UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Technology
Robotics and Computer Engineering Curriculum

Léon Glorieux

Evaluation of image sequence (video) compression with an embedded processing system for future space applications

Master's Thesis (30 ECTS)

Supervisor(s): Msc. Quazi Saimoon Islam
Msc. Ric Dengel

Tartu 2023

Evaluation of image sequence (video) compression with an embedded processing system for future space applications

Abstract:

In recent years, the accessibility to Graphics Processing Unit (GPU)s and hardware accelerators for space missions has increased drastically with implementable Commercial Off-The-Shelf (COTS) solutions. As a result, the onboard computing power has increased for many space missions, and the performances of the different video compression methods have been improved. With such evolution in computing resources, it is crucial to analyze the different methods and record their performances, to find efficient compression methods in this new configuration. To find efficient compression methods, we developed a benchmark, to assess multiple methods on their compression ratios, power consumption, execution time and the difference of quality between the input and output images on a low resource configuration, adapted for space missions. From the analysis led in this thesis 4, on the standard codecs, the more lossless the compression, the higher the benefits from the acceleration. The machine learning approach shows promising results for the future, and the Consultative Committee for Space Data Systems (CCSDS) 122 despite the GPU acceleration was outperformed by Advanced Video Coding (H.264) and High-Efficiency Video Coding (H.265).

Keywords GPU, hardware accelerator, COTS, CCSDS, H.265, H.264

CERCS: T320 Space technology, T111 Imaging, image processing

Manussüsteemiga pildijada (video) kompresseerimise hindamine tulevaste kosmoserakenduste jaoks

Lühikokkuvõte:

Viimastel aastatel on ligipääs graafikakaartidele (GPU) ja riistvarakiirenditele kosmosemissioonide jaoks suurenenud märkimisväärselt tänu rakendatavatele kommertslahendustele (COTS). Selle tulemusena on paljude missioonide pardal olev arvutusvõimsus suurenenud ning erinevate kompressioonimeetodite jõudlus on paranenud. Selle mõjul on oluline analüüsida erinevaid meetodeid, et tuvastada tõhusaimad meetodid. Selleks arendasime võrdlusalused mitme meetodi omavaheliseks võrdluseks järgneva alusel: kompressioonisuhe, energiatarve, täitmisaeg ja kvaliteedi erinevus sisend- ja väljundpiltide vahel madala ressursikonfiguratsiooni puhul, mis on kohandatud kosmosemissioonide jaoks. Analüüsist selgub, et standardsete koodekite puhul mida suurema kaoga kompresseeritakse, seda suuremad on kiirendusest tulenevad eelised. Masinõppe lähenemine näitab lootustandvaid tulemusi ning CCSDS 122, hoolimata GPU kiirendusest, jäi alla H.264 ja H.265-le.

Võtmesõnad: GPU, riistvaraline kiirendus, COTS, CCSDS, H.265, H.264

CERCS: T320 Kosmosetehnoloogia, T111 Pilditehnika

Contents

1 Introduction **10**
- 1.1 Background . . . 10
- 1.2 Problem statement . . . 10
- 1.3 Goals . . . 11

2 Literature Review **12**
- 2.1 General Notions . . . 12
  - 2.1.1 Lossy and Lossless compression . . . 12
  - 2.1.2 Compression methods . . . 12
  - 2.1.3 Standard video codecs . . . 13
- 2.2 Specific Notions of Space application . . . 14
  - 2.2.1 On-board processing . . . 14
  - 2.2.2 Space standard . . . 15
- 2.3 Other compression benchmarks . . . 16

3 Methodology **17**
- 3.1 Test Facilities and Hardware Resources . . . 17
- 3.2 Software . . . 19
- 3.3 Tools . . . 20
- 3.4 Benchmarking methods . . . 20
  - 3.4.1 CCSDS 122 . . . 20
  - 3.4.2 H.264 . . . 21
  - 3.4.3 H.265 . . . 22
  - 3.4.4 Machine learning approach . . . 22
- 3.5 Metrics . . . 23
  - 3.5.1 Computational Time . . . 23
  - 3.5.2 Quality . . . 23
  - 3.5.3 Compression ratio . . . 24
  - 3.5.4 Power . . . 24
- 3.6 Implementation . . . 24
  - 3.6.1 CCSDS 122 . . . 24
  - 3.6.2 H.264 and H.265 . . . 25
  - 3.6.3 Machine Learning . . . 25
  - 3.6.4 Testbench Pipeline . . . 26

4 Results and analysis **27**
- 4.1 H.264 and H.265 tests . . . 27
  - 4.1.1 H.264 . . . 27
- 4.2 H.265 test . . . 30

4.3 CCSDS test 34
4.4 CompressAI models test 35

5 Discussion **36**

6 Conclusion **37**

References **43**

Lisad **44**
I. Additional data 44
A/ Figures 44
B/ Code 45
C/ Tables 47
D/ Tools 50
II. Licence 52

List of Tables

1 H.264 Peak Signal-to-Noise Ratio (PSNR) difference between CPU and NVENC (from annex tables 5 and 6) 29
2 H.264 PSNR difference between CPU and GPU (from annex table 7 and 8) 33
3 CCSDS compression ratio table 35
4 Benchmark of machine learning models on video compression 35
5 H.264 PSNR statistics on the GPU performances 47
6 H.264 PSNR statistics on the CPU performances 48
7 H.265 PSNR statistics on the GPU performances 49
8 H.265 PSNR statistics on the CPU performances 50

List of Figures

1 Example compression on a photo [1]. 12
2 Space Bunker images 17
3 "Orin System on Chip (SoC) Block Diagram" [2]. 19
4 " Three-Level 2-d Discrete Wavelet Transform (DWT) Decomposition of an Image" [3]. 21
5 CCSDS Shematics 21
6 "Overview of the benchmark process". 26
7 Graph of the evolution of the compression ratio compared to the change in the quantization parameter. 27
8 Graph of the evolution of the execution time compared to the change in the quantization parameter. 28
9 Graph of the evolution of the PSNR image quality compared to the change in the quantization parameter. 29
10 H.264 Graph of the evolution of the mean Power consumption compared to the change in the quantization parameter. 30
11 Graph of the evolution of the compression ratio compared to the change in the quantization parameter. 31
12 Graph of the evolution of the execution time compared to the change in the quantization parameter. 32
13 Graph of the evolution of the PSNR image quality compared to the change in the quantization parameter. 33
14 H.265 Graph of the evolution of the mean Power consumption compared to the change in the quantization parameter. 34
15 Jetson TOP (JTOP) panel showing no GPU use while running CompressAI models 36
16 "Kuupkulgur Field of View (FOV)". 44
17 265 Graph of the evolution of the Energy consumption compared to the change in the quantization parameter. 45
18 H.264 Graph of the evolution of the Energy consumption compared to the change in the quantization parameter. 45

List of Acronyms

H.264 Advanced Video Coding

H.265 High-Efficiency Video Coding

H.266 Versatile Video Coding

VP9 Video Codec 9 (designed by Google)

AV1 AOMedia Video 1

NASA National Aeronautics and Space Administration

ESA European Space agency

GPU Graphics Processing Unit

CPU Central Processing Unit

COTS Commercial Off-The-Shelf

SoC System on Chip

SSD Solid-State Drive

SD Secure Digital

AI Artificial intelligence

JAXA Japan Aerospace Exploration Agency

ISRO Indian Space Research Organisation

CCSDS Consultative Committee for Space Data Systems

NN Neural Network

AAC Advanced Audio Coding

MKV Matroska Video

AVI Audio Video Interleave

WebM Web Multimedia

MP4 MPEG-4(Moving Picture Experts Group-4)

GPU4S Graphical Processing Unit for Space

OBPMark On Board Processing Benchmark

FPGA Field-Programmable Gate Array

LEO Low Earth Orbit

BSC Barcelona Supercomputing Center

UPC Universitat Politècnica de Catalunya

DVC Deep Video Compression

FVC Feature-space Video Coding

VCT Video Compression Transformer

MSU Moscow State University

FOV Field of View

CUDA Compute Unified Device Architecture

NVENC NVIDIA Encoder

NVDEC NVIDIA Decoder

ROS Robot Operating System

FFMPEG Fast Forward Moving Picture Experts Group

DWT Discrete Wavelet Transform

PSNR Peak Signal-to-Noise Ratio

MSE Mean Squared Error

bps Bits per Second

crf Constant Rate Factor

DLA Deep Learning Accelerator

CNN Convolutional Neural Network

PVA Programmable Vision Accelerator

HDR ISP High Dynamic Range Image Signal Processor

JTOP Jetson TOP

qp Quantization Parameter

ssh Secure Shell

bps bit size of each pixel

SSIM Structural Similarity

DCT Discrete Cosine Transform

CTU coding Tree Unit

HD High Definition

1 Introduction

This thesis is a benchmark of multiple video compression methods on a space-relevant System on Chip (SoC). This work will cover multiple video compression methods which are interesting to benchmark for future possible applications in the space industry. The Kuupkulgur[4] Rover development originates this study, led in parallel to the project.

1.1 Background

Space exploration has been evolving very fast in the past years with very ambitious projects such as Artemis [5] from National Aeronautics and Space Administration (NASA) whose final objective would be to build a lunar camp on the south pole of the moon for exploration, scientific objectives, in-situ resource utilization, and may be used as a hub for further space exploration in the future. For exploration space missions, multiple rovers[6] such as Sojourner, Spirit and Opportunity, Curiosity, and Perseverance have been sent in the past and provided excellent results. On such predicate, multiple project developments have started around small rovers [7][4] [8]. Their objective is to provide a small cost-efficient platform to perform scientific experiments, test payloads and explore. With such applications, all resources available to the rovers are vital. Therefore, it is crucial to optimise, all the processes to have a viable end product. This benchmark continues the effort in the line of optimisation of the different processes for space systems with work on image sequence/video compression methods analysis, which may be used by the Kuupkulgur [4], a project conducted at Tartu Observatory, in the University of Tartu[9].

1.2 Problem statement

Despite the evolution of technology, many space missions get through an information bottleneck limiting its amount and quality. Nevertheless, in such an unforgiving environment, many subsystems need to run at all times, and some others must be active at specific moments to ensure the mission's success. Thus, it is crucial to set up efficient data compression methods to save resources while exchanging as much information as possible within an acceptable level of quality. For this specific problem, the Consultative Committee for Space Data Systems (CCSDS) designed standards which are updated regularly, to adapt to the changing technologies. Recently, a lot of effort has been oriented towards the availability of Graphics Processing Unit (GPU) and hardware accelerators in space [10][11]. These efforts can be seen in multiple projects, such as:

- **On Board Processing Benchmark (OBPMark) [10]:** a framework built to test the space standards on both Central Processing Unit (CPU) and GPU, for comparisons

- **Graphical Processing Unit for Space (GPU4S) [11]:** a research project, to build an optimized, and affordable computing system implementing GPU for space.

With the increasing accessibility of GPU, the computing power of many space missions is expected to increase. As GPUs may become a norm, it is crucial to test methods designed for space on both CPU, GPU and hardware accelerators, but also test other computationally expensive methods on relevant hardware platforms.

1.3 Goals

As GPUs have become more accessible, the average onboard computing power has increased significantly [12] [13] and the standing point of image/video compression for space has become unclear. Therefore the purpose of this work is to test and examine the performance of CCSDS 122 [3] (popular video compression standard for space missions) on a space relevant environment and compare these to the performances of Advanced Video Coding (H.264), the most used video compression method, High-Efficiency Video Coding (H.265) its more computationally demanding descendant, and finally, pre-trained models of a rigorous Neural Network benchmark project named compressAI [14]. Objectives of the thesis include the following:

- Find the boundaries of GPU and Hardware accelerators compared to CPU for image/video compression
- Provide a baseline based on performances for compression method selection in future space missions
- Develop a pipeline to test other methods in the future

2 Literature Review

2.1 General Notions

2.1.1 Lossy and Lossless compression

A very fundamental notion about compression that must be understood is the difference between lossy and lossless compression. Both of those methods share a common point of finding patterns, redundancy or similarities that allow data that would reduce the overall size of the data. Lossy differs from lossless in the way that it approximates the data without requiring a perfect replacement. For this reason, it can achieve higher compression ratios but also sacrifices some levels of detail according to the degree approximation as shown in the example in figure 1.

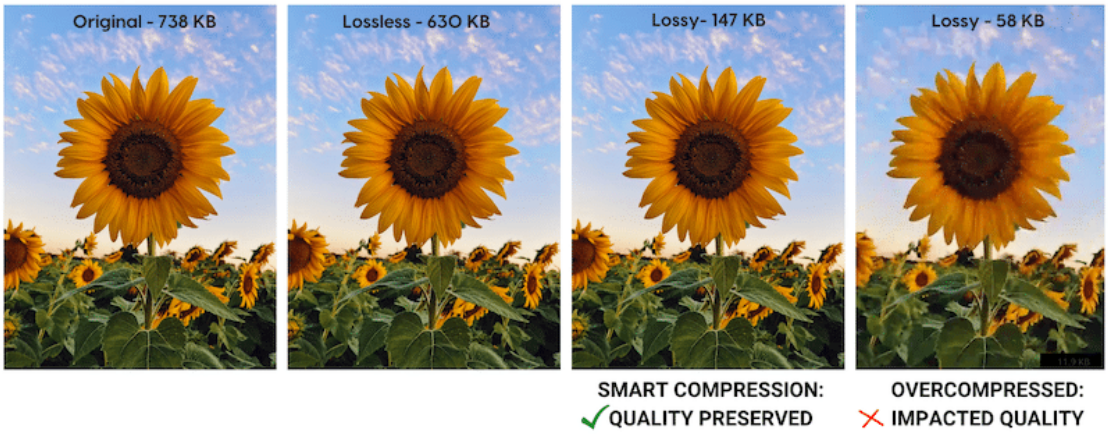


Figure 1. Example compression on a photo [1].

2.1.2 Compression methods

To compress a video, there are various steps which are performed on all the different standards. Each of these steps can be implemented in multiple ways, with different approaches and varying results, but the general techniques remain the same [15] [16]:

- **Partitioning:** Splits the image into multiple blocks
- **Prediction:** predicts similar information from previous data.
  - **Spatial Redundancy:** Finds redundancy within a frame from patterns, neighbouring pixels, etc.

  - **Temporal Redundancy:** Finds similarities between images to avoid re-encoding the same components multiple times.

- **Quantization:** Regroups similar information into batches
- **Entropy Coding:** Uses the distribution to assign smaller representations to the most frequent patterns [17].
- **Motion Compensation:** Predict pixels/patterns positions, to reduce motion blur

The decision to apply one of these methods over another often comes down to a trade-off between computational intensity, compression ratios (between the original and encoded file) and the associated efficiencies.

2.1.3 Standard video codecs

As the amount of possible technique combinations is large, standard codecs have emerged to help. A codec (from coding/decoding) represents all the techniques applied on a file from the original to the encoded version, and vice-versa.

Among all codecs, research into relevant standards[18],[19],[20], [21], [22], [23], [14] have yielded the following:

- **H.264/AVC [24]:** H.264, is a widely used video compression standard. It offers efficient compression while maintaining high video quality. It has been commonly utilized in applications such as video streaming, Blu-ray discs, and video conferencing.
- **H.265/HEVC [25]:** H.265, is a successor to H.264 and provides better compression efficiency. It is particularly used for High Definition (HD) content until 4K.
- **Versatile Video Coding (H.266)/VVC [26]:** H.266, represents the latest advancement of the H.264 and H.265 family. It aims to further improve compression efficiency compared to H.265, making it suitable for high-resolution video content, including 8K and beyond.
- **Video Codec 9 (designed by Google) (VP9) [27]:** VP9 is an open-source video compression standard developed by Google. It competes with H.265 and offers efficient compression, making it popular for web-based video streaming services, especially on platforms like YouTube.
- **AOMedia Video 1 (AV1) [28]:** AV1 is another open-source video codec designed for efficient compression and high video quality. It is developed by the Alliance for Open Media. AV1 is gaining popularity for online video streaming.

Since H.264 is a widely used codec, it was chosen as a first test for standards which are not designed for space. Then H.265 was chosen over VP9 to test the processing limits with a similar implementation to H.264 for comparisons. VP9 was not selected as more computing resources than H.265 are required.
Specifics and details about the implementation of formats fall outside the scope of this study. Nevertheless, for a better global understanding, an overview has been provided. Formats regroup a standardized video codec, an audio codec and metadata to define the file. Multiple formats exist, offering a range of different codec combinations. Some popular formats are:

- **MPEG-4(Moving Picture Experts Group-4) (MP4):** H.264 video codec, Advanced Audio Coding (AAC) audio codec
- **Web Multimedia (WebM):** VP9 video codec, Opus or Vorbis audio codec
- **Audio Video Interleave (AVI):** can accommodate various codecs
- **Matroska Video (MKV):** can contain various video and audio codecs

2.2 Specific Notions of Space application

2.2.1 On-board processing

As mentioned in [11] the existing Commercial Off-The-Shelf (COTS) hardware computing systems are not perfectly fit to the constraints for many space applications. Then, it becomes necessary to develop a customized system if the risks become too large. However, the skills required and fabrication methods are mostly held by the companies on the market. It is possible to have partnerships, but the processes and hardware cannot be disclosed, making it very difficult to choose the right partner for a mission. When a computer is developed for space applications, it gets very expensive as one must test all the components, respect physical conditions while maintaining a minimum performance level, and run all the necessary processes of the mission for a functional end product. The physical constraints that must be respected are :

- **Limited Power:** (most missions rely on solar panels and must avoid overheating)
- **Resistance to vibrations:** (most dangers at launch)
- **Radiation hardening:** (for cosmic rays and solar radiations)
- **Vacuum:** (outgassing and pressure problems)
- **Extreme temperatures :** From launch to end of life, the systems are subject to cold temperatures, down to 270 °C in outer space, and up to very high temperatures,

up to 2000°C and more. These temperatures are mitigated to avoid damaging the hardware. The larger the working temperature range, the less constrained the design is.

- **Mass and size :** (essentially for the launch)

As a result, most designs are currently built on CPU or Field-Programmable Gate Array (FPGA) on separate devices, to have the most suited computing system for a mission. Nevertheless, it is highly expensive to develop and can't be used for everything. In order to respond to the growing interest toward GPUs and hardware accelerators, NASA performed a first benchmark of the different COTS designs in 2018 [12]. For small missions and Low Earth Orbit (LEO), it was deemed possible to use them as cheaper options. Then, it became clear that using GPU for space technologies was crucial for enhancing the computing performances of avionics and the GPU4S [11] was designed by the scientific community. It provides a hardware framework, access to relatively cheap GPUs and a computing system adapted for most missions. The "Case-Study for Integration of COTS SoC Devices in Reliable Space Systems for On-Board Processing" [13] recently published under Barcelona Supercomputing Center (BSC), European Space agency (ESA) and Universitat Politècnica de Catalunya (UPC) showed that the NVIDIA SoC provide reliable alternatives for space applications with additional embedded software accelerators. Since GPUs and hardware acceleration has become easily accessible for many space projects, it is necessary to run tests of the existing compression methods and provide more information on the COTS on-board processing performances.

2.2.2 Space standard

Despite the quick evolution of the processing systems and their computing capabilities, heavy calculations or processes remain a limiting factor for the use of certain methods. Consequently, these methods have been applied only within specific scenarios. In the past H.264 computations used to be too heavy. With the progression of the processing power, it has become a reference standard for a multiplicity of applications. The global norm is transitioning to H.265 which continues to present challenges due to its high computational requirements.
As this computing limit exists on Earth, it is an even larger issue in space, with processing units that must fit into very specific criteria, as in list 2.2.1. To ensure the availability of codecs adapted to the performances of space computing systems, the CCSDS built a series of standards to be used on data compression, for space applications. They offer a good baseline to find the right compromise between compression and computing power. The CCSDS[29] standards are used by many actors in the space industry, such as the ESA, NASA, Japan Aerospace Exploration Agency (JAXA), Indian Space Research Organisation (ISRO), Roscosmos, etc. All these actors in the space industry use the

CCSDS standards to maintain compatibility and consistency with each other. Those standards cover a variety of topics related to communication, data handling, and security. They are openly accessible on [29]. The CCSDS standards relevant for this body of work on data compression are:

- **CCSDS 121:** [30] a 1D lossless compression based on Rice [31] algorithm
- **CCSDS 122:** [3] a 2D with lossless and lossy compression based on DWT and Bit planes
- **CCSDS 123:** [32] a 3D Multi/hyperspectral compression based on predictions

2.3 Other compression benchmarks

Video compression is a thriving topic among computer scientists and the amount of data we use continues to increase daily. Therefore, a lot of attention is put toward compression methods. The standard for most public use scenarios is on the H.264 computing level. Nowadays, the transition to H.265 level has yet to be done, and the H.266 already exists. To increase significantly the compressions, using a more efficient codec is necessary, but it may requires an upgrade of the global hardware. Therefore, testing the different configurations is necessary, leading to multiple benchmarks and comparisons of the methods such as [33] [19] [20], and others. Recently, with the evolution of the field of machine learning, new methods appeared with the help of different frameworks, such as TensorFlow and PyTorch. After training many models, new models such as Deep Video Compression (DVC) [18], Feature-space Video Coding (FVC) [22], Video Compression Transformer (VCT) [23] displayed very convincing results. After these new developments, the scientific community needed a rigorous comparison of the machine learning and the standard codecs (such as in section 2.1.3). Multiple benchmarking projects were developed as a result, such as

CompressAI, an end-to-end compression benchmark platform for research [14], and the Moscow State University (MSU) benchmark [34] implementing their Neural Network (NN) method paper [21]. As these evaluations were not directly related to space, the evaluations were performed on top-notch computing systems for the training and tests. In our case, we will implement the CCSDS 122 [3] space standard with the OBPMark [10] framework as a base. This setup will be used as a reference to compare the performances of the standard with other compression methods using a platform constrained by space requirements.

3 Methodology

It must be noted that for this body of work, further mentions of the term "video" express the idea of a sequence of images.

3.1 Test Facilities and Hardware Resources

All the work performed in this study was realised at Tartu Observatory [9] of the University of Tartu. In particular, the Lunar analog environment facility visible in figure 2a and figure 2b. This "simulated" environment serves multiple purposes such as testing the mechanics and gathering camera and sensor data for test platforms (such as the KuupKulgur rover would ex: appendix figure 16). All the images used for the compression tests presented in section 3.4 were captured with the onboard front camera visible in figure 2b. The camera model is the Waveshare IMX219-160 Camera Module for Jetson Nano, with IMX219 Sensor 8 Megapixels 160° Field of View (FOV) Wide Viewing Angle and supports a maximum resolution of 3280 × 2464 [35].

(a) Lunar environment. (b) Lunar rover prototype [36].

Figure 2. Space Bunker images

To process all those images, we needed a computing system. According to the NASA paper [12], affordable and efficient COTS options are appearing on the global market, notably on the Nvidia side with the Jetson computing boards. As mentioned in [12], recent models offer very good computing power with CPUs, GPUs and hardware acceleration, providing a solid framework for accelerated Artificial intelligence (AI) with the Compute Unified Device Architecture (CUDA) cores, but also the codec accelerators (NVIDIA Encoder (NVENC) and NVIDIA Decoder (NVDEC)) granting the access to GPU and hardware accelerated H.265 and H.264. The platform also offers physical benefits such as a good radiation characteristics [13]. With that in mind, we initiated the project with a Jetson Xavier and upgraded to the Orin AGX as the possibility to use the

Jetson Orin AGX dev kit was demonstrated in an ESA case study presented in October [13]. For this SoC, the hardware architecture is visible in figure 3, where you see the 3 groups of 4 CPUs, the 2 GPUs and the 5 Hardware accelerators:

- **Deep Learning Accelerator (DLA):** accelerator for Deep Learning tasks such as Convolutional Neural Network (CNN)s
- **Programmable Vision Accelerator (PVA):** accelerator for computer vision tasks
- **High Dynamic Range Image Signal Processor (HDR ISP):** accelerator for processing image signals with high dynamic range
- **Video Decoder:** accelerator for decoding (NVENC)
- **Video Encoder:** accelerator for encoding (NVDEC)

It is important to note that when the device runs under 15W, 8 of the 12 CPUs are turned off. At last, the device was provided with a 128 GB Solid-State Drive (SSD) for the data storage.

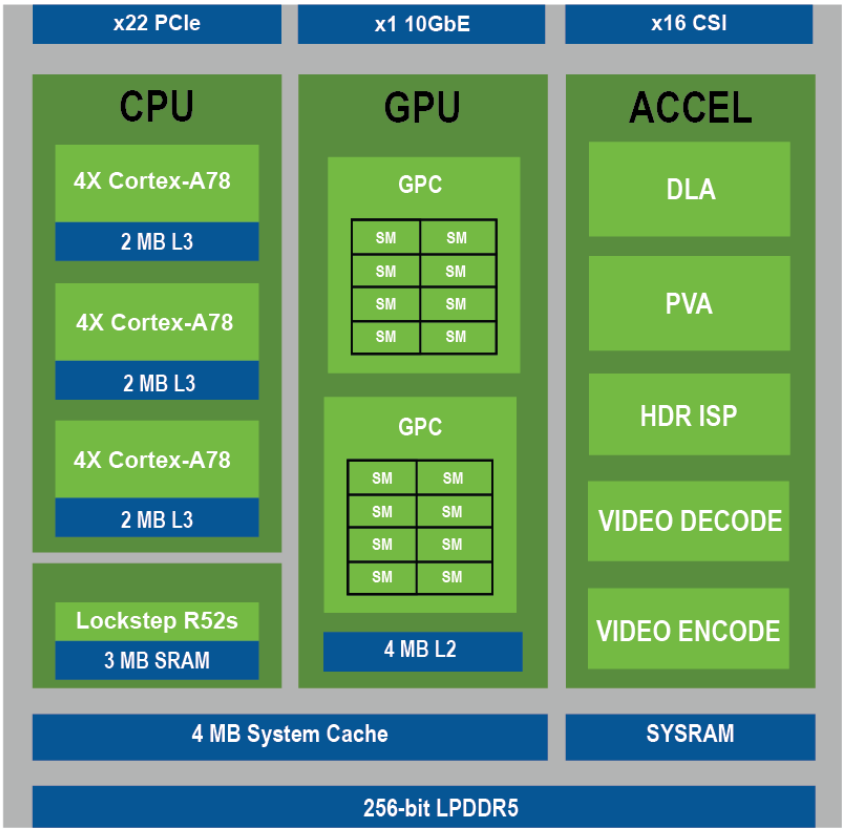


Figure 3. "Orin SoC Block Diagram" [2].

3.2 Software

To start the work, the Jetson AGX Orin Dev kit 32 GB was flashed with Jetpack 5.1.2. All the work was done on Ubuntu 20.04 on an arm.64 architecture. For the software architecture, Robot Operating System (ROS) [37] was installed to develop and connect the different features of the rover. Then, for the video compression work, the accelerated Gstreamer [38], [39] and Fast Forward Moving Picture Experts Group (FFMPEG) [40] [41]libraries were installed to run the standard codecs and accelerators available on the Jetson. For the Machine learning implementations, the PyTorch library with its required dependencies. The Anaconda framework was also installed to limit the risks. the versions of the different components mentioned above are:

- ROS Noetic
- Ubuntu 20.04 Focal

- GStreamer 1.14.1
- FFMPEG 4.2.7
- Python 3.11
- PyTorch 2.1.1

3.3 Tools

Through the development of the project, many tools were used to reach the end product. A list of the most important tools for the development is available in the appendix 6

3.4 Benchmarking methods

3.4.1 CCSDS 122

The first compression method to be tested is the CCSDS 122 standard [3]. This standard is specifically designed for image compression. For a benchmark related to video compression, it is the most important method to assess as it is a widely used standard for space missions.
To summarize the mechanism, the data is processed by using Discrete Wavelet Transform (DWT) [42], which turns the image data into a batch of coefficients. Then, the DWT is applied three times, to separate the data into filters, each defining some information about the picture as in figure 4. Once the image is separated, the coefficients are grouped in a block as in figure 5a. Those blocks are concatenated to build segments, which are fed into the bit plane encoder. This encoder sorts the coefficient into planes according to the number of bits necessary to encode the value of the coefficient. Afterwards, the encoded bit planes are concatenated to the previous planes. This operation is repeated until the number of bits necessary to describe the information is 0 or the available byte space to write information is reached. In the case memory runs out, lossy compression is performed as the planes containing the least information in the segment are truncated.

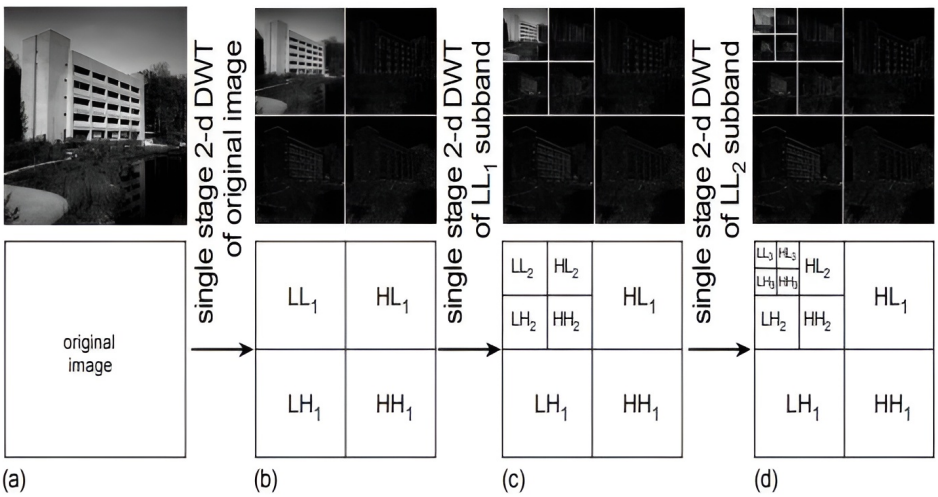
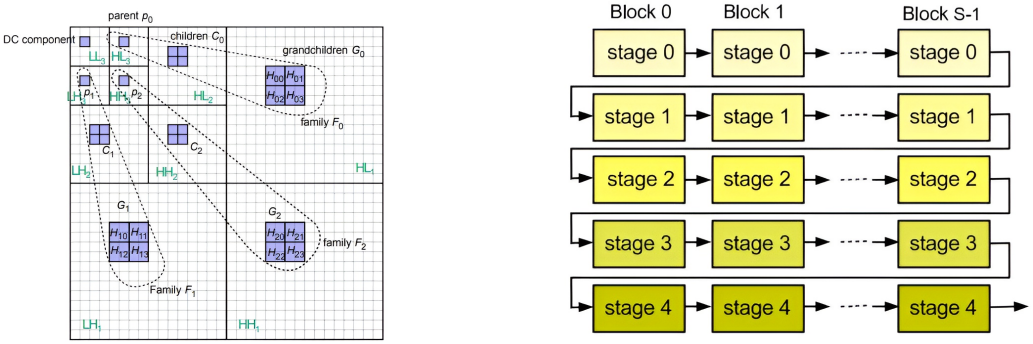


Figure 4. " Three-Level 2-d DWT Decomposition of an Image" [3].



(a) "Schematic of Wavelet-Transformed Image " [3].

(b) "Overview of the Structure of a Coded Segment within one bit plane " [3].

Figure 5. CCSDS Shematics

3.4.2 H.264

The H.264 codec, was chosen among all codecs for multiple reasons:

- High-definition formats are encoded with it
- Good video quality is achieved at low bitrates (broadcasting).
- The standard is supported by most devices

- Hardware accelerators are available on the Jetson AGX Orin [2], and visible in figure 3

To explain the mechanism, the encoding process follows the steps defined in section 3.4. The source video undergoes splitting into 16x16 or 4x4 blocks, used for intraframe and interframe prediction. Then, a Discrete Cosine Transform (DCT) is applied to the residual of the predictions. Afterwards, the coefficients resulting from the DCT are quantized according to the Quantization Parameter (qp). At last, the quantized block of coefficients is encoded into bits with variable length encoding or arithmetic coding

3.4.3 H.265

Once H.264 was chosen, his successor H.265 had to be tested too as it offers better performances at the cost of heavier processing. Then the Jetson Orin AGX[2] also possesses hardware accelerators (visible in figure 3) for it. The differences with H.264 in the encoding mechanism are:

- **Partitioning:** The blocks diversity has been highly increased, and these blocks are further partitioned in coding Tree Unit (CTU)s
- **Predictions:**Predictions are not simply intraframe and interframe, often, images are predicted solely with intraframe prediction, to avoid large extrapolations

3.4.4 Machine learning approach

As mentioned in the introduction section 2.3, machine learning implementations have developed a lot in recent years. Multiple methods have been tailored and trained to present better results than the previous codecs (such as in section 2.1.3). As mentioned in the section 3, the Jetson AGX Orin [2] hardware was built with GPU and hardware acceleration, with CUDA cores which enable AI acceleration. Therefore, the CompressAI [14] [43] framework has been tested in this thesis. The CompressAI project has designed a library for end-to-end machine learning based compression research, complementing this body of work with a fully developed framework offering multiple image and video compression methods built from research articles [44] [45] [46] [47][44]. Also, CompressAI has mentioned plans to expand the framework and implement new models. The existing pre-trained models in the CompressAI library are available in the "model zoo":

- FactorizedPrior [45]
- ScaleHyperprior [45]
- MeanScaleHyperprior [47]
- JointAutoregressiveHierarchicalPriors [47]

- Cheng2020Anchor [46]
- Cheng2020Attention [46]
- ScaleSpaceFlow [44]

3.5 Metrics

To ensure the relevance of the different tests performed in this benchmark, the tests were realized in the 15 W power mode of the Jetson AGX Orin. It is the closest mode to eventual implementations of the KuupKulgur rover for any eventual Lunar mission. The main metrics used are the time for the encoding, the quality of the output image, the compression ratio and the power consumption. These four elements are the most representative of the efficiency of a compression method for space application. It is important to note that this body of work sets a baseline for video encoding. Decoding was not deemed to be a computation cost as the videos have been assumed to flow from space toward Earth.

3.5.1 Computational Time

Ideally, the measured time for the different methods implemented in all case scenarios is the execution time for the encoding. However, for the H.264 and H.265, and CompressAI, there was no tool available to get such accuracy. Therefore, the time measurement will cover the total execution time for each method following the processes mentioned previously.

3.5.2 Quality

There exist multiple methods to assess the quality of an image. In this benchmark, the Peak Signal-to-Noise Ratio (PSNR) has been programmed to assess it. The PSNR is the ratio between the maximum power of a signal, and the noise induced through the processing, computed with:

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{\text{MSE}}{\text{MAX}^2} \right)$$

with:

- **PSNR:** In decibel (dB)
- **MAX:** The maximum possible pixel value of the image (e.g., 255 for an 8-bit image).

- **Mean Squared Error (MSE):** The Mean Squared Error between the original and reconstructed image.

After compressing, the outputs will be decompressed, and this output image will be compared with the original to assess the noise. This process will be executed for all scenarios.

3.5.3 Compression ratio

The compression measurement will remain simple. The output compressed file will be compared to the original file size. The Compression Ratio is given by:

$$\text{Compression Ratio (\%)} = \left(1 - \frac{\text{Size of Uncompressed Data (kB)}}{\text{Size of Compressed Data (kB)}}\right) \times 100$$

3.5.4 Power

The power consumption of the compression methods is difficult to measure properly. To get the most accurate measurement possible, we will use the Jetson-stats [48] interface from Jetson-stats. This tool provides information about all the different resources of the Jetson in real-time. From those, we retrieve the instantaneous power consumption through the complete duration of the process to compute a mean power consumption. To avoid interferences from other sources using the power of the jetson, the benchmark has been entirely executed via Secure Shell (ssh) from a remote computer. From experimentations during the development, connecting a screen to the Jetson AGX Orin[2] changes the total power consumption. At last, as the platform was designed, it appeared that reading and writing the inputs and outputs in an SSD instead of an Secure Digital (SD) storage reduces the instantaneous power consumption by a Watt. Since the limits of the storage capacity of the SoC had been reached, no further tests have been done.

3.6 Implementation

After selecting the method, and the metrics, this section will cover the process to set them up physically.

3.6.1 CCSDS 122

To implement the CCSDS 122 standard on the Jetson, the OBPMark [10] framework was set up. This framework is a benchmarking platform available on GitHub [49], which offers multiple CCSDS standards for compression and a GPU4S [11] benchmark. Using this project work as a reference, the framework was modified to fit the purpose of this work. The main modification for this body of work has been about video compression.

Since the source structure was built to encode single images, it has been modified to convert a sequence of images instead. The program originally generated compressed binary files for each image, therefore it has been changed to gather all these binary files within a folder. Ideally, after being compressed, images should have been decompressed within an image output folder. Nevertheless, this method proved to be challenging as the official codecs for CCSDS 122 available on the CCSDS page [29] could not provide results from the encoded files. Also, the WhiteDwarf [50] decoder used by OBPMark, was not accessible in time for the tests. For the CCSDS tests, the bit size of each pixel (bps) parameter was used to change the compression from lossless to lossy. The different possibilities of tuning have been 8 and 16, for the different bit sizes of 1 byte and 2 bytes of each pixel.

3.6.2 H.264 and H.265

The FFMPEG [40] library provides an optimised version of the H.264 and H.265 encoders on CPU. This library provides multiple features to tune the available codecs with further descriptions in the FFMPEG documentation[40]. In this library, the parameters work on both H.264 and H.265. Therefore, to assess the differences, all the parameters are kept the same between the two codecs. For the tests, Constant Rate Factor (crf) has been the most important tuning parameter. It has offered the possibility to change the quantization parameter, to variate the compression, from near lossless to lossy with parameters scaling from 0 to 51. Concerning the GPU and hardware acceleration, the GStreamer[38] library was initialized similarly as the CPU version with tuning on the qp_range, varying from 0 to 51. These two library commands, available in the appendix tables 6, and 6 produce a compressed ".264" or "265" file, which can be decompressed into an image folder by the decompressing code 6 provided in the appendix.

3.6.3 Machine Learning

As mentioned in section 3.1, the Jetson Orin provides an efficient AI framework with the GPU and hardware acceleration. Therefore to evaluate what the system is capable of, the CompressAI framework was implemented. It provides a well-documented platform implementing multiple pre-trained models, which can be used off the shelf, with Python 3.6+, PyTorch 1.7+ and the codes. For this paper, we made tests with all the models available trained on the Vimeo images [51]. Therefore, the results could be improved through specific training on topic relevant datasets. Only image compression models worked in the designed architecture. Therefore sequential image compression has been implemented as with the CCSDS 122 in section 3.6.1.

3.6.4 Testbench Pipeline

Many video compression benchmarks of the classical standards discussed in section 2.1.3 provide fixed results and method comparisons. Recent projects with the NN method provided the feature to train your models and analyze the results. It is a very interesting tool that shall be expanded to video compression as a whole. In that fashion, this project aims to set up a framework with specific inputs and outputs to help concatenate future metrics, compression methods or parameters. An overview of the developed pipeline can be seen in figure 6.

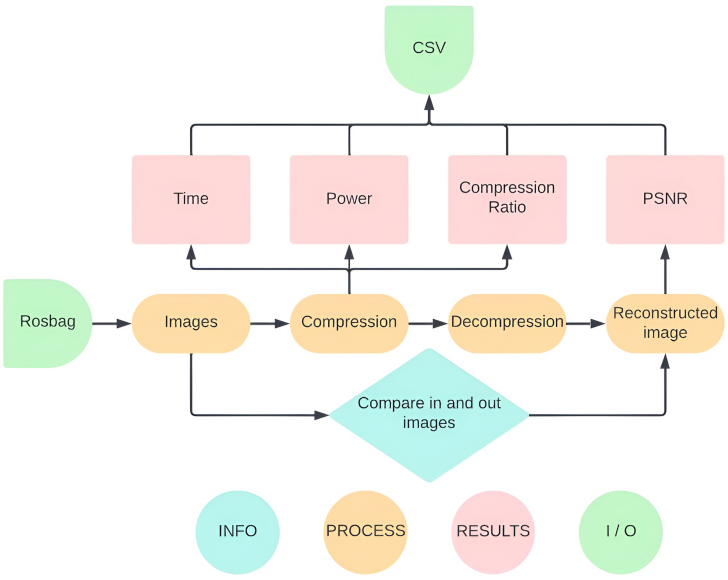


Figure 6. "Overview of the benchmark process".

As in figure 6, the different methods have been implemented such that they output a compressed file or a folder to test the compression ratio. Then, after decompression, they output a folder of images which is used to measure the PSNR quality. Through the compression, the execution time is measured from the duration of the subprocess calling each method. At last, power is measured using Jetson TOP (JTOP) [48] with multi-threading to enable the simultaneous execution of both the compression method and JTOP.

4 Results and analysis

4.1 H.264 and H.265 tests

The H.264, and H.265 codecs have been tested on 900 image videos (30 seconds at 30 frames per second). The information has been plotted using the evolution of the quantization parameter from 5 to 50, using qp values from 0 to 9 to represent them. As H.264 and H.265 are originally lossy methods, the higher the qp value, the lossier the compression. Nevertheless, at qp0, the compression can be considered near-lossless [40] [39] The tests performed On H.264, and H.265 , have been conducted with CPU and NVENC accelerator.

4.1.1 H.264

In the graph representing the evolution of the compression ratio related to the quantization parameter in figure 7, the compression ratio decreases exponentially until qp6. Afterwards, there are minor improvements. Also, there is a disparity between the different videos which disappears as the compression becomes lossier.

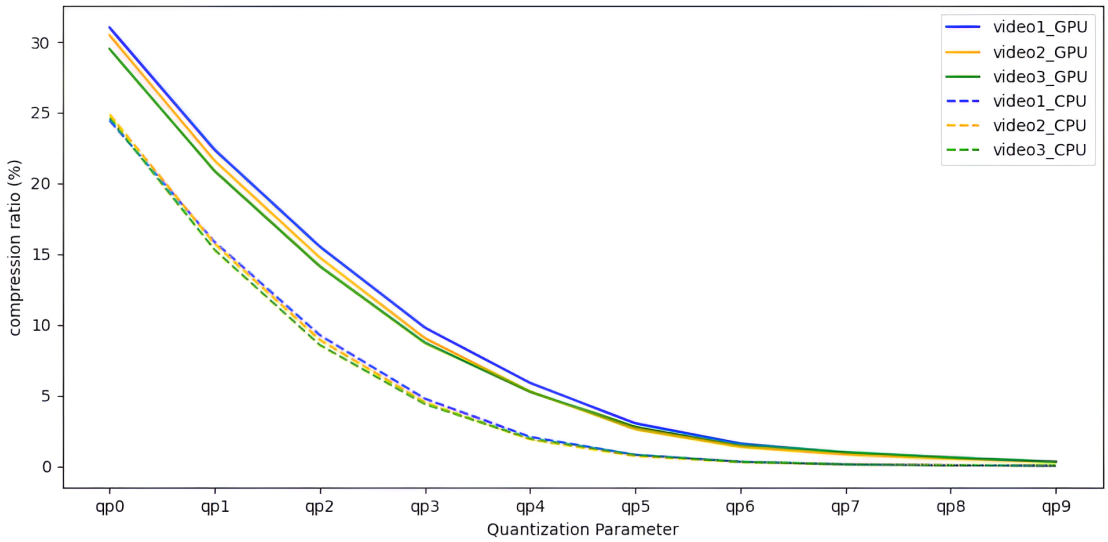


Figure 7. Graph of the evolution of the compression ratio compared to the change in the quantization parameter.

The figure 8 represents the evolution of the execution time against quantization parameters. It can be observed that the NVENC accelerated computing time remains around 58 seconds despite the change in quantization. On another hand, the CPU computation execution time almost decreases by 4 times along the scale.

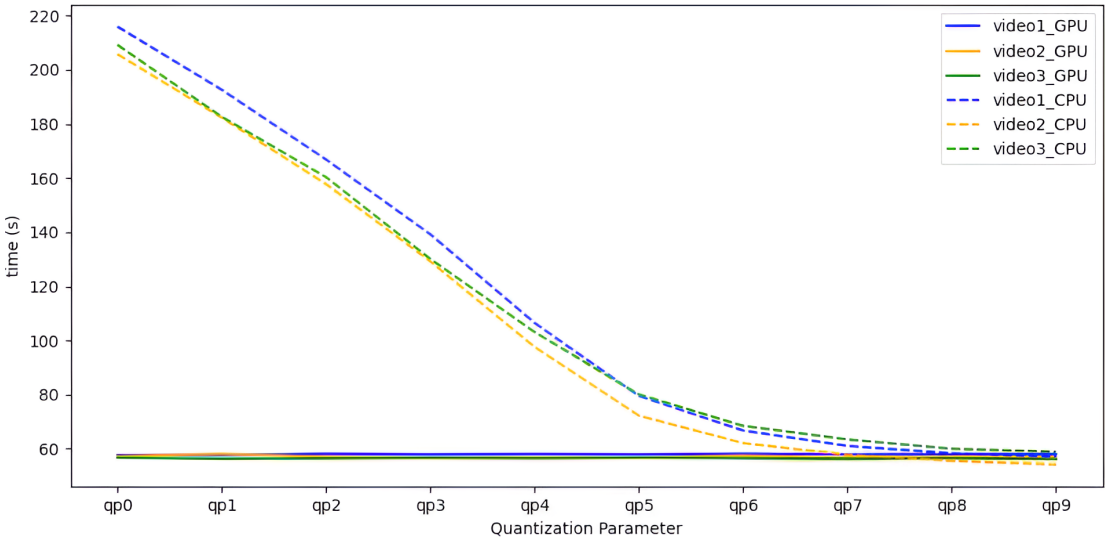


Figure 8. Graph of the evolution of the execution time compared to the change in the quantization parameter.

Analysis into the PSNR measurements of the benchmark on the PSNR measurements as seen in figure 9, the image quality of the CPU decreases slowly on the first 4 steps of quantization with less than 0.1 dB/step. Afterwards, it decreases by more than 0.6 dB/step. This change displays a threshold where the quality starts dropping significantly after reaching quantization parameters superior to 20. Then, the NVENC accelerator displays surprising results. Acceleration, should not be the source of such a difference as it computes in parallel. As the PSNR computation is the same for all the methods, the offset must be from the GStreamer implementation. From the table 1, results show that the image quality is very unstable on GStreamer accelerated, compared to FFMPEG CPU.

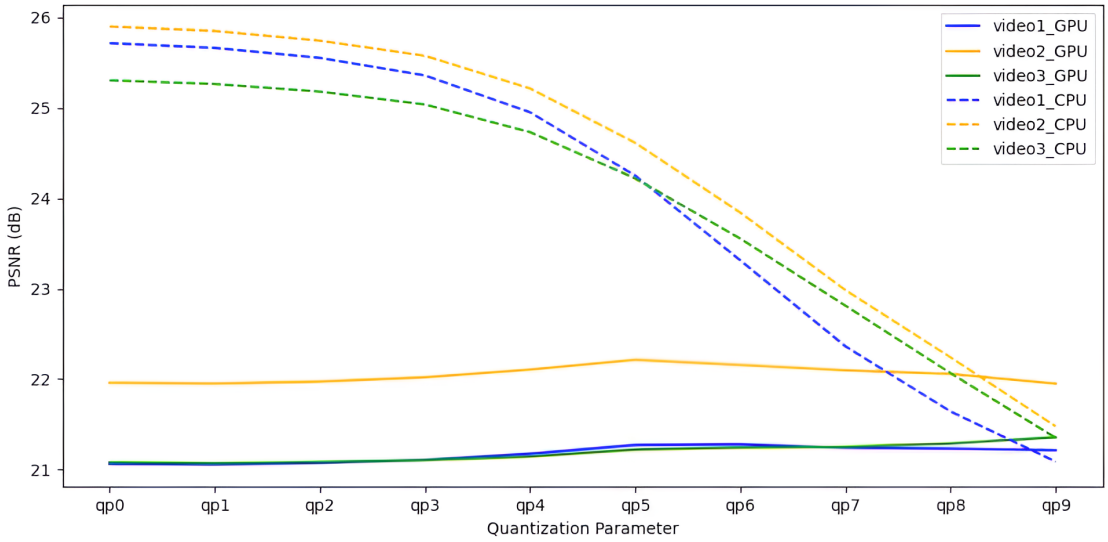


Figure 9. Graph of the evolution of the PSNR image quality compared to the change in the quantization parameter.

Table 1. H.264 PSNR difference between CPU and NVENC (from annex tables 5 and 6)

video 1	**Mean**	**Standard deviation**	**Maximum**	**Minimum**
NVENC_qp0	21.06	2.77	27.80	12.82
CPU_qp0	25.72	0.62	27.6	24.4
NVENC_qp9	21.21	2.32	25.66	13.58
CPU_qp9	21.09	1.80	24.81	14.18

Finally, the power measurements on the H.264 methods visible in the figure 10 show that the CPU compression consumes more power than NVENC accelerated compression. Using the information from the execution time figure 8, it is clear that the CPU uses more energy (appendix figure 17) overall as its consumption and execution time are either equal or higher than NVENC implementation.

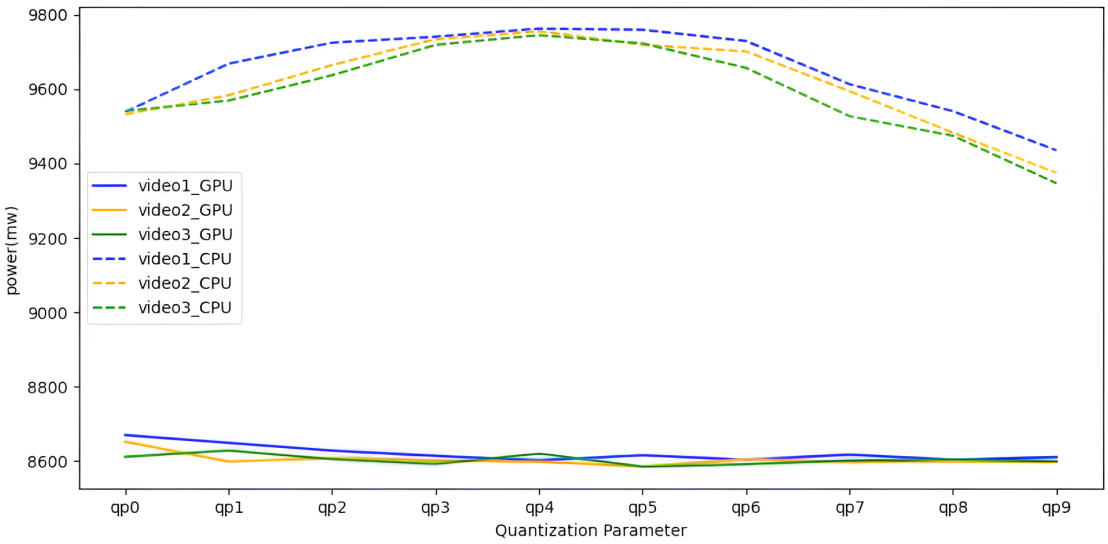


Figure 10. H.264 Graph of the evolution of the mean Power consumption compared to the change in the quantization parameter.

4.2 H.265 test

In the graph representing the evolution of the compression ratio related to the quantization parameter from figure 11, the same trends as with H.264 in figure 11 appear. The main difference lies in the compression ratio being about 5% smaller than the later on qp0. This difference decreases gradually with the augmentation of qp and disappears after qp6. Therefore compressing with H.265 on a very lossy setting does not hold more interest than compressing with H.264 on the file size.

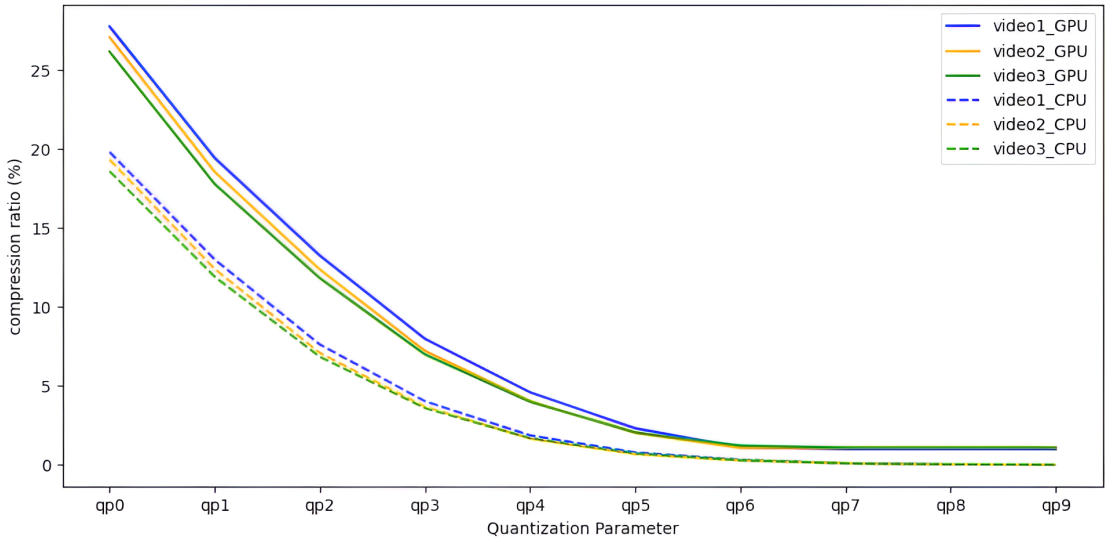


Figure 11. Graph of the evolution of the compression ratio compared to the change in the quantization parameter.

Then, the figure 12, representing the evolution of the execution time according to the quantization shows similar results as with the H.264 figure 8. Between the two codecs, the NVENC accelerated processing time remains the same (around 58 seconds). The main difference comes from the CPU execution time, exploding with more than 10 time increase on qp0, and about 7 times higher at qp9. This change shows a limit of the CPU capabilities for H.265 as it is not a viable solution time-wise.

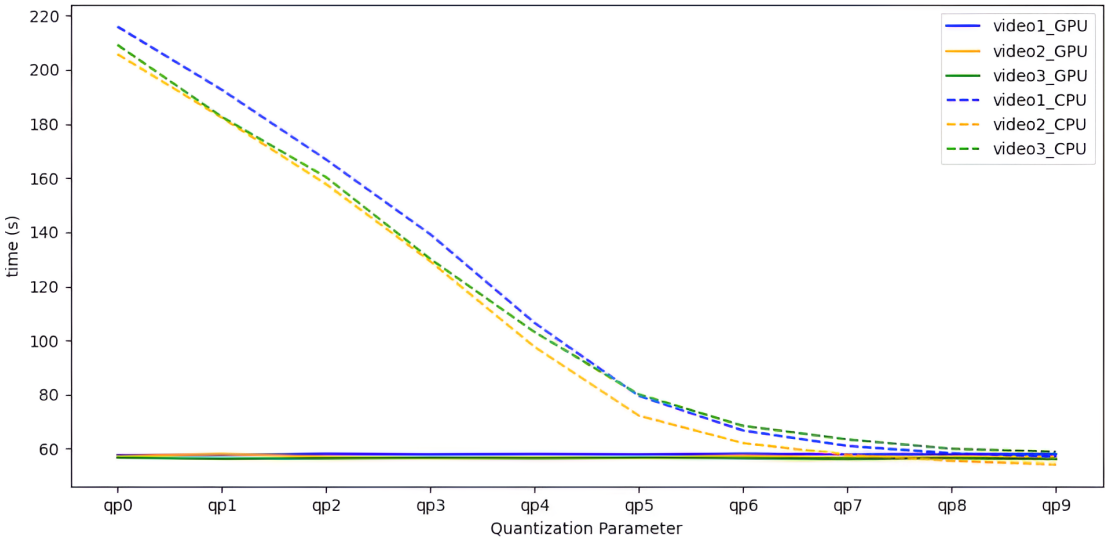


Figure 12. Graph of the evolution of the execution time compared to the change in the quantization parameter.

Then, looking into the PSNR measurements of the benchmark, the trend of the image quality is the same as with the H.264 trend from the figure 9. Then, NVENC acceleration similarly shows unstable results in the table 2 as in table 1. Comparing these two tables, interesting information appears as the qp0 of the two tables are the same on two digits, but differences appear as quantization increases since qp0 generate a near lossless compression.

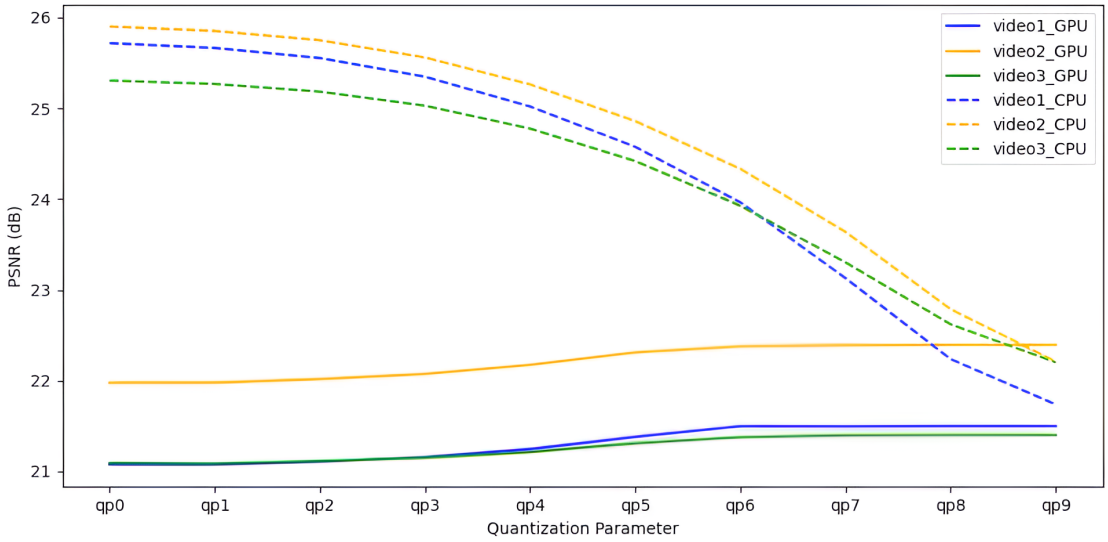


Figure 13. Graph of the evolution of the PSNR image quality compared to the change in the quantization parameter.

Table 2. H.264 PSNR difference between CPU and GPU (from annex table 7 and 8)

video 1	**Mean**	**Standard deviation**	**Maximum**	**Minimum**
NVENC_qp0	21.06	2.77	27.80	12.82
CPU_qp0	25.71	0.62	27.6	24.4
NVENC_qp9	21.50	2.73	26.78	13.39
CPU_qp9	21.74	1.96	25.33	13.78

At last, the power measurements on the H.265 methods visible in the figure 14 shows that the NVENC accelerated compression consumes more power than CPU. The power consumption has remained stable between 8.7, and 8.6 W, whereas the CPU decreases by nearly 0.4 W over the evolution of qp.

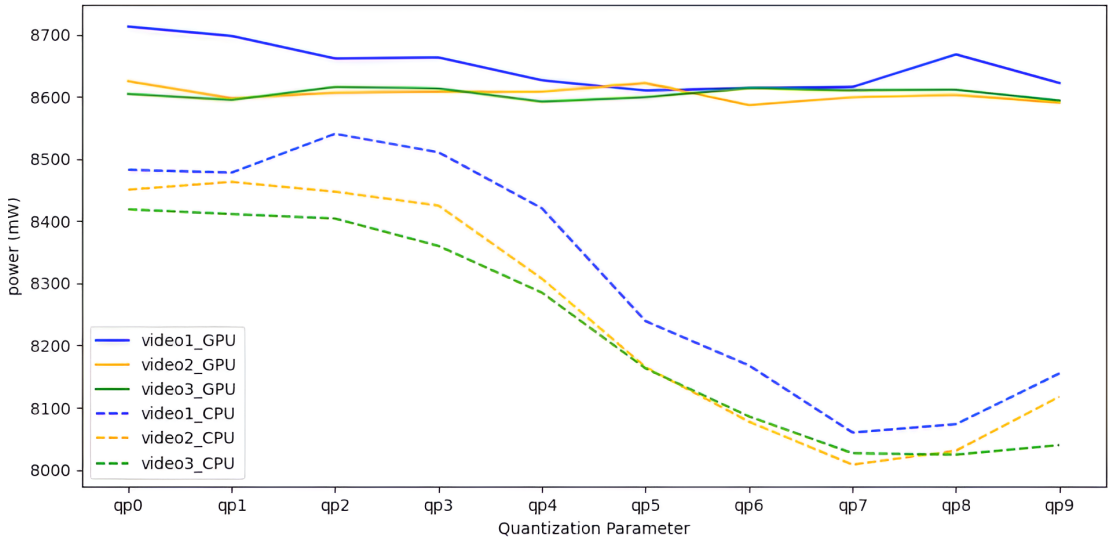


Figure 14. H.265 Graph of the evolution of the mean Power consumption compared to the change in the quantization parameter.

4.3 CCSDS test

The CCSDS 122 codec has not been tested with 900 image videos, but it has repeatedly been tested on 300 images (for a 10 seconds video at 30 images per second). This choice has been induced by the variability of the results. Therefore, the presented graphs have come from the average performance on 4 tests for the 300 images videos. Through the table, results show that the compression ratios and the execution time have been improved by 8 % and 10 to 27 seconds, at the cost of 700 mW by the CUDA implemented GPU. The induced changes have not been as striking as in the figures 12 and 8, but the capacity of the CUDA implemented GPU to improve the different compression methods is verified.

Table 3. CCSDS compression ratio table

parameters	compression ration (%)	execution time (s)	Power (mW)
GPU_lossless	68.05	290.94	9010.06
CPU_lossless	60.26	301.71	8385.22
GPU_lossy	33.27	149.68	9121.59
CPU_lossy	25.6	176.79	8461.52

4.4 CompressAI models test

From the models available in the CompressAI[14] zoo, the Video compressing model [44] was not available, and within the image compression section, 2 of the models (cheng2020_anchor [46] and mbt2018 [47]) were not working. After running all remaining models on a test set of 220 images (for a video of 7.3 seconds), the following results were gathered in the table below:

Table 4. Benchmark of machine learning models on video compression

Model	Compre (%)	Time (s)	Energy (Wh)	Quality (dB)
bmshj2018_factorized	1.13	2150.12	5.49	31.79
bmshj2018_hyperprior	0.88	2265.45	5.78	31.88
mbt2018_mean	0.77	2359.71	6.03	31.92
cheng2020_attn	0.58	11472.72	29.38	31.98

From these results, we can see that the models hold very good PSNR Quality and compression ratios compared to the standard methods. On another hand, the execution time results show no possibility for video compression used. The average power measured displays similar results to the H.264 on CPU from the figure 10. After the methods have been analyzed on JTOP, the information in figure 15 showed the process to be running entirely on CPU. As seen in the H.264, and H.265 tests, the GPU and hardware accelerated implementation can be the source of significant improvements in computing time.

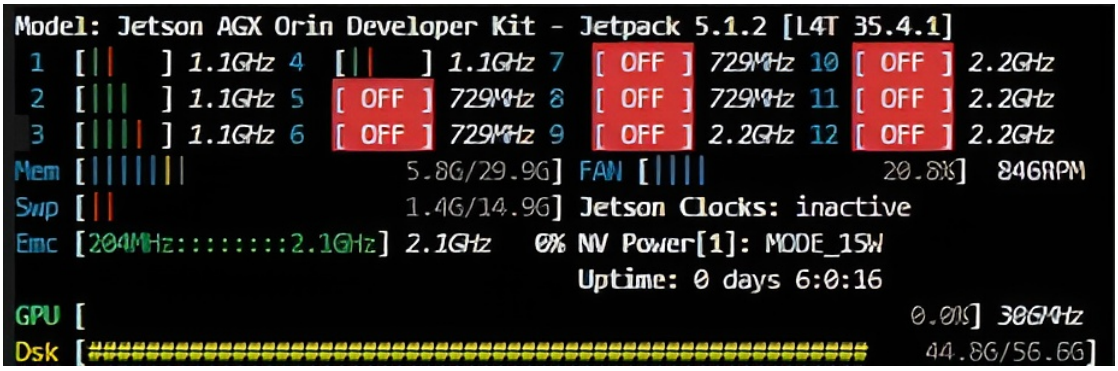


Figure 15. JTOP panel showing no GPU use while running CompressAI models

5 Discussion

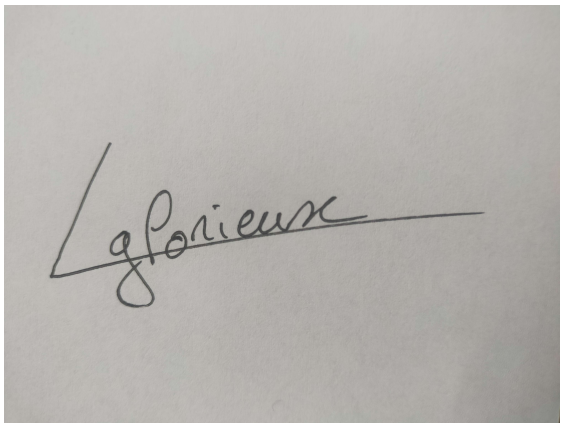
As mentioned in the test section 4, the machine learned approach has held amazing results for quality with low compression ratios. Even though time has shown to be a very large downside to the different approaches, the results have displayed the applicability of the models in the domain of onboard image processing for a CPU base. On another hand, the recorded accelerations on the H.264 and H.265 codecs have offered a very promising view of the machine learning methods on GPU. Through this Body of work, the CCSDS 122 showed very limited results, visible in table 3. Whereas, H.264 and H.265 showed significant evolutions upon acceleration. This situation shows that the CCSDS processes cannot be parallelized much, and the efficiency is limited in its evolution at the moment. Nevertheless, it is still used by many space systems, and will still need to be decoded by new machines. The results shown by H.264 in the test section 4 showcased that on a NVENC accelerated configuration, CCSDS 122 standard was largely bested on both compression ratios and execution time. Nevertheless, for lossless compressions on CPU, tests showed the two methods to hold very close results to one another. Therefore, without a GPU or hardware accelerator in the on-board computing transitioning to H.264 does not seem to provide any improvement. From the Literature Review section 2, H.265 have been selected expecting to reach the limits of the computing system. The limits of the codec were shown on the CPU implementation, due to the high execution time. Nevertheless, the tests have shown the expectations to be wrong since the compressions, execution time, and power consumptions offered by the method were superior to the CCSDS 122 on GPU. A grey area has remained on the image quality difference between the two.

6 Conclusion

In this body of work, multiple compression methods have been studied, evaluated on four metrics: time, power, quality, and compression, and compared on their results. A pipeline offering an accessible compression testing ground has been built on Python regrouping different implementations of the studied codecs. Using the metrics designed in the body of work to assess the different implementations showed different results. The newer approach of machine learning applied in this benchmark showed promising results in quality countered by the large execution time. The H.264 codec showed improvements in execution time and compression on CPU, which were significantly higher on NVENC acceleration compared to the CCSDS 122 standard. The most outstanding method tested in this thesis is the H.265, which required too many resources to provide improvements to the CCSDS standard on CPU. On GPU, the results were better in all aspects compared to the other methods tested in this benchmark. From the results obtained in this thesis, the machine learning implementation has a lot of potential to unleash with GPU or NVENC SoC destined for space applications. Also, the limits of the architecture were not found with the H.265 standard method. Making further tests on the next generation H.266 may offer interesting applications or more information on the limits of the classic codecs. For future improvements of the benchmark, adding the White Dwarf [50] CCSDS 122 decompressor would enable more accurate comparisons of the quality of the different standards. The quality assessment could also be improved with the implementation of Structural Similarity (SSIM), and pair it with the PSNR for more informative quality results. Lastly, investigate deeper into the FFMPEG or GStreamer frameworks evolution for the implementation of other accelerated standards, and consolidate the current structure.

Acknowledgments

First, I would like to thank my supervisors Saimoon Quazi Islam, and Ric Dengel for supporting me through the project and giving me advice and motivation. They allowed me to learn more about the fundamentals of research, and video compression and allowed me to work on the Kuupkulgur project alongside them in the field of space-embedded systems which interests me a lot. Their support and guidance introduced me to multiple new skills and good working habits I acquired through the project. I want to thank Tarvi Tepandi for the guidance, the support on the software and the efforts to set the office in a working mood. Then, I want to express my thanks to the members of the Office of Autonomous Work of Tartu Observatory, for their guidance, and for bringing a good mood along the way, which was a key to the work motivation. I want to thank the Tartu Observatory and for inviting me to work on their research projects, sponsoring me, and providing me with a work environment of quality and good equipment. I want to thank Sten Salumets for translating the abstract to Estonian language, and for introducing me to the observatory where I had the chance to build this thesis. I also want to thank Indrek Sunter for reviewing the Thesis, but also teaching me about space systems. Of course, I want to thank the University of Tartu for the chance to study in the robotics, computer engineering and space fields through the masters I want to thank ECAM Lyon University for the engineering education program, which enabled me to develop the necessary base skills and English levels for this work. At last, I want to thank my friends in Tartu and Lyon, and my family for motivating me, cheering me up and encouraging me to continue pushing forward.



References

[1] M. Larmier, "Lossy vs lossless image compression: What's the difference?." https://imagify.io/blog/lossless-vs-lossy-image-compression/.

[2] NVIDIA, "Jetson agx orin developer kit datasheet." https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwiYi7-UgeyCAxWKFxAIHXWICEsQFnoECA8QAQ&url=https%3A%2F%2Fstatic6.arrow.com%2Faropdfconversion%2Farrowresources%2F3e0f93dc18d2fbe2cceb9b8218d496713cc9b980%2Fjetson-agx-orin-developer-kit-datasheet-nvidia-a4-2203098-r1-web.pdf&usg=AOvVaw2ngNuKO7VdDTUzUu5RdvXA&opi=89978449.

[3] C. C. for Space Data Systems, "Image data compression," tech. rep., CCSDS, 2017.

[4] M.-L. Plats, "Scientists and students to develop the first estonian lunar rover." https://ut.ee/en/content/scientists-and-students-develop-first-estonian-lunar-rover.

[5] N. O. B. Dunbar, "Artemis." https://www.nasa.gov/specials/artemis/, 2023.

[6] T. Mateo Sanguino, "50 years of rovers for planetary exploration: A review for future directions," *Robotics and Autonomous Systems*, vol. 94, 05 2017.

[7] Astrobotics, "Cube rover overview." https://www.astrobotic.com/lunar-delivery/rovers/cuberover/, 2023.

[8] N. J. P. Laboratory, "Cadre." https://www.jpl.nasa.gov/missions/cadre, 2023.

[9] T. Observatory, "Tartu observatory web page." https://kosmos.ut.ee/en.

[10] L. Kosmidis, I. Rodriguez-Ferrandez, A. Jover-Alvarez, K. Förste, and D. Steenari, "Obpmark (on-board processing benchmarks) – open source computational performance benchmarks for space applications," in *OBDP2021 - 2nd European Workshop on On-Board Data Processing (OBDP2021) , 14-17 June 2021 (Session 5)*, zenodo, 2021.

[11] L. Kosmidis, I. Rodriguez, A. Jover-Alvarez, S. Alcaide, J. Lachaize, O. Notebaert, A. Certain, and D. Steenari, "Gpu4s: Major project outcomes, lessons learnt and way forward," in *2021 Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pp. 1314–1319, 2021.

[12] W. Powell, M. Campola, T. Sheets, A. Davidson, and S. Welsh, "Commercial off-the-shelf gpu qualification for space applications." https://ntrs.nasa.gov/api/citations/20180006906/downloads/20180006906.pdf.

[13] European Space Agency, "Case-study for integration of cots soc devices - 06-10-23_v3," 2023.

[14] J. Bégaint, F. Racapé, S. Feltman, and A. Pushparaja, "Compressai: a pytorch library and evaluation platform for end-to-end compression research," 2020.

[15] Vignesh.S.Krishnan, "A detailed overview of popular video compression techniques." https://imagekit.io/blog/video-compression-techniques/, 2023.

[16] Lumenci, "How video codecs works." https://www.lumenci.com/post/how-video-codecs-works, 2023.

[17] D. Van Anda, "Understanding compression methods: From arithmetic coding to zstandard." https://itnext.io/understanding-compression-methods-df970a9133c9, 2023.

[18] G. Lu, W. Ouyang, D. Xu, X. Zhang, C. Cai, and Z. Gao, "Dvc: An end-to-end deep video compression framework," 2019.

[19] I. Mansri, N. Doghmane, N. Kouadria, S. Harize, and A. Bekhouch, "Comparative evaluation of vvc, hevc, h.264, av1, and vp9 encoders for low-delay video applications," in *2020 Fourth International Conference on Multimedia Computing, Networking and Applications (MCNA)*, pp. 38–43, 2020.

[20] P. Akyazi and T. Ebrahimi, "Comparison of compression efficiency between hevc/h.265, vp9 and av1 based on subjective quality assessments," in *2018 Tenth International Conference on Quality of Multimedia Experience (QoMEX)*, pp. 1–6, 2018.

[21] A. Antsiferova, S. Lavrushkin, M. Smirnov, A. Gushchin, D. Vatolin, and D. Kulikov, "Video compression dataset and benchmark of learning-based video-quality metrics," in *Advances in Neural Information Processing Systems* (S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, eds.), vol. 35, pp. 13814–13825, Curran Associates, Inc., 2022.

[22] Z. Hu, G. Lu, and D. Xu, "Fvc: A new framework towards deep video compression in feature space," 2021.

[23] F. Mentzer, G. Toderici, D. Minnen, S.-J. Hwang, S. Caelles, M. Lucic, and E. Agustsson, "Vct: A video compression transformer," 2022.

[24] T. Wiegand, G. 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, no. 7, pp. 560–576, 2003.

[25] 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, no. 12, pp. 1649–1668, 2012.

[26] B. Bross, Y.-K. Wang, Y. Ye, S. Liu, J. Chen, G. J. Sullivan, and J.-R. Ohm, "Overview of the versatile video coding (vvc) standard and its applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 10, pp. 3736–3764, 2021.

[27] D. Mukherjee, J. Han, J. Bankoski, R. Bultje, A. Grange, J. Koleszar, P. Wilkins, and Y. Xu, "A technical overview of vp9 – the latest open-source video codec," in *SMPTE 2013 Annual Technical Conference and Exhibition*, pp. 1–17, 2013.

[28] J. Han, B. Li, D. Mukherjee, C.-H. Chiang, A. Grange, C. Chen, H. Su, S. Parker, S. Deng, U. Joshi, Y. Chen, Y. Wang, P. Wilkins, Y. Xu, and J. Bankoski, "A technical overview of av1," *Proceedings of the IEEE*, vol. 109, no. 9, pp. 1435–1462, 2021.

[29] S. Victor, "Consultative committee for space data systems." `https://public.ccsds.org/Publications/BlueBooks.aspx`.

[30] C. C. for Space Data Systems, "Lossless data compression," tech. rep., CCSDS, 2020.

[31] R. Rice, P.-S. Yeh, and W. Miller, *Algorithms for high-speed universal noiseless coding*.

[32] C. C. for Space Data Systems, "Low-complexity lossless and near-lossless multispectral and hyperspectral image compression," tech. rep., CCSDS, 2019.

[33] M. Alvarez, E. Salami, A. Ramirez, and M. Valero, "Hd-videobench. a benchmark for evaluating high definition digital video applications," in *2007 IEEE 10th International Symposium on Workload Characterization*, pp. 120–125, 2007.

[34] A. Antsiferova, S. Lavrushkin, M. Smirnov, A. Gushchin, D. S. Vatolin, and D. Kulikov, "MSU_VQM_Compression_Benchmark," in *Advances in Neural Information Processing Systems*, 2022. NeurIPS'2022.

[35] Waveshare, "Imx219-160 camera." `https://www.waveshare.com/wiki/IMX219-160_Camera`.

[36] V. Allik, "Space bunker." https://kosmos.ut.ee/en/space-bunker.

[37] ROS, "Robot operating system (ros)." `https://www.ros.org/`.

[38] NVIDIA Corporation, "Accelerated gstreamer user guide." https://developer.download.nvidia.com/embedded/L4T/r32_Release_v1.0/Docs/Accelerated_GStreamer_User_Guide.pdf?t=eyJscyI6ImdzZW8iLCJsc2QiOiJodHRwczovL3d3dy5nb29nbGUuY29tLyJ9, 2019.

[39] NVIDIA Corporation, "Accelerated gstreamer user guide." https://docs.nvidia.com/jetson/archives/r35.2.1/DeveloperGuide/text/SD/Multimedia/AcceleratedGstreamer.html, 2022.

[40] FFmpeg Project, "FFmpeg." https://trac.ffmpeg.org/wiki/Encode/H.264, 2000.

[41] NVIDIA Corporation, "Accelerated decode with ffmpeg." https://docs.nvidia.com/jetson/archives/r35.1/DeveloperGuide/text/SD/Multimedia/AcceleratedDecodeWithFfmpg.html, 2022.

[42] U. of Saint Andrews, "Discrete wavelet transform tutorial." https://www.st-andrews.ac.uk/~wjh/dataview/tutorials/dwt.html. Very informative non mathematical introductory to DWT, author may be William J. Heitler, as wjh saint andrews in url refers to him on google.

[43] I. InterDigital, "Compressai documentation and models." https://interdigitalinc.github.io/CompressAI/index.html, 2021.

[44] E. Agustsson, D. Minnen, N. Johnston, J. Balle, S. J. Hwang, and G. Toderici, "Scale-space flow for end-to-end optimized video compression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[45] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," 2018.

[46] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, "Learned image compression with discretized gaussian mixture likelihoods and attention modules," 2020.

[47] D. Minnen, J. Ballé, and G. Toderici, "Joint autoregressive and hierarchical priors for learned image compression," 2018.

[48] NVIDIA, "Jetson Stats." https://developer.nvidia.com/embedded/community/jetson-projects/jetson_stats, 2023. jtop.

[49] D. Steenari and D. L. Kosmidis, "Obpmark git repository." https://github.com/OBPMark.

[50] E. S. Agency, "Whitedwarf." https://essr.esa.int/project/whitedwarf, 2017.

[51] T. Xue, B. Chen, J. Wu, D. Wei, and W. T. Freeman, "Video enhancement with task-oriented flow," *International Journal of Computer Vision (IJCV)*, vol. 127, no. 8, pp. 1106–1125, 2019.

[52] OpenAI, "Chatgpt." https://www.openai.com, 2023. Accessed: December 22, 2023.

Lisad

I. Additional data

A/ Figures

Figure 16. "Kuupkulgur FOV".

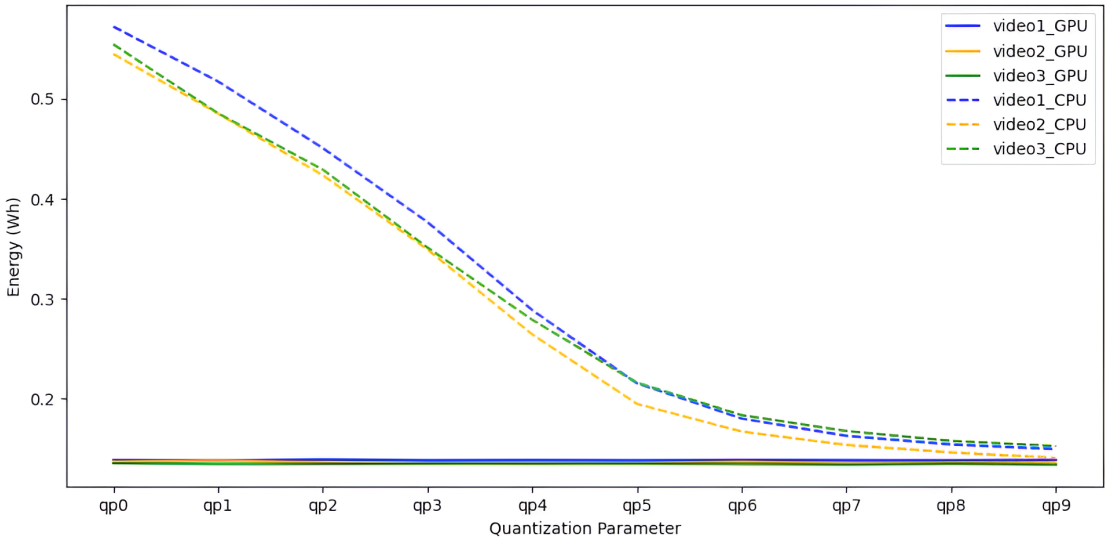


Figure 17. 265 Graph of the evolution of the Energy consumption compared to the change in the quantization parameter.

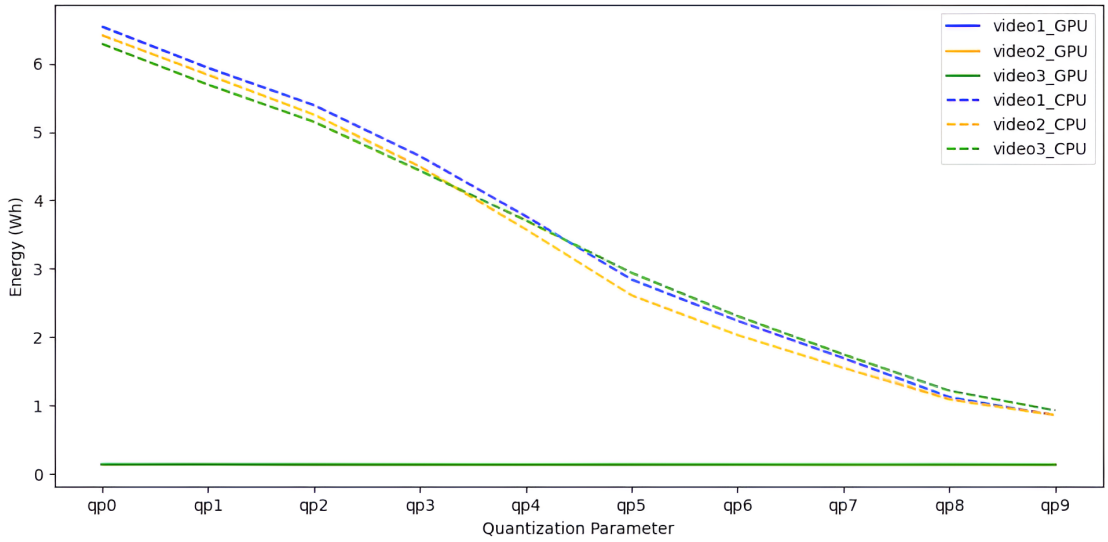


Figure 18. H.264 Graph of the evolution of the Energy consumption compared to the change in the quantization parameter.

B/ Code

FFMPEG CPU command detail:

```
cmd = [
    "gst-launch-1.0",
    "multifilesrc", "location=" + input_folder + bag_name + "/\%d.png"
        , "index=1",
    "caps=image/png,framerate=(fraction)30/1",
    "!", "pngdec",
    "!", "videoconvert",
    "!", "video/x-raw, format=(string)I420",
    "!", "nvvidconv",
    "!", "video/x-raw(memory:NVMM), format=(string)I420",
    "!", libenc(chose encoder), "qp-range=" + qp , "preset-level=" +
       preset_level,
    "!", "video/x-h" + method(264/265) + ", stream-format=(string)byte
       -stream, alignment=(string)au",
    "!", "filesink", "location=" + output_file_name, "-e"
]
```

GStreamer accelerated GPU

```
cmd = [
    "gst-launch-1.0",
    "multifilesrc", "location=" + input_folder + bag_name + "/\%d.png"
        , "index=1",
    "caps=image/png,framerate=(fraction)30/1",
    "!", "pngdec",
    "!", "videoconvert",
    "!", "video/x-raw, format=(string)I420",
    "!", "nvvidconv",
    "!", "video/x-raw(memory:NVMM), format=(string)I420",
    "!", libenc(chose encoder), "qp-range=" + qp , "preset-level=" +
       preset_level,
    "!", "video/x-h" + method(264/265) + ", stream-format=(string)byte
       -stream, alignment=(string)au",
    "!", "filesink", "location=" + output_file_name, "-e"
]
```

H.264 and H.265 decoding

```
gstreamer_command = [
    "gst-launch-1.0",
    "filesrc", "location=" + output_file_name,
    "!", "h" + method(chose 264/265) + "parse",
```

```
    "!", "nvv4l2decoder",
    "!", "nvvidconv",
    "!", "videoconvert",
    "!", "pngenc",
    "!", "multifilesink", "location=" + output_file_name + "_img/%d.
       png"
]
```

C/ Tables

Table 5. H.264 PSNR statistics on the GPU performances

Quantization	Mean	Standard deviation	Maximum	Minimum
qp0	21.06	2.77	27.80	12.82
qp1	21.05	2.76	27.85	12.82
qp2	21.07	2.78	28.00	12.83
qp3	21.11	2.81	28.12	12.85
qp4	21.17	2.86	28.06	12.88
qp5	21.27	2.89	27.36	12.93
qp6	21.28	2.80	26.51	13.03
qp7	21.24	2.61	26.06	13.21
qp8	21.23	2.49	26.06	13.53
qp9	21.21	2.32	25.66	13.58

Table 6. H.264 PSNR statistics on the CPU performances

Quantization	**Mean**	**Standard deviation**	**Maximum**	**Minimum**
qp0	25.72	0.62	27.64	24.44
qp1	25.66	0.61	24.41	27.56
qp2	25.55	0.60	27.40	24.32
qp3	25.36	0.61	27.27	24.10
qp4	24.95	0.65	23.44	27.02
qp5	24.25	0.79	26.73	21.55
qp6	23.31	1.09	26.28	19.04
qp7	22.36	1.44	25.89	16.58
qp8	21.64	1.69	25.53	14.75
qp9	21.09	1.78	24.81	14.18

Table 7. H.265 PSNR statistics on the GPU performances

Quantization	**Mean**	**Standard deviation**	**Maximum**	**Minimum**
qp0	21.08	2.77	27.82	12.82
qp1	21.07	2.77	27.88	12.82
qp2	21.11	2.79	28.00	12.83
qp3 21.16	2.82	28.13	12.85	
qp4	21.25	2.85	28.17	12.87
qp5	21.38	2.89	27.75	12.93
qp6	21.50	2.86	27.15	13.07
qp7	21.50	2.76	26.89	13.29
qp8	21.50	2.73	26.81	13.40
qp9	21.50	2.73	26.78	13.39

Table 8. H.265 PSNR statistics on the CPU performances

Quantization	Mean	Standard deviation	Maximum	Minimum
qp0	25.72	0.62	27.63	24.44
qp1	25.67	0.61	27.54	24.41
qp2	25.55	0.60	27.40	24.34
qp3	25.35	0.59	27.18	24.13
qp4	25.02	0.62	26.90	23.66
qp5	24.58	0.77	26.73	21.78
qp6	23.96	1.06	26.37	19.39
qp7 23.13	1.42	26.20	16.93	
qp8	22.24	1.78	25.51	14.34
qp9	21.74	1.96	25.33	13.78

D/ Tools

- **EDUVPN:** VPN. Enables remote access to the Jetson AGX Orin.
- **GitHub:** Code source. Multiple repositories with programs working off the shelf for different features.
- **Gitlab:** Project Hub. All codes were saved on the different branches.
- **Python Wiki:** Programming tool. Find information on Python functions.
- **Geeks for Geeks:** Programming tool. Information about code implementations on multiple coding languages.
- **W3Schools:**Programming tool. Information about code implementations on multiple coding languages.
- **NumPy:**Programming tool. Information about Numpy libraries.

- **Stack Overflow:** Debugging tool. mostly used after errors appeared, or dependency issues
- **Ask Ubuntu:** Debugging tool. Ubuntu dependencies problems or installations
- **NVIDIA Blogs:** Debugging tool. Provides information on the hardware set-up 3
- **FFMPEG:** Video compression tool. Provides information on the parameters and commands to use FFMPEG codecs.
- **GStreamer:** Video compression tool. Provides information on the parameters and commands to use codecs.
- **Grammarly:** Writing assistance tool. grammar internet plugin.
- **IEEE:** Information source.
- **Elsevier:** Information source.
- **ResearchGate:** Information source
- **Arxiv:** Information source.
- **Sci-Hub:** Article access tool. Unlocks part of the pay-to-read papers
- **Medium:** Information collection . Blogs provide quick and relatively detailed information.
- **Cambridge Dictionary:** Dictionary.
- **Obsidian:** Blog tool. The first entries were made on Obsidian before changing to One Note for the free online service
- **OneNote:** Blog Tool.
- **Wikipedia:** Information tool. First approach to new topics.
- **Trello:** KANBAN tool. Build a schedule
- **Lucid Chart:** Chart designing tool. Used to build the figure 6
- **ChatGPT 3.5[52]:** ChatGPT was used on many layers of the project. In the course of developing this thesis, it was used for debugging, providing first draft information for more accurate research, terminal commands for Linux and Windows finding latex commands,

III. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Léon Glorieux**,

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

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Evaluation of image sequence (video) compression with an embedded processing system for future space applications,

   supervised by Quazi Saimoon Islam and Ric Dengel.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Léon Glorieux
23/12/2023