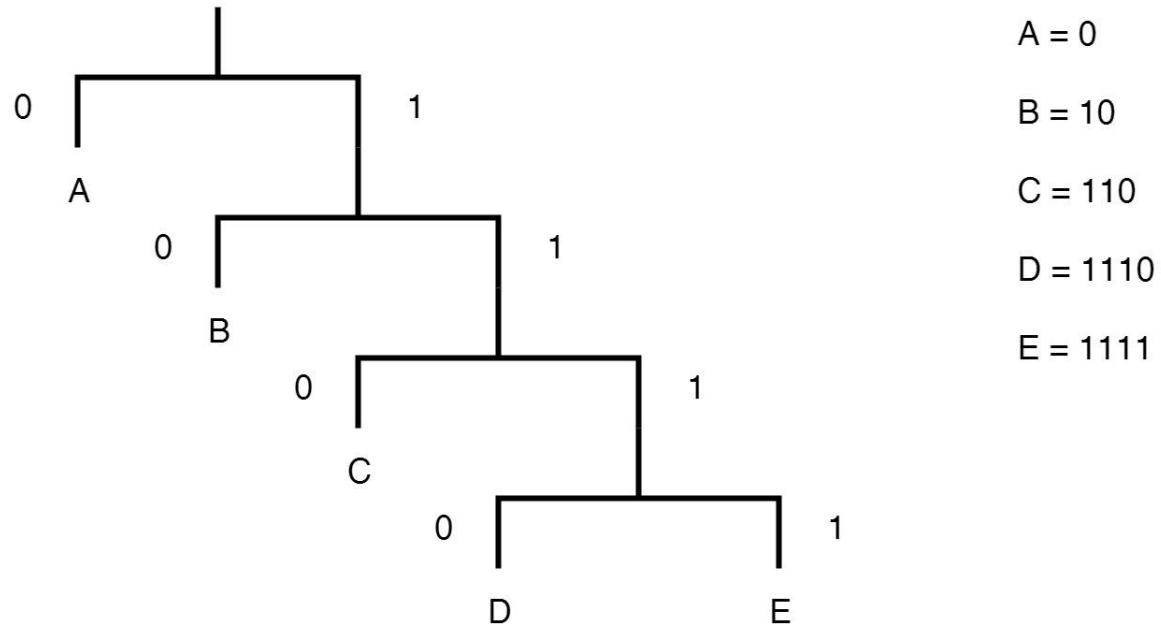


Figure 3. Example of the Huffman code

2.2.2. Lossy Compression

Lossy compression algorithms are used for overcoming limitations of lossless algorithms and achieving more extensive compression rates. In many usage cases, lossless data handling has no major benefits over systems with lossy compression algorithms. This depends on the available resources and the characteristics of the content's consumers. For example, humans do not perceive high-frequency information that is present in many picture, video and audio files [7, p. 432]. With the cost of information loss, greater compression rates can be achieved. Even if loss of data is evident, the content might still be usable, and the reduced file sizes would lighten the storage and bandwidth requirements for some applications.

Lossy compression algorithms often use quantization techniques where a range of values is represented by one value. In other words, quantization is the process of mapping a continuous or discrete scalar or vector into a set of symbols that can be represented with finite number of bits [9]. For example, in case of colours, all tones of red could be represented with one certain tone of red. This many-to-one mapping increases compression rate but is irreversible as all original colours cannot be recreated from the compressed data. Additional information on quantization and other lossy information compressing techniques that are used on images and videos can be found in Appendix 3.

2.3. Video Compression

Video media is an excellent target for compression as it is one of the most information-heavy media formats. The compression of video data usually consists of encoding video media, transmitting or storing the compressed signal and finally decoding it when a decompressed version is being presented to the consumer [6, p. 3]. Similar to images, some loss of information is acceptable without resulting in perceivable loss of quality for a human eye. Achieving the compressed data representation trough encoding process of a video is usually much more complex and uses more signal processing operations than video decoding. Thus, playing back the compressed video in real time can be done on hardware that is less powerful than the hardware that is used in real-time encoding. Decoders and encoders are often compounded into *codecs* [10].

Videos can be viewed as sequences of images which makes video compression somewhat similar to image compression. This means that most of the techniques used in image compression algorithms can also be used for each frame in video media. For example, it is used in motion JPEG video format where each frame is compressed through JPEG image compression algorithm. In addition to traditional image compression techniques, video sequences have other attributes that could be used for achieving greater compression. Typically, a video or sequence of frames has some correlation between successive images. In many cases, it is more efficient to send the differences between neighbouring frames rather than the full frames themselves [7, p. 441]. This idea is used in most international video compression standards, including H.261 and H.263. The block diagram for H.261 can be seen on Figure 4 [7, p. 442].

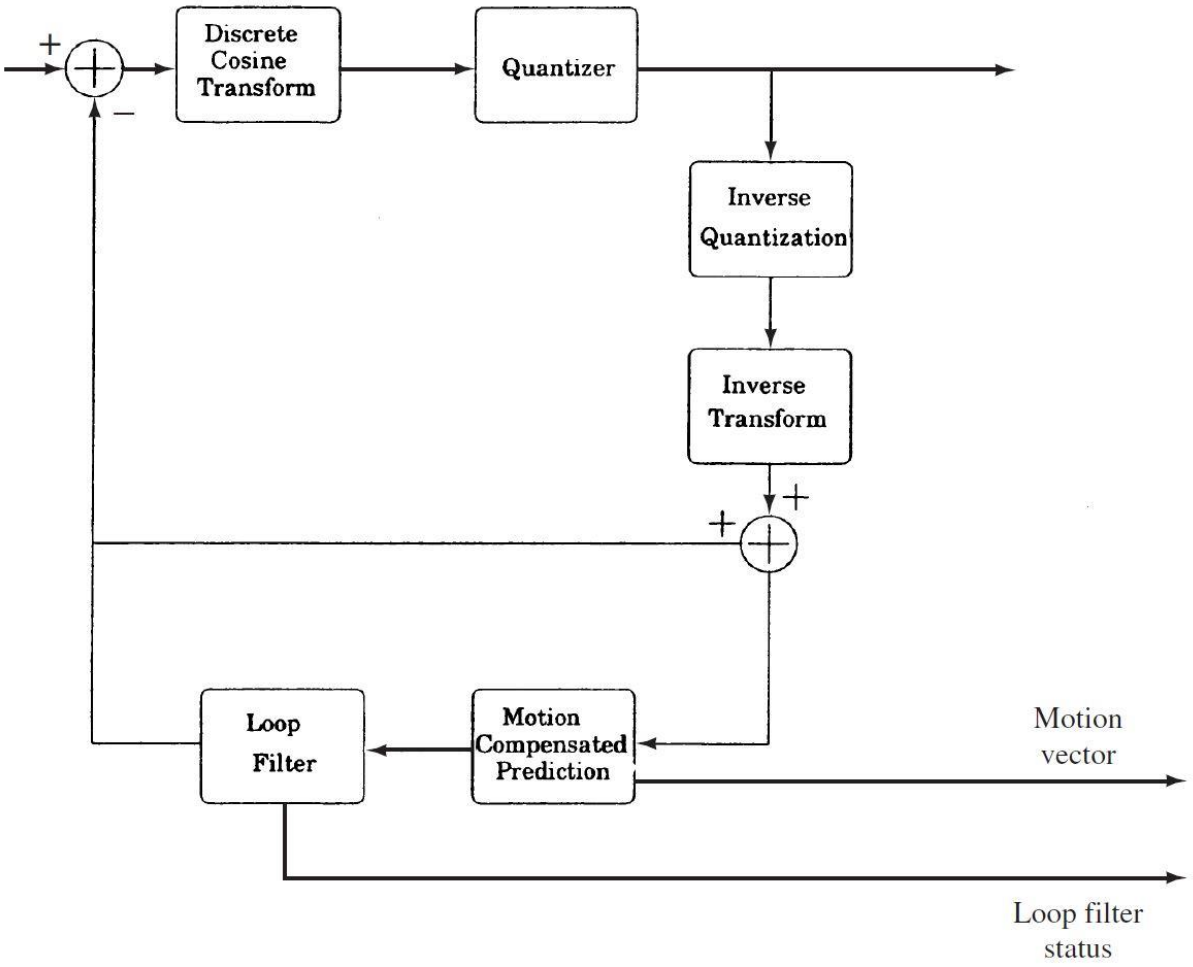


Figure 4. Block diagram of the H.261 video compression algorithm

In the case of H.261, each frame is divided into 8 by 8 pixel blocks where previous frame is used to predict the values of the pixels in the block that is being encoded [7, p. 441]. Possible offsets between frames are also taken into account as the information in identical pixel locations might shift. Motion-compensated prediction is done by analysing computationally reasonable area around the block from where the closest match is used as the predicted value. This offset between the block that is used for prediction and the block that is being encoded is transmitted to the decoder as a motion vector. For preventing sharp transitions, a loop filter neutralizes high-frequency components otherwise present in the difference [7, p. 441]. Next, this difference is transformed with DCT, followed by uniform quantization. Larger quantizer step sizes should be selected for higher compression rates, and smaller quantizer step sizes for lower compression rates.

H.263, the successor to H.261, has improved motion-compensated prediction over older standard. It also includes increased number of formats and greater error resilience. Better prediction is achieved by interpolation between neighbouring pixels in the frame, thus creating

a picture that allows motion-compensated displacement with half a pixel steps instead of full pixel steps [7, p. 442]. Additionally, references are allowed outside the picture in an unrestricted motion vector mode, where outside areas are generated by duplicating boundary pixels of the image. Error propagation is prevented by independent segment decoding mode that allows the frame to be broken into segments that can be decoded independently [7, p. 442]. This also gives greater control over the quality of regions in the reconstruction.

Both H.261 and H.263 standards can be seen as simplified versions of MPEG video compression. MPEG-1 standard defines I frames, P frames and B frames [7, p. 443]. I frame is coded without predictions from previous frames. The frames that are placed periodically in the video sequence allow random access to any point of the video without decompressing all the previous frames. Video can be decoded from the first I frame available. P frames are generated from the predictions of previous frames similarly to the prediction method used in H.263. The B frame is combined from predictions of previous I or P frame and the nearest future I or P frame. As B frames are not used for predicting other frames, they are usually allowed to have more errors than I or P frames.

Various I, P and B frames are usually grouped into the smallest random access units in the video sequence – *group of pictures* or GOP in short [7, p. 443]. The first frame in a GOP is I frame or B frame that uses motion-compensated prediction from the following I frame. On Figure 5 [7, p. 443], there is an example of a GOP sequence with I, P and B frames. The sequence starts with I frame and has several B frames. Most of the compression comes from P frames that are predicted from previously mentioned I and B frames.

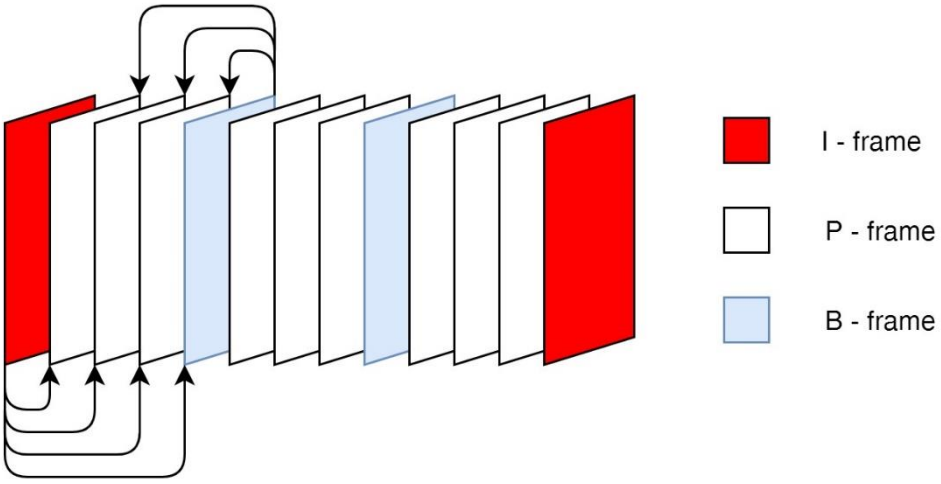


Figure 5. Example GOP arrangement

2.3.1. Video Transcoding

Video encoding is usually seen as creating coded bitstream from RAW, uncompressed video. Transcoding, on the other hand, is converting one coded signal to another [11]. Possible results for transcoding a video signal include changing the file size and format. File size is crucial in limited storage capacity systems and limited bandwidth networks, where keeping and transporting excess information is expensive. Overcoming these restrictions have contributed in the creation of formats that are able to pack video media into small files with relatively light degradation in the perceptible quality. For some applications like video editing, where unpacking highly compressed files is more demanding for the CPU than uncompressing less packed video formats, motivations for transcoding might be completely different [11].

2.4. Video Container Formats

After video encoding, the compressed media is usually held in a container along with encoded audio and some container-specific information [12, p. 35]. This way, instead of separate video and audio files, there is a single file that holds all the information for video and audio playback. Different video containers have different number of codecs supported. For example, MPG container uses only MPEG codecs for holding video, but MOV can hold data in various audio and video codecs [13]. In addition to codecs, video containers have information on the audio and video files they are holding. A brief overview of widely used container formats can be seen on Table 1.

Container Name	Additional Information
WebM	Royalty-free Matroska-based audio-video container that is sponsored by Google. Available codecs for this container are VP8, VP9, Vorbis, Opus [12, 14].
MP4	Standardised by The Moving Picture Experts Group (MPEG). Available codecs include H.264, H.265, AAC, MP3 [12].
QuickTime File Format	Supports same codecs as MP4 but is not standardised and thus has less support. Available codecs include H.264, H.265, AAC, MP3 [12].
AVI	One of the oldest container formats that has wide range of codecs available [12].
ASF	Was created by Windows especially for streaming media. ASF container usually contains WMV or WMA codecs [12].
Ogg	An open-source container that is usually used with Vorbis codec for audio and Theora for video [12].
Matroska	An open-source container that attempts to support all the codecs available [12].

WAV	A lossless patent-free container developed by Microsoft and IBM [12].

Table 1. Brief overview of widely used container formats

2.5. Development of Video Coding Standards

In order to achieve interoperability among different hardware and software, international video coding standards have been defined by organisations like International Standards Organization and International Telecommunications Union. The ratification of ITU-T H.261 in the year of 1988 was the starting point of rapid standardization activities in the television, film, computer, communication, and signal processing fields [6, p. 56]. Throughout the years, MPEG-1, H.263, MPEG-2, MPEG-4 Part 2, AVC/H.264, and HEVC/H.265, the descendants of ITU-T H.261 standard, have been described and approved. The evolution of ITU-T and ISO-IEC standards can be seen on Figure 6 [8]. Each new standard has brought higher quality for lower bitrates and thus made feasible video conferencing, video streaming, higher resolution video, etc. available for consumers.

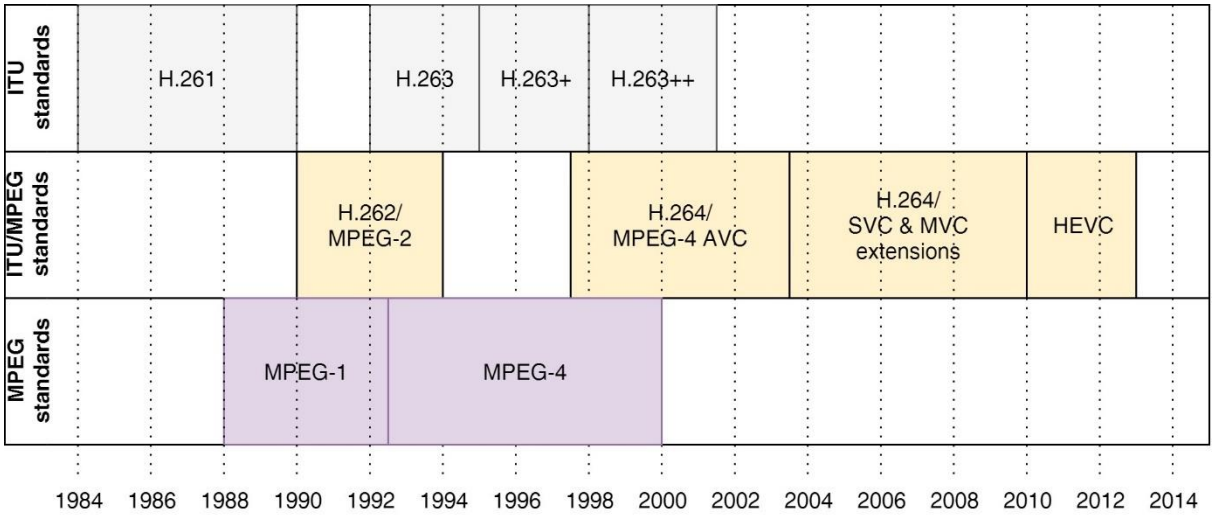


Figure 6. Evolution of ITU-T and ISO-IEC standards

The High Efficiency Video Coding or H.265 is a recent joint video coding standard following the previous H.264, also known as The Advanced Video Coding standard which is currently one of the most popular coding standards for the recording, compression and distribution of high-definition video media [6, p. 84, 8]. H.265 is designed to address increased video resolution and development of parallel processing architectures [20], and is capable of achieving twice the compression efficiency of AVC without losing perceivable video quality [6, p. 84]. On the other hand, the complexity of HEVC decoder is quite close to AVC decoder, thus software solutions' decoding times are similar to identical hardware. More information on

the differences of AVC and HEVC can be found from Table 2 [15], additional descriptions and explanations are given in Appendixes 4 and 5.

Category	AVC (h.264)	HEVC (h.265)
Names	MPEG 4 Part 10 AVC	MPEG-H, HEVC, Part 2
Key Improvement	• 40-50% bit rate reduction compared to MPEG-2 • Led the growth of HD content delivery for broadcast and online	• 40-50% the bit rate reduction at the same visual quality compared to H.264 • Potential to realize UHD, 2K, 4K for broadcast and online
Industry Adoption	Dominant and accepted video codec for Terrestrial, Cable, Satellite and TV broadcast. Widely used across Blu-Ray, security systems, videoconferencing, mobile video, media players, video chat etc.	Implementation demonstration across NAB, IBC and other events starting 2012 from companies e.g. ATEME, Broadcom, Thomson, Harmonic (Cisco), Ericsson, Qualcomm etc. Increased R&D across encoder and decoder vendors for software and hardware-based solutions
Specification	• Up to 4K (4,096×2,304) • Up to 59.94 frames per second • 21 profiles; 17 levels	• Up to 8K (8192×4320) • Up to 300 frames per second • 3 approved profiles, draft for additional 5; 13 levels
Progression	Successor to MPEG-2 Part	Successor to MPEG 4 AVC, H.264
Drawbacks	Unrealistic for UHD content delivery due to high bit rate requirements. Frame rate support restricted to 59.94	Computationally expensive (~ 300%) due to larger prediction units and expensive motion estimation (intra prediction with more nodes, asymmetric partitions in inter prediction)
Compression Model	• Hybrid spatial-temporal prediction model • Flexible partition of Macro Block (MB), sub MB for motion estimation • Intra prediction (extrapolate already decoded neighbouring pixels for prediction) • Introduced multi-view extension • 9 directional modes for intra prediction	• Enhanced hybrid spatial-temporal prediction model • Flexible partitioning, introduces Coding Tree Units (Coding, Prediction and Transform Units -CU, PU, TU) • 35 directional modes for intra prediction

	• Macro Blocks structure with maximum size of 16x16 • Entropy coding is CABAC and CAVLC	• Superior parallel processing architecture, enhancements in multi-view coding extension • CTU supporting larger block structure (64x64) with more variable sub partition structures • Entropy coding is only CABAC

Table 2. A comparison between AVC and HEVC

Outside of ITU-T and ISO-IEC open video coding standardization, some companies individually develop their own video codecs that are often based partly on their own secretly kept technologies [3]. VP9, the successor to VP8 and main competitor to HEVC, is a royalty-free video codec developed by Google [16]. As a part of WebM project, it is a part of speeding up the pace of video compression innovation [1]. Different publications from recent years prove the VP9 to be inferior to HEVC standard with an average bitrate overhead at the same objective quality for HD and Full HD resolutions [3, 17, 18]. The compression efficiency of other royalty-free codecs like Daala is not very stable [16].

2.6. Available Video Encoding Software for HEVC

Contrary to the HEVC encoder, the complexity of the decoder is similar to AVC decoder [19]. Thus, decoding HEVC is feasible for software solutions on most hardware used for playing AVC video media, but encoding needs some additional computational resources. There are several software products available for Windows and Linux personal computers and servers. Overview on a selection of software encoding tools is given in Table 3.

Coder	**OS support**	**Found Price**	**4K HEVC Coding**	**Additional Information**
FFmpeg [20]	Windows, Linux	**Free**	Dependent on used library	• Supports x265 [21], Intel QuickSync [22] and NVENC API [23] • A command line tool
Handbrake [24]	Windows, Linux	**Free**	Dependent on used library	• Supports x265 [21] and Intel QuickSync [22] • Has GUI

Cinemartin Cinec [25]	Windows	**2.994 €** [28] (With Nvidia hardware acceleration)	Promises up to two times real-time encoding (only with gtx960, Quadro M4000 and upward [28])	Has plugins for popular Adobe Premiere and After Effects
MainConcept HEVC Encoder SDK [26]	Windows, Linux	Has various licensed software packages, **no listed prices**	Promises up to **60 frames per second** encoding for 8-bit and 10-bit profiles [29]	• Works with Intel Quick Sync [27] • Works on a Nvidia Quadro based solutions [28] • Product is a low-level API
BEAMR 5 [29]	Windows, Linux	Prices depend on the scale, **no listed prices**	Promises up to **60 frames per second** with **36 cores** [32]	Promises high scalability
TITAN Live [30]	Linux, Windows	Depends on the scale, **no listed prices**	More than one CPU is needed for real time 4K encoding [33]	Promises absorbing peak loads; is a pure CPU solution
Ittiam i265 [31]	Windows, Linux	Has 2 encoding products, **no listed prices**	Promises up to **60 frames per second** on CPUs [34]	• i.265-PQ for high-quality encoding • i.265-HS for high-speed encoding
Wowza Streaming Engine [32]	Windows, Linux	Has several payment plans, perpetual license price is **1995 $**	Depends on the system it is deployed on.	• Supports NVENC, but only on high-end server grade Nvidia GPUs • Some problems cooperating with new Nvidia drivers [32] • While NVENC is used, CUDA cores sit at idle

Table 3. Software solutions for HEVC encoding

2.7. Available Video Encoding Hardware for HEVC

The increased complexity of HEVC encoding algorithm over AVC has motivated several hardware production companies into designing dedicated accelerators to speed up HEVC encoding times and making real-time H.265 encoding more accessible. In Table 4, there are

shown different solutions for HEVC encoding that do not rely solely on the CPU cores of the system.

Coder	OS support	Price	4K HEVC Coding	Additional Information
NVENC [33]	Windows, Linux	HEVC capable devices range **from hundreds up to thousands** of dollars [34, 35]	• 2nd generation Maxwell GPUs introduced HEVC encoding up to 8 bit **4096x4096** resolution • From Pascal architecture, capable encoding up to 10 bit **8192x8192**	• NVENC supports h.264 and h265 [33]; • Appendix 6 shows the popularity of HEVC capable GPUs amongst Steam users [36]; • Consumer grade GPUs support up to 2 simultaneous video streams, enterprise grade Quadro series GPUs support more than 2 simultaneous streams
QuickSinc [37]	Windows, Linux	Supported intel devices range from **below hundred up to thousands** of dollars [38, 39]	• 6th generation Intel Core processors support 8-bit hardware accelerated HEVC coding (**up to 60 frames per second**) • 7th generation processors added 10-bit HEVC coding capability [40]	• Hardware acceleration capabilities can be used through Media Software Development Kit by Intel [41] • From Kaby Lake architecture h.264, h.265, JPEG, VP8 and VP9 encoding and additionally MPEG2 decoding is supported
Kyrion CM [42]	Is a black-box solution that sits in	Used Kyrion CM5000 devices have been priced **over 6000 $** [43]	An encoder device supporting single channel or dual channel	Based on ATEME 5th generation STREAM compression engine

	IP and ASI networks		UHD HEVC encoding through cascading [44]	
RealityCodec H.265/HEVC video encoder FPGA IP core [45]	An IP core that could be used in various solutions	Sold to enterprises and product designers who pay license among annual support and maintenance fees, no listed prices	• **Up to 60 frames per second** on XILINX Virtex-US+ VU11P devices • Lesser framerates and resolutions are also supported for Kintex-US and Zynq-US+ devices	One of the FPGA design benefits over ASIC-based systems is the programmability which allows adding improvements over time to the system; designed by NGCodec company
Harmonic VIBE 4K [46]	Is a black-box solution that sits in IP network	Hardware and licenses are sold separately, no listed price	**Up to 59.94 frames per second**	• This system is designed for live broadcasts capable of processing uncompressed UHD signals • In addition to VIBE 4K, Harmonic has a quite wide product line with devices like Electra X2 that supports FHD encoding and is mainly targeted to broadcast and multiscreen services [47]
Ericsson Keepixo AL2000 [48]	Is a black-box solution for TV service providers	Depends on the scale, no listed prices	Only FHD video media encoding	Supports multiscreen encoding

VITEC MGW ACE [49, 50]	Is a black-box solution for video streaming	Device has been priced for approximately **10 000 $** [50]	Only FHD video media encoding	Promises artefacts-free content delivery

Table 4. Hardware solutions for HEVC encoding

2.8. Licencing

AVC and HEVC are not royalty-free like some of their competitors, including VP8 and VP9. From November 22 in the year of 2016, an independent licensing administrator, HEVC Advance, started an initiative where no royalties are needed for software implementations of HEVC on consumer mobile devices and personal computers [51]. Implementing HEVC on all hardware solutions in mobile devices, televisions and other consumer devices have royalty rates based on regions they are intended to. For example, main profile royalty on hardware HEVC solutions for mobile devices is 0.40 USD in countries from region 1 and 0.20 USD in countries from region 2 [52].

3. Methodology

3.1. Research

Market research (*in social sciences known as primary research*) was conducted in order to understand video transcoding needs of the industry. Research was done by contacting specialists in different fields of video media creation, handling and presenting. Communication was mostly done in free form and stored through notes, emails, and audio files that can be seen in Appendix 1. First-hand experience on GPU programming was collected by participating at PUMPS summer school that took place in Barcelona Supercomputing Centre in July 2016.

3.2. Prototype Design and Testing

Goals for the prototype system were:

- to have HEVC transcoding capability;
- scalability for increasing transcoding performance;
- high control over transcoding parameters;
- gathering information on transcoding performance of the system.

3.2.1. Used Hardware

Important demands on hardware for the prototype and reference systems:

- all chosen hardware components had to be available in the timeframe of this thesis;
- at least one system had to have capabilities for hardware accelerated transcoding;
- at least one system had to have capabilities for CPU based transcoding;
- had to support goals of the prototype.

To better understand the effects of various hardware combinations, three different hardware systems called #1, #2 and #3 were used for testing purposes. System #1 had two Nvidia GPUs for hardware accelerated transcoding. Other CPU based transcoding configurations were used as reference for assessing the performance of the hardware accelerated system. The list of relevant hardware components installed in these systems is shown in Table 5. Reference

systems' configurations were chosen to simulate typical software based video transcoding solutions, where sequences are transcoded on high-end CPUs.

Component	**Test Rig #1**	**Test Rig #2**	**Test Rig #3** [53]
CPU	**Intel(R) Core(TM) i7-6700** CPU @ 3.40GHz	**Intel(R) Core(TM) i7-4790K** CPU @ 4.00GHz	**2 times Intel(R) Xeon(R) CPU E5-2660 v2** @ 2.20GHz (10 cores, 20 threads)
Memory	**2 times 16GiB DIMM DDR4** Synchronous 2133 MHz (32GiB in total)	**2 times 8GiB DIMM DDR3** Synchronous 1600 MHz (16GiB in total)	**64 GiB RAM**
Storage	**120GB KINGSTON** SV300S3 (up to 450MB/s Read and 450MB/s Write speeds [54])	**120GB KINGSTON** SV300S3 (up to 450MB/s Read and 450MB/s Write speeds [54])	**1TB HDD** (~800GB usable)
Other	• **B150M MORTAR** motherboard (**MS-7972**) • **GeForce GTX 1060** • **GeForce GTX 970**	**MS-7850**	**4x QDR InfiniBand**

Table 5. Used test rigs' hardware configurations

System #1 was used for testing video media transcoding on i7-6700 CPU, GeForce GTX 1060 and GeForce GTX 970 GPU, while on system #2 and #3 i7-4790K and E5-2660 v2 CPUs were tested. On systems #1 and #2, the hardware was fully accessible during the testing period, allowing components to be added and removed. On these systems, the same SSD was used to minimize the effect that using different storage device would have on the test results. Hardware for system #3, which is a node from the Rocket cluster owned by the High-Performance Computing Centre of the University of Tartu, was not accessible during the testing period and thus the configuration was not altered. This should be kept in mind while viewing test results. For some tests on system #1, 24 GiB of RAM out of 32 GiB was used to create a RAM Drive for simulating faster read speeds that would be possible on 120 GB SSD, installed into the system.

3.2.2. Used Software

Important demands on software for the prototype and reference systems:

- all software chosen had to be available in the timeframe of this thesis;
- software had to be similarly configurable for hardware accelerated and CPU based transcoding solutions;
- had to support goals of the prototype.

Software stayed mostly the same on systems #1 and #2, however it was completely different on system #3. Software for system #3 was not allowed to be altered as it was also used for other tests and simulations. Test script across the systems stayed the same with slight alterations made for the #3 system, where some input and output directories were changed. Relevant software used on each system is listed in Table 6. As GPUs were tested only on system #1, all software development kits and necessary drivers were not needed on #2 and #3 systems. Instructions for setting up FFmpeg for NVENC coding are presented in Appendix 7.

Component	**Test Rig #1**	**Test Rig #2**	**Test Rig #3** [53]
OS	**Ubuntu 17.04**	**Ubuntu 17.04**	**CentOS** (release 7.3.1611)
Converter	**FFmpeg** (compiled with libraries needed for the project)	**FFmpeg** (compiled with libraries needed for the project)	**FFmpeg-3.2.2** (prebuilt release preinstalled into the system, containing libx264 and libx265 libraries)
SDKs	• **CUDA SDK** (contains Linux display driver) • **NVENC SDK** (software development kit needed for compiling FFMPEG with NVENC support)	-	-
Libraries	• **Libx264 and libx265** for transcoding into AVC and HEVC standards on CPU • **CUDA utility** (a light-weight library to communicate with the CUDA display driver)	**Libx264 and libx265** for transcoding into AVC and HEVC standards on CPU	**Libx264 and libx265** for transcoding into AVC and HEVC standards on CPU (comes with prebuilt release)

Custom scripts	**Transcoding script** ***transcodeTestScript***	**Transcoding script** ***transcodeTestScript***	**Transcoding script** ***transcodeTestScript***

Table 6. Used test rigs' software configurations

FFmpeg was chosen for its high configurability, thorough documentation, active development and support. NVENC, libx264 and libx265 were chosen over competitive solutions for the following reasons:

- lib264, libx265 and NVENC are compatible with highly configurable FFmpeg tool;
- high configurability that is similar to each other;
- lib264 and libx265 have been used in various researches as good implementations of AVC and HEVC algorithms [3, 55, 56].

3.2.3. Overview of Transcoding Tests

Transcoding tests were set to run on 3 CPUs and 2 GPUs installed into test systems listed in Table 5. The main objective for the results from these tests were to show the transcoding rates achieved by transcoding video media into HEVC standard on Nvidia GPUs, on two high-end consumer CPUs, and on slightly older generation server CPU. Additional tests were done in order to show possible bottlenecks, comparison to older AVC standard and the scalability of transcoding performance.

In total, 12 test runs were planned and completed that produced 13 logfiles for storing achieved coding rates. For tests runs 1 to 11, each initial YUV file was coded into AVC and HEVC formats. Each AVC file created was additionally coded into HEVC format. In other words, seven YUV sequences were encoded into AVC format, then these AVC sequences were transcoded into HEVC format, and finally seven YUV files were also coded into HEVC format. In total, this makes 7 original YUV files, 7 files in AVC format and 14 files in HEVC format. For minimizing the effects of random dips and peaks in encoding speeds, each coding test was run 3 times, tripling the number of videos coded in test runs 1 to 11. As each test run resulted in 63 new video sequences, 7 original YUV files were transcoded into 693 MP4 files containing video sequences coded into AVC and HEVC formats. Visualisation of the files used and created during the 12 test runs are shown on Figure 7.

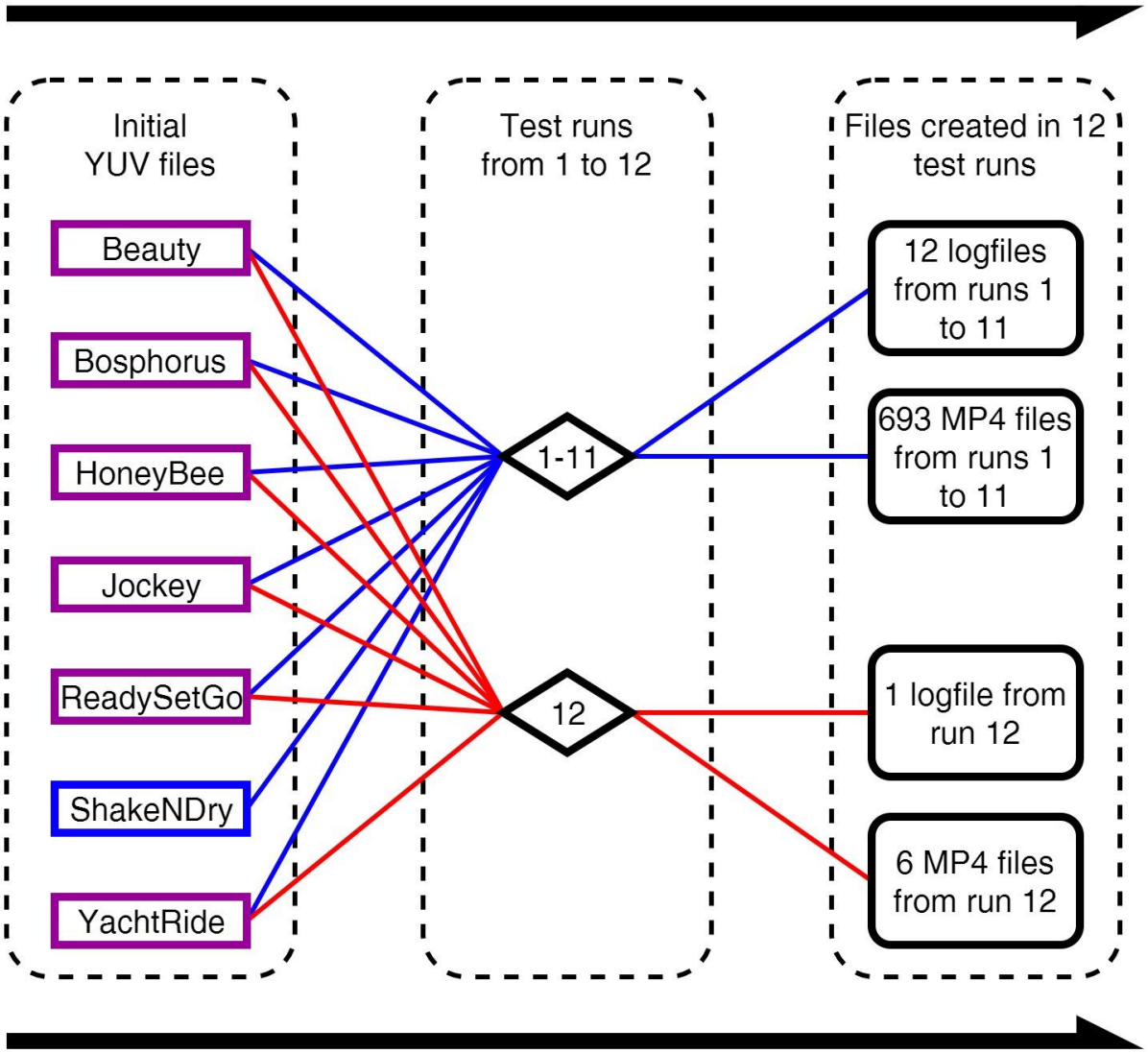


Figure 7. Scheme showing files used and created during 12 test runs

The 12th test run used 6 YUV that were transcoded into 6 MP4 files in HEVC format, making the total number of videos coded 699. In this test, two YUV files were run through HEVC algorithm simultaneously. One of the YUV files was not used as it had less frames than other six sequences. This test was run three times by coding two YUV videos simultaneously in each run. For the 12th test, YUV sequences were not coded with AVC algorithm, nor were videos in AVC format transcoded into HEVC format. The descriptions of test runs can be seen in Appendix 8.

3.2.4. Setting Up the Tests

For running transcoding tests, *transcodeTestScript*, which is a Linux shell script, was created for converting test sequences on #1, #2 and #3 configurations into AVC and HEVC coding formats. Terminal interface for the *transcodeTestScript* can be seen on Figure 8. There are

options for running transcoding tests only on CPU or GPU by typing in number 1 or number 2, run test concurrently on CPU and then on GPU by typing in number 3, and finally on two GPUs simultaneously by typing number 4. Typing in 0 will exit the script. Additional encoding test, converting multiple YUV sequences simultaneously into HEVC sequences on a single GeForce GTX 1060 GPU, was done outside the *transcodeTestScript*, as it needed different approach due to FFmpeg and *transcodeTestScript* restrictions on available transcoding configuration.

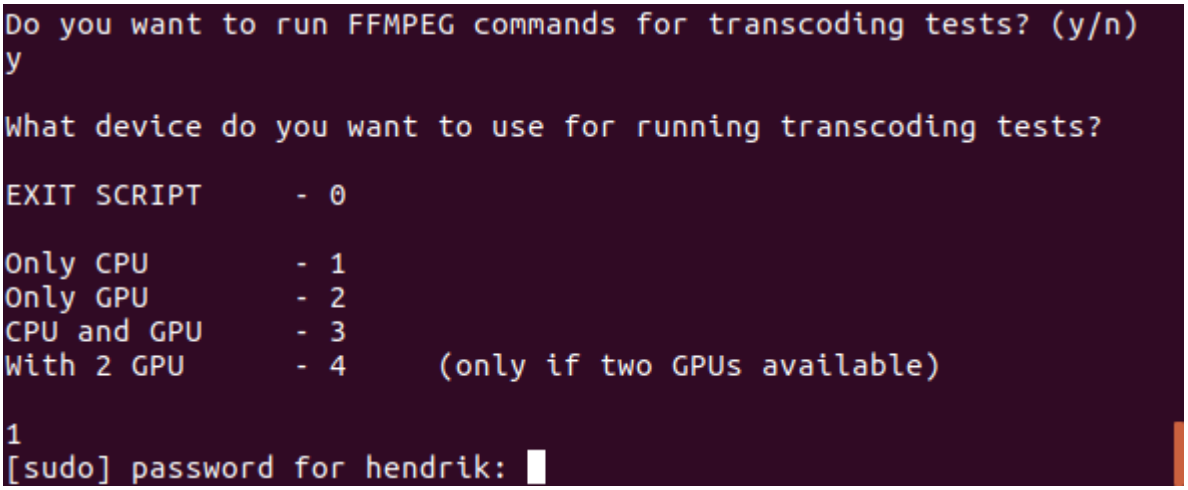


Figure 8. Terminal interface for transcodeTestScript

The *transcodeTestScript* uses prebuilt FFmpeg framework as the core engine for doing all the transcoding tests. It was compiled with libx264 and libx265 libraries for running AVC and HEVC transcoding tests on CPUs and with NVENC library for running these tests on GPUs. Different FFmpeg transcoding commands used by *transcodeTestScript* can be seen in Appendix 9.

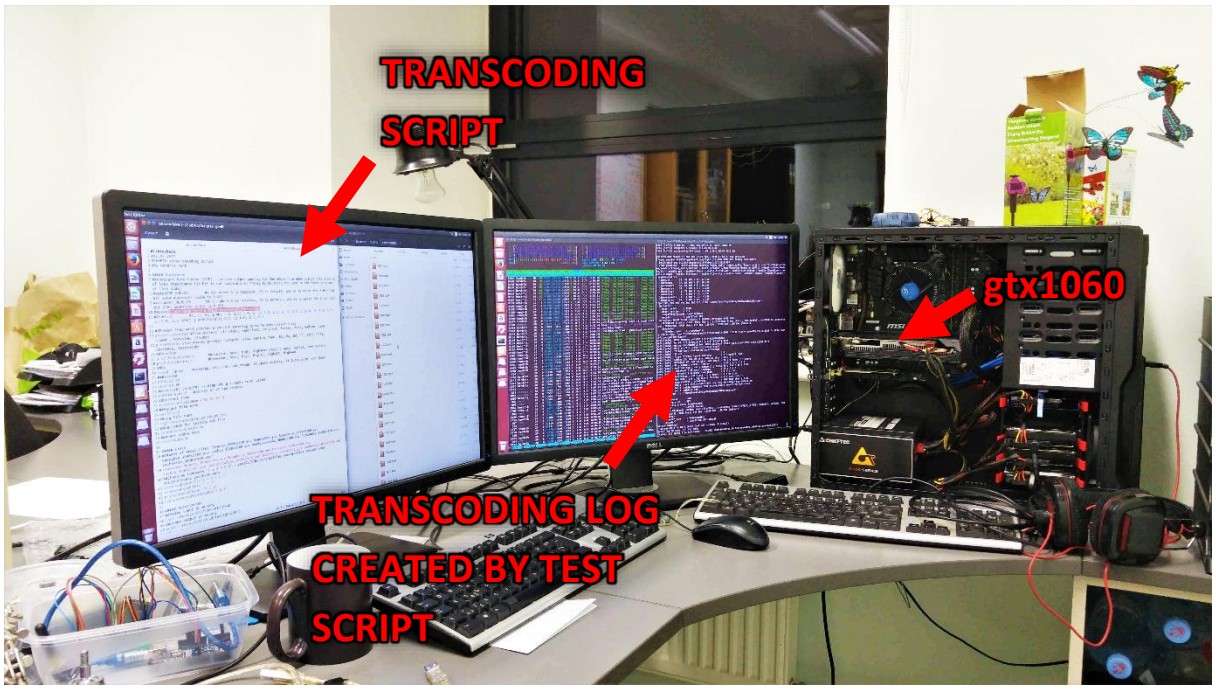


Figure 9. Test Rig #1 in action

For achieving similar file sizes and video quality, bitrates and encoding pre-sets were defined for each transcoding command. Bitrate for coding video media into AVC was set to be double the bitrate used for coding the same video media into HEVC, as the coding efficiency is theoretically twice as high for the newer standard. The pre-set for all coding tests was set to *slow* in order to achieve reasonably high coding efficiency across all tests. Setting it the same for all runs also made the computational complexity comparable between tests conducted on CPUs and GPUs. Other parameters affecting the file size and the quality were left as default values set by the pre-set. With these settings, the subjective video quality was expected to be similar for comparable videos coded on CPUs and GPUs. File sizes were expected to differ slightly, as the design of HEVC and AVC encoders is not standardised and they are probably optimized differently. Parameters used for transcoding together with their descriptions are shown in Appendix 10.

Each transcoded sequence was assigned a general *CVNN* code, created for distinguishing transcoded videos created by *transcodeTestScript*. The characters in the code represent information on the coded data in the following way:

- C – can be number from 0 to 9, representing the used transcoding command;
- V – can be number from 1 to 7, representing the test sequence used;

- NN – can be number from 01 to 99, representing the NNth iteration of coding test run on a test sequence with a specific transcoding command.

For example, *CVNN* code *1203* is the third iteration of *Bosphorus* test sequence being encoded from YUV to AVC standard. The mapping of transcoding commands to *CVNN* codes can be seen in Appendix 11.

3.2.5. Test Videos

Main requirements for choosing test sequences included:

- 4K resolution for addressing the increased video resolution focus on HEVC;
- RAW video format that sets the same starting point for AVC and HEVC coding;
- relatively short video clip lengths, for representing differences in transcoding speeds without the additional encoding time of longer clips.

The seven 3840x2160 8bit YUV RAW test sequences chosen for transcoding tests, also used by the Ultra Video Group from Tampere University of Technology for testing their Kvazaar HEVC encoder [57], fulfilled these requirements. The names, *CVNN* codes and properties of test sequences can be seen in Appendix 12. The bit depth of 8 bits was chosen, as 8-bit encoding was supported on all test systems. All seven test sequences also have 10 bit versions.

3.2.6. Presenting the Results

All transcoding results from *transcodeTestScript* were formatted into graphs by *encodingResultsGraphCreator* script written in Python 3.6. Meanings of the markings used to differentiate results presented on the graphs are shown in Appendix 13.

4. Results

4.1. Observations from Primary Research

List of specialists that were contacted, their positions and communication methods used with them can be seen in Appendix 14. Information collected from private communication is presented in Appendix 1. Notable observations from personal communication and first-hand experience from PUMPS 2016:

- Intermediate codecs like ProRes are used for video editing in professional workflows;
- Final video products are often coded into h.262 for TV work, mostly to ensure compatibility over wide array of devices;
- Lossless compression is preferred for editing, as it allows better control over lightning and colour changes;
- For content creators, high compression rates are typically not used in recording and editing phases, as editing compressed files is more demanding for the CPU and might slow down the editing flow;
- HEVC is believed to take over the place of AVC firstly at online content, but its implementation in other areas is restricted by its higher license fees and devices currently in use, not supporting the newer standard;
- Easiest adoption for HEVC is for online content as most consumers can access it through PCs and mobile devices;
- Bandwidth constraints and increasing quality of media, consumed and created on mobile devices, have made mobile industry, with their short development cycle, one of the leading forces in the adoption of HEVC related software and hardware solutions [58];
- Television channels like Kanal 2 send their media to television service providers like Starman or Telia in relatively low or uncompressed formats;
- Television service providers already have some HEVC capable devices, but not enough to completely shift to HEVC right away;
- The devices used by television service providers have to be considered reliable, well-tested and typically from a known manufacturer;

- Hardware accelerated transcoding devices are expensive, typically single device cost is measured in tens of thousands of euros;
- Unused source media is typically not stored in larger video media productions like television channels or in smaller media productions like the UTTV, which is the university television of the University of Tartu;
- ERR and Kanal 2 use ProMedia Carbon transcoder that, in the time of writing this thesis, did not support HEVC standard [59];
- Some of the transcoding has moved to cloud services where typically software transcoders are used;
- Using cloud services can reduce costs, but results in having less control over the media uploaded to the cloud;
- Hardware transcoders are considered to be best suited for real-time or power-constrained solutions;
- Designing a CUDA-optimized FFmpeg-like framework with a GUI is a complex problem that would need a team of developers to create and manage;
- Transcoding devices capable of doing over 4 times the real time 4K transcoding would make media creators like Eesti Meedia consider investing into the proposed system.
- Having higher picture quality with lower bitrate is considered as a competitive advantage in the case of delivering video content to end user;
- For media creators, highly compressed media is indispensable for sending and receiving information like news stories, into and from remote locations with poor network connections and low bandwidths;
- UTTV uses FFmpeg as its core transcoder engine;
- The transcoding rate for most of Europe (according to PAL) for real-time coding is considered to be 25 frames per second [60].

4.2. The Transcoding Performance of the Prototype

Figure 10 shows the encoding differences between 3 CPUs running at their maximum number of threads. Best transcoding speeds across transcoding tests were achieved with the newest i7 6700 CPU. Average coding rates for i7 6700 are given by first two groups of bars on the chart. Differences between tested CPUs were mostly under 1 frame per second. While reading video sequences from RAM drives, i7 6700 was able to achieve approximately 1 frame per second faster transcoding rates in the cases of coding YUV sequences into AVC and AVC sequences

into HEVC. While coding YUV to HEVC, using RAM drive had a smaller effect on the transcoding speed. Overall, all tested CPU configurations were capable of fairly similar transcoding rates and using faster storage had small effect on their coding performance.

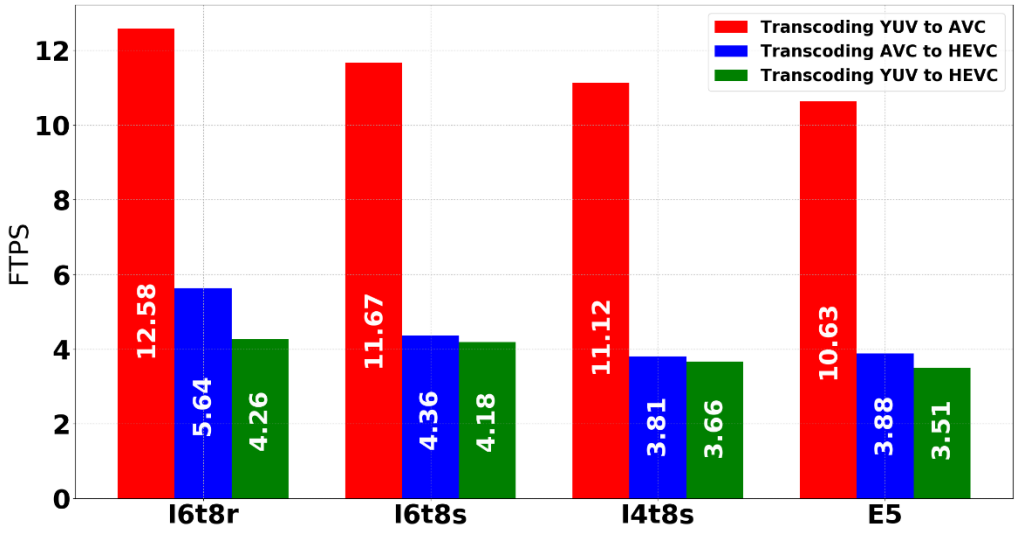


Figure 10. Differences between transcoding rates on tested CPU configurations

Figure 11 shows the transcoding rates achieved when converting 4K 8 bit YUV sequences into AVC, YUV sequences into HEVC, and AVC sequences into HEVC on 3 different CPUs and on 2 different GPUs in multi GPU and single GPU configurations. It can be seen that transcoding speeds for converting sequences into HEVC on gtx1060 were about 25 times faster than the fastest reference configuration equipped with i7 6700 CPU. Over 4 times, the real-time transcoding speeds were achieved with the prototype system only using single gtx1060 hardware accelerator. Older gtx970 was about 40 percent slower, but still capable of achieving approximately 15 times faster transcoding rates than the fastest reference CPU based configuration. Newer NVENC accelerator chip on gtx1060 is capable of outperforming competitors that promise up to 60 frames per second 4K transcoding speeds.

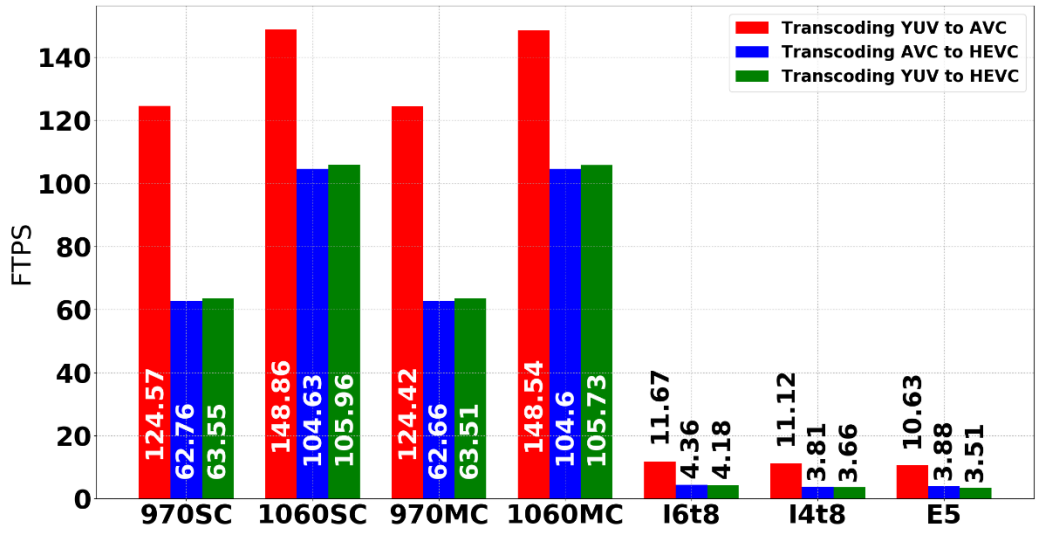


Figure 11. Transcoding speeds for converting sequences into AVC and HEVC

Processing videos simultaneously on gtx960 and on gtx1060, both inserted in the prototype, combined their transcoding speeds. This shows that doubling the number of NVENC GPUs increases the performance linearly. Similar differences between CPUs and GPUs are also present when YUV videos are coded into AVC coding, however the transcoding rates are even higher. For coding AVC, close to 6 times the real-time transcoding rate was achieved on a single gtx1060.

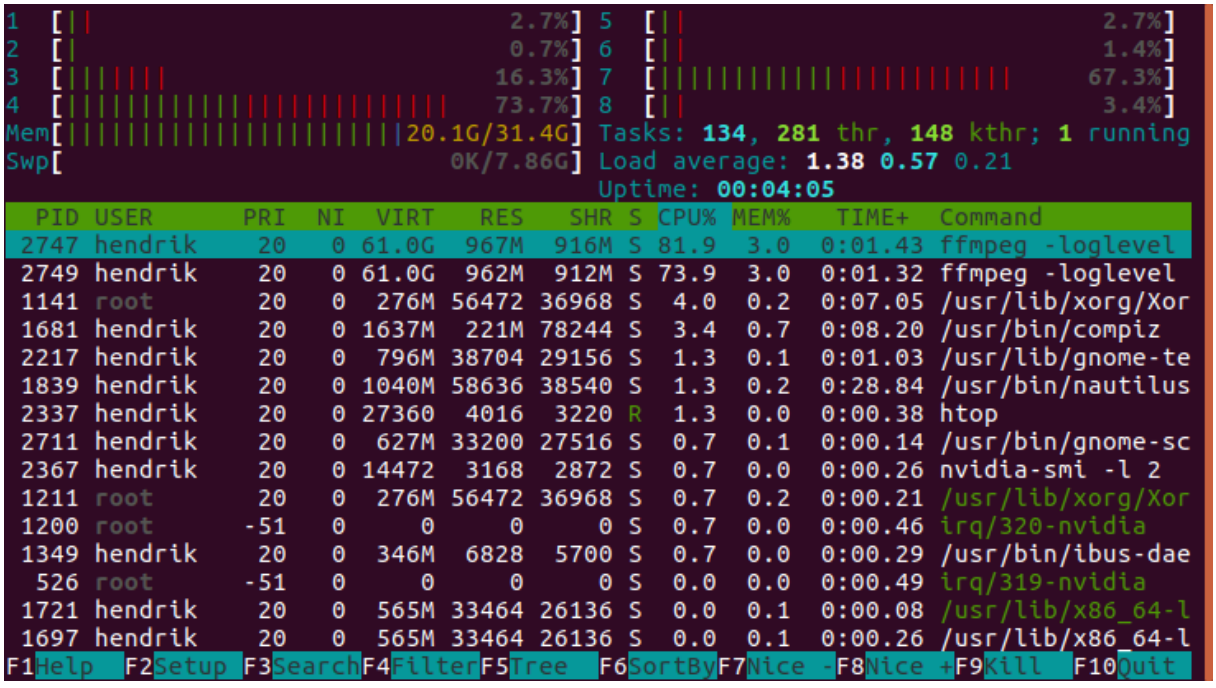


Figure 12. Most CPU resources are available for other tasks when GPU is running transcoding tests

```
|    0      1119    G   /usr/lib/xorg/Xorg                              173MiB |
|    0      1644    G   /usr/bin/compiz                                 121MiB |
|    0      2464    C   ffmpeg                                          569MiB |
+-----------------------------------------------------------------------------+
Mon Apr 17 21:50:14 2017
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 375.39                 Driver Version: 375.39                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  GeForce GTX 106...  Off  | 0000:01:00.0      On |                  N/A |
| 28%   60C    P2    41W / 120W |    875MiB /  6063MiB |     17%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID  Type  Process name                               Usage      |
|=============================================================================|
|    0      1119    G   /usr/lib/xorg/Xorg                              179MiB |
|    0      1644    G   /usr/bin/compiz                                 121MiB |
|    0      2464    C   ffmpeg                                          569MiB |
+-----------------------------------------------------------------------------+
```

Figure 13. GPU gtx1060 has low utilization of CUDA cores when coding is done on NVENC chip

The performance drop caused from using SSDs, for reading sequences into GPU, can be seen on Figure 14. Reading data into GPU from SSD had a significant impact on transcoding results. The first group of bars from the left show transcoding rates in the case were video sequences were read into the GPU in alternate order. The same test video was read no more than one time in a row and had to be always read from the SSD. SSD bottlenecked the data transfer with its up to 450MB per second read speeds.

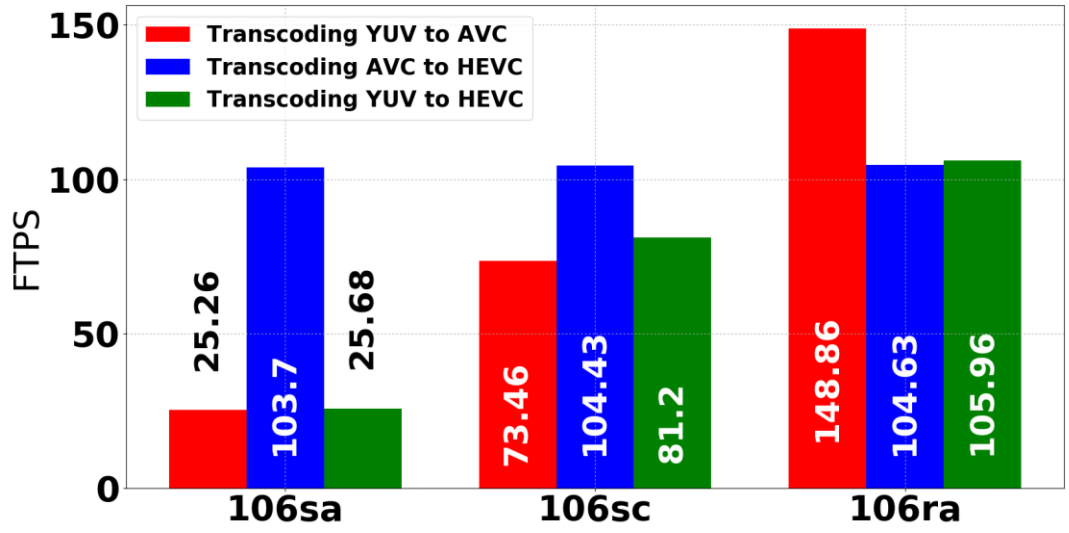


Figure 14. The effect of using a RAM drive instead of SSD for transcoding tests on GPUs

Second group of bars show transcoding tests where video sequences were read into the GPU concurrently, resulting in three times higher coding rates for converting YUV sequences. For the first test run, the video sequence was read into the GPU from the SSD but for concurrent runs it was cashed in RAM that has higher data transfer rates.

Third group of bars show transcoding tests where video sequences were read into the GPU from a RAM drive. This configuration completely removed the bottleneck caused by relatively low SSD read rates. For achieving 150 frames transcoded per second rates with 7.5 GB sequence containing 600 frames, it has to be read into the transcoder approximately 1.9 GB per second. It should be noted that transcoding AVC to HEVC was not influenced by the slower SSD speeds because the sequences in AVC coding are significantly smaller than RAW YUV files.

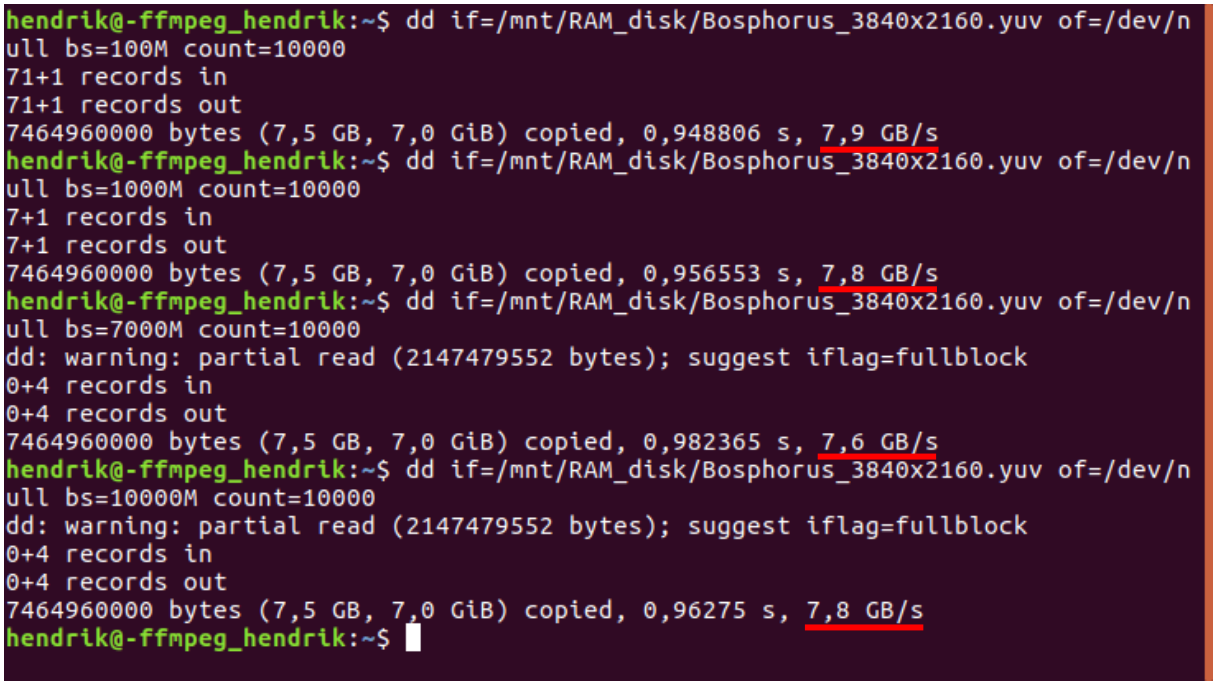


Figure 15. RAM drive read tests done on Test Rig #1

Figure 16 has the results for transcoding test run on the 4 core 8 thread i7 6700 CPU with 2, 4 and 8 threads. It can be seen that the transcoding performance improves significantly when going from 2 threads to 4 threads, but the increase is smaller when jumping from four threads to 8 threads. This can be explained by noting that jumping from 2 threads to 4 threads also increases the number of cores used, but going from 4 threads to 8 only increases the number of threads.

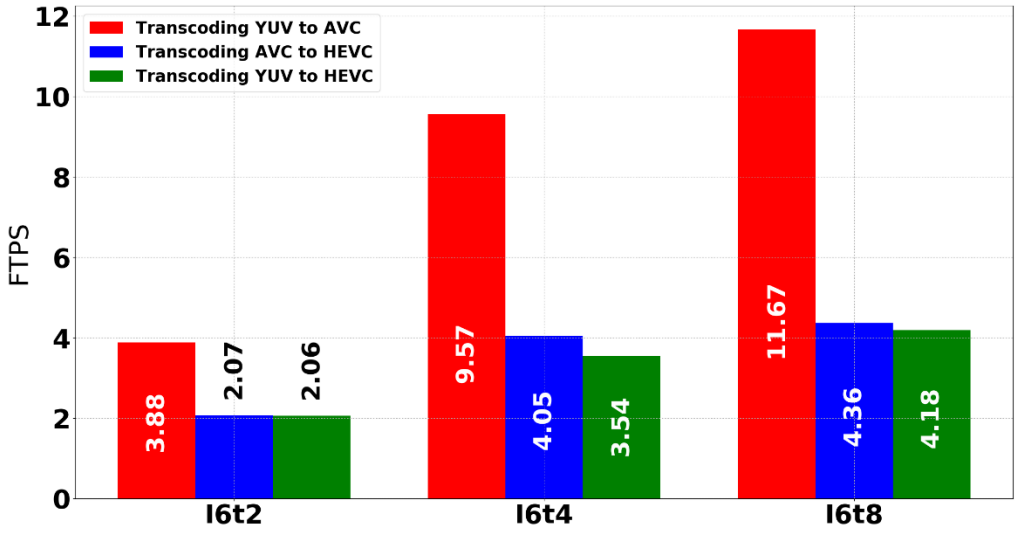


Figure 16. The scaling of transcoding performance across the number of threads used

Figure 17. CPU i6700 under full load while coding HEVC on 8 threads

The transcoding speed for coding two sequences simultaneously on one gtx1060 can be seen by having a look at the first column from the left on Figure 18. This is the transcoding rate averaged for one video out of two streams encoded. Adding the speeds of two streams shows the overall transcoding rate gtx1060 operated at. This is a bit over 106 frames transcoded per second. Approximately the same transcoding speed is also achieved when a single stream was coded on the same GPU. Result for a single stream is presented in the second column. From the

third column, it can be seen that newer gtx1060 GPU is capable of encoding two video sequences simultaneously into HEVC in almost the same rate gtx970 can transcode only one stream.

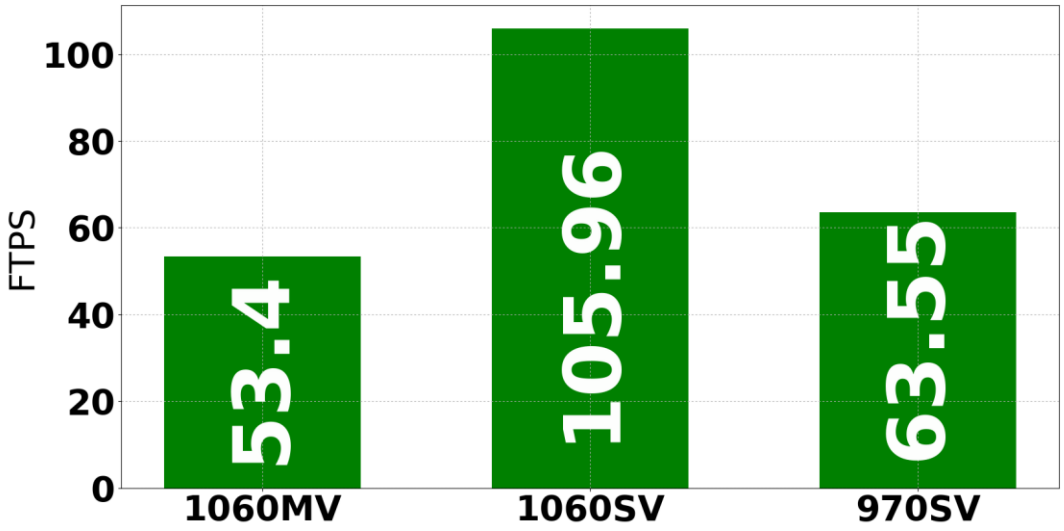


Figure 18. The performance difference from transcoding two sequences simultaneously on gtx1060 (left column) compared to transcoding single sequences on gtx1060 (middle column) and gtx970 (right column)

5. Analysis

Test runs showed superiority of the prototype system equipped with NVENC GPUs over the reference configurations where transcoding was done on CPUs. Furthermore, using multiple accelerators increased the performance gain by combining the transcoding rates of the GPUs. Between generations, the performance has almost doubled from older gtx970 to newer gtx1060, and probably will increase even more for next generations of Nvidia GPUs. The fact that NVENC is currently supporting only AVC and HEVC encoding is the main downside, when compared to hardware accelerators and transcoding software supporting higher variety of coding algorithms. A solution for this could be using idle CUDA cores, unused by NVENC, for achieving higher variety of video coding formats.

Compared to existing CPU based solutions that require at least 36 real CPU cores to achieve up to 60 frames per second HEVC transcoding speeds for 4K media, using a system with single gtx970 for the same result is probably less expensive. Four high-end 8 core Intel CPUs can each cost as much or more than one gtx970. When adding few hardware accelerators can postpone the costly upgrade or rebuild of the whole system for several years, it is probably considered as a better investment. Research has shown that even some Estonian media companies might be interested in investing into a well-priced solution that is capable of doing over 4 times the real-time 4K transcoding into HEVC.

It is common knowledge that finding and addressing the needs of potential customers is crucial in order to have success with any product. For video coding, the readiness and the need for new and efficient coding algorithms varies a lot while targeting different target markets. In mobile industry, where the life cycle of consumer device is short, adopting new software solution and decoder chip is faster than for other industries like cable television networks where most of the intermediate devices for media decoding are in use for longer periods. Making software HEVC coders royalty-free for mobile and PC devices has furthermore simplified its integration to online content. From this, it can be predicted that HEVC will firstly take over high-quality video content for streaming sports, events and other live performances, online. Online video media channels will be the first part of the industry, having the need to upgrade their systems to support reliable faster than real-time HEVC transcoding rates.

To media companies like Eesti Meedia where the content contains real-time online streaming, adopting HEVC can be a powerful tool for targeting consumers with various bandwidth

restrictions and consumers that are upgrading to devices supporting higher resolutions, larger screens, etc. Pioneers here get a competitive edge over competitors that are still using older coding standards that are not capable of outputting higher quality video at the same bitrate. For achieving this, large quantities of video media transcoding have to be handled. Most solutions currently used by many of these companies are not optimized for UHD HEVC transcoding and need to be upgraded or replaced. This creates an opportunity for manufacturers to sell their products in large quantities to media companies replacing their systems in the near future. As the prototype shown, system based on NVENC accelerator chip outperforms many of its competitors and thus could be chosen over them as an upgrade.

HEVC adoption seems also inevitable in areas like cable television, although being slower than for online content, as UHD TVs are already in production and have increased consumer demand for high-quality 4K content. Content creators like Kanal 2 probably will not have the need for highly compressed 4K media as the main compression for end consumers is typically handled by a television service provider like Starman. This need for TV channels might occur when source materials are sent from areas with underdeveloped networks and need to travel through low bandwidth connections, but if the transfer is not time-critical, CPU based solutions are sufficient. For small-time content creators like game streamers and vloggers who want to stream high-quality or even UHD content from their home networks, building transcoding servers or buying most dedicated hardware platforms is not feasible. Reasonably, many gamers already have HEVC capable GPUs as seen from Steam hardware reports. Thus, a lot of value could be found from intuitive and easy-to-use software layer that can use NVENC capabilities of the GPU for transcoding video media into HEVC standard in fast rates.

5.1.1. Considerations for Future Development

Based on the tests and the results, following points should be considered for future development:

- test results already proved high coding capability of the prototype, but composing stable real-life working solution needs further resources to be put into product development;
- additional funding for development, testing and hiring a team of motivated hardware engineers and software developers should be found for continuing with the project;
- organisations like EAS and Buildit could be approached for funding the project;
- storage media might bottleneck the performance for large-source files;

- if more than two transcoding streams per Nvidia GPU are needed, Quadro cards should be used as consumer grade Nvidia GPUs only support up to two parallel streams on one GPU;
- developing own hardware accelerator chip under unknown brand might exclude customers valuing strong brand name and reputation over cost and performance, thus using components from companies like Nvidia would be preferred;
- after brand establishment in the industry, product with a custom designed accelerator chip may be the future direction to take if it results in additional competitive advantage;
- when coming out with new products, it should be kept in mind that HEVC is a licensed coding standard;
- to compete with currently available solutions, end product should be capable, in addition to AVC and HEVC, transcoding source media into other well-known standards and formats like VP9 and MPEG 2;
- CUDA cores could be used to add the support for well-known standards in systems equipped with Nvidia GPUs;
- tests on video transcoding can be done on the CUDA cores for finding out additional transcoding capabilities the product would have, outside NVENC;
- using CUDA cores simultaneously with NVENC chip might have positive effect on the coding performance and should also be tested;
- systems targeted to large media corporations and various service providers should be put through comprehensive stress tests, as the failure of these systems could be extremely costly for these clients.

5.1.2. Suggested Product Lines

Based on the performance of the prototype and the needs found out from the research, three product lines are suggested for future development. All of these products are independent and not reliant on the development of the other two. Large media corporations and service providers with existing server solutions, having inferior performance or fewer features, could be targeted with the first two product lines. Third line should appeal to smaller media enterprisers, content creators, etc. Main value points for these solutions would be:

- competitive transcoding performance at competitive price;
- the reduced complexity achieved by having easily implemented solutions;
- comprehensive testing assuring the reliability of the products;

- ongoing development with all-time support, keeping solutions up to date and providing help when needed.

Line 1 – a black box for inserting into server rack that:

- is a complete solution housing hardware accelerator chip alongside CPU for managing the workflow, fast input-output connections for low latency and high bandwidth, internal memory for cashing incoming and outgoing data, integrated power supply capable of powering the system under continuous high loads, etc.;
- should have simple setup and integration in widely used server solutions to make the transition process for new clients fast and seamless;
- should have hardware configurations for price sensitive clients who do not mind consumer grade components or lower performances, and also configurations with all professional server grade components for clients valuing reliability and performance over cost.

Depiction of a line 1 product is shown on Figure 1. It can be seen that user only needs to set the coding parameters and send in video streams for transcoding. Output streams in desired format are received without the user knowing what has taken place inside the device. The black box itself contains N amount of NVENC capable Nvidia GPUs that are connected through hardware layer containing all necessary components, like CPU for directing the workflow and fast memory for cashing data. Software layer, responsible for giving orders to the GPUs, is built on top of an operating system communicating with all the hardware components.

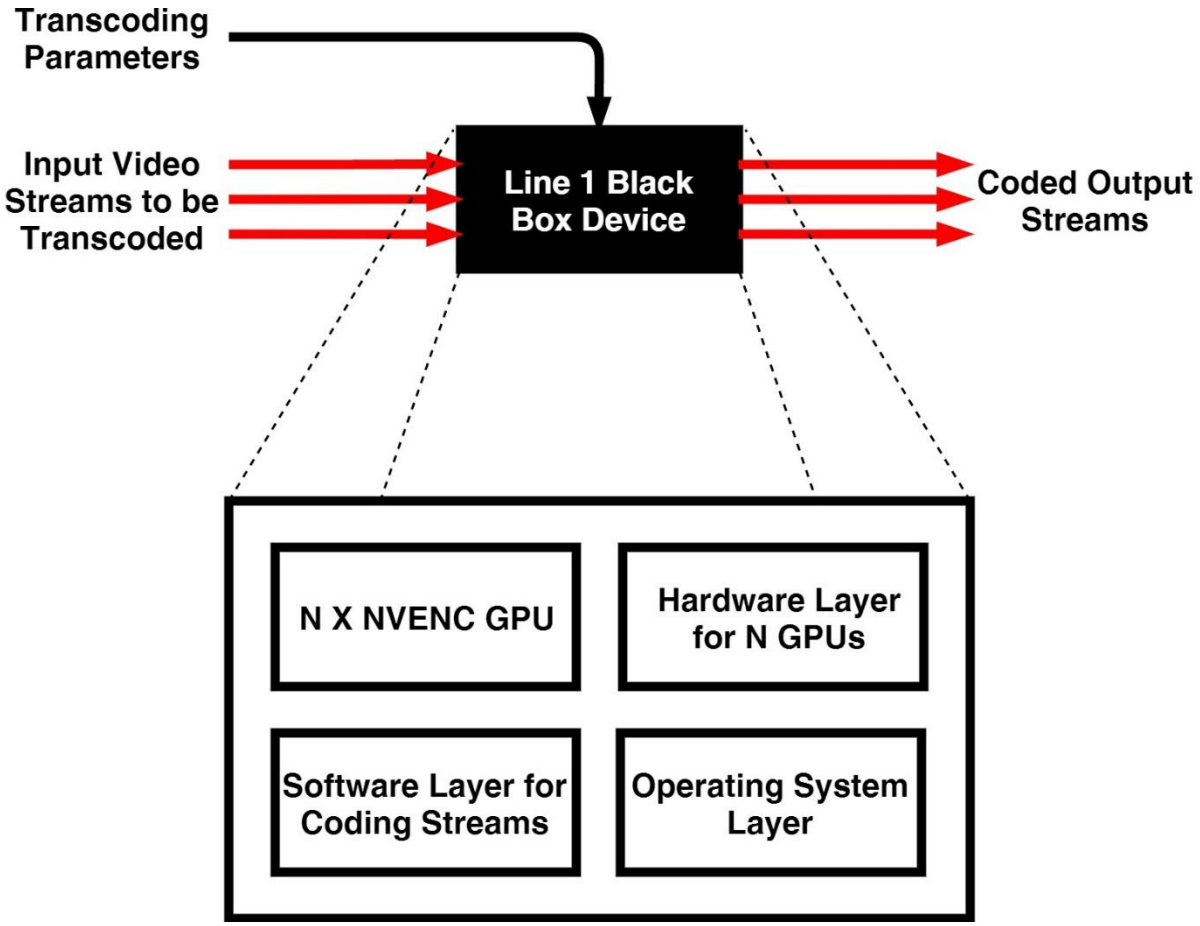


Figure 19. A depiction of line 1 black box device

Line 2 – a software layer alongside user interface for servers already containing HEVC ready NVENC GPUs that:

- is a video transcoding workflow management solution suitable for server systems with integrated HEVC ready NVENC GPUs;
- should be easy to install, use and update in existing server systems, to make the transition process for new clients fast and seamless;
- should have additional features like the capability of detecting bottlenecks in video transcoding workflow that can be used for maximising system performance, adding additional value for the client.

Line 3 – software layer alongside user interface for PCs already containing HEVC ready NVENC GPUs that:

- is a software layer with easy-to-navigate GUI, allowing NVENC GPUs to be used for accelerating and offloading video media transcoding from CPUs;

- should be easy to install, use, update and also compatible with mainstream operating systems to maximize the number of potential customers;
- should preferably be compatible with popular streaming platforms and video editing software to maximize the number of potential customers.

6. Conclusions

Based on the test results and conducted research, it can be said that the purpose of the work was achieved. Test results showed that the prototype was capable of achieving similar or even greater transcoding rates in order to create HEVC compressed video, than state of the art solutions from competitors. Pricewise, equipping existing system with few Nvidia GPUs is cheaper than buying off the shelf hardware transcoder that is priced ten times as much. This said, there are some deficiencies for NVENC chip, as currently it can only be used to encode source files with AVC or HEVC algorithms. In addition, to compete in the variety category, the support for additional coding algorithms could be implemented through available CUDA cores on the GPUs. This together with the other suggestions presented in the analysis section of this work for developing feasible NVENC accelerated product lines shows what can be done in order to compete with commercially successful solutions used in various target applications like streaming events, providing online video media service, small time content creation in high quality, etc.

Future development of the project requires additional funding, as the software layers for proposed product lines have to be developed and possible hardware combinations thoroughly tested. Funding for this project could be found through organisations like EAS and Buildit that specialize in kickstarting new businesses.

Acknowledgements

I would first like to thank my supervisors Heiki Kasemägi and Vahur Zadin for their support, advice and for keeping me motivated through interesting conversations and cheerful attitude. Furthermore, I would like to acknowledge my reviewer Aleksander Tõnnisson who found time in his busy schedule to review this thesis. He was the first that came into mind for reviewing this project because of his comprehensive technical background and first-hand knowledge on start-ups.

In addition to supervisors and reviewer, I would like to express my deep gratitude to Erkki Põllu, Aimar Prous, Priit Pärg, Mait Tafenau, Peeter Linnap, Rihhard Krüüner, Toomas Petersell and Jarno Vanne who found time to educate me in their fields of expertise. For helpful conversations on discussing the test results and providing me with hardware for *Test Rig #1*, I would like to thank Veiko Vunder. Additionally, many thanks go to Dr. Gholamreza Anbarjafari, Pejman Rasti and Martin Johanson for providing me access to their hardware devices.

Next, I thank my parents who have provided me an excellent environment and opportunities for fully concentrating onto my studies. I would like to specially thank my biggest supporter Ingel, who has motivated me throughout my studies in the University of Tartu.

References

[1] M. Uhrina, J. Bienik and M. Vaculik, “Coding efficiency of HEVC/H.265 and VP9 compression standards for high resolutions,” in *2016 26th International Conference Radioelektronika ({RADIOELEKTRONIKA})*, 2016.

[2] A. Ichigaya, “Standardization Trends in Video Coding,” *Broad Cast Technology*, 2016.

[3] D. Grois, D. Marpe, A. Mulayoff, B. Itzhaky and O. Hadar, “Performance comparison of H.265/MPEG-HEVC, VP9, and H.264/MPEG-AVC encoders,” in *2013 Picture Coding Symposium ({PCS})*, 2013.

[4] M. Abomhara, O. O. Khalifa, O. Zakaria, A. A. Zaidan, B. B. Zaidan and A. Rame, “Video Compression Techniques: An Overview,” *Journal of Applied Sciences,* vol. 10, pp. 1834-1840, dec 2010.

[5] J. Ozer, “How to Choose an Enterprise Video Encoder,” Streaming Media, Jun 2012. [Online]. Available: http://www.streamingmedia.com/Articles/Editorial/Featured-Articles/How-to-Choose-an-Enterprise-Video-Encoder-83037.aspx. [Accessed 13 May 2017].

[6] S. Akramullah, Digital Video Concepts, Methods, and Metrics, Springer Nature, 2014.

[7] K. Sayood, Introduction to Data Compression, Elsevier Science, 2012.

[8] J. Zeng, O. C. Au, W. Dai, Y. Kong, L. Jia and W. Zhu, “A tutorial on image/video coding standards,” in *2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, 2013.

[9] M. A. T. Figueiredo, “Scalar and Vector Quantization,” 2008.

[10] ProsoundNetwork, “Understanding Video Codecs,” AVNetwork, 18 Jul 2008. [Online]. Available: http://www.avnetwork.com/article.aspx?articleid=107681. [Accessed 13 May 2017].

[11] A. Vetro, C. Christopoulos and H. Sun, “Video transcoding architectures and techniques: an overview,” *{IEEE} Signal Processing Magazine,* vol. 20, pp. 18-29, mar 2003.

[12] S. Hudson, “Audio and Video Basics,” in *JavaScript Creativity,* Springer Nature, 2014, pp. 35-54.

[13] O. Lindblom, “How to Choose the Right Codec and Container for Your Video Workflow,” *Videomaker Magazine,* 2015.

[14] WebM, “About WebM,” [Online]. Available: http://www.webmproject.org/about/. [Accessed 16.03.2017].

[15] N. Narang, “What is the difference between HEVC (H.265) and H.264 (MPEG-4 AVC),” Future Of Media, Media Concepts, 1 Oct 2013. [Online]. Available: http://www.mediaentertainmentinfo.com/2013/10/4-concept-series-what-is-the-difference-between-hevc-h-265-and-h-264-mpeg-4-avc.html/. [Accessed 15 May 2017].

[16] M. P. Sharabayko and N. G. Markov, “Contemporary video compression standards: H.265/HEVC, VP9, VP10, Daala,” in *2016 International Siberian Conference on Control and Communications ({SIBCON}),* 2016.

[17] D. Grois, N. Tung and D. Marpe, “Coding Efficiency Comparison of AV1/VP9, H. 265/MPEG-HEVC, and H. 264/MPEG-AVC Encoders,” 2016.

[18] J. Kufa and T. Kratochvil, “Comparison of H.265 and VP9 coding efficiency for full HDTV and ultra HDTV applications,” in *2015 25th International Conference Radioelektronika ({RADIOELEKTRONIKA}),* 2015.

[19] F. Bossen, B. Bross, K. Suhring and D. Flynn, “HEVC Complexity and Implementation Analysis,” *{IEEE} Transactions on Circuits and Systems for Video Technology,* vol. 22, pp. 1685-1696, dec 2012.

[20] FFmpeg, “About FFmpeg,” FFmpeg, [Online]. Available: https://ffmpeg.org/about.html. [Accessed 12 Apr 2017].

[21] Multicoreware, “x265 HEVC Encoder,” Multicoreware, 2017. [Online]. Available: https://multicorewareinc.com/video/. [Accessed 12 Apr 2017].

[22] HandBrake, “Features,” HandBrake, [Online]. Available: https://handbrake.fr/features.php. [Accessed 12 Apr 2017].

[23] FFmpeg, “HWAccelIntro,” FFmpeg, 2017. [Online]. Available: https://trac.ffmpeg.org/wiki/HWAccelIntro. [Accessed 12 Apr 2017].

[24] HandBrake, “HandBrake The open source video transcoder,” HandBrake, [Online]. Available: https://handbrake.fr/. [Accessed 12 Apr 2017].

[25] Cinemartin, “Cinec,” Cinemartin, 2017. [Online]. Available: http://www.cinemartin.com/cinec/. [Accessed 12 Apr 2017].

[26] MainConcept, “HEVC/H.265,” MainConcept, 2017. [Online]. Available: http://www.mainconcept.com/eu/products/for-developers/video/hevch265.html. [Accessed 12 Apr 2017].

[27] MainConcept, “HEVC/H.265,” 2016. [Online]. Available: http://www.mainconcept.com/fileadmin/user_upload/datasheets/HEVC_SDK_DATASHEET.pdf. [Accessed 12 Apr 2017].

[28] NeuLion, Inc., “NeuLion's MainConcept Business to Accelerate HEVC Encoding by Supporting NVIDIA Quadro GPU for 4K Live Broadcasting,” NeuLion, Inc., 11 Sept 2016. [Online]. Available: http://www.marketwired.com/press-release/neulions-mainconcept-business-accelerate-hevc-encoding-supporting-nvidia-quadro-gpu-tsx-nln-2157251.htm. [Accessed 3 May 2017].

[29] BEAMR, “BEAMR 5,” BEAMR, 2017. [Online]. Available: http://beamr.com/vanguard-by-beamr-content-adaptive-hevc-codec-sdk. [Accessed 12 Apr 2017].

[30] TITAN Live, “Software-Based Live Video Compression for Distribution applications,” 2016. [Online]. Available: https://ateme-czd4fhlsmdr8.netdna-ssl.com/wp-

content/uploads/2016/09/ATEME_TITAN_Live_Datasheet-May2016.pdf. [Accessed 12 Apr 2017].

[31] Ittiam, "H.265/HEVC," Ittiam, 2017. [Online]. Available: http://www.ittiam.com/products/software-ips/video/h265-hevc/. [Accessed 12 Apr 2017].

[32] Wowza, "Server specifications for NVIDIA NVENC and NVIDIA CUDA acceleration with Wowza Transcoder," Wowza, 06 Mar 2017. [Online]. Available: https://www.wowza.com/docs/server-specifications-for-nvidia-nvenc-and-nvidia-cuda-acceleration-with-wowza-transcoder. [Accessed 10 May 2017].

[33] Nvidia, "NVIDIA VIDEO CODEC SDK," Nvidia, 2017. [Online]. Available: https://developer.nvidia.com/nvidia-video-codec-sdk. [Accessed 14 Apr 2017].

[34] Amazon, "ZOTAC GeForce GTX 1060 AMP Edition, ZT-P10600B-10M, 6GB GDDR5 PCI Express 3.0 Dual-link DVI, Display Port, HDMI IceStorm Cooling Gaming Graphics Card," ZOTAC, 2017. [Online]. Available: https://www.amazon.com/ZOTAC-ZT-P10600B-10M-Dual-link-IceStorm-Graphics/dp/B01IDCOOLC/ref=sr_1_49?s=pc&ie=UTF8&qid=1493580464&sr=1-49&keywords=gtx+1060. [Accessed 30 Apr 2017].

[35] Amazon, "NVIDIA Tesla P100 GPU computing processor - Tesla P100 - 16 GB - Centernex update," HHPP, 2017. [Online]. Available: https://www.amazon.com/NVIDIA-Tesla-P100-computing-processor/dp/B06WV7HFWV/ref=sr_1_1?s=electronics&ie=UTF8&qid=1493581331&sr=1-1&keywords=tesla+p100. [Accessed 30 Apr 2017].

[36] Steam, "Steam Hardware & Software Survey: March 2017," Steam, March 2017. [Online]. Available: http://store.steampowered.com/hwsurvey/videocard/. [Accessed 24 Apr 2017].

[37] Intel, "Intel® Quick Sync Video," Intel, [Online]. Available: http://www.intel.com/content/www/us/en/architecture-and-technology/quick-sync-video/quick-sync-video-general.html. [Accessed 15 Apr 2017].

[38] Amazon, “Intel Boxed Pentium Processor G4500 FC-LGA14C 3.5 1 LGA 1151 BX80662G4500,” Intel, 2017. [Online]. Available: https://www.amazon.com/Intel-Pentium-Processor-FC-LGA14C-BX80662G4500/dp/B015VPX190/ref=sr_1_22?ie=UTF8&qid=1493581756&sr=8-22&keywords=skylake. [Accessed 30 Apr 2017].

[39] Amazom, “Intel CM8064501549928 XEON PROCESSOR E7-8890 V3, 18C, 2.5 GHZ, 45M CACHE, DDR4 UP TO 1866 MHZ, T,” Intel, 2017. [Online]. Available: https://www.amazon.com/Intel-CM8064501549928-PROCESSOR-E7-8890-CACHE/dp/B011DMKTAE/ref=sr_1_1?ie=UTF8&qid=1493582210&sr=8-1&keywords=E7-8890+v4. [Accessed 30 Apr 2017].

[40] T. S. Ganesh and C. Ian, “Home>CPUs,” AnandTech, 30 Aug 2016. [Online]. Available: http://www.anandtech.com/show/10610/intel-announces-7th-gen-kaby-lake-14nm-plus-six-notebook-skus-desktop-coming-in-january. [Accessed 15 Apr 2017].

[41] Intel, “Intel® Media Software Development Kit 2016,” 2016. [Online]. Available: https://software.intel.com/sites/default/files/mediasdk_release_notes.pdf. [Accessed 15 Apr 2017].

[42] Ateme, “Kyrion CM,” 2016. [Online]. Available: https://www.ateme.com/products/encoding-and-transcoding/kyrion-cm. [Accessed 12 Apr 2017].

[43] Kitplus, “ATEME KYRION CM5000,” Kitplus, 24 Jun 2016. [Online]. Available: https://www.kitplus.com/FORSALE/Converters_and_Processing/Encoders___Decoders/ATEME/KYRION_CM5000/172386.html. [Accessed 12 Apr 2017].

[44] Ateme, “Kyroion CM5000,” [Online]. Available: https://ateme-czd4fhlsmdr8.netdna-ssl.com/wp-content/uploads/2016/09/ATEME_KYRION_CM5000_Datasheet-May2016.pdf. [Accessed 12 Apr 2017].

[45] NGCodec, “REALITYCODEC H.265/HEVC VIDEO ENCODER FPGA IP CORE,” NGCodec, [Online]. Available: https://ngcodec.com/reality-codec-fpga. [Accessed 13 Apr 2017].

[46] Harmonic, “Vibe 4K,” Harmonic, 2017. [Online]. Available: https://www.harmonicinc.com/products/product-detail/vibe-4k/. [Accessed 13 Apr 2017].

[47] Harmonic, “Encoding, Transcoding & Multiplexing,” Harmonic, 2017. [Online]. Available: https://www.harmonicinc.com/products/encoding-transcoding-multiplexing/. [Accessed 13 Apr 2017].

[48] Ericsson, “Ericsson Keepixo AL2000 Family,” Ericsson, 2017. [Online]. Available: https://www.ericsson.com/ourportfolio/products/keepixo-al2000-family?nav=productcategory007%7Cfgb_101_752. [Accessed 14 Apr 2017].

[49] Vitec, “MGW ACE,” Vitec, 2017. [Online]. Available: https://www.vitec.com/products/encoders/portable-encoders/product/show/mgw-ace/. [Accessed 30 Apr 2017].

[50] B&H, “VITEC MGW ACE Compact HEVC/H.265 Hardware Encoder,” Vitec, 2017. [Online]. Available: https://www.bhphotovideo.com/c/product/1170882-REG/vitec_14846_mgw_ace_compact_hevc_h_265.html. [Accessed 30 Apr 2017].

[51] HEVC Advance, “HEVC Advance Announces 'Royalty Free' HEVC Software,” PR Newswire, 22 Nov 2016. [Online]. Available: http://www.prnewswire.com/news-releases/hevc-advance-announces-royalty-free-hevc-software-300367212.html. [Accessed 11 Apr 2017].

[52] HEVC Advance, “HEVC Advance,” 2016. [Online]. Available: http://www.hevcadvance.com/pdfnew/RoyaltyRatesSummary.pdf. [Accessed 11 Apr 2017].

[53] High Performance Computing Center of the University of Tartu, “High Performance Computing Center, University of Tartu,” University of Tartu, [Online]. Available: http://www.hpc.ut.ee/rocket_cluster. [Accessed 24 Apr 2017].

[54] Kingston Technology, “SSDNow V300 Drive,” Kingston Technology, 2017. [Online]. Available: https://www.kingston.com/en/ssd/consumer/sv300s3. [Accessed 28 Apr 2017].

[55] T. Shen, Y. Lu, Z. Wen, L. Zou, Y. Chen and J. Wen, “Ultra Fast H.264/AVC to HEVC Transcoder,” in *2013 Data Compression Conference*, 2013.

[56] High Efficiency Video Coding (HEVC), Springer-Verlag GmbH, 2014.

[57] Ultra Video Group, “Test Sequences,” Ultra Video Group, [Online]. Available: http://ultravideo.cs.tut.fi/#testsequences. [Accessed 24 Apr 2017].

[58] J. Sui, “Why Smartphones are the Initial Products on HEVC Rollout Timeline,” Digital Tech Consulting, INC, 10 Jan 2016. [Online]. Available: http://www.dtcreports.com/weeklyriff/2016/01/10/why-smartphones-the-initial-products-on-hevc-rollouts-timeline/. [Accessed 5 May 2017].

[59] ProMedia, “ProMedia Carbon File-Based Transcoder,” ProMedia, 2017. [Online]. Available: http://www.promediacarbon.tv/. [Accessed 1 May 2017].

[60] Diffen, “NTSC vs. PAL,” Diffen, [Online]. Available: http://www.diffen.com/difference/NTSC_vs_PAL. [Accessed 6 May 2017].

[61] P. G. Howard and J. S. Vitter, “Arithmetic coding for data compression,” *Proceedings of the {IEEE}*, vol. 82, pp. 857-865, jun 1994.

[62] G. K. Wallace, “The JPEG still picture compression standard,” *{IEEE} Transactions on Consumer Electronics*, vol. 38, pp. xviii--xxxiv, 1992.

[63] T. Wiegand, G. J. Sullivan, G. Bjontegaard and A. Luthra, “Overview of the H.264/AVC video coding standard,” *{IEEE} Transactions on Circuits and Systems for Video Technology*, vol. 13, pp. 560-576, jul 2003.

[64] G. J. Sullivan, J.-R. Ohm, W.-J. Han and T. Wiegand, “Overview of the High Efficiency Video Coding (HEVC) Standard,” *{IEEE} Transactions on Circuits and Systems for Video Technology*, vol. 22, pp. 1649-1668, dec 2012.

[65] Coding Horror, “Our Brave New World of 4K Displays,” Coding Horror, 18 Aug 2015. [Online]. Available: https://blog.codinghorror.com/our-brave-new-world-of-4k-displays/. [Accessed 29 Apr 2017].

[66] M. Alvarez-Mesa, C. C. Chi, B. Juurlink, V. George and T. Schierl, “Parallel video decoding in the emerging HEVC standard,” in *2012 {IEEE} International Conference on Acoustics, Speech and Signal Processing ({ICASSP})*, 2012.

[67] S. Radicke, J.-U. Hahn, Q. Wang and C. Grecos, “Many-Core HEVC Encoding Based on Wavefront Parallel Processing and GPU-accelerated Motion Estimation,” in *E-Business and Telecommunications*, Springer International Publishing, 2015, pp. 393-417.

Non-exclusive Licence to Reproduce Thesis and Make Thesis Public

I, Hendrik Türk,

herewith grant the University of Tartu a free permit (non-exclusive licence) to:

reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

make available to the public via the university's web environment, including via the DSpace digital archives, as of 30.05.2022 until expiry of the term of validity of the copyright,

NVENC-BASED VIDEO TRANSCODER PROTOTYPE,

supervised by Heiki Kasemägi and Vahur Zadin,

I am aware of the fact that the author retains these rights.

This is to certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, 16.05.2017

APPENDIX

Appendix 1

Links to Thesis Files

All files

Link:

https://drive.google.com/drive/folders/0B4v5fFF7Pti1dWxNbnk5aU80ZXc?usp=sharing

Description: All following folders with subfolders.

Scripts created

Link:

https://drive.google.com/drive/folders/0B4v5fFF7Pti1VVZWVjlXTk82Zms?usp=sharing

Description: Script responsible of creating graphs from logfiles is a *.py* file and script used for running tests is in a Linux shell script format.

Private communication

Link:

https://drive.google.com/drive/folders/0B4v5fFF7Pti1Sjg1TWNvWVVUQzQ?usp=sharing

Description: Email conversations and notes made on meetings are in separate *.txt* files. Recorded meetings with Starman and Kanal 2 are stored as .wav files.

A selection of files from transcoding tests on GPU and CPU

Link:

https://drive.google.com/drive/folders/0B4v5fFF7Pti1MVV1SXotZUhVenM?usp=sharing

Description: As the settings remained the same for all the files transcoded on GPUs and files transcoded on CPUs, only one video is present for transcoding commands having the same settings.

Appendix 2

Additional Information on Lossless Coding

Arithmetic coding relies on the fact that there is an infinite amount of numbers between two numbers [7, p. 427], for example between 0 and 1. A unique number from the interval can be assigned to a signal being encoded. As this number acting as a label is unique, it could be used to encode the whole signal [61, p. 521]. With longer messages, the number of bits needed to specify that interval grows. Arithmetic coding differs from Huffman coding by dividing the entire message to one number instead of separating the input into component symbols and replacing each with a code.

In addition to the entropy encoding, it is also important to know the relations and the nature of the encoded message. Most compressed signals are not made up of independently occurring information, but have dependencies [7, p. 428]. When assuming that pixel values in an image occur independently, the probability estimates for each pixel values are small and approximately the same. In natural pictures, it is quite probable that the values of the neighbouring pixels are similar [7, p. 428]. This knowledge can be used in order to achieve greater compression in the encoded signal than it would be achievable in a fully independent system. It should be noted that using the whole history is usually not feasible and therefore the context is made out of some subset of the history.

For images, context-based coding might become problematic, as the images have factors, for example noise caused by the sensor of the capturing device, that disrupt the reoccurrence of patterns [7, p. 429]. Solution for this is to generate a prediction for the value to be encoded and encode the prediction error. With small errors, the number to be encoded can be represented with lesser bits than the actual pixel value. Predictive coding has been used in popular lossless compression algorithms like JPEG-LS [7, p. 430]. Neighbouring pixels N, W, NE and NW that are used for finding prediction for pixel X in JPEG-LS algorithm can be seen on Figure 20.

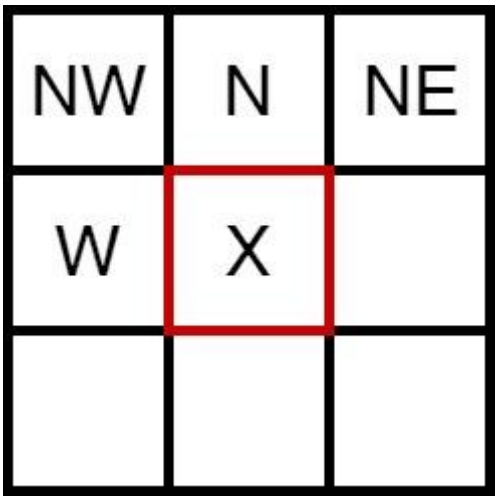


Figure 20. Neighbouring pixel differences are used for finding the prediction for pixel X

Another lossless coding techniques used on pictures are LZSS and LZW dictionary coding algorithms. LZSS attempts to replace a string of symbols with a reference to a dictionary location for the same string [7, p. 431]. The dictionary is a sliding window that contains the last symbols encoded or decoded. With larger dictionary sizes, it takes longer to search the whole dictionary, and representing dictionaries take more bits. LZW is similar to LZSS, but instead of symbols it uses longer strings in the dictionary [7, p. 432]. These two techniques form the basis of GIF and PNG compression schemes.

Appendix 3

Additional Information on Lossy Coding

Quantization techniques are divided into scalar and vector quantization [7, p. 432]. Both use codebooks that can be fixed or transmitted with the compressed data. In case of an image being compressed, scalar quantization maps the source image pixel values to the codebook pixel values pixel by pixel [7, p. 432]. On the other hand, instead of individual pixels, vector quantization maps blocks of pixels to the codebook vectors [7, p. 433]. This has more flexibility as it allows different-size blocks and different-size codebooks. In addition, pixels within a block are correlated which tends to minimize the number of code words needed to represent the vectors well. Differences between pixel mappings are illustrated on Figure 21. In the case of JPEG images, uniform scalar quantizers are used [7, p. 433].

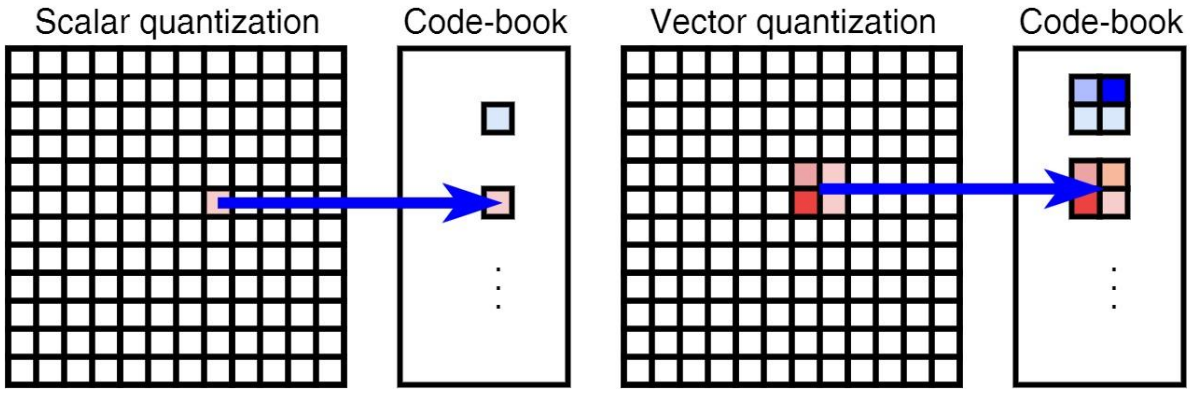


Figure 21. Scalar and vector quantization pixel mapping differences

Predictive coding is additional step used in some lossy data compression algorithms. In many data sets, the knowledge of previous samples can give information about the next sample. This has been exploited in coding algorithms like DPCM where predictor part of the algorithm obtains an estimate of the current sample based on the values of previous reconstructed samples [7, p. 435]. Next, the difference between the prediction and the actual value is quantized. After encoding, this signal, consisting of prediction errors, is sent to the receiver where it can be decoded trough these error values. Typically, a linear predictor, where predicted value is obtained as a weighted sum of past reconstructed values, is used in DPCM [7, p. 435]. Prediction techniques are also used in video coding algorithms where instead of sending full frames, only the differences are sent.

Lossy information compressing is often done through converting the data into a form where compression is easier. This type of transforming process is used in many picture and video coding algorithms. For example, JPEG, H.263 and MPEG2 all use DCT transform coding algorithm [7, p. 436]. In case of JPEG, DCT operates on 8x8 pixel blocks one at a time. The transform is done in a zig-zag sequence that starts from top left pixel of the 64-pixel block and end in the bottom right corner [62]. This zig-zagging can be seen on Figure 22 [62]. In its core, DCT is a Fourier-related transform similar to DFT, but using only real numbers [62].

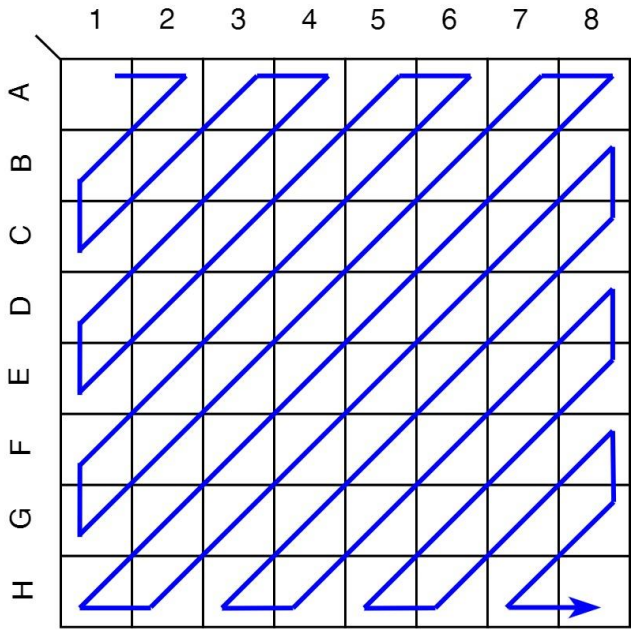


Figure 22. Zig-Zag sequencing DCT uses in JPEG compression algorithm

Compressing images at low rates with previously described transform coding algorithms introduces visible block like artefacts. This is often countered with subband and wavelet coding techniques that smooth out block boundaries without increasing the number of transform coefficients [7, p. 438]. The implementation of both of these schemes is similar. Firstly, the input is filtered through a bank of filters that cover the entire frequency range of the signal. The bandwidth of each filter is a fraction of the bandwidth of the original signal. Next, the output of filters is subsampled and the output values are quantized by encoded quantization labels. For decoding, the decoder firstly decodes the input signal, then upsamples the signal by inserting zeros, and finally filters the signal with reconstruction filters [7, p. 438].

Appendix 4

A Brief Overview of AVC Standard

H.264, often called Advanced Video Coding, is a video coding standard of the ITU-T Video Coding Experts Group and the ISO/IEC Moving Picture Experts Group [63, p. 1]. ITU-T has approved it as H.264 and ISO/IEC as AVC or MPEG-4 part 10. H.264 was developed as a response to growing coding efficiency needs in high definition TV and other services [63, p. 1]. AVC was originally designed [63, p. 2] for being:

- broadcasted trough wide variety of communication channels, including cable, DSL and satellite;
- storage on optical and magnetic devices;
- used in conversational services over various networks, including DSL, wireless, etc.;
- used in multimedia streaming services;
- used in Multimedia Messaging Services.

Some of the most important differences from previous ITU-T and ISO-IEC video encoding standards include enhanced motion prediction capability, use of small block-size exact-match transform, adaptive in-loop deblocking filter and enhanced entropy coding methods [63, p. 18].

Like all other ITU-T and ISO-IEC video standards, only the bitstream structure and syntax is standardized in H.264, as well as constraints on the bitstream and its mapping for the generation of decoded pictures. [63, p. 1]. This guarantees that every decoder confirming to the standard would reproduce similar output from the encoded bitstream, but there are no guarantees for end-to-end reproduction quality. In the same time, there is a lot of freedom for optimizing implementations appropriate to specific applications.

The coded video sequence of AVC either consists of progressive or interlaced frames. The timing differences between progressive or interlaced frames are mostly solved through geometric concepts in the coding representation [63, p. 6]. As interlaced frames with regions of moving objects tend to show a reduced statistical dependency, in some cases it might be more efficient to compress each frame separately. It is typically more efficient to code none-moving

video sections in frame mode and the moving sections in the field mode [63, p. 8]. Line scanning differences between interlaced and progressive frames can be seen on Figure 23.

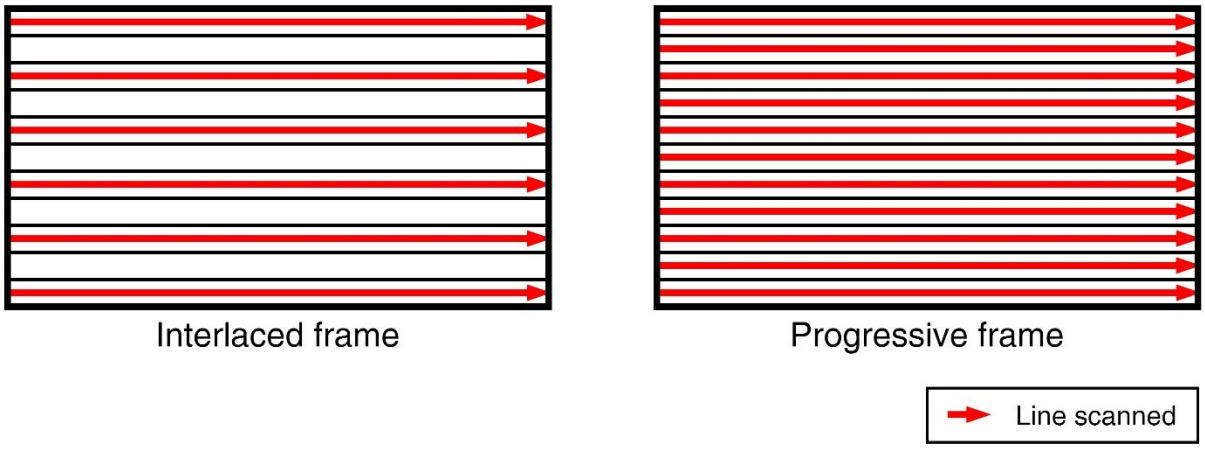


Figure 23. Line scanning differences between interlaced and progressive frames

AVC design allows encoders to combine interlaced fields together in frame mode, to code interlaced fields separately in field mode or to combine the two fields together and compress them as a single frame, but when coding the frame to split the pairs of two vertically adjacent macroblocks into either pair of two field or frame macroblocks before coding them [63, p. 8]. The choice between these options is made adaptively for each frame in a video sequence. Main differences when a frame is coded in two fields instead of a single frame is firstly different zig-zag scan of transform coefficients, furthermore the utilization of reference fields rather than reference frames for motion compensation, and also because the field rows are spatially twice as far apart as frame rows, the strong deblocking strength is not used for filtering horizontal edges of macroblocks in fields [63, p. 8]. Each frame is split into smaller 16x16 pixel fixed-size macroblocks, forming slices and slice groups, that are encoded independently. Frames are made out of one or several slices of macroblocks. An example of slicing can be seen on Figure 24 [62, p. 7]. Similarly to its predecessors, since H.261, the design of H.264 follows the block-

based video coding approach shown on Figure 25 [62, p. 8], where can be seen the schema for processing each macroblock.

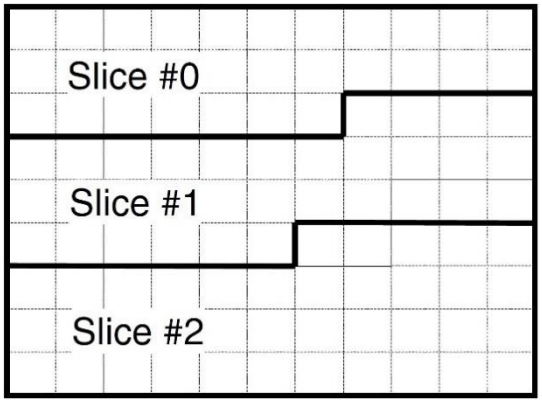


Figure 24. An example of dividing a frame into slices in AVC standard

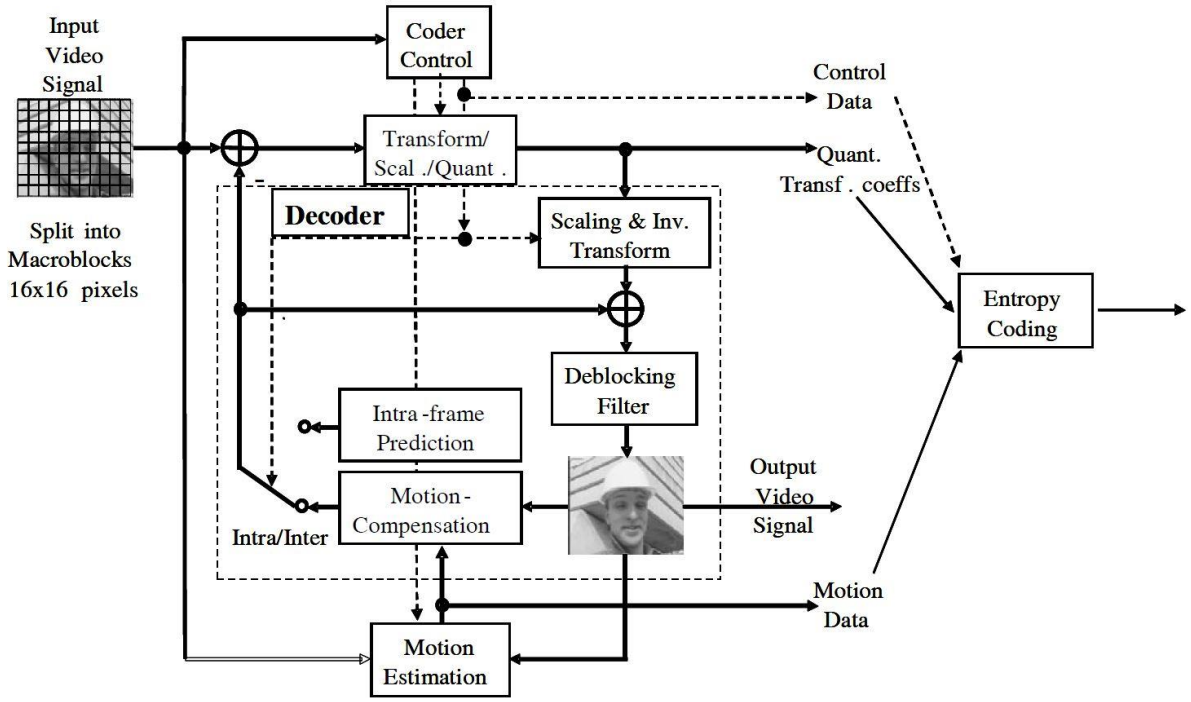


Figure 25. Block-based hybrid video coding approach used in ITU-T and ISO-IEC video standards since H.261

Brightness and colour information for AVC standard is kept in *YCbCr* colour space where Y component represents the brightness, Cb and Cr components represent the extent to which the colour deviates from grey towards blue and red. With *YCbCr* it is possible for AVC to exploit the human visual system, as eyes are more sensitive to Y component than Cb or Cr components

[63, p. 7]. For doing this a sampling structure, in which the chroma component has one fourth of the number of samples compared to the luma component, is used. This sampling, also used in MPEG-2, is called 4:2:0.

Slicing allows an efficient parallel processing of macroblocks. The information in each slice can be correctly decoded without the use of data from other slices. Information from different slices might be needed for applying deblocking filters across slice boundaries. Flexible macroblock ordering technique is used to split the frame into many macroblock scanning patterns [63, p. 7]. In the case of the simplest pattern, the whole picture consists of a single slice. Each slice can be coded using different coding types for creating [63, p. 8]:

- I slice, where all macroblocks are coded using intra prediction;
- P slice, where in addition to intra prediction, some can be also coded using inter prediction with at most one motion compensated prediction signal per prediction block;
- B slice, where in addition to intra prediction, some can be also coded using inter prediction with two motion compensated prediction signal per prediction block;
- SP slice, where switching between different precoded P slice pictures becomes possible;
- SI slice, where an exact match of a macroblock in an SP slice is allowed for random access and error recovery purposes.

Each 16x16 pixel macroblock can be encoded using blocks of pixels that are already encoded within the current frame. This type of intra prediction in AVC does not cross slice boundaries, in order to keep all slices independent of each other [63, p. 10]. Additionally, there is inter frame coding, where macroblocks can be coded using blocks of pixels in previous or future frames. 16x16 pixel macroblocks are often partitioned into 8x8, 8x16 or 16x8 blocks that can be furthermore broken into 4x4, 4x8 or 8x4 blocks in order to make estimations more precise [63,

p. 10]. The segmentation macroblocks and 8x8 pixel partitions for motion compensation can be seen on Figure 26 [62, p. 10].

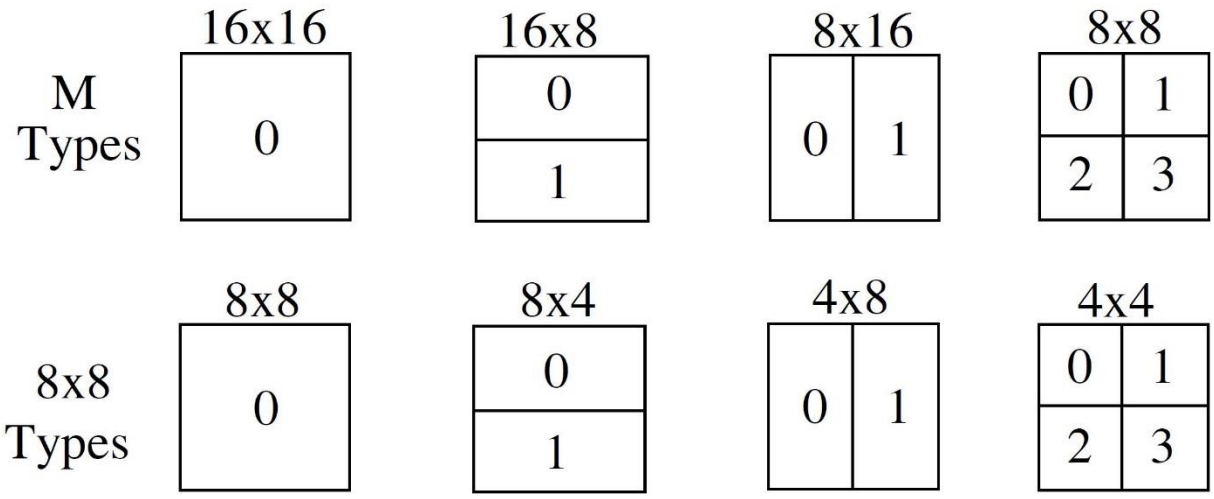


Figure 26. The segmentation of 16x16 macroblocks and 8x8 partitions for motion compensation

In H.264 standard, transform coding is applied to 4x4 blocks, where integer transform with similar properties to 4x4 DCT is used [63, p. 12]. The process includes firstly a forward transform, then zig-zag scanning, scaling and rounding, followed by entropy coding. There are two methods of entropy encoding used, from which the simpler entropy encoding method uses a single infinite-extent codeword table for all syntax elements except the quantized transform coefficients [63, p. 13]. A more complicated Context-Adaptive Variable Length Coding, where coding tables are switched depending on already transmitted syntax elements, is also used as it is more efficient for transmitting quantized transform coefficients.

As most of the encoding is done on macroblocks, the encoded video might end up with unwanted blocky artefacts. AVC standard uses in-loop deblocking filter for removing visible block structures [63, p. 14]. If the difference between block edges is relatively large, it is quite likely a blocking artefact and should therefore be reduced. However, when the difference becomes large enough, that it cannot be explained by the coarseness of the quantization used in the encoding, the edge is more likely to reflect the actual behaviour of the source picture and is not smoothed over. This way the blockiness is reduced without significantly influencing the sharpness of the image.

Appendix 5

A Brief Overview of HEVC Standard

Similarly to AVC, the newer HEVC still has the basic hybrid coding architecture of prior ITU-T and ISO-IEC video coding standards with enhancements in its coding stages. The emergence of higher than HD video and the increasing diversity of video media services have created the need for more efficient encoding than is possible by one of the most popular video standards – AVC [64, p. 1649]. The design focus for HEVC is increased video resolution and increased use of parallel processing architectures in addition to all existing AVC applications [64, p. 1649]. The pixel area comparison of commonly used resolution formats can be seen on Figure 27 [65].

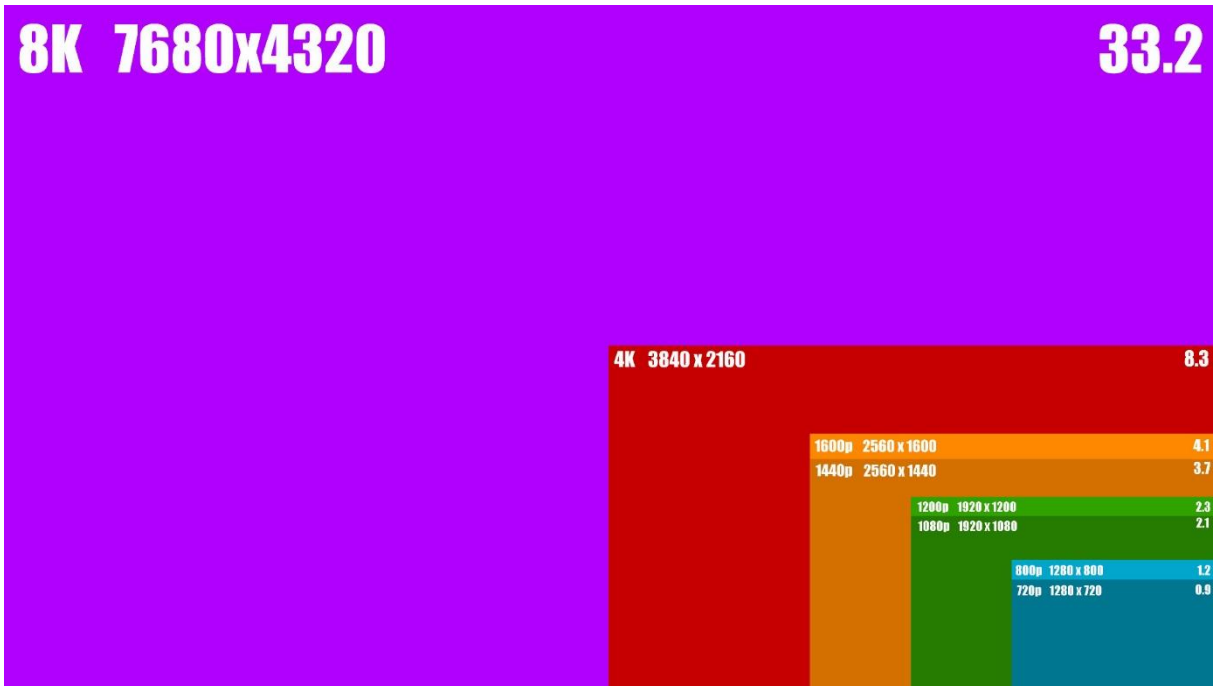


Figure 27. The pixel area comparison of common resolutions

In order to efficiently encode motion vectors for larger frame sizes, each frame should be partitioned into larger blocks than is possible with macroblocks in AVC, but on the other hand, small details are still important and should be also attended to. The solution and major difference, compared to older standards like H.264, is the replacement of macroblock coding unit with a more adaptive quadtree structure based on a coding tree unit, also called CTU [19]. CTU is a logical unit that consists of Y, Cb, Cr samples and associated syntax elements. Each of these samples form a coding tree block or CTB for short. Similarly to AVC, HEVC typically uses YCbCr colour space with 4:2:0 sampling [64, p. 1654]. CTU can be in 64x64, 32x32 or

16x16 sample sizes throughout the video. Because human eye is more sensitive to luminance, luma CTBs are kept the same as CTU, but each chroma CTB components have one fourth of the CTU's sample size [64, p. 1655]. An example block scheme showing a typical design of HEVC encoder is shown on Figure 28 [64, p. 1651].

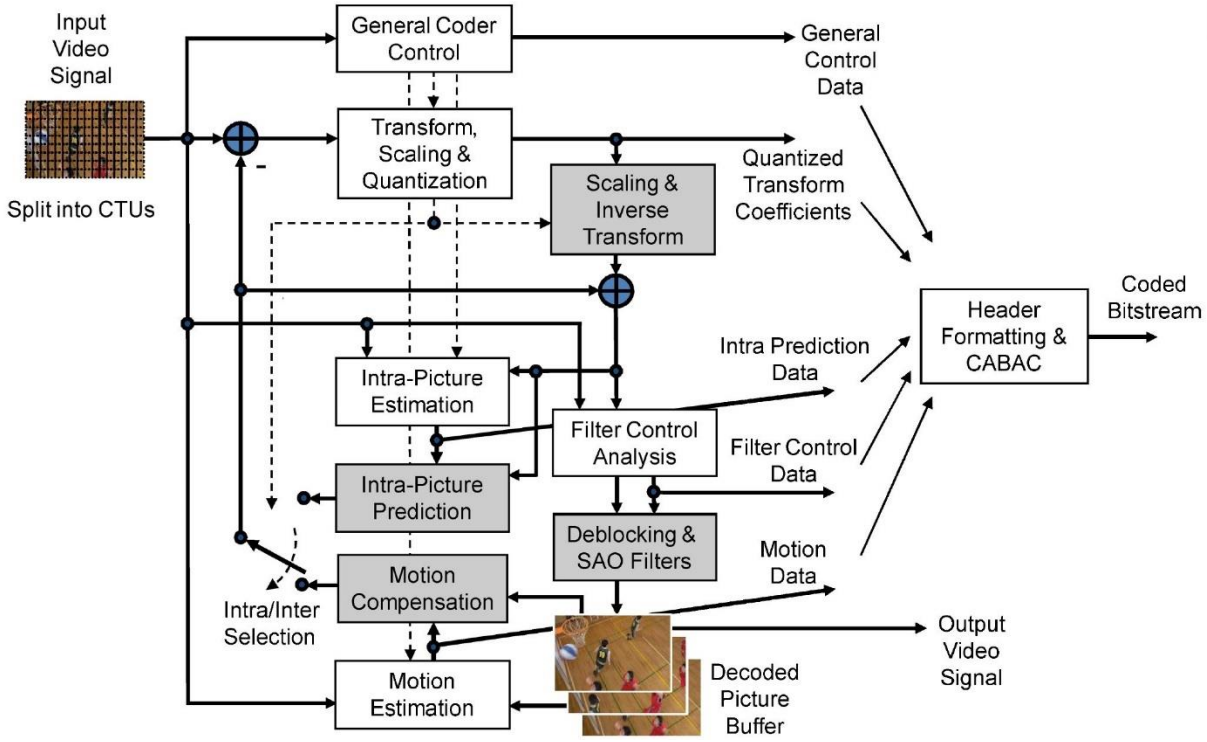


Figure 28. An exemplary block scheme of a HEVC encoder

Compared to the macroblocks, from previous ITU-T and ISO-IEC video coding standards, that uses fixed array size of 16x16 luma samples [64, p. 1655], the variable size CTBs can be selected according to the memory and computational requirements. Encoding larger blocks is especially beneficial for higher resolution video content as it allows higher compression rates in areas of frames where smaller blocks would not capture significantly more information. CTBs are furthermore divided into coding blocks called CBs [64, p. 1655]. CB sizes all the way from the same sample size as CTB to as small as 8x8 are supported by HEVC. The decision whether to perform inter-frame or intra-frame prediction is done at CB level where the prediction type is coded in coding unit that consists of Y, Cb and Cr coding blocks with associated syntax elements [64, p. 1655].

Although CB is excellent for prediction type decision, it could still be too large for storing motion vectors for very small objects. Thus, each CB can be split into PBs based on the temporal and spatial predictability [64, p. 1661]. The difference between predicted and actual image is transform coded trough DCT like transformation, for which each CB has to be split into

transform blocks, also known as TBs [64, p. 1661]. PBs and TBs do not have to be aligned, thus it is possible to do single transforms across residuals from multiple PBs. An example of subdividing CTBs into CBs and CBs into TBs can be seen on Figure 29 [63, p. 1656].

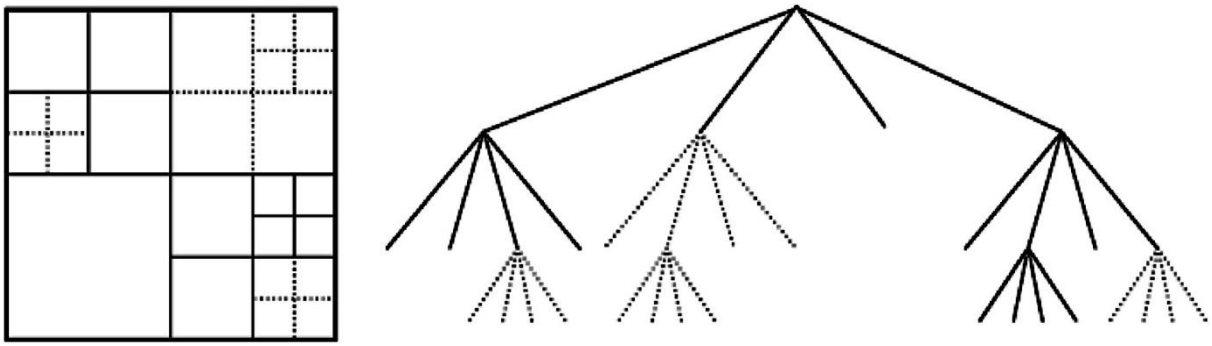


Figure 29. Subdivision of CTB into CBs (indicated by solid lines) and TBs (indicated by dotted lines)

CTUs ordered in a sequence trough raster scan are called slices in HEVC. Each frame can contain one or several self-contained slices, that can be decoded without any information from neighbouring slices with the exception of in-loop deblocking and SAO filtering near the edges [64, p. 1656]. The way frames are divided into slices, tiles and threads can be seen on Figure 30 [64, p. 1656]. Deblocking filter in HEVC standard is applied to the edges that are aligned on an 8x8 sample grid for luma and chroma samples, unlike older AVC standard, where deblocking filter was applied on 4x4 sample grid basis [64, p. 1663]. SAO is a nonlinear filtering operation which enhances the signal representation in smooth areas and around the edges [64, p. 1664].

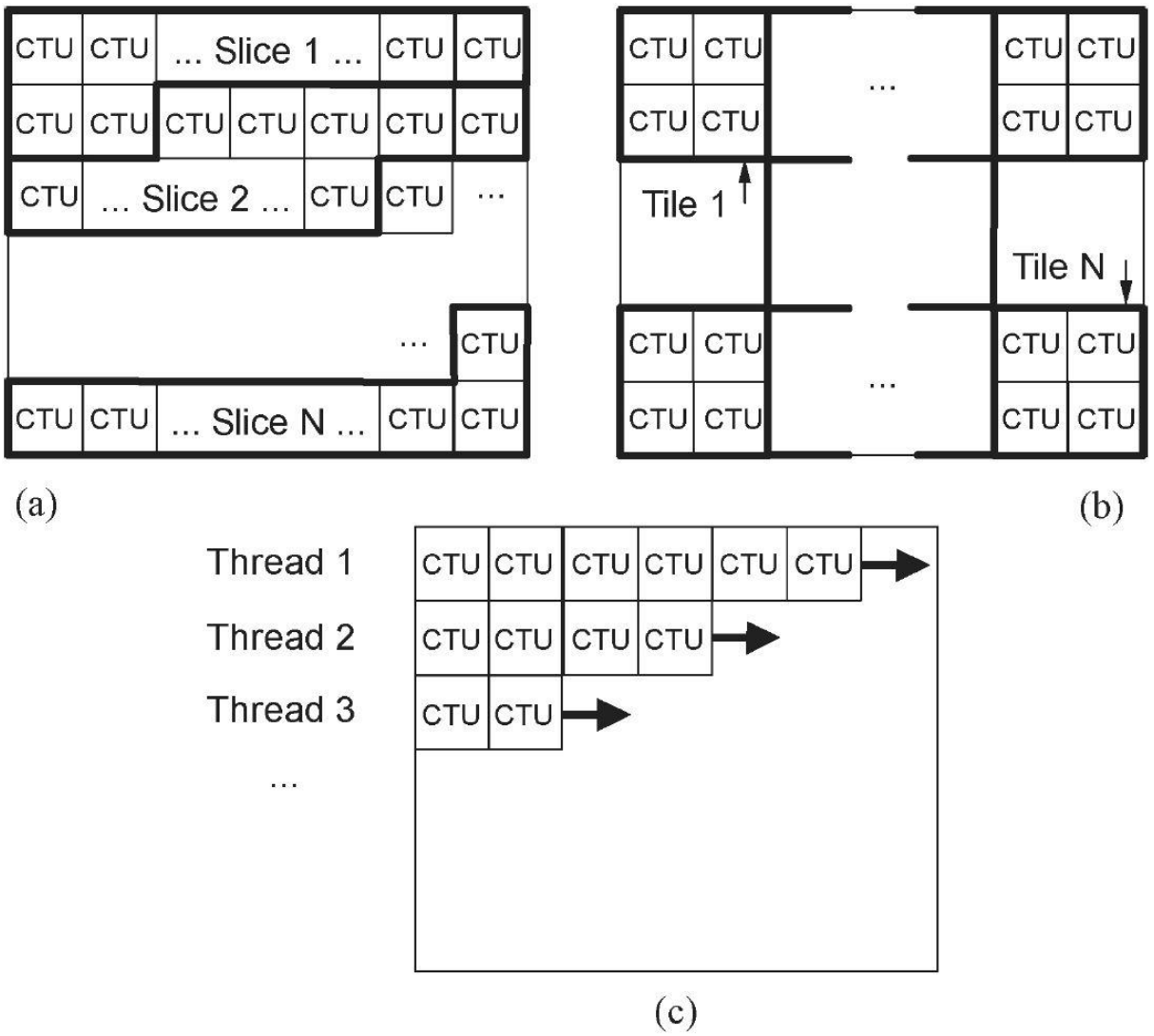


Figure 30. The division of picture into slices (a), tiles(b) and threads(c)

The parallel processing in previous video codecs, like in H.264, has been through slice-level or block-level parallelism [66]. Although, by adding more slices in a frame the coding can be easily parallelized, the cost of coding efficiency is reduced due to breaking up the training of the context models and the inability to cross slice boundaries for context selection [66]. Additionally, the pixels from neighbouring slices cannot be used in the prediction stage and also a header with additional start code information needs to be present in the bitstream [66]. Parallel processing of CTU blocks on the other hand has the downside of breaking some dependencies on tile borders [67, p. 397].

In addition to slice-level and block-level parallelism, HEVC also has wavefront parallel processing method available for even better parallel processing performance [67, p. 398]. WPP fully maintains all data dependencies of block-level parallelism without the efficiency loss of slice-level parallel processing. Because all CTUs are processed in an order in which their inter-

dependencies never need to be broken [67, p. 398]. The dependencies and the structure of WPP processing method is shown on Figure 31 [67, p. 398]. It can be seen that the scheme scales well with the number of available processing cores and with higher video resolutions.

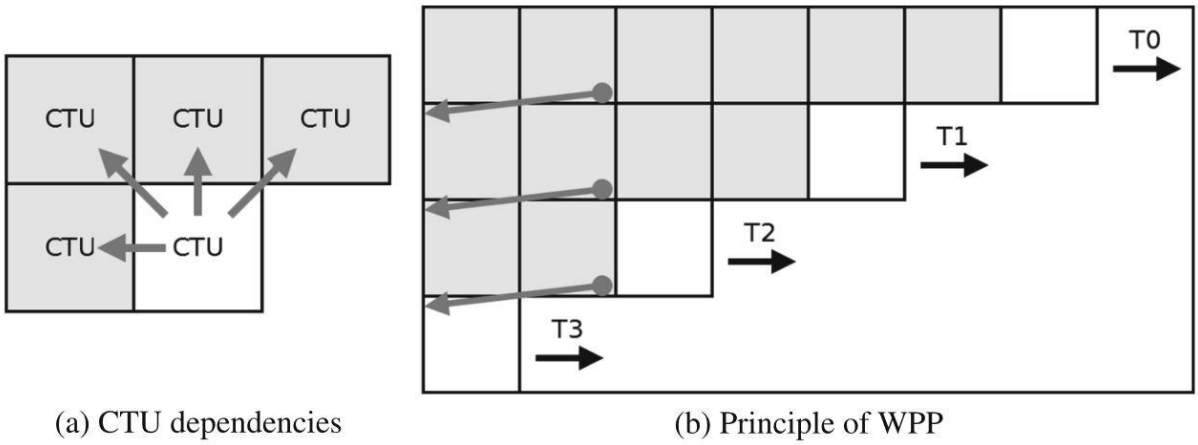


Figure 31. The dependencies between CTUs and the structure of WPP processing method

Appendix 6

Average Percentages of GPUs Used by Steam Users

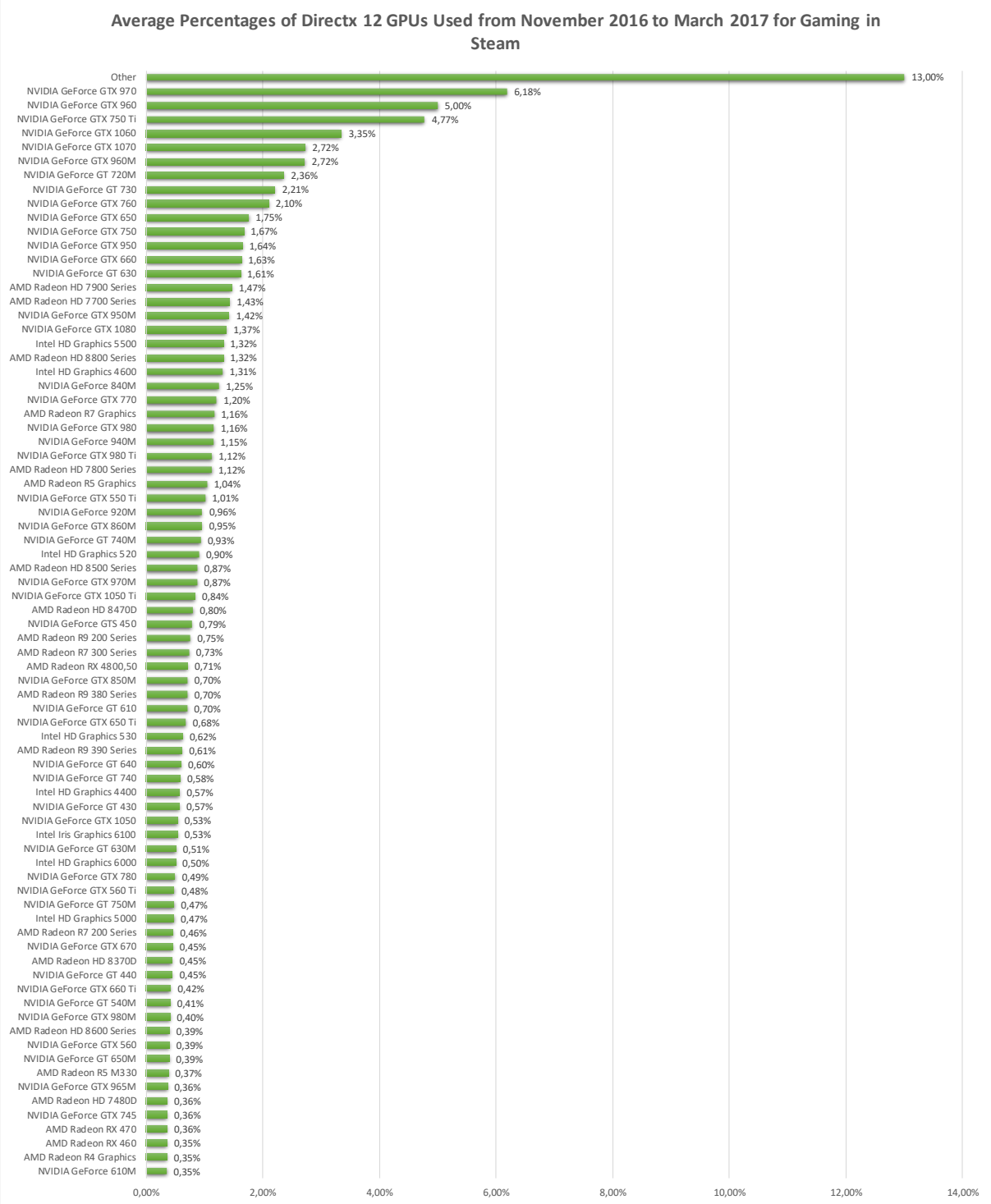
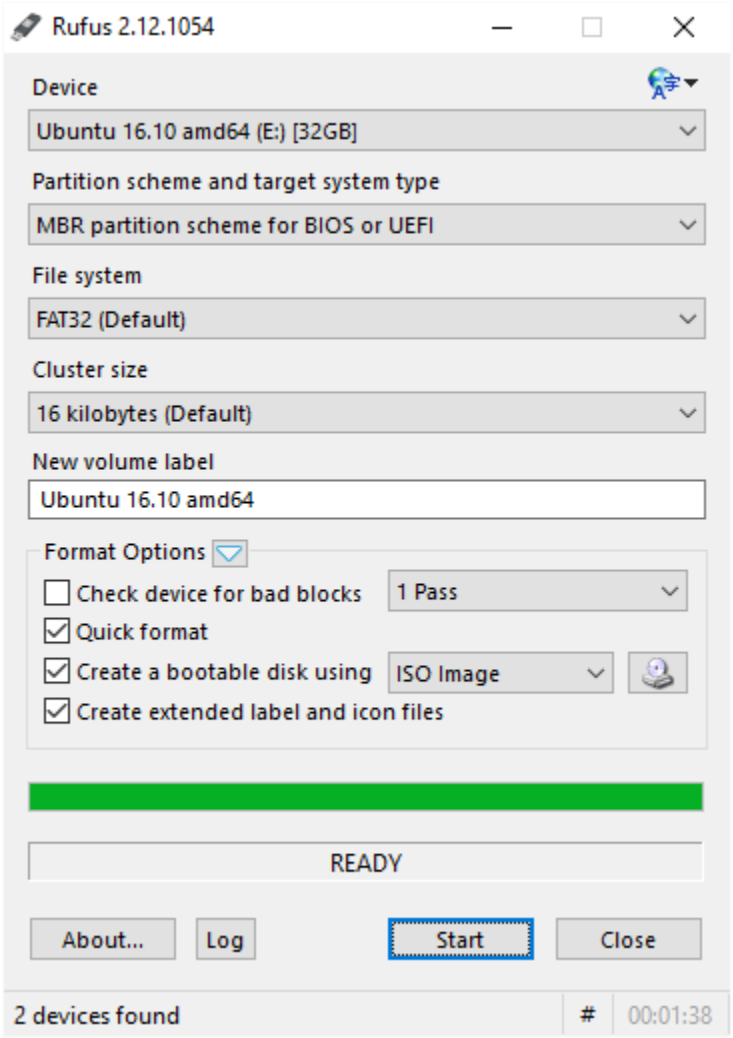


Figure 32. The average percentages of GPUs used from November 2016 to March 2017 by Steam users

Appendix 7

FFmpeg Set-up Instructions

1. DOWNLOAD Ubuntu OR SIMILAR
    a. http://ftp.estpak.ee/pub/centos/7/isos/x86_64/CentOS-7-x86_64-Everything-1611.iso (31.01.2017)
    b. https://www.ubuntu.com/download/desktop (06.02.2017)
2. SET-UP USB STICK
    a. DOWNLOAD Rufus https://rufus.akeo.ie/downloads/rufus-2.12.exe (31.01.2017)
    b. CREATE BOOTABLE USB WITH Rufus TOOL



3. INSTALL Ubuntu
    a. Set user password (fmpeg2017mag for example)
    b. Login to user account and finish setup
4. CONFIGURE NVENC AND INSTALL CUDA

a. http://developer.download.nvidia.com/compute/redist/ffmpeg/1511-patch/FFMPEG-with-NVIDIA-Acceleration-on-Ubuntu_UG_v01.pdf (06.02.2017)
b. (more information on FFmpeg using Nvidia Video Codec SDK) https://developer.nvidia.com/ffmpeg (07.02.2017)
c. (CUDA SDK download page) https://developer.nvidia.com/cuda-downloads (16.02.2017)
d. (Additional information on setting up CUDA) http://docs.nvidia.com/cuda/cuda-installation-guide-linux/#axzz4YmjMOKbj (16.02.2017)

5. INSTALL FFmpeg
    a. (additional information on hardwae acceleration for FFmpeg) https://trac.ffmpeg.org/wiki/HWAccelIntro (16.02.2017)
    b. (FFmpeg compillation quides for CentOS and Ubuntu) https://trac.ffmpeg.org/wiki/CompilationGuide/Centos (31.01.2017) https://trac.ffmpeg.org/wiki/CompilationGuide/Ubuntu (06.02.2017)

Comments:

- For installing CUDA and configuring NVENC appropriate GPU should be installed into the computer.
- Some information in the available guides is outdated (mostly because of the rapid development of new hardware, drivers and software). Combining guides presented in "4. CONFIGURE NVENC AND INSTALL CUDA" section resulted in working FFmpeg with NVENC capabilities for encoding videos into HEVC standard.
- HDD and SSD storage devices might botlleneck possible maximum transcoding speededs, one solution to overcome it is to use faster RAM Drives: (tutorial for creating RAMdisk) http://www.hecticgeek.com/2015/12/create-ram-disk-ubuntu-linux/ (17.04.2017)

Appendix 8

The Descriptions and Objectives of Conducted Test Runs

TEST RUN 1

Objective:

To gather information on the CPU coding speeds when 2 CPU threads are used on i7-6700

Description:

Transcoding test sequences into AVC and HEVC formats by using 2 threads on i7-6700 CPU (same SSD was used for reading input files to the *transcodeTestScript* and storing resulting output files; three coding iterations on each test sequence were run alternately)

TEST RUN 2

Objective:

To gather information on the CPU coding speeds when 4 CPU threads are used on i7-6700

Description:

Transcoding test sequences into AVC and HEVC formats by using 4 threads on i7-6700 CPU (same SSD was used for reading input files to the *transcodeTestScript* and storing resulting output files; three coding iterations on each test sequence were run alternately)

TEST RUN 3

Objective:

To gather information on the CPU coding speeds when 8 CPU threads are used on i7-6700

Description:

Transcoding test sequences into AVC and HEVC formats by using 8 threads on i7-6700 CPU (same SSD was used for reading input files to the *transcodeTestScript* and storing resulting output files; three coding iterations on each test sequence were run concurrently)

TEST RUN 4

Objective:

To gather information on the difference of CPU coding speeds when RAM Drive is used instead of SSD for reading input files into the script

Description:

Transcoding test sequences into AVC and HEVC formats by using 8 threads on i7-6700 CPU (RAM Drive was used for reading input files to the *transcodeTestScript* and SSD for storing resulting output files; three coding iterations on each test sequence were run alternately)

TEST RUN 5

Objective:

To gather information on the difference of CPU coding speeds when an older generation i7-4790K CPU is used

Description:

Transcoding test sequences into AVC and HEVC formats by using 8 threads on i7-4790K CPU (same SSD was used for reading input files to the *transcodeTestScript* and storing resulting output files; three coding iterations on each test sequence were run concurrently)

TEST RUN 6

Objective:

To gather information on the difference of CPU coding speeds when an older generation server CPU is used

Description:

Transcoding test sequences into AVC and HEVC formats by using all available threads on E5-2660 v2 CPU (server storage was used for reading input files to the *transcodeTestScript* and storing resulting output files; three coding iterations on each test sequence were run concurrently)

TEST RUN 7

Objective:

To gather information on the coding speeds of a GPU with NVENC hardware acceleration, when input data is read into the script from a single SSD and three transcoding iterations on test videos are done concurrently

Description:

Transcoding test sequences into AVC and HEVC formats by using single GeForce GTX 1060 GPU (same SSD was used for reading input files to the *transcodeTestScript* and storing resulting output files; three coding iterations on each test sequence were run concurrently)

TEST RUN 8

Objective:

To gather information on the coding speeds of a GPU with NVENC hardware acceleration, when input data is read into the script from a single SSD and three transcoding iterations on test videos are done alternately

Description:

Transcoding test sequences into AVC and HEVC formats by using single GeForce GTX 1060 GPU (same SSD was used for reading input files to the *transcodeTestScript* and storing resulting output files; three coding iterations on each test sequence were run alternately)

TEST RUN 9

Objective:

To gather information on the coding speeds of a GPU with NVENC hardware acceleration, when input data is read into the script from a RAM Drive and resulting output files are written onto the SSD

Description:

Transcoding test sequences into AVC and HEVC formats by using single GeForce GTX 1060 GPU (RAM Drive was used for reading input files to the *transcodeTestScript* and SSD for storing resulting output files; three coding iterations on each test sequence were run alternately)

TEST RUN 10

Objective:

To gather information on the coding speeds of an older GPU with NVENC hardware acceleration, when input data is read into the script from a RAM Drive and resulting output files are written onto the SSD

Description:

Transcoding test sequences into AVC and HEVC formats by using single GeForce GTX 970 GPU (RAM Drive was used for reading input files to the *transcodeTestScript* and SSD for storing resulting output files; three coding iterations on each test sequence were run alternately)

TEST RUN 11

Objective:

To gather information on the coding speeds of multiple GPU run simultaneously with NVENC hardware acceleration, when input data is read into the script from a RAM Drive and resulting output files are written onto the SSD

Description:

Transcoding test sequences into AVC and HEVC formats by using GeForce GTX 970 and GeForce GTX 1060 GPUs simultaneously (RAM Drive was used for reading input files to the *transcodeTestScript* and SSD for storing resulting output files; three coding iterations on each test sequence were run alternately; the file processed at one time was the same on both GPUs; this is the only test run outputting 2 logfiles)

TEST RUN 12

Objective:

To gather information on the coding speeds of multiple sequences coded on a single GPU with NVENC hardware acceleration, when input data is read into the script from a RAM Drive and resulting output files are written onto the same RAM Drive

Description:

Transcoding two test sequences simultaneously into HEVC format by using single GeForce GTX 1060 GPU (RAM Drive was used for reading input files to the *transcodeTestScript* and for storing resulting output files; results were gathered coding in total six test sequences in three runs, where two sequences were encoded at one time)

Appendix 9

FFmpeg Commands Used for Transcoding Tests

Commands in Table 7 are marked with yellow, green and blue accents. Yellow accent marks the necessary meta-information needed for encoding YUV sequences, green highlights all configuration parameters that set the settings for output media, and blue shows the part responsible for logging the information of transcoding speeds.

Transcoding Process	**Transcoding Device**	**Corresponding FFmpeg Command**
From YUV to AVC	i7-6700 CPU	ffmpeg -loglevel verbose -f rawvideo -vcodec rawvideo -s 3840x2160 -r 120 -i $inDir/$2 -c:v libx264 -preset $presetLibx -threads $nrOfThreads -b:v $bAVC -y $outDir/$3 2>&1 | tee -a $logDir/$logFile
	i7-4790K CPU	
	E5-2660 v2 CPU	
	GeForce GTX 1060 GPU	ffmpeg -loglevel verbose -f rawvideo -vcodec rawvideo -s 3840x2160 -r 120 -i $inDir/$2 -c:v h264_nvenc -gpu $gpuSelect -preset $presetNvenc -b:v $bAVC -y $outDir/$3 2>&1 | tee -a $logDir/$logFile
	GeForce GTX 970 GPU	
From AVC to HEVC	i7-6700 CPU	ffmpeg -loglevel verbose -i $inDir/$4 -c:v libx265 -preset $presetLibx -threads 0 -x265-params pools=$nrOfPools -b:v $bHEVC -y $outDir/$3 2>&1 | tee -a $logDir/$logFile
	i7-4790K CPU	
	E5-2660 v2 CPU	
	GeForce GTX 1060 GPU	ffmpeg -loglevel verbose -i $inDir/$4 -c:v hevc_nvenc -gpu $gpuSelect -preset $presetNvenc -b:v $bHEVC -y $outDir/$3 2>&1 | tee -a $logDir/$logFile
	GeForce GTX 970 GPU	
From YUV to HEVC	i7-6700 CPU	ffmpeg -loglevel verbose -f rawvideo -vcodec rawvideo -s 3840x2160 -r 120 -i $inDir/$2 -c:v libx265 -preset
	i7-4790K CPU	

	E5-2660 v2 CPU	$presetLibx -threads 0 -x265-params pools=$nrOfPools -b:v $bHEVC -y $outDir/$3 2>&1 | tee -a $logDir/$logFile
	GeForce GTX 1060 GPU	ffmpeg -loglevel verbose -f rawvideo -vcodec rawvideo -s 3840x2160 -r 120 -i $inDir/$2 -c:v hevc_nvenc -gpu $gpuSelect -preset $presetNvenc -b:v $bHEVC -y $outDir/$3 2>&1 | tee -a $logDir/$logFile
	GeForce GTX 970 GPU	
	GeForce GTX 1060 GPU (test was run separately, outside the *transcodeTestScript*)	find /mnt/RAM_disk/ -type f -name *.yuv -print | sed 's/.yuv$//' | xargs -n 1 -I@ -P 2 ffmpeg -loglevel verbose -f rawvideo -vcodec rawvideo -s 3840x2160 -r 120 -i "@.yuv" -c:v hevc_nvenc -gpu $gpuSelect -preset $presetNvenc -b:v $bHEVC "@.mp4" 2>&1 | tee -a $logDir/$logFile

Table 7. FFmpeg commands used for transcoding tests on CPUs and GPUs

Appendix 10

Parameters Used in Transcoding Commands

Parameter	Description	Parameter Values Used on APIs		
		libx264	**libx265**	**NVENC**
-c:v …	Parameter for selecting video coder	libx264	libx265	h264_nvenc, hevc_nvenc
-b:v …M	Parameter for selecting video target bitrate (M stands for Megabits per second)	19.2	9.6	19.2, 9.6
-preset …	determines the speed for encoding process at the expense of compression efficiency	slow	slow	slow
-gpu …	Parameter for selecting the GPU for video coding	-	-	0, 1
-threads …	Set umber of encoding threads used	2, 4, 8	-	-
-pools …	Set umber of encoding threads used	-	2, 4, 8	-

Table 8. Parameters used in transcoding commands and their description

Appendix 11

CVNN Codes Corresponding to Test Objectives

Command Description	CVNN
Sequence converted from YUV to AVC with libx264 on a CPU	1VNN
Sequence converted from AVC to HEVC with libx265 on a CPU	2VNN
Sequence converted from YUV to HEVC with libx265 on a CPU	3VNN
Sequence converted from YUV to AVC with h264_nvenc on a GPU	4VNN
Sequence converted from AVC to HEVC with hevc_nvenc on a GPU	5VNN
Sequence converted from YUV to HEVC with hevc_nvenc on a GPU	6VNN
Sequence converted from YUV to AVC with h264_nvenc on 2 GPUs	7VNN
Sequence converted from AVC to HEVC with hevc_nvenc on 2 GPUs	8VNN
Sequence converted from YUV to HEVC with hevc_nvenc on 2 GPUs	9VNN
Two sequences converted simultaneously from YUV to HEVC with hevc_nvenc on a GPU	0VNN

Table 9. CVNN codes corresponding to test objectives

Appendix 12

Names and Properties of Test Sequences

Test Sequence	CVNN	Properties
Beauty	C1NN	• YUV format • RAW container • 3840x2160 resolution (4K) • Bit depth of 8 bits • 600 frames, *300 frames • 120 frames per second • 7.5 GB, *3.7 GB
Bosphorus	C2NN	
HoneyBee	C3NN	
Jockey	C4NN	
ReadySetGo	C5NN	
ShakeNDry*	C6NN	
YachtRide	C7NN	

Table 10. Names and Properties of Test Sequences

Appendix 13

Markings Used on the Graphs Presenting Test Results

Device	gtx1060	gtx970	i7 6700	i7-4790K	E5-2660 v2
Abbreviation	1060	970	I6	I4	E5
YUV Files on SSD	s	s	s	s	-
YUV Files on RAM Drive	r	r	r	r	-
Videos Processed Alternately	a	-	-	-	-
Videos Processed Concurrently	c	-	-	-	-
2 CPU Threads	-	-	t2	t2	-
4 CPU Threads	-	-	t4	t4	-
8 CPU Threads	-	-	t8	t8	-
Multi GPU Configuration	MC	MC	-	-	-
Single GPU Configuration	SC	SC	-	-	-
Multiple Videos on One Device	MV	MV	-	-	-
Single Video on One Device	SV	SV	-	-	-

Table 11. Markings used on the graphs presenting test results

Appendix 14

Specialists Contacted During Research

Specialist Contacted	Position	Communication Method
Erkki Põllu	Head of Network and Service at Starman	By email and face-to-face meeting
Aimar Prous	Technical and Production Director at Kanal 2	By email and face-to-face meeting
Priit Pärg	Head of Video Editing Department at ERR	By email
Mait Tafenau	IT Manager at Delfi	By email and face-to-face meeting
Peeter Linnap and Rihhard Krüüner	Media and Advertising at Tartu Art College	By email and a phone call
Toomas Petersell	Senior Specialist for Video Equipment, University of Tartu	By email and face-to-face meeting
Jarno Vanne	Assistant Professor at Tampere University of Technology and a Member in the Ultra Video Group	By email

Table 12. Specialists contacted during the primary research phase

Appendix 15

Certification Received from PUMPS 2016

Figure 33. Certification from PUMPS 2016